trame

Kitware Inc.

Jun 03, 2022

CONTENTS

This is the auto-generated API documentation for trame, but the project website is here with additional guides. All classes are listed in the genindex.

CHAPTER

ONE

TRAME

trame.**start**(*layout=None*, *name=None*, *favicon=None*, *on_ready=None*, *port=None*, *debug=False*)

Start the web server for your application

Parameters

- **layout** (*None | str | trame.layouts.**) – UI content that should be used for your application
- **name** (*None | str*) – "Title" that you can see in your tab browser. This will be filled automatically if a trame.layouts.* layout was provided.
- **favicon** (*None | str*) – Relative path to a png image that should be used as favicon
- **port** (*None | Number*) – Port on which the server should run on. Default is 8080. This overrides a port from the command line `--port/-p` option.
- **on_ready** (*None | function*) – Function called once the server is ready
- **debug** (*bool*) – Whether to print debugging information

```
>>> start(on_ready=initialize)
```

trame.**stop**()

trame.**port**()

trame.**state = <trame.internal.state.core.State object>**

This object provides pythonic access to the state

For instance, these getters are the same:

```
>>> field, = get_state("field")
>>> field = state.field
```

As are these setters:

```
>>> update_state("field", value)
>>> state.field = value
```

`get_state()` should be used instead if more than one argument is to be passed, and `update_state()` should be used instead to specify additional arguments (e.g. `force=True`).

The state may also be accessed and updated similar to dictionaries:

```
>>> value = state["field"]
>>> state["field"] = value
>>> state.update({"field": value})
```

This object may be imported via

```
>>> from trame import state
```

trame.**update_state**(*key*, *value=None*, *force=False*)

Updating the current application state that is shared with the Web UI

Parameters

- **key** (*str*) – The key for the value we wish to update
- **value** (*Any*) – The new value
- **force** (*bool*) – Set to True when you want to force push a new or same value to the client.

```
>>> update_state("workload_finished",  True)
```

update_state() may not detect a change if the same reference is passed even if its content has change. You have the option to let the system know that you want to force the update.

```
>>> a = { "x": 1 }
>>> update_state("a", a)
>>> a["x"] = 2
>>> update_state("a", a, force=True)
```

Sometime you may want to update a set of variables at once without triggering any @change callback. To do so, just provide a dictionary. Even if no @change is called, the client will receive the updated modified change.

```
>>> change_set = { "a": 1, "b": 2 }
>>> update_state(change_set)
```

trame.**get_state**(**names*)

Return the list of values of the given state keys or the full state dictionary if no key names were provided.

Parameters **names** (*list[str]*) – List of names of state values to retreive

Return type List[Any] | dict[str, Any]

Returns Either a list of values matching the given state property names or the full state dict

```
>>> greeting, name  = get_state("greeting", "name")
>>> f'{greeting}, {name}!'
"Hello, Trame!"
```

```
>>> greeting, = get_state("greeting")
>>> greeting
"Hello"
```

```
>>> full_state = get_state()
>>> full_state.get("greeting")
"Hello"
```

trame.**flush_state**(**args*)

Force push selected keys of the server state to the client

Parameters **args** (*list[str]*) – Which keys to flush

```
>>> flush_state('myNestedDict')
```

trame.**is_dirty**(**args*)

Check if a set of keys in an @change have been modified

Parameters **args** (*list[str]*) – Which keys to check for modification

Returns True if any of the keys in *args* are modified

```
>>> @change('sound_settings', 'picture_settings')
... def show_changed_settings(sound_settings, picture_settings, **kwargs):
...     if is_dirty('sound_settings'):
...         print(sound_settings)
...     if is_dirty('picture_settings'):
...         print(picture_settings)
```

trame.**is_dirty_all**(**args*)

See whether all keys in an @change have been modified

Parameters **args** (*list[str]*) – Which keys to check for modification

Returns True if all of the keys in *args* are modified

```
>>> @change('sound_settings', 'picture_settings')
... def save_changed_settings(sound_settings, picture_settings, **kwargs):
...     if is_dirty_all('sound_settings', 'picture_settings'):
...         print("Cannot save both sound and picture settings at once")
...         raise
```

trame.**change**(**_args*, ***_kwargs*)

The @change decorator allows us to register a function so that it will be automatically called when any of the given list of state names gets modified.

The decorated function is passed the full state as `**kwargs` when possible. This means you should have a method profile similar to `fn(..., **kwargs)`

Parameters **_args** (*list[str]*) – List of names that your function should listen to

```
>>> @change('settings')
... def show_settings(settings, user, **kwargs):
...     print(settings, "for", user)
```

trame.**trigger**(*name*)

The @trigger decorator allows you to register a function as a trigger with a given name.

Parameters **name** (*str*) – Name which this trigger function should listen to.

<v-btn @click=”blue_button_clicked”>Blue Button</v-btn>

```
>>> @trigger('blue_button_clicked')
... def log_clicks():
...     print("The blue button was clicked")
```

trame.**controller = <trame.internal.triggers.controller.Controller object>**

The controller is a container for function proxies

The function proxies may be used as callbacks even though the function has not yet been defined. The function may also be re-defined. For example:

```
>>> from trame import controller as ctrl
>>> layout = SinglePage("Controller test")
>>> with layout.toolbar:
...     vuetify.VSpacer()
...     vuetify.VBtn("Click Me", click=ctrl.on_click)  # not yet defined
```

```
>>> ctrl.on_click = lambda: print("Hello, Trame!")  # on_click is now defined
```

This can be very useful for large projects where the functions may be defined in separate files after the UI has been constructed, or for re-defining callbacks when conditions in the application change.

trame.**update_layout**(*layout*)

Flush layout to the client

Parameters **layout** (*str | trame.layouts.**) – UI content for your application

```
>>> layout.title.set_text("Workload finished!")
>>> update_layout(layout)
```

trame.**get_cli_parser**()

Run or add args to CLI parser

Returns Parser from argparse

```
>>> parser = get_cli_parser()
>>> parser.add_argument("-o", "--output", help="Working directory")
>>> args, unknown = parser.parse_known_args()
>>> print(args.output)
```

trame.**setup_dev**(**reload_list*, *clear_changes=False*, *clear_triggers=True*)

Set up a development environment for the trame app if –dev was passed as a command line argument. If enabled, a reload button will appear at the bottom of the web browser which will reload the modules that were passed as arguments.

Parameters

- **reload_list** (*python modules*) – positional arguments of the modules to reload when the reload button is pressed.
- **clear_changes** (*bool*) – whether or not to clear changes on reload
- **clear_triggers** (*bool*) – whether or not to clear triggers on reload

Return type bool

Returns whether the program is running in dev mode or not

class trame.**RemoteFile**(*local_path=None*, *remote_url=None*, *local_base=None*)

Bases: trame.internal.utils.remote_data.AbstractRemoteFile

fetch()

class trame.**GoogleDriveFile**(*local_path=None*, *google_id=None*, *local_base=None*)

Bases: trame.internal.utils.remote_data.AbstractRemoteFile

fetch()

class trame.**AssetManager**(*base_path*)

Bases: object

base64(*key*, *file_path=None*)

url(*key*, *file_path*)

txt(*key*, *file_path*)

property assets

get_assets(**keys*)

class trame.**Singleton**(*cls: Type[trame.internal.utils.singleton.T]*)

Bases: Generic[trame.internal.utils.singleton.T]

Singleton decorator

CHAPTER

TWO

TRAME.HTML

This module provides the fundamental elements for HTML and the root class for the `trame.html` submodules.

2.1 trame.html.deckgl

You can find more information about using deck.gl with trame .

class trame.html.deckgl.**Deck**(*name=None*, *deck=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Deck.gl component. See vue-deck docs for more info.

Parameters

- **name** (*str | None*) – Identifier for this element in shared state. Generated if not given
- **deck** – pydeck instance to display
- **mapboxApiKey** – See vue-deck docs for more info
- **tooltip** – See vue-deck docs for more info
- **customLibraries** – See vue-deck docs for more info

static to_jsonInput(*deck*)

Serialize pydeck instance

update(*deck=None*, ***kwargs*)

Change the deck this component displays

Parameters **deck** – pydeck instance to display

2.2 trame.html.markdown

You can find more information about using Markdown with trame .

class trame.html.markdown.**Markdown**(***kwargs*)

Bases: *trame.html.AbstractElement*

Create a markdown viewer element

Parameters **v_model** – Variable name in state

```
>>> component = Markdown(v_model=("document", "**Bold**"))
```

```
>>> content = """
...
... # My document
... 1. First
... 2. Second
... 3. Third
...
... Hello "trame"
... """
>>> component = Markdown(v_model=("document2", content))
```

2.3 trame.html.paraview

These auto-generated docs only show this module's objects, which rely on keyword arguments (***kwargs*) for configuration. You can find more information about using Paraview with trame , or more about these components in the vue-vtk-js documentation .

class trame.html.paraview.**VtkView**(*children=None*, *ref='view'*, ***kwargs*)

Bases: *trame.html.AbstractElement*

reset_camera(***kwargs*)

Move camera to center actors within the frame

class trame.html.paraview.**VtkRemoteView**(*view*, *ref='view'*, ***kwargs*)

Bases: *trame.html.AbstractElement*

The VtkRemoteView component relies on the server for rendering by sending images to the client by binding your vtkRenderWindow to it. This component gives you control over the image size and quality to reduce latency while interacting.

```
>>> remote_view = vtk.vtkRemoteView(
...   view=...,               # Instance of the view (required)
...                           # - VTK: vtkRenderWindow
...                           # - Paraview: viewProxy
...   ref=...,                # Identifier for this component
...   interactive_quality=60, # [0, 100] 0 for fastest render, 100 for best quality
...   interactive_ratio=...,  # [0.1, 1] Image size scale factor while interacting
...   interactor_events=(     # Enable vtk.js interactor events for method binding
...     "events",
...     ['EndAnimation'],
...   ),
...   EndAnimation=end,       # Bind method to the enabled event
...
...   box_selection=True,     # toggle selection box rendering
...   box_selection_change=fn # Bind method to get rect selection
... )
```

static push_image(*view*)

Force image *view* to be pushed to the client

update(***kwargs*)

Force image to be pushed to client

reset_camera(***kwargs*)

replace_view(*new_view*, ***kwargs*)

resize(***kwargs*)

class trame.html.paraview.**VtkLocalView**(*view*, *ref='view'*, ***kwargs*)

Bases: *trame.html.AbstractElement*

The VtkLocalView component relies on the server for defining the vtkRenderWindow but then only the geometry is exchanged with the client. The server does not need a GPU as no rendering is happening on the server. The vtkRenderWindow is only used to retrieve the scene data and parameters (coloring by, representations, ...). By relying on the same vtkRenderWindow, you can easily switch from a `VtkRemoteView` to a `VtkLocalView` or vice-versa. This component gives you controls on how you want to map mouse interaction with the camera. The default setting mimic default VTK interactor style so you will rarely have to override to the `interactor_settings`.

```
>>> local_view = vtk.VtkLocalView(
...   view=...,                 # Instance of the view (required)
...                             # - VTK: vtkRenderWindow
...                             # - Paraview: viewProxy
...   ref=...,                  # Identifier for this component
...   context_name=...,         # Namespace for geometry cache
...   interactor_settings=..., # Options for camera controls. See below.
...   interactor_events=(       # Enable vtk.js interactor events for method binding
...     "events",
...     ['EndAnimation'],
...    ),
...   EndAnimation=end,        # Bind method to the enabled event
...
...   box_selection=True,      # toggle selection box rendering
...   box_selection_change=fn # Bind method to get rect selection
... )
```

update(***kwargs*)

Force geometry to be pushed

reset_camera(***kwargs*)

Move camera to center actors within the frame

replace_view(*new_view*, ***kwargs*)

resize(***kwargs*)

class trame.html.paraview.**VtkRemoteLocalView**(*view*, *enable_rendering=True*, ***kwargs*)

Bases: *trame.html.AbstractElement*

The VtkRemoteLocalView component is a blend of VtkLocalView and VtkRemoteView where the user can choose dynamically which mode they want to be in. When instantiating a VtkRemoteLocalView several variables and triggers will be created for you to more easily control your view.

```
>>> rl_view = vtk.VtkRemoteLocalView(
...   view=...,                 # Instance of the view (required)
...                             # - VTK: vtkRenderWindow
...                             # - Paraview: viewProxy
...   # Just VtkRemoteLocalView params
...   namespace=...,            # Prefix for variables and triggers. See below.␣
↪(required)
...   mode="local",             # Decide between local or remote. See below.
...
```

(continues on next page)

(continued from previous page)

```
...   # VtkRemoteView params
...   **remote_view_params,
...
...   # VtkLocalView params
...   **local_view_params,
... )
```

update_geometry(*reset_camera=False*)
Force update to geometry

update_image(*reset_camera=False*)
Force update to image

set_local_rendering(*local=True*, ***kwargs*)

set_remote_rendering(*remote=True*, ***kwargs*)

update(*reset_camera=False*, ***kwargs*)

replace_view(*new_view*, ***kwargs*)

reset_camera(***kwargs*)

resize(***kwargs*)

property view
Get linked vtkRenderWindow instance

class trame.html.paraview.**VtkAlgorithm**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

class trame.html.paraview.**VtkCellData**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

class trame.html.paraview.**VtkDataArray**(***kwargs*)
Bases: *trame.html.AbstractElement*

class trame.html.paraview.**VtkFieldData**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

class trame.html.paraview.**VtkGeometryRepresentation**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

class trame.html.paraview.**VtkGlyphRepresentation**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

class trame.html.paraview.**VtkMesh**(*name*, *dataset=None*, *field_to_keep=None*, *point_arrays=None*, *cell_arrays=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

set_dataset(*dataset*)
Change this mesh's internal dataset and update shared state

update(***kwargs*)
Propagate changes in internal data to shared state

class trame.html.paraview.**VtkPointData**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

class trame.html.paraview.**VtkPolyData**(*name*, *children=None*, *dataset=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

set_dataset(*dataset*)
Change this polydata's internal dataset and update shared state

update()
Propagate changes in internal data to shared state

class trame.html.paraview.**VtkReader**(***kwargs*)
Bases: *trame.html.AbstractElement*

class trame.html.paraview.**VtkShareDataset**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

2.4 trame.html.simput

These auto-generated docs only show this module's objects, which rely on keyword arguments (***kwargs*) for configuration. You can find more information in the modules section .

class trame.html.simput.**Simput**(*ui_manager*, *domains_manager=None*, *prefix=None*, *children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Simput data management component. This must be set as the root of a layout to provide children with Simput data. See simput docs for more info.

Parameters

- **ui_manager** – See simput docs for more info
- **domains_manager** – See simput docs for more info
- **prefix** (*str | None*) – Constructing a Simput component will set several variables, optionally prefixed by a namespace
- **query** (*str*) – String filtering
- **children** (*str | list[trame.html.*] | trame.html.* | None*) – The children nested within this element

```
>>> layout.root = simput.Simput(ui_manager, prefix="myForm")
```

property controller
Simput helper object

apply(***kwargs*)
Flush modified properties so they can be pushed to their concrete objects

reset(***kwargs*)
Unapply properties

push(*id=None*, *type=None*, *domains=None*, *proxy=None*, ***kwargs*)
Ask server to push data, ui, or constraints

update(*change_set*, ***kwargs*)
List of properties and value to update

```
>>> change_set = [
... {"id":"12", "name":"Radius", "value": 0.75},
... {"id": "12", "name":"Resolution", "value": 24}
... ]
```

refresh(*id=0*, *property=''*, ***kwargs*)

property changeset
All unapplied changesets

property has_changes
Does the changeset have content?

property auto_update
Whether to automatically apply changes

class trame.html.simput.**SimputItem**(*children=None*, *extract=[]*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Simput data display component. This must be child of a Simput component to have access to Simput data. See simput docs for more info.

Parameters

- **itemId** (*str*) – The simput id of the data to display
- **extract** (*list[str]*) – Columns to make available from this component to its children
- **no_ui** (*bool*) – Whether to show simput template UI
- **children** (*str | list[trame.html.*] | trame.html.* | None*) – The children nested within this element

Events

Parameters dirty (*function*) – Function to call when itemId is changed

2.5 trame.html.vega

You can find more information about using Vega with trame .

class trame.html.vega.**VegaEmbed**(*name=None*, *chart=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vega component. See vega docs for more info.

Parameters

- **chart** – The chart to display. Defaults to None.
- **name** – The identifier of this components data in shared state. Generated if not given.

static altair_to_spec(*chart*, ***kwargs*)
Serialize altair chart

update(*chart=None*, ***kwargs*)
Change which chart is displayed

Parameters chart – The chart to display. Defaults to None.

2.6 trame.html.vtk

These auto-generated docs only show this module's objects, which rely on keyword arguments (***kwargs*) for configuration. You can find more information in the trame modules section , or more about these components in the vue-vtk-js documentation .

class trame.html.vtk.**VtkView**(*children=None*, *ref='view'*, ***kwargs*)

Bases: *trame.html.AbstractElement*

reset_camera(***kwargs*)

Move camera to center actors within the frame

class trame.html.vtk.**VtkRemoteView**(*view*, *ref='view'*, ***kwargs*)

Bases: *trame.html.AbstractElement*

The VtkRemoteView component relies on the server for rendering by sending images to the client by binding your vtkRenderWindow to it. This component gives you control over the image size and quality to reduce latency while interacting.

```
>>> remote_view = vtk.vtkRemoteView(
...   view=...,               # Instance of the view (required)
...                           # - VTK: vtkRenderWindow
...                           # - Paraview: viewProxy
...   ref=...,                # Identifier for this component
...   interactive_quality=60, # [0, 100] 0 for fastest render, 100 for best quality
...   interactive_ratio=...,  # [0.1, 1] Image size scale factor while interacting
...   interactor_events=(     # Enable vtk.js interactor events for method binding
...     "events",
...     ['EndAnimation'],
...   ),
...   EndAnimation=end,       # Bind method to the enabled event
...
...   box_selection=True,     # toggle selection box rendering
...   box_selection_change=fn # Bind method to get rect selection
... )
```

static push_image(*view*)

Force image *view* to be pushed to the client

update(***kwargs*)

Force image to be pushed to client

reset_camera(***kwargs*)

replace_view(*new_view*, ***kwargs*)

resize(***kwargs*)

class trame.html.vtk.**VtkLocalView**(*view*, *ref='view'*, ***kwargs*)

Bases: *trame.html.AbstractElement*

The VtkLocalView component relies on the server for defining the vtkRenderWindow but then only the geometry is exchanged with the client. The server does not need a GPU as no rendering is happening on the server. The vtkRenderWindow is only used to retrieve the scene data and parameters (coloring by, representations, ...). By relying on the same vtkRenderWindow, you can easily switch from a `VtkRemoteView` to a `VtkLocalView` or vice-versa. This component gives you controls on how you want to map mouse interaction with the camera. The default setting mimic default VTK interactor style so you will rarely have to override to the `interactor_settings`.

```
>>> local_view = vtk.VtkLocalView(
...   view=...,                # Instance of the view (required)
...                            # - VTK: vtkRenderWindow
...                            # - Paraview: viewProxy
...   ref=...,                 # Identifier for this component
...   context_name=...,        # Namespace for geometry cache
...   interactor_settings=..., # Options for camera controls. See below.
...   interactor_events=(      # Enable vtk.js interactor events for method binding
...     "events",
...     ['EndAnimation'],
...    ),
...   EndAnimation=end,       # Bind method to the enabled event
...
...   box_selection=True,     # toggle selection box rendering
...   box_selection_change=fn # Bind method to get rect selection
... )
```

update(***kwargs*)
Force geometry to be pushed

reset_camera(***kwargs*)
Move camera to center actors within the frame

replace_view(*new_view*, ***kwargs*)

resize(***kwargs*)

class trame.html.vtk.**VtkRemoteLocalView**(*view*, *enable_rendering=True*, ***kwargs*)
Bases: *trame.html.AbstractElement*

The VtkRemoteLocalView component is a blend of VtkLocalView and VtkRemoteView where the user can choose dynamically which mode they want to be in. When instantiating a VtkRemoteLocalView several variables and triggers will be created for you to more easily control your view.

```
>>> rl_view = vtk.VtkRemoteLocalView(
...   view=...,                # Instance of the view (required)
...                            # - VTK: vtkRenderWindow
...                            # - Paraview: viewProxy
...   # Just VtkRemoteLocalView params
...   namespace=...,           # Prefix for variables and triggers. See below.␣
↪(required)
...   mode="local",            # Decide between local or remote. See below.
...
...   # VtkRemoteView params
...   **remote_view_params,
...
...   # VtkLocalView params
...   **local_view_params,
... )
```

update_geometry(*reset_camera=False*)
Force update to geometry

update_image(*reset_camera=False*)
Force update to image

set_local_rendering(*local=True*, ***kwargs*)

set_remote_rendering(*remote=True*, ***kwargs*)

update(*reset_camera=False*, ***kwargs*)

replace_view(*new_view*, ***kwargs*)

reset_camera(***kwargs*)

resize(***kwargs*)

property view
Get linked vtkRenderWindow instance

class trame.html.vtk.**VtkAlgorithm**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

class trame.html.vtk.**VtkCellData**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

class trame.html.vtk.**VtkDataArray**(***kwargs*)
Bases: *trame.html.AbstractElement*

class trame.html.vtk.**VtkFieldData**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

class trame.html.vtk.**VtkGeometryRepresentation**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

class trame.html.vtk.**VtkGlyphRepresentation**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

class trame.html.vtk.**VtkMesh**(*name*, *dataset=None*, *field_to_keep=None*, *point_arrays=None*, *cell_arrays=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

set_dataset(*dataset*)
Change this mesh's internal dataset and update shared state

update(***kwargs*)
Propagate changes in internal data to shared state

class trame.html.vtk.**VtkPointData**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

class trame.html.vtk.**VtkPolyData**(*name*, *children=None*, *dataset=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

set_dataset(*dataset*)
Change this polydata's internal dataset and update shared state

update()
Propagate changes in internal data to shared state

class trame.html.vtk.**VtkReader**(***kwargs*)
Bases: *trame.html.AbstractElement*

class trame.html.vtk.**VtkShareDataset**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

2.7 trame.html.vuetify

You can find more information about using Vuetify with trame .

trame.html.vuetify.**dataframe_to_grid**(*dataframe*, *options={}*)
Transform a dataframe for use with a VDataTable

Parameters

- **dataframe** – A pandas dataframe
- **options** – Control which columns are sortable, filterable, grouped, aligned, etc. A dictionary where keys are the columns from the dataframe and values are Vuetify DataTableHeader objects. See more info .

```
>>> headers, rows = vuetify.dataframe_to_grid(dataframe)
>>> VDataTable(
...     headers=("table_headers", headers),
...     items=("table_rows", rows))
```

class trame.html.vuetify.**VApp**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VApp component. See more info and examples .

Parameters **id** – Sets the DOM id on the component

class trame.html.vuetify.**VAppBar**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VAppBar component. See more info and examples .

Parameters

- **absolute** – Applies position: absolute to the component.
- **app** – See description .
- **bottom** – Aligns the component towards the bottom.
- **clipped_left** – Designates that the application's *v-navigation-drawer* that is positioned on the left is below the app-bar.
- **clipped_right** – Designates that the application's *v-navigation-drawer* that is positioned on the right is below the app-bar.
- **collapse** – Puts the toolbar into a collapsed state reducing its maximum width.
- **collapse_on_scroll** – Puts the app-bar into a collapsed state when scrolling.
- **color** – See description .
- **dark** – See description .
- **dense** – Reduces the height of the toolbar content to 48px (96px when using the **prominent** prop).
- **elevate_on_scroll** – Elevates the app-bar when scrolling.
- **elevation** – See description .
- **extended** – Use this prop to increase the height of the toolbar _without_ using the *extension* slot for adding content. May be used in conjunction with the **extension-height** prop, and any

of the other props that affect the height of the toolbar, e.g. **prominent**, **dense**, etc., **WITH THE EXCEPTION** of **height**.

- **extension_height** – Specify an explicit height for the *extension* slot.
- **fade_img_on_scroll** – When using the **src** prop or *img* slot, will fade the image when scrolling.
- **fixed** – Applies **position: fixed** to the component.
- **flat** – Removes the toolbar's box-shadow.
- **floating** – Applies **display: inline-flex** to the component.
- **height** – Designates a specific height for the toolbar. Overrides the heights imposed by other props, e.g. **prominent**, **dense**, **extended**, etc.
- **hide_on_scroll** – Hides the app-bar when scrolling. Will still show the *extension* slot.
- **inverted_scroll** – Hides the app-bar when scrolling down and displays it when scrolling up.
- **light** – Applies the light theme variant to the component.
- **max_height** – Sets the maximum height for the component.
- **max_width** – Sets the maximum width for the component.
- **min_height** – Sets the minimum height for the component.
- **min_width** – Sets the minimum width for the component.
- **outlined** – Removes elevation (box-shadow) and adds a *thin* border.
- **prominent** – Increases the height of the toolbar content to 128px.
- **rounded** – See description .
- **scroll_off_screen** – Hides the app-bar when scrolling. Will **NOT** show the *extension* slot.
- **scroll_target** – Designates the element to target for scrolling events. Uses *window* by default.
- **scroll_threshold** – The amount of scroll distance down before **hide-on-scroll** activates.
- **shaped** – Applies a large border radius on the top left and bottom right of the card.
- **short** – Reduce the height of the toolbar content to 56px (112px when using the **prominent** prop).
- **shrink_on_scroll** – Shrinks a **prominent** toolbar to a **dense** or **short** (default) one when scrolling.
- **src** – Image source. See *v-img* for details
- **tag** – Specify a custom tag used on the root element.
- **tile** – Removes the component's **border-radius**.
- **value** – Controls whether the component is visible or hidden.
- **width** – Sets the width for the component.

class trame.html.vuetify.**VAppBarNavIcon**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VAppBarNavIcon component. See more info and examples .

class trame.html.vuetify.**VAppBarTitle**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VAppBarTitle component. See more info and examples .

class trame.html.vuetify.**VAlert**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VAlert component. See more info and examples .

Parameters

- **border** – Puts a border on the alert. Accepts **top | right | bottom | left**.
- **close_icon** – Change the default icon used for **dismissible** alerts.
- **close_label** – See description .
- **color** – See description .
- **colored_border** – Applies the defined **color** to the alert's border.
- **dark** – See description .
- **dense** – Decreases component's height.
- **dismissible** – Adds a close icon that can hide the alert.
- **elevation** – See description .
- **height** – Sets the height for the component.
- **icon** – Designates a specific icon.
- **light** – Applies the light theme variant to the component.
- **max_height** – Sets the maximum height for the component.
- **max_width** – Sets the maximum width for the component.
- **min_height** – Sets the minimum height for the component.
- **min_width** – Sets the minimum width for the component.
- **mode** – See description .
- **origin** – See description .
- **outlined** – Makes the background transparent and applies a thin border.
- **prominent** – Displays a larger vertically centered icon to draw more attention.
- **rounded** – See description .
- **shaped** – Applies a large border radius on the top left and bottom right of the card.
- **tag** – Specify a custom tag used on the root element.
- **text** – Applies the defined **color** to text and a low opacity background of the same.
- **tile** – Removes the component's border-radius.
- **transition** – See description .
- **type** – Specify a **success**, **info**, **warning** or **error** alert. Uses the contextual color and has a pre-defined icon.
- **value** – Controls whether the component is visible or hidden.
- **width** – Sets the width for the component.

Events

Parameters **input** – The updated bound model

class trame.html.vuetify.**VAutocomplete**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VAutocomplete component. See more info and examples .

Parameters

- **allow_overflow** – Allow the menu to overflow off the screen
- **append_icon** – Appends an icon to the component, uses the same syntax as *v-icon*
- **append_outer_icon** – Appends an icon to the outside the component's input, uses same syntax as *v-icon*
- **attach** – Specifies which DOM element that this component should detach to. String can be any valid querySelector and Object can be any valid Node. This will attach to the root *v-app* component by default.
- **auto_select_first** – When searching, will always highlight the first option
- **autofocus** – Enables autofocus
- **background_color** – Changes the background-color of the input
- **cache_items** – Keeps a local _unique_ copy of all items that have been passed through the **items** prop.
- **chips** – Changes display of selections to chips
- **clear_icon** – Applied when using **clearable** and the input is dirty
- **clearable** – Add input clear functionality, default icon is Material Design Icons **mdi-clear**
- **color** – See description .
- **counter** – Creates counter for input length; if no number is specified, it defaults to 25. Does not apply any validation.
- **counter_value** –
- **dark** – See description .
- **deletable_chips** – Adds a remove icon to selected chips
- **dense** – Reduces the input height
- **disable_lookup** – Disables keyboard lookup
- **disabled** – Disables the input
- **eager** – Will force the components content to render on mounted. This is useful if you have content that will not be rendered in the DOM that you want crawled for SEO.
- **error** – Puts the input in a manual error state
- **error_count** – The total number of errors that should display at once
- **error_messages** – Puts the input in an error state and passes through custom error messages. Will be combined with any validations that occur from the **rules** prop. This field will not trigger validation
- **filled** – Applies the alternate filled input style
- **filter** – See description .

- **flat** – Removes elevation (shadow) added to element when using the **solo** or **solo-inverted** props
- **full_width** – Designates input type as full-width
- **height** – Sets the height of the input
- **hide_details** – Hides hint and validation errors. When set to *auto* messages will be rendered only if there's a message (hint, error message, counter value etc) to display
- **hide_no_data** – Hides the menu when there are no options to show. Useful for preventing the menu from opening before results are fetched asynchronously. Also has the effect of opening the menu when the *items* array changes if not already open.
- **hide_selected** – Do not display in the select menu items that are already selected
- **hide_spin_buttons** – Hides spin buttons on the input when type is set to *number*.
- **hint** – Hint text
- **id** – Sets the DOM id on the component
- **item_color** – Sets color of selected items
- **item_disabled** – Set property of **items**'s disabled value
- **item_text** – Set property of **items**'s text value
- **item_value** – See description .
- **items** – Can be an array of objects or array of strings. When using objects, will look for a text, value and disabled keys. This can be changed using the **item-text**, **item-value** and **item-disabled** props. Objects that have a **header** or **divider** property are considered special cases and generate a list header or divider; these items are not selectable.
- **label** – Sets input label
- **light** – Applies the light theme variant to the component.
- **loader_height** – Specifies the height of the loader
- **loading** – Displays linear progress bar. Can either be a String which specifies which color is applied to the progress bar (any material color or theme color - **primary**, **secondary**, **success**, **info**, **warning**, **error**) or a Boolean which uses the component **color** (set by color prop - if it's supported by the component) or the primary color
- **menu_props** – Pass props through to the *v-menu* component. Accepts either a string for boolean props *menu-props="auto, overflowY"*, or an object *:menu-props="{ auto: true, overflowY: true }"*
- **messages** – Displays a list of messages or message if using a string
- **multiple** – Changes select to multiple. Accepts array for value
- **no_data_text** – Display text when there is no data
- **no_filter** – Do not apply filtering when searching. Useful when data is being filtered server side
- **open_on_clear** – When using the **clearable** prop, once cleared, the select menu will either open or stay open, depending on the current state
- **outlined** – Applies the outlined style to the input
- **persistent_hint** – Forces hint to always be visible
- **persistent_placeholder** – Forces placeholder to always be visible

- **placeholder** – Sets the input's placeholder text
- **prefix** – Displays prefix text
- **prepend_icon** – Prepends an icon to the component, uses the same syntax as *v-icon*
- **prepend_inner_icon** – Prepends an icon inside the component's input, uses the same syntax as *v-icon*
- **readonly** – Puts input in readonly state
- **return_object** – Changes the selection behavior to return the object directly rather than the value specified with **item-value**
- **reverse** – Reverses the input orientation
- **rounded** – Adds a border radius to the input
- **rules** – Accepts a mixed array of types *function*, *boolean* and *string*. Functions pass an input value as an argument and must return either *true* / *false* or a *string* containing an error message. The input field will enter an error state if a function returns (or any value in the array contains) *false* or is a *string*
- **search_input** – Search value. Can be used with *.sync* modifier.
- **shaped** – Round if *outlined* and increase *border-radius* if *filled*. Must be used with either *outlined* or *filled*
- **single_line** – Label does not move on focus/dirty
- **small_chips** – Changes display of selections to chips with the **small** property
- **solo** – Changes the style of the input
- **solo_inverted** – Reduces element opacity until focused
- **success** – Puts the input in a manual success state
- **success_messages** – Puts the input in a success state and passes through custom success messages.
- **suffix** – Displays suffix text
- **type** – Sets input type
- **validate_on_blur** – Delays validation until blur event
- **value** – The input's value
- **value_comparator** – See description .

Events

Parameters

- **blur** – Emitted when the input is blurred
- **change** – Emitted when the input is changed by user interaction
- **click_append** – Emitted when appended icon is clicked
- **click_append_outer** – Emitted when appended outer icon is clicked
- **click_clear** – Emitted when clearable icon clicked
- **click_prepend** – Emitted when prepended icon is clicked
- **click_prepend_inner** – Emitted when prepended inner icon is clicked

- **focus** – Emitted when component is focused
- **input** – The updated bound model
- **keydown** – Emitted when **any** key is pressed
- **update_error** – The *error.sync* event
- **update_list_index** – Emitted when menu item is selected using keyboard arrows
- **update_search_input** – The *search-input.sync* event

class trame.html.vuetify.**VAvatar**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VAvatar component. See more info and examples .

Parameters

- **color** – See description .
- **height** – Sets the height for the component.
- **left** – See description .
- **max_height** – Sets the maximum height for the component.
- **max_width** – Sets the maximum width for the component.
- **min_height** – Sets the minimum height for the component.
- **min_width** – Sets the minimum width for the component.
- **right** – See description .
- **rounded** – See description .
- **size** – Sets the height and width of the component.
- **tile** – Removes the component's **border-radius**.
- **width** – Sets the width for the component.

class trame.html.vuetify.**VBadge**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VBadge component. See more info and examples .

Parameters

- **avatar** – Removes badge padding for the use of the *v-avatar* in the **badge** slot.
- **bordered** – Applies a **2px** by default and **1.5px** border around the badge when using the **dot** property.
- **bottom** – Aligns the component towards the bottom.
- **color** – See description .
- **content** – Any content you want injected as text into the badge.
- **dark** – See description .
- **dot** – Reduce the size of the badge and hide its contents
- **icon** – Designates a specific icon used in the badge.
- **inline** – Moves the badge to be inline with the wrapping element. Supports the usage of the **left** prop.

- **label** – The **aria-label** used for the badge
- **left** – Aligns the component towards the left.
- **light** – Applies the light theme variant to the component.
- **mode** – See description .
- **offset_x** – Offset the badge on the x-axis.
- **offset_y** – Offset the badge on the y-axis.
- **origin** – See description .
- **overlap** – Overlaps the slotted content on top of the component.
- **tile** – Removes the component's border-radius.
- **transition** – See description .
- **value** – Controls whether the component is visible or hidden.

class trame.html.vuetify.**VBanner**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VBanner component. See more info and examples .

Parameters

- **app** – When used inside of *v-main*, will calculate top based upon application *v-toolbar* and *v-system-bar*.
- **color** – See description .
- **dark** – See description .
- **elevation** – See description .
- **height** – Sets the height for the component.
- **icon** – Designates a specific icon.
- **icon_color** – Designates a specific icon color.
- **light** – Applies the light theme variant to the component.
- **max_height** – Sets the maximum height for the component.
- **max_width** – Sets the maximum width for the component.
- **min_height** – Sets the minimum height for the component.
- **min_width** – Sets the minimum width for the component.
- **mobile_breakpoint** – Sets the designated mobile breakpoint for the component.
- **outlined** – Removes elevation (box-shadow) and adds a *thin* border.
- **rounded** – See description .
- **shaped** – Applies a large border radius on the top left and bottom right of the card.
- **single_line** – Forces the banner onto a single line.
- **sticky** – See description .
- **tag** – Specify a custom tag used on the root element.
- **tile** – Removes the component's **border-radius**.
- **value** – Controls whether the component is visible or hidden.

- **width** – Sets the width for the component.

class trame.html.vuetify.**VBottomNavigation**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VBottomNavigation component. See more info and examples .

Parameters

- **absolute** – Applies **position: absolute** to the component.
- **active_class** – See description .
- **app** – See description .
- **background_color** – Changes the background-color for the component.
- **color** – See description .
- **dark** – See description .
- **fixed** – Applies **position: fixed** to the component.
- **grow** – See description .
- **height** – Sets the height for the component.
- **hide_on_scroll** – Will transition the navigation off screen when scrolling up.
- **horizontal** – See description .
- **input_value** – Controls whether the component is visible or hidden. Supports the **.sync** modifier.
- **light** – Applies the light theme variant to the component.
- **mandatory** – Forces a value to always be selected (if available).
- **max_height** – Sets the maximum height for the component.
- **max_width** – Sets the maximum width for the component.
- **min_height** – Sets the minimum height for the component.
- **min_width** – Sets the minimum width for the component.
- **scroll_target** – Designates the element to target for scrolling events. Uses *window* by default.
- **scroll_threshold** – The amount of scroll distance down before **hide-on-scroll** activates.
- **shift** – See description .
- **tag** – Specify a custom tag used on the root element.
- **value** – See description .
- **width** – Sets the width for the component.

Events

Parameters

- **change** – The value of currently selected button. If no value is assigned, will be the current index of the button.
- **update_input_value** – The event used for *input-value.sync*.

class trame.html.vuetify.**VBottomSheet**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VBottomSheet component. See more info and examples .

Parameters

- **activator** – Designate a custom activator when the *activator* slot is not used. String can be any valid querySelector and Object can be any valid Node.
- **attach** – Specifies which DOM element that this component should detach to. String can be any valid querySelector and Object can be any valid Node. This will attach to the root *v-app* component by default.
- **close_delay** – Milliseconds to wait before closing component.
- **content_class** – Applies a custom class to the detached element. This is useful because the content is moved to the beginning of the *v-app* component (unless the **attach** prop is provided) and is not targetable by classes passed directly on the component.
- **dark** – See description .
- **disabled** – Disables the ability to open the component.
- **eager** – Will force the components content to render on mounted. This is useful if you have content that will not be rendered in the DOM that you want crawled for SEO.
- **fullscreen** – Changes layout for fullscreen display.
- **hide_overlay** – Hides the display of the overlay.
- **inset** – Reduces the sheet content maximum width to 70%.
- **internal_activator** – Detaches the menu content inside of the component as opposed to the document.
- **light** – Applies the light theme variant to the component.
- **max_width** – Sets the maximum width for the component.
- **no_click_animation** – Disables the bounce effect when clicking outside of a *v-dialog*'s content when using the **persistent** prop.
- **open_delay** – Milliseconds to wait before opening component.
- **open_on_focus** –
- **open_on_hover** – Designates whether component should activate when its activator is hovered.
- **origin** – See description .
- **overlay_color** – Sets the overlay color.
- **overlay_opacity** – Sets the overlay opacity.
- **persistent** – Clicking outside of the element or pressing **esc** key will not deactivate it.
- **retain_focus** – Tab focus will return to the first child of the dialog by default. Disable this when using external tools that require focus such as TinyMCE or vue-clipboard.
- **return_value** –
- **scrollable** – See description .
- **transition** – See description .
- **value** – Controls whether the component is visible or hidden.

- **width** – Sets the width for the component.

class trame.html.vuetify.**VBreadcrumbs**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VBreadcrumbs component. See more info and examples .

Parameters

- **dark** – See description .
- **divider** – Specifies the dividing character between items.
- **items** – An array of objects for each breadcrumb.
- **large** – Increase the font-size of the breadcrumb item text to 16px (14px default).
- **light** – Applies the light theme variant to the component.

class trame.html.vuetify.**VBreadcrumbsItem**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VBreadcrumbsItem component. See more info and examples .

Parameters

- **active_class** – See description .
- **append** – See description .
- **disabled** – Removes the ability to click or target the component.
- **exact** – See description .
- **exact_active_class** – See description .
- **exact_path** – See description .
- **href** – Designates the component as anchor and applies the **href** attribute.
- **link** – Designates that the component is a link. This is automatic when using the **href** or **to** prop.
- **nuxt** – See description .
- **replace** – See description .
- **ripple** – See description .
- **tag** – Specify a custom tag used on the root element.
- **target** – Designates the target attribute. This should only be applied when using the **href** prop.
- **to** – See description .

class trame.html.vuetify.**VBreadcrumbsDivider**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VBreadcrumbsDivider component. See more info and examples .

Parameters **tag** – Specify a custom tag used on the root element.

class trame.html.vuetify.**VBtn**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VBtn component. See more info and examples .

Parameters

- **absolute** – Applies **position: absolute** to the component.
- **active_class** – See description .
- **append** – See description .
- **block** – Expands the button to 100% of available space.
- **bottom** – Aligns the component towards the bottom.
- **color** – See description .
- **dark** – See description .
- **depressed** – Removes the button box shadow.
- **disabled** – Removes the ability to click or target the component.
- **elevation** – See description .
- **exact** – See description .
- **exact_active_class** – See description .
- **exact_path** – See description .
- **fab** – Designates the button as a floating-action-button. Button will become _round_.
- **fixed** – Applies **position: fixed** to the component.
- **height** – Sets the height for the component.
- **href** – Designates the component as anchor and applies the **href** attribute.
- **icon** – Designates the button as icon. Button will become _round_ and applies the **text** prop.
- **input_value** – Controls the button's active state.
- **large** – Makes the component large.
- **left** – Aligns the component towards the left. This should be used with the **absolute** or **fixed** props.
- **light** – Applies the light theme variant to the component.
- **link** – Designates that the component is a link. This is automatic when using the **href** or **to** prop.
- **loading** – Adds a loading icon animation.
- **max_height** – Sets the maximum height for the component.
- **max_width** – Sets the maximum width for the component.
- **min_height** – Sets the minimum height for the component.
- **min_width** – Sets the minimum width for the component.
- **nuxt** – See description .
- **outlined** – Makes the background transparent and applies a thin border.
- **plain** – Removes the default background change applied when hovering over the button.
- **replace** – See description .
- **retain_focus_on_click** – Don't blur on click.
- **right** – Aligns the component towards the right. This should be used with the **absolute** or **fixed** props.

- **ripple** – See description .
- **rounded** – Applies a large border radius on the button.
- **shaped** – Applies a large border radius on the top left and bottom right of the card.
- **small** – Makes the component small.
- **tag** – Specify a custom tag used on the root element.
- **target** – Designates the target attribute. This should only be applied when using the **href** prop.
- **text** – Makes the background transparent. When using the **color** prop, the color will be applied to the button text instead of the background.
- **tile** – Removes the component's **border-radius**.
- **to** – See description .
- **top** – Aligns the content towards the top.
- **type** – Set the button's **type** attribute.
- **value** – Controls whether the component is visible or hidden.
- **width** – Sets the width for the component.
- **x_large** – Makes the component extra large.
- **x_small** – Makes the component extra small.

class trame.html.vuetify.**VBtnToggle**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VBtnToggle component. See more info and examples .

Parameters

- **active_class** – The **active-class** applied to children when they are activated.
- **background_color** – Changes the background-color for the component.
- **borderless** – Removes the group's border.
- **color** – See description .
- **dark** – See description .
- **dense** – Reduces the button size and padding.
- **group** – See description .
- **light** – Applies the light theme variant to the component.
- **mandatory** – Forces a value to always be selected (if available).
- **max** – Sets a maximum number of selections that can be made.
- **multiple** – Allow multiple selections. The **value** prop must be an _array_.
- **rounded** – Round edge buttons
- **shaped** – Applies a large border radius on the top left and bottom right of the card.
- **tag** – Specify a custom tag used on the root element.
- **tile** – Removes the component's border-radius.
- **value** – The designated model value for the component.

- **value_comparator** – Apply a custom value comparator function

Events

Parameters **change** – Emitted when the input is changed by user interaction

class trame.html.vuetify.**VCalendar**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VCalendar component. See more info and examples .

Parameters

- **categories** – Specifies what categories to display in the *category* view. This controls the order of the categories as well. If the calendar uses events any categories specified in those events not specified in this value are dynamically rendered in the view unless *category-hide-dynamic* is true.
- **category_days** – The number of days to render in the *category* view.
- **category_for_invalid** – The category to place events in that have invalid categories. A category is invalid when it is not a string. By default events without a category are not displayed until this value is specified.
- **category_hide_dynamic** – Sets whether categories specified in an event should be hidden if it's not defined in *categories*.
- **category_show_all** – Set whether the *category* view should show all defined *categories* even if there are no events for a category.
- **category_text** – If categories is a list of objects, you can use this to determine what property to print out as the category text on the calendar. You can provide a function to do some logic or just define the prop name. It's similar to item-text on v-select
- **color** – See description .
- **dark** – See description .
- **day_format** – Formats day of the month string that appears in a day to a specified locale
- **end** – The ending date on the calendar (inclusive) in the format of *YYYY-MM-DD*. This may be ignored depending on the *type* of the calendar.
- **event_category** – Set property of *event*'s category. Instead of a property a function can be given which takes an event and returns the category.
- **event_color** – A background color for all events or a function which accepts an event object passed to the calendar to return a color.
- **event_end** – Set property of *event*'s end timestamp.
- **event_height** – The height of an event in pixels in the *month* view and at the top of the *day* views.
- **event_margin_bottom** – Margin bottom for event
- **event_more** – Whether the more 'button' is displayed on a calendar with too many events in a given day. It will say something like '5 more' and when clicked generates a *click:more* event.
- **event_more_text** – The text to display in the more 'button' given the number of hidden events.

- **event_name** – Set property of *event*'s displayed name, or a function which accepts an event object passed to the calendar as the first argument and a flag signalling whether the name is for a timed event (true) or an event over a day.
- **event_overlap_mode** – One of *stack*, *column*, or a custom render function
- **event_overlap_threshold** – A value in minutes that's used to determine whether two timed events should be placed in column beside each other or should be treated as slightly overlapping events.
- **event_ripple** – Applies the *v-ripple* directive.
- **event_start** – Set property of *event*'s start timestamp.
- **event_text_color** – A text color for all events or a function which accepts an event object passed to the calendar to return a color.
- **event_timed** – If Dates or milliseconds are used as the start or end timestamp of an event, this prop can be a string to a property on the event that is truthy if the event is a timed event or a function which takes the event and returns a truthy value if the event is a timed event.
- **events** – An array of event objects with a property for a start timestamp and optionally a name and end timestamp. If an end timestamp is not given, the value of start will be used. If no name is given, you must provide an implementation for the *event* slot.
- **first_interval** – The first interval to display in the *day* view. If *intervalMinutes* is set to 60 and this is set to 9 the first time in the view is 9am.
- **first_time** – The first time to display in the *day* view. If specified, this overwrites any *firstInterval* value specified. This can be the number of minutes since midnight, a string in the format of *HH:mm*, or an object with number properties hour and minute.
- **hide_header** – If the header at the top of the *day* view should be visible.
- **interval_count** – The number of intervals to display in the *day* view.
- **interval_format** – Formats time of day string that appears in the interval gutter of the *day* and *week* view to specified locale
- **interval_height** – The height of an interval in pixels in the *day* view.
- **interval_minutes** – The number of minutes the intervals are in the *day* view. A common interval is 60 minutes so the intervals are an hour.
- **interval_style** – Returns CSS styling to apply to the interval.
- **interval_width** – The width of the interval gutter on the left side in the *day* view.
- **light** – Applies the light theme variant to the component.
- **locale** – The locale of the calendar.
- **locale_first_day_of_year** – Sets the day that determines the first week of the year, starting with 0 for **Sunday**. For ISO 8601 this should be 4.
- **max_days** – The maximum number of days to display in the custom calendar if an *end* day is not set.
- **min_weeks** – The minimum number of weeks to display in the *month* or *week* view.
- **month_format** – Formats month string that appears in a day to specified locale
- **now** – Override the day & time which is considered now. This is in the format of *YYYY-MM-DD hh:mm:ss*. The calendar is styled according to now.

- **short_intervals** – If true, the intervals in the *day* view will be 9 AM as opposed to 09:00 AM
- **short_months** – Whether the short versions of a month should be used (Jan vs January).
- **short_weekdays** – Whether the short versions of a weekday should be used (Mon vs Monday).
- **show_interval_label** – Checks if a given day and time should be displayed in the interval gutter of the *day* view.
- **show_month_on_first** – Whether the name of the month should be displayed on the first day of the month.
- **show_week** – Whether week numbers should be displayed when using the *month* view.
- **start** – The starting date on the calendar (inclusive) in the format of *YYYY-MM-DD*. This may be ignored depending on the *type* of the calendar.
- **type** – A string which is one of *month*, *week*, *day*, *4day*, *custom-weekly*, *custom-daily*, and *category*. The custom types look at the *start* and *end* dates passed to the component as opposed to the *value*.
- **value** – A date in the format of *YYYY-MM-DD* which determines what span of time for the calendar.
- **weekday_format** – Formats day of the week string that appears in the header to specified locale
- **weekdays** – Specifies which days of the week to display. To display Monday through Friday only, a value of *[1, 2, 3, 4, 5]* can be used. To display a week starting on Monday a value of *[1, 2, 3, 4, 5, 6, 0]* can be used.

Events

Parameters

- **change** – The range of days displayed on the calendar changed. This is triggered on initialization. The event passed is an object with start and end date objects.
- **click_date** – The click event on the day of the month link. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **click_day** – The click event on a day. The event passed is the day object. Native mouse event is passed as a second argument.
- **click_day_category** – The click event on a day in the *category* view. The event passed is the day object. Native mouse event is passed as a second argument.
- **click_event** – The click event on a specific event. The event passed is the day & time object.
- **click_interval** – The click event at a specific interval label in the *day* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **click_more** – The click event on the *X more* button on views with too many events in a day. Native mouse event is passed as a second argument.
- **click_time** – The click event at a specific time in the *day* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **click_time_category** – The click event at a specific time in the *category* view. The event passed is the day & time object. Native mouse event is passed as a second argument.

- **contextmenu_date** – The right-click event on the day of the month link. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **contextmenu_day** – The right-click event on a day. The event passed is the day object. Native mouse event is passed as a second argument.
- **contextmenu_day_category** – The right-click event on a day in the *category* view. The event passed is the day object. Native mouse event is passed as a second argument.
- **contextmenu_event** – The right-click event on an event. The event passed is the day & time object.
- **contextmenu_interval** – The right-click event at a specific interval label in the *day* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **contextmenu_time** – The right-click event at a specific time in the *day* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **contextmenu_time_category** – The right-click event at a specific time in the *category* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **input** – An alias to the *click:date* event used to support v-model.
- **mousedown_day** – The mousedown event on a day. The event passed is the day object. Native mouse event is passed as a second argument.
- **mousedown_day_category** – The mousedown event on a day in the *category* view. The event passed is the day object. Native mouse event is passed as a second argument.
- **mousedown_event** – The mousedown event on an event. The event passed is the day & time object.
- **mousedown_interval** – The mousedown event at a specific interval label in the *day* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **mousedown_time** – The mousedown event at a specific time in the *day* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **mousedown_time_category** – The mousedown event at a specific time in the *category* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **mouseenter_day** – The mouseenter event on a day. The event passed is the day object. Native mouse event is passed as a second argument.
- **mouseenter_day_category** – The mouseenter event on a day in the *category* view. The event passed is the day object. Native mouse event is passed as a second argument.
- **mouseenter_event** – The mouseenter event on an event. The event passed is the day & time object.
- **mouseenter_interval** – The mouseenter event at a specific interval label in the *day* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **mouseenter_time** – The mouseenter event at a specific time in the *day* view. The event passed is the day & time object. Native mouse event is passed as a second argument.

- **mouseenter_time_category** – The mouseenter event at a specific time in the *category* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **mouseleave_day** – The mouseleave event on a day. The event passed is the day object. Native mouse event is passed as a second argument.
- **mouseleave_day_category** – The mouseleave event on a day in the *category* view. The event passed is the day object. Native mouse event is passed as a second argument.
- **mouseleave_event** – The mouseleave event on an event. The event passed is the day & time object.
- **mouseleave_interval** – The mouseleave event at a specific interval label in the *day* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **mouseleave_time** – The mouseleave event at a specific time in the *day* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **mouseleave_time_category** – The mouseleave event at a specific time in the *category* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **mousemove_day** – The mousemove event on a day. The event passed is the day object. Native mouse event is passed as a second argument.
- **mousemove_day_category** – The mousemove event on a day in the *category* view. The event passed is the day object. Native mouse event is passed as a second argument.
- **mousemove_event** – The mousemove event on an event. The event passed is the day & time object.
- **mousemove_interval** – The mousemove event at a specific interval label in the *day* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **mousemove_time** – The mousemove event at a specific time in the *day* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **mousemove_time_category** – The mousemove event at a specific time in the *category* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **mouseup_day** – The mouseup event on a day. The event passed is the day object. Native mouse event is passed as a second argument.
- **mouseup_day_category** – The mouseup event on a day in the *category* view. The event passed is the day object. Native mouse event is passed as a second argument.
- **mouseup_event** – The mouseup event on an event. The event passed is the day & time object.
- **mouseup_interval** – The mouseup event at a specific interval label in the *day* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **mouseup_time** – The mouseup event at a specific time in the *day* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **mouseup_time_category** – The mouseup event at a specific time in the *category* view. The event passed is the day & time object. Native mouse event is passed as a second argument.

- **moved** – One of the functions *next*, *prev*, and *move* was called. The event passed is the day object calculated for the movement.
- **touchend_day** – The touchend event on a day. The event passed is the day object. Native mouse event is passed as a second argument.
- **touchend_day_category** – The touchend event on a day in the *category* view. The event passed is the day object. Native mouse event is passed as a second argument.
- **touchend_event** – The touchend event on am view. The event passed is the day & time object.
- **touchend_interval** – The touchend event at a specific interval label in the *day* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **touchend_time** – The touchend event at a specific time in the *day* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **touchend_time_category** – The touchend event at a specific time in the *category* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **touchmove_day** – The touchmove event on a day. The event passed is the day object. Native mouse event is passed as a second argument.
- **touchmove_day_category** – The touchmove event on a day in the *category* view. The event passed is the day object. Native mouse event is passed as a second argument.
- **touchmove_event** – The touchmove event on an *event* view. The event passed is the day & time object.
- **touchmove_interval** – The touchmove event at a specific interval label in the *day* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **touchmove_time** – The touchmove event at a specific time in the *day* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **touchmove_time_category** – The touchmove event at a specific time in the *category* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **touchstart_day** – The touchstart event on a day. The event passed is the day object. Native mouse event is passed as a second argument.
- **touchstart_day_category** – The touchstart event on a day in the *category* view. The event passed is the day object. Native mouse event is passed as a second argument.
- **touchstart_event** – The touchstart event on an event` view. The event passed is the day & time object.
- **touchstart_interval** – The touchstart event at a specific interval label in the *day* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **touchstart_time** – The touchstart event at a specific time in the *day* view. The event passed is the day & time object. Native mouse event is passed as a second argument.
- **touchstart_time_category** – The touchstart event at a specific time in the *category* view. The event passed is the day & time object Native mouse event is passed as a second argument..

class trame.html.vuetify.**VCalendarDaily**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VCalendarDaily component. See more info and examples .

Parameters

- **color** – See description .
- **dark** – See description .
- **day_format** – Formats day of the month string that appears in a day to a specified locale
- **end** – The ending date on the calendar (inclusive) in the format of *YYYY-MM-DD*. This may be ignored depending on the *type* of the calendar.
- **first_interval** – The first interval to display in the *day* view. If *intervalMinutes* is set to 60 and this is set to 9 the first time in the view is 9am.
- **first_time** – The first time to display in the *day* view. If specified, this overwrites any *firstInterval* value specified. This can be the number of minutes since midnight, a string in the format of *HH:mm*, or an object with number properties hour and minute.
- **hide_header** – If the header at the top of the *day* view should be visible.
- **interval_count** – The number of intervals to display in the *day* view.
- **interval_format** – Formats time of day string that appears in the interval gutter of the *day* and *week* view to specified locale
- **interval_height** – The height of an interval in pixels in the *day* view.
- **interval_minutes** – The number of minutes the intervals are in the *day* view. A common interval is 60 minutes so the intervals are an hour.
- **interval_style** – Returns CSS styling to apply to the interval.
- **interval_width** – The width of the interval gutter on the left side in the *day* view.
- **light** – Applies the light theme variant to the component.
- **locale** – The locale of the calendar.
- **max_days** – The maximum number of days to display in the custom calendar if an *end* day is not set.
- **now** – Override the day & time which is considered now. This is in the format of *YYYY-MM-DD hh:mm:ss*. The calendar is styled according to now.
- **short_intervals** – If true, the intervals in the *day* view will be 9 AM as opposed to 09:00 AM
- **short_weekdays** – Whether the short versions of a weekday should be used (Mon vs Monday).
- **show_interval_label** – Checks if a given day and time should be displayed in the interval gutter of the *day* view.
- **start** – The starting date on the calendar (inclusive) in the format of *YYYY-MM-DD*. This may be ignored depending on the *type* of the calendar.
- **weekday_format** – Formats day of the week string that appears in the header to specified locale
- **weekdays** – Specifies which days of the week to display. To display Monday through Friday only, a value of *[1, 2, 3, 4, 5]* can be used. To display a week starting on Monday a value of *[1, 2, 3, 4, 5, 6, 0]* can be used.

class trame.html.vuetify.**VCalendarWeekly**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VCalendarWeekly component. See more info and examples .

Parameters

- **color** – See description .
- **dark** – See description .
- **day_format** – Formats day of the month string that appears in a day to a specified locale
- **end** – The ending date on the calendar (inclusive) in the format of *YYYY-MM-DD*. This may be ignored depending on the *type* of the calendar.
- **hide_header** – If the header at the top of the *day* view should be visible.
- **light** – Applies the light theme variant to the component.
- **locale** – The locale of the calendar.
- **locale_first_day_of_year** – Sets the day that determines the first week of the year, starting with 0 for **Sunday**. For ISO 8601 this should be 4.
- **min_weeks** – The minimum number of weeks to display in the *month* or *week* view.
- **month_format** – Formats month string that appears in a day to specified locale
- **now** – Override the day & time which is considered now. This is in the format of *YYYY-MM-DD hh:mm:ss*. The calendar is styled according to now.
- **short_months** – Whether the short versions of a month should be used (Jan vs January).
- **short_weekdays** – Whether the short versions of a weekday should be used (Mon vs Monday).
- **show_month_on_first** – Whether the name of the month should be displayed on the first day of the month.
- **show_week** – Whether week numbers should be displayed when using the *month* view.
- **start** – The starting date on the calendar (inclusive) in the format of *YYYY-MM-DD*. This may be ignored depending on the *type* of the calendar.
- **weekday_format** – Formats day of the week string that appears in the header to specified locale
- **weekdays** – Specifies which days of the week to display. To display Monday through Friday only, a value of *[1, 2, 3, 4, 5]* can be used. To display a week starting on Monday a value of *[1, 2, 3, 4, 5, 6, 0]* can be used.

class trame.html.vuetify.**VCalendarMonthly**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VCalendarMonthly component. See more info and examples .

Parameters

- **color** – See description .
- **dark** – See description .
- **day_format** – Formats day of the month string that appears in a day to a specified locale
- **end** – The ending date on the calendar (inclusive) in the format of *YYYY-MM-DD*. This may be ignored depending on the *type* of the calendar.

- **hide_header** – If the header at the top of the *day* view should be visible.
- **light** – Applies the light theme variant to the component.
- **locale** – The locale of the calendar.
- **locale_first_day_of_year** – Sets the day that determines the first week of the year, starting with 0 for **Sunday**. For ISO 8601 this should be 4.
- **min_weeks** – The minimum number of weeks to display in the *month* or *week* view.
- **month_format** – Formats month string that appears in a day to specified locale
- **now** – Override the day & time which is considered now. This is in the format of *YYYY-MM-DD hh:mm:ss*. The calendar is styled according to now.
- **short_months** – Whether the short versions of a month should be used (Jan vs January).
- **short_weekdays** – Whether the short versions of a weekday should be used (Mon vs Monday).
- **show_month_on_first** – Whether the name of the month should be displayed on the first day of the month.
- **show_week** – Whether week numbers should be displayed when using the *month* view.
- **start** – The starting date on the calendar (inclusive) in the format of *YYYY-MM-DD*. This may be ignored depending on the *type* of the calendar.
- **weekday_format** – Formats day of the week string that appears in the header to specified locale
- **weekdays** – Specifies which days of the week to display. To display Monday through Friday only, a value of *[1, 2, 3, 4, 5]* can be used. To display a week starting on Monday a value of *[1, 2, 3, 4, 5, 6, 0]* can be used.

class trame.html.vuetify.**VCard**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VCard component. See more info and examples .

Parameters

- **active_class** – See description .
- **append** – See description .
- **color** – See description .
- **dark** – See description .
- **disabled** – Removes the ability to click or target the component.
- **elevation** – See description .
- **exact** – See description .
- **exact_active_class** – See description .
- **exact_path** – See description .
- **flat** – Removes the card's elevation.
- **height** – Sets the height for the component.
- **hover** – See description .
- **href** – Designates the component as anchor and applies the **href** attribute.

- **img** – See description .
- **light** – Applies the light theme variant to the component.
- **link** – Designates that the component is a link. This is automatic when using the **href** or **to** prop.
- **loader_height** – Specifies the height of the loader
- **loading** – Displays linear progress bar. Can either be a String which specifies which color is applied to the progress bar (any material color or theme color - **primary**, **secondary**, **success**, **info**, **warning**, **error**) or a Boolean which uses the component **color** (set by color prop - if it's supported by the component) or the primary color
- **max_height** – Sets the maximum height for the component.
- **max_width** – Sets the maximum width for the component.
- **min_height** – Sets the minimum height for the component.
- **min_width** – Sets the minimum width for the component.
- **nuxt** – See description .
- **outlined** – Removes elevation (box-shadow) and adds a *thin* border.
- **raised** – See description .
- **replace** – See description .
- **ripple** – See description .
- **rounded** – See description .
- **shaped** – Applies a large border radius on the top left and bottom right of the card.
- **tag** – Specify a custom tag used on the root element.
- **target** – Designates the target attribute. This should only be applied when using the **href** prop.
- **tile** – Removes the component's **border-radius**.
- **to** – See description .
- **width** – Sets the width for the component.

class trame.html.vuetify.**VCardActions**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VCardActions component. See more info and examples .

Parameters **tag** – Specify a custom tag used on the root element.

class trame.html.vuetify.**VCardSubtitle**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VCardSubtitle component. See more info and examples .

Parameters **tag** – Specify a custom tag used on the root element.

class trame.html.vuetify.**VCardText**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VCardText component. See more info and examples .

Parameters **tag** – Specify a custom tag used on the root element.

class trame.html.vuetify.**VCardTitle**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VCardTitle component. See more info and examples .

Parameters **tag** – Specify a custom tag used on the root element.

class trame.html.vuetify.**VCarousel**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VCarousel component. See more info and examples .

Parameters

- **active_class** – The **active-class** applied to children when they are activated.
- **continuous** – Determines whether carousel is continuous
- **cycle** – Determines if the carousel should cycle through images.
- **dark** – See description .
- **delimiter_icon** – Sets icon for carousel delimiter
- **height** – Sets the height for the component
- **hide_delimiter_background** – Hides the bottom delimiter background.
- **hide_delimiters** – Hides the carousel's bottom delimiters.
- **interval** – The duration between image cycles. Requires the **cycle** prop.
- **light** – Applies the light theme variant to the component.
- **mandatory** – Forces a value to always be selected (if available).
- **max** – Sets a maximum number of selections that can be made.
- **multiple** – Allow multiple selections. The **value** prop must be an _array_.
- **next_icon** – The displayed icon for forcing pagination to the next item.
- **prev_icon** – The displayed icon for forcing pagination to the previous item.
- **progress** – Displays a carousel progress bar. Requires the **cycle** prop and **interval**.
- **progress_color** – Applies specified color to progress bar.
- **reverse** – Reverse the normal transition direction.
- **show_arrows** – Displays arrows for next/previous navigation.
- **show_arrows_on_hover** – Displays navigation arrows only when the carousel is hovered over.
- **tag** – Specify a custom tag used on the root element.
- **touch** – Provide a custom **left** and **right** function when swiped left or right.
- **touchless** – Disable touch support.
- **value** – The designated model value for the component.
- **value_comparator** – Apply a custom value comparator function
- **vertical** – Uses a vertical transition when changing windows.
- **vertical_delimiters** – Displays carousel delimiters vertically.

Events

Parameters **change** – Emitted when the component value is changed by user interaction

class trame.html.vuetify.**VCarouselItem**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VCarouselItem component. See more info and examples .

Parameters

- **active_class** – See description .
- **append** – See description .
- **disabled** – Removes the ability to click or target the component.
- **eager** – Will force the components content to render on mounted. This is useful if you have content that will not be rendered in the DOM that you want crawled for SEO.
- **exact** – See description .
- **exact_active_class** – See description .
- **exact_path** – See description .
- **href** – Designates the component as anchor and applies the **href** attribute.
- **link** – Designates that the component is a link. This is automatic when using the **href** or **to** prop.
- **nuxt** – See description .
- **replace** – See description .
- **reverse_transition** – Sets the reverse transition
- **ripple** – See description .
- **tag** – Specify a custom tag used on the root element.
- **target** – Designates the target attribute. This should only be applied when using the **href** prop.
- **to** – See description .
- **transition** – See description .
- **value** – The value used when the component is selected in a group. If not provided, the index will be used.

class trame.html.vuetify.**VCheckbox**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VCheckbox component. See more info and examples .

Parameters

- **append_icon** – Appends an icon to the component, uses the same syntax as *v-icon*
- **background_color** – Changes the background-color of the input
- **color** – See description .
- **dark** – See description .
- **dense** – Reduces the input height
- **disabled** – Disable the input
- **error** – Puts the input in a manual error state

- **error_count** – The total number of errors that should display at once
- **error_messages** – Puts the input in an error state and passes through custom error messages. Will be combined with any validations that occur from the **rules** prop. This field will not trigger validation
- **false_value** – Sets value for falsy state
- **hide_details** – Hides hint and validation errors. When set to *auto* messages will be rendered only if there's a message (hint, error message, counter value etc) to display
- **hide_spin_buttons** – Hides spin buttons on the input when type is set to *number*.
- **hint** – Hint text
- **id** – Sets the DOM id on the component
- **indeterminate** – Sets an indeterminate state for the checkbox
- **indeterminate_icon** – The icon used when in an indeterminate state
- **input_value** – The **v-model** bound value
- **label** – Sets input label
- **light** – Applies the light theme variant to the component.
- **messages** – Displays a list of messages or message if using a string
- **multiple** – Changes expected model to an array
- **off_icon** – The icon used when inactive
- **on_icon** – The icon used when active
- **persistent_hint** – Forces hint to always be visible
- **prepend_icon** – Prepends an icon to the component, uses the same syntax as *v-icon*
- **readonly** – Puts input in readonly state
- **ripple** – See description .
- **rules** – Accepts a mixed array of types *function*, *boolean* and *string*. Functions pass an input value as an argument and must return either *true* / *false* or a *string* containing an error message. The input field will enter an error state if a function returns (or any value in the array contains) *false* or is a *string*
- **success** – Puts the input in a manual success state
- **success_messages** – Puts the input in a success state and passes through custom success messages.
- **true_value** – Sets value for truthy state
- **validate_on_blur** – Delays validation until blur event
- **value** – The input's value
- **value_comparator** – Apply a custom value comparator function

Events

Parameters

- **change** – Emitted when the input is changed by user interaction
- **click_append** – Emitted when appended icon is clicked

- **click_prepend** – Emitted when prepended icon is clicked
- **update_error** – The *error.sync* event
- **update_indeterminate** – The **indeterminate.sync** event.

class trame.html.vuetify.**VSimpleCheckbox**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VSimpleCheckbox component. See more info and examples .

Parameters

- **color** – See description .
- **dark** – See description .
- **disabled** – Disables simple checkbox.
- **indeterminate** – Sets an indeterminate state for the simple checkbox.
- **indeterminate_icon** – The icon used when in an indeterminate state.
- **light** – Applies the light theme variant to the component.
- **off_icon** – The icon used when inactive.
- **on_icon** – The icon used when active.
- **ripple** – See description .
- **value** – A boolean value that represents whether the simple checkbox is checked.

Events

Parameters **input** – The updated bound model

class trame.html.vuetify.**VChip**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VChip component. See more info and examples .

Parameters

- **active** – Determines whether the chip is visible or not.
- **active_class** – See description .
- **append** – See description .
- **close** – Adds remove button
- **close_icon** – Change the default icon used for **close** chips
- **close_label** – See description .
- **color** – See description .
- **dark** – See description .
- **disabled** – Disables the chip, making it un-selectable
- **draggable** – Makes the chip draggable
- **exact** – See description .
- **exact_active_class** – See description .
- **exact_path** – See description .
- **filter** – Displays a selection icon when selected

- **filter_icon** – Change the default icon used for **filter** chips
- **href** – Designates the component as anchor and applies the **href** attribute.
- **input_value** – Controls the **active** state of the item. This is typically used to highlight the component.
- **label** – Removes circle edges
- **large** – Makes the component large.
- **light** – Applies the light theme variant to the component.
- **link** – Explicitly define the chip as a link
- **nuxt** – See description .
- **outlined** – Removes background and applies border and text color
- **pill** – Remove *v-avatar* padding
- **replace** – See description .
- **ripple** – See description .
- **small** – Makes the component small.
- **tag** – Specify a custom tag used on the root element.
- **target** – Designates the target attribute. This should only be applied when using the **href** prop.
- **text_color** – Applies a specified color to the control text
- **to** – See description .
- **value** – See description .
- **x_large** – Makes the component extra large.
- **x_small** – Makes the component extra small.

Events

Parameters

- **click_close** – Emitted when close icon is clicked
- **input** – The updated bound model
- **update_active** – Emitted when close icon is clicked, sets active to *false*

class trame.html.vuetify.**VChipGroup**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VChipGroup component. See more info and examples .

Parameters

- **active_class** – The **active-class** applied to children when they are activated.
- **center_active** – Forces the selected chip to be centered
- **color** – See description .
- **column** – Remove horizontal pagination and wrap items as needed
- **dark** – See description .
- **light** – Applies the light theme variant to the component.

- **mandatory** – Forces a value to always be selected (if available).
- **max** – Sets a maximum number of selections that can be made.
- **mobile_breakpoint** – Sets the designated mobile breakpoint for the component.
- **multiple** – Allow multiple selections. The **value** prop must be an _array_.
- **next_icon** – Specify the icon to use for the next icon
- **prev_icon** – Specify the icon to use for the prev icon
- **show_arrows** – Force the display of the pagination arrows
- **tag** – Specify a custom tag used on the root element.
- **value** – The designated model value for the component.
- **value_comparator** – Apply a custom value comparator function

Events

Parameters **change** – Emitted when the component value is changed by user interaction

class trame.html.vuetify.**VColorPicker**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VColorPicker component. See more info and examples .

Parameters

- **canvas_height** – Height of canvas
- **dark** – See description .
- **disabled** – Disables picker
- **dot_size** – Changes the size of the selection dot on the canvas
- **elevation** – See description .
- **flat** – Removes elevation
- **hide_canvas** – Hides canvas
- **hide_inputs** – Hides inputs
- **hide_mode_switch** – Hides mode switch
- **hide_sliders** – Hides sliders
- **light** – Applies the light theme variant to the component.
- **mode** – Sets mode of inputs. Available modes are 'rgba', 'hsla', and 'hexa'. Can be synced with the *.sync* modifier.
- **show_swatches** – Displays color swatches
- **swatches** – Sets the available color swatches to select from - This prop only accepts rgba hex strings
- **swatches_max_height** – Sets the maximum height of the swatches section
- **value** – Current color. This can be either a string representing a hex color, or an object representing a RGBA, HSLA, or HSVA value
- **width** – Sets the width of the color picker

Events

Parameters

- **input** – Selected color. Depending on what you passed to the *value* prop this is either a string or an object
- **update_color** – Selected color. This is the internal representation of the color, containing all values.
- **update_mode** – Selected mode

class trame.html.vuetify.**VContent**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VContent component. See more info and examples .

Parameters **tag** – Specify a custom tag used on the root element.

class trame.html.vuetify.**VCombobox**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VCombobox component. See more info and examples .

Parameters

- **allow_overflow** – Allow the menu to overflow off the screen
- **append_icon** – Appends an icon to the component, uses the same syntax as *v-icon*
- **append_outer_icon** – Appends an icon to the outside the component's input, uses same syntax as *v-icon*
- **attach** – Specifies which DOM element that this component should detach to. String can be any valid querySelector and Object can be any valid Node. This will attach to the root *v-app* component by default.
- **auto_select_first** – When searching, will always highlight the first option
- **autofocus** – Enables autofocus
- **background_color** – Changes the background-color of the input
- **cache_items** – Keeps a local _unique_ copy of all items that have been passed through the **items** prop.
- **chips** – Changes display of selections to chips
- **clear_icon** – Applied when using **clearable** and the input is dirty
- **clearable** – Add input clear functionality, default icon is Material Design Icons **mdi-clear**
- **color** – See description .
- **counter** – Creates counter for input length; if no number is specified, it defaults to 25. Does not apply any validation.
- **counter_value** –
- **dark** – See description .
- **deletable_chips** – Adds a remove icon to selected chips
- **delimiters** – Accepts an array of strings that will trigger a new tag when typing. Does not replace the normal Tab and Enter keys.
- **dense** – Reduces the input height
- **disable_lookup** – Disables keyboard lookup

- **disabled** – Disables the input
- **eager** – Will force the components content to render on mounted. This is useful if you have content that will not be rendered in the DOM that you want crawled for SEO.
- **error** – Puts the input in a manual error state
- **error_count** – The total number of errors that should display at once
- **error_messages** – Puts the input in an error state and passes through custom error messages. Will be combined with any validations that occur from the **rules** prop. This field will not trigger validation
- **filled** – Applies the alternate filled input style
- **filter** – See description .
- **flat** – Removes elevation (shadow) added to element when using the **solo** or **solo-inverted** props
- **full_width** – Designates input type as full-width
- **height** – Sets the height of the input
- **hide_details** – Hides hint and validation errors. When set to *auto* messages will be rendered only if there's a message (hint, error message, counter value etc) to display
- **hide_no_data** – Hides the menu when there are no options to show. Useful for preventing the menu from opening before results are fetched asynchronously. Also has the effect of opening the menu when the *items* array changes if not already open.
- **hide_selected** – Do not display in the select menu items that are already selected
- **hide_spin_buttons** – Hides spin buttons on the input when type is set to *number*.
- **hint** – Hint text
- **id** – Sets the DOM id on the component
- **item_color** – Sets color of selected items
- **item_disabled** – Set property of **items**'s disabled value
- **item_text** – Set property of **items**'s text value
- **item_value** – See description .
- **items** – Can be an array of objects or array of strings. When using objects, will look for a text, value and disabled keys. This can be changed using the **item-text**, **item-value** and **item-disabled** props. Objects that have a **header** or **divider** property are considered special cases and generate a list header or divider; these items are not selectable.
- **label** – Sets input label
- **light** – Applies the light theme variant to the component.
- **loader_height** – Specifies the height of the loader
- **loading** – Displays linear progress bar. Can either be a String which specifies which color is applied to the progress bar (any material color or theme color - **primary**, **secondary**, **success**, **info**, **warning**, **error**) or a Boolean which uses the component **color** (set by color prop - if it's supported by the component) or the primary color
- **menu_props** – Pass props through to the *v-menu* component. Accepts either a string for boolean props *menu-props="auto, overflowY"*, or an object *:menu-props="{ auto: true, overflowY: true }"*

- **messages** – Displays a list of messages or message if using a string
- **multiple** – Changes select to multiple. Accepts array for value
- **no_data_text** – Display text when there is no data
- **no_filter** – Do not apply filtering when searching. Useful when data is being filtered server side
- **open_on_clear** – When using the **clearable** prop, once cleared, the select menu will either open or stay open, depending on the current state
- **outlined** – Applies the outlined style to the input
- **persistent_hint** – Forces hint to always be visible
- **persistent_placeholder** – Forces placeholder to always be visible
- **placeholder** – Sets the input's placeholder text
- **prefix** – Displays prefix text
- **prepend_icon** – Prepends an icon to the component, uses the same syntax as *v-icon*
- **prepend_inner_icon** – Prepends an icon inside the component's input, uses the same syntax as *v-icon*
- **readonly** – Puts input in readonly state
- **return_object** – Changes the selection behavior to return the object directly rather than the value specified with **item-value**
- **reverse** – Reverses the input orientation
- **rounded** – Adds a border radius to the input
- **rules** – Accepts a mixed array of types *function*, *boolean* and *string*. Functions pass an input value as an argument and must return either *true* / *false* or a *string* containing an error message. The input field will enter an error state if a function returns (or any value in the array contains) *false* or is a *string*
- **search_input** – Search value. Can be used with *.sync* modifier.
- **shaped** – Round if *outlined* and increase *border-radius* if *filled*. Must be used with either *outlined* or *filled*
- **single_line** – Label does not move on focus/dirty
- **small_chips** – Changes display of selections to chips with the **small** property
- **solo** – Changes the style of the input
- **solo_inverted** – Reduces element opacity until focused
- **success** – Puts the input in a manual success state
- **success_messages** – Puts the input in a success state and passes through custom success messages.
- **suffix** – Displays suffix text
- **type** – Sets input type
- **validate_on_blur** – Delays validation until blur event
- **value** – The input's value
- **value_comparator** – See description .

Events

Parameters

- **blur** – Emitted when the input is blurred
- **change** – Emitted when the input is changed by user interaction
- **click_append** – Emitted when appended icon is clicked
- **click_append_outer** – Emitted when appended outer icon is clicked
- **click_clear** – Emitted when clearable icon clicked
- **click_prepend** – Emitted when prepended icon is clicked
- **click_prepend_inner** – Emitted when prepended inner icon is clicked
- **focus** – Emitted when component is focused
- **input** – The updated bound model
- **keydown** – Emitted when **any** key is pressed
- **update_error** – The *error.sync* event
- **update_list_index** – Emitted when menu item is selected using keyboard arrows
- **update_search_input** – The *search-input.sync* event

class trame.html.vuetify.**VDataIterator**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VDataIterator component. See more info and examples .

Parameters

- **checkbox_color** –
- **custom_filter** – Function to filter items
- **custom_group** – Function used to group items
- **custom_sort** – Function used to sort items
- **dark** – See description .
- **disable_filtering** – Disables filtering completely
- **disable_pagination** – Disables pagination completely
- **disable_sort** – Disables sorting completely
- **expanded** – Array of expanded items. Can be used with *.sync* modifier
- **footer_props** – See description .
- **group_by** – Changes which item property should be used for grouping items. Currently only supports a single grouping in the format: *group* or *['group']*. When using an array, only the first element is considered. Can be used with *.sync* modifier
- **group_desc** – Changes which direction grouping is done. Can be used with *.sync* modifier
- **hide_default_footer** – Hides default footer
- **item_key** – The property on each item that is used as a unique key
- **items** – The array of items to display

- **items_per_page** – Changes how many items per page should be visible. Can be used with *.sync* modifier. Setting this prop to *-1* will display all items on the page
- **light** – Applies the light theme variant to the component.
- **loading** – If *true* and no items are provided, then a loading text will be shown
- **loading_text** – Text shown when *loading* is true and no items are provided
- **locale** – See description .
- **mobile_breakpoint** – Used to set when to toggle between regular table and mobile view
- **multi_sort** – If *true* then one can sort on multiple properties
- **must_sort** – If *true* then one can not disable sorting, it will always switch between ascending and descending
- **no_data_text** – Text shown when no items are provided to the component
- **no_results_text** – Text shown when *search* prop is used and there are no results
- **options** –
- **page** –
- **search** – Text input used to filter items
- **selectable_key** – The property on each item that is used to determine if it is selectable or not
- **server_items_length** – Used only when data is provided by a server. Should be set to the total amount of items available on server so that pagination works correctly
- **single_expand** – Changes expansion mode to single expand
- **single_select** – Changes selection mode to single select
- **sort_by** – Changes which item property (or properties) should be used for sort order. Can be used with *.sync* modifier
- **sort_desc** – Changes which direction sorting is done. Can be used with *.sync* modifier
- **value** – Used for controlling selected rows

Events

Parameters

- **current_items** – Emits the items provided via the **items** prop, every time the internal **computedItems** is changed.
- **input** – Array of selected items
- **item_expanded** – Event emitted when an item is expanded or closed
- **item_selected** – Event emitted when an item is selected or deselected
- **page_count** – Emits when the **pageCount** property of the **pagination** prop is updated
- **pagination** – Emits when something changed to the *pagination* which can be provided via the *pagination* prop
- **toggle_select_all** – Emits when the *select-all* checkbox in table header is clicked. This checkbox is enabled by the **show-select** prop
- **update_expanded** – The *.sync* event for *expanded* prop
- **update_group_by** – Emits when the **group-by** property of the **options** property is updated

- **update_group_desc** – Emits when the **group-desc** property of the **options** prop is updated
- **update_items_per_page** – Emits when the **items-per-page** property of the **options** prop is updated
- **update_multi_sort** – Emits when the **multi-sort** property of the **options** prop is updated
- **update_must_sort** – Emits when the **must-sort** property of the **options** prop is updated
- **update_options** – Emits when one of the **options** properties is updated
- **update_page** – Emits when the **page** property of the **options** prop is updated
- **update_sort_by** – Emits when the **sort-by** property of the **options** prop is updated
- **update_sort_desc** – Emits when the **sort-desc** property of the **options** prop is updated

class trame.html.vuetify.**VDataFooter**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VDataFooter component. See more info and examples .

Parameters

- **disable_items_per_page** – Disables items-per-page dropdown
- **disable_pagination** – Disables pagination buttons
- **first_icon** – First icon
- **items_per_page_all_text** – Text for 'All' option in items-per-page dropdown
- **items_per_page_options** – Array of options to show in the items-per-page dropdown
- **items_per_page_text** – Text for items-per-page dropdown
- **last_icon** – Last icon
- **next_icon** – Next icon
- **options** – DataOptions
- **page_text** –
- **pagination** – DataPagination
- **prev_icon** – Previous icon
- **show_current_page** – Show current page number between prev/next icons
- **show_first_last_page** – Show first/last icons

Events

Parameters **update_options** – The *.sync* event for *options* prop

class trame.html.vuetify.**VDataTable**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VDataTable component. See more info and examples .

Parameters

- **calculate_widths** – Enables calculation of column widths. *widths* property will be available in select scoped slots
- **caption** – Set the caption (using <*caption*>)
- **checkbox_color** – Set the color of the checkboxes (showSelect must be used)

- **custom_filter** – Function to filter items
- **custom_group** – Function used to group items
- **custom_sort** – Function used to sort items
- **dark** – See description .
- **dense** – Decreases the height of rows
- **disable_filtering** – Disables filtering completely
- **disable_pagination** – Disables pagination completely
- **disable_sort** – Disables sorting completely
- **expand_icon** – Icon used for expand toggle button.
- **expanded** – Array of expanded items. Can be used with *.sync* modifier
- **fixed_header** – Fixed header to top of table. **NOTE:** Does not work in IE11
- **footer_props** – See description .
- **group_by** – Changes which item property should be used for grouping items. Currently only supports a single grouping in the format: *group* or *['group']*. When using an array, only the first element is considered. Can be used with *.sync* modifier
- **group_desc** – Changes which direction grouping is done. Can be used with *.sync* modifier
- **header_props** – See description .
- **headers** – An array of objects that each describe a header column. See the example below for a definition of all properties
- **headers_length** – Can be used in combination with *hide-default-header* to specify the number of columns in the table to allow expansion rows and loading bar to function properly
- **height** – Set an explicit height of table
- **hide_default_footer** – Hides default footer
- **hide_default_header** – Hide the default headers
- **item_class** – Property on supplied *items* that contains item's row class or function that takes an item as an argument and returns the class of corresponding row
- **item_key** – The property on each item that is used as a unique key
- **items** – The array of items to display
- **items_per_page** – Changes how many items per page should be visible. Can be used with *.sync* modifier. Setting this prop to *-1* will display all items on the page
- **light** – Applies the light theme variant to the component.
- **loader_height** – Specifies the height of the loader
- **loading** – If *true* and no items are provided, then a loading text will be shown
- **loading_text** – Text shown when *loading* is true and no items are provided
- **locale** – See description .
- **mobile_breakpoint** – Used to set when to toggle between regular table and mobile view
- **multi_sort** – If *true* then one can sort on multiple properties

- **must_sort** – If *true* then one can not disable sorting, it will always switch between ascending and descending
- **no_data_text** – Text shown when no items are provided to the component
- **no_results_text** – Text shown when *search* prop is used and there are no results
- **options** –
- **page** – The current displayed page number (1-indexed)
- **search** – Text input used to filter items
- **selectable_key** – The property on each item that is used to determine if it is selectable or not
- **server_items_length** – Used only when data is provided by a server. Should be set to the total amount of items available on server so that pagination works correctly
- **show_expand** – Shows the expand toggle in default rows
- **show_group_by** – Shows the group by toggle in the header and enables grouped rows
- **show_select** – Shows the select checkboxes in both the header and rows (if using default rows)
- **single_expand** – Changes expansion mode to single expand
- **single_select** – Changes selection mode to single select
- **sort_by** – Changes which item property (or properties) should be used for sort order. Can be used with *.sync* modifier
- **sort_desc** – Changes which direction sorting is done. Can be used with *.sync* modifier
- **value** – Used for controlling selected rows

Events

Parameters

- **click_row** – Emits when a table row is clicked. This event provides 2 arguments: the first is the item data that was clicked and the second is the other related data provided by the *item* slot. **NOTE:** will not emit when table rows are defined through a slot such as *item* or *body*.
- **contextmenu_row** – Emits when a table row is right-clicked. The item for the row is included. **NOTE:** will not emit when table rows are defined through a slot such as *item* or *body*.
- **current_items** – Emits the items provided via the **items** prop, every time the internal **computedItems** is changed.
- **dblclick_row** – Emits when a table row is double-clicked. The item for the row is included. **NOTE:** will not emit when table rows are defined through a slot such as *item* or *body*.
- **input** – Array of selected items
- **item_expanded** – Event emitted when an item is expanded or closed
- **item_selected** – Event emitted when an item is selected or deselected
- **page_count** – Emits when the **pageCount** property of the **pagination** prop is updated
- **pagination** – Emits when something changed to the *pagination* which can be provided via the *pagination* prop

- **toggle_select_all** – Emits when the *select-all* checkbox in table header is clicked. This checkbox is enabled by the **show-select** prop
- **update_expanded** – The *.sync* event for *expanded* prop
- **update_group_by** – Emits when the **group-by** property of the **options** property is updated
- **update_group_desc** – Emits when the **group-desc** property of the **options** prop is updated
- **update_items_per_page** – Emits when the **items-per-page** property of the **options** prop is updated
- **update_multi_sort** – Emits when the **multi-sort** property of the **options** prop is updated
- **update_must_sort** – Emits when the **must-sort** property of the **options** prop is updated
- **update_options** – Emits when one of the **options** properties is updated
- **update_page** – Emits when the **page** property of the **options** prop is updated
- **update_sort_by** – Emits when the **sort-by** property of the **options** prop is updated
- **update_sort_desc** – Emits when the **sort-desc** property of the **options** prop is updated

class trame.html.vuetify.**VEditDialog**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VEditDialog component. See more info and examples .

Parameters

- **cancel_text** – Sets the default text for the cancel button when using the **large** prop
- **dark** – See description .
- **eager** – Will force the components content to render on mounted. This is useful if you have content that will not be rendered in the DOM that you want crawled for SEO.
- **large** – Attaches a submit and cancel button to the dialog
- **light** – Applies the light theme variant to the component.
- **persistent** – Clicking outside or pressing **esc** key will not dismiss the dialog
- **return_value** –
- **save_text** – Sets the default text for the save button when using the **large** prop
- **transition** – See description .

Events

Parameters

- **cancel** – Emits when editing is canceled
- **close** – Emits when edit-dialog close button is pressed
- **open** – Emits when editing is opened
- **save** – Emits when edit-dialog save button is pressed

class trame.html.vuetify.**VDataTableHeader**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VDataTableHeader component. See more info and examples .

Parameters

- **checkbox_color** –

- **disable_sort** – Toggles rendering of sort button
- **every_item** – Indicates if all items in table are selected
- **headers** – Array of header items to display
- **mobile** – Renders mobile view of headers
- **options** – Options object. Identical to the one on *v-data-table*
- **show_group_by** – Shows group by button
- **single_select** – Toggles rendering of select-all checkbox
- **some_items** – Indicates if one or more items in table are selected
- **sort_by_text** – Sets the label text used by the default sort-by selector when *v-data-table* is rendering the mobile view
- **sort_icon** – Icon used for sort button

class trame.html.vuetify.**VSimpleTable**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VSimpleTable component. See more info and examples .

Parameters

- **dark** – See description .
- **dense** – Decreases paddings to render a dense table
- **fixed_header** – Sets table header to fixed mode
- **height** – Sets the height for the component
- **light** – Applies the light theme variant to the component.

class trame.html.vuetify.**VDatePicker**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VDatePicker component. See more info and examples .

Parameters

- **active_picker** – Determines which picker in the date or month picker is being displayed. Allowed values: *'DATE'*, *'MONTH'*, *'YEAR'*
- **allowed_dates** – Restricts which dates can be selected
- **color** – See description .
- **dark** – See description .
- **day_format** – Allows you to customize the format of the day string that appears in the date table. Called with date (ISO 8601 **date** string) arguments.
- **disabled** – Disables interaction with the picker
- **elevation** – See description .
- **event_color** – Sets the color for event dot. It can be string (all events will have the same color) or *object* where attribute is the event date and value is boolean/color/array of colors for specified date or *function* taking date as a parameter and returning boolean/color/array of colors for that date
- **events** – Array of dates or object defining events or colors or function returning boolean/color/array of colors

- **first_day_of_week** – Sets the first day of the week, starting with 0 for Sunday.
- **flat** – Removes elevation
- **full_width** – Forces 100% width
- **header_color** – Defines the header color. If not specified it will use the color defined by <code>color</code> prop or the default picker color
- **header_date_format** – Allows you to customize the format of the month string that appears in the header of the calendar. Called with date (ISO 8601 **date** string) arguments.
- **landscape** – Orients picker horizontal
- **light** – Applies the light theme variant to the component.
- **locale** – Sets the locale. Accepts a string with a BCP 47 language tag.
- **locale_first_day_of_year** – Sets the day that determines the first week of the year, starting with 0 for **Sunday**. For ISO 8601 this should be 4.
- **max** – Maximum allowed date/month (ISO 8601 format)
- **min** – Minimum allowed date/month (ISO 8601 format)
- **month_format** – Formatting function used for displaying months in the months table. Called with date (ISO 8601 **date** string) arguments.
- **multiple** – Allow the selection of multiple dates
- **next_icon** – Sets the icon for next month/year button
- **next_month_aria_label** –
- **next_year_aria_label** –
- **no_title** – Hide the picker title
- **picker_date** – Displayed year/month
- **prev_icon** – Sets the icon for previous month/year button
- **prev_month_aria_label** –
- **prev_year_aria_label** –
- **range** – Allow the selection of date range
- **reactive** – Updates the picker model when changing months/years automatically
- **readonly** – Makes the picker readonly (doesn't allow to select new date)
- **scrollable** – Allows changing displayed month with mouse scroll
- **selected_items_text** – See description .
- **show_adjacent_months** – Toggles visibility of days from previous and next months
- **show_current** – Toggles visibility of the current date/month outline or shows the provided date/month as a current
- **show_week** – Toggles visibility of the week numbers in the body of the calendar
- **title_date_format** – Allows you to customize the format of the date string that appears in the title of the date picker. Called with date (ISO 8601 **date** string) arguments.
- **type** – Determines the type of the picker - *date* for date picker, *month* for month picker
- **value** – Date picker model (ISO 8601 format, YYYY-mm-dd or YYYY-mm)

- **weekday_format** – Allows you to customize the format of the weekday string that appears in the body of the calendar. Called with date (ISO 8601 **date** string) arguments.
- **width** – Width of the picker
- **year_format** – Allows you to customize the format of the year string that appears in the header of the calendar. Called with date (ISO 8601 **date** string) arguments.
- **year_icon** – Sets the icon in the year selection button

Events

Parameters

- **change** – Reactive date picker emits *input* even when any part of the date (year/month/day) changes, but *change* event is emitted only when the day (for date pickers) or month (for month pickers) changes. If *range* prop is set, date picker emits *change* when both [from, to] are selected.
- **input** – The updated bound model
- **update_active_picker** – The *.sync* event for *active-picker* prop
- **update_picker_date** – The *.sync* event for *picker-date* prop

class trame.html.vuetify.**VDialog**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VDialog component. See more info and examples .

Parameters

- **activator** – Designate a custom activator when the *activator* slot is not used. String can be any valid querySelector and Object can be any valid Node.
- **attach** – Specifies which DOM element that this component should detach to. String can be any valid querySelector and Object can be any valid Node. This will attach to the root *v-app* component by default.
- **close_delay** – Milliseconds to wait before closing component.
- **content_class** – Applies a custom class to the detached element. This is useful because the content is moved to the beginning of the *v-app* component (unless the **attach** prop is provided) and is not targetable by classes passed directly on the component.
- **dark** – See description .
- **disabled** – Disables the ability to open the component.
- **eager** – Will force the components content to render on mounted. This is useful if you have content that will not be rendered in the DOM that you want crawled for SEO.
- **fullscreen** – Changes layout for fullscreen display.
- **hide_overlay** – Hides the display of the overlay.
- **internal_activator** – Detaches the menu content inside of the component as opposed to the document.
- **light** – Applies the light theme variant to the component.
- **max_width** – Sets the maximum width for the component.
- **no_click_animation** – Disables the bounce effect when clicking outside of a *v-dialog*'s content when using the **persistent** prop.
- **open_delay** – Milliseconds to wait before opening component.

- **open_on_focus** –
- **open_on_hover** – Designates whether component should activate when its activator is hovered.
- **origin** – See description .
- **overlay_color** – Sets the overlay color.
- **overlay_opacity** – Sets the overlay opacity.
- **persistent** – Clicking outside of the element or pressing **esc** key will not deactivate it.
- **retain_focus** – Tab focus will return to the first child of the dialog by default. Disable this when using external tools that require focus such as TinyMCE or vue-clipboard.
- **return_value** –
- **scrollable** – See description .
- **transition** – See description .
- **value** – Controls whether the component is visible or hidden.
- **width** – Sets the width for the component.

Events

Parameters

- **click_outside** – Event that fires when clicking outside an active dialog.
- **input** – The updated bound model
- **keydown** – Event that fires when key is pressed. If dialog is active and not using the **persistent** prop, the **esc** key will deactivate it.

class trame.html.vuetify.**VDivider**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VDivider component. See more info and examples .

Parameters

- **dark** – See description .
- **inset** – Adds indentation (72px) for **normal** dividers, reduces max height for **vertical**.
- **light** – Applies the light theme variant to the component.
- **vertical** – Displays dividers vertically

class trame.html.vuetify.**VExpansionPanels**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VExpansionPanels component. See more info and examples .

Parameters

- **accordion** – Removes the margin around open panels
- **active_class** – The **active-class** applied to children when they are activated.
- **dark** – See description .
- **disabled** – Disables the entire expansion-panel
- **flat** – Removes the expansion-panel's elevation and borders
- **focusable** – Makes the expansion-panel headers focusable

- **hover** – Applies a background-color shift on hover to expansion panel headers
- **inset** – Makes the expansion-panel open with a inset style
- **light** – Applies the light theme variant to the component.
- **mandatory** – Forces a value to always be selected (if available).
- **max** – Sets a maximum number of selections that can be made.
- **multiple** – Allow multiple selections. The **value** prop must be an _array_.
- **popout** – Makes the expansion-panel open with an popout style
- **readonly** – Makes the entire expansion-panel read only.
- **tag** – Specify a custom tag used on the root element.
- **tile** – Removes the border-radius
- **value** – Controls the opened/closed state of content in the expansion-panel. Corresponds to a zero-based index of the currently opened content. If the *multiple* prop (previously *expand* in 1.5.x) is used then it is an array of numbers where each entry corresponds to the index of the opened content. The index order is not relevant.
- **value_comparator** – Apply a custom value comparator function

class trame.html.vuetify.**VExpansionPanel**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VExpansionPanel component. See more info and examples .

Parameters

- **active_class** – See description .
- **disabled** – Disables the expansion-panel content
- **readonly** – Makes the expansion-panel content read only.

Events

Parameters **change** – Toggles the value of the selected panel

class trame.html.vuetify.**VExpansionPanelHeader**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VExpansionPanelHeader component. See more info and examples .

Parameters

- **color** – See description .
- **disable_icon_rotate** – Removes the icon rotation animation when expanding a panel
- **expand_icon** – Set the expand action icon
- **hide_actions** – Hide the expand icon in the content header
- **ripple** – See description .

class trame.html.vuetify.**VExpansionPanelContent**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VExpansionPanelContent component. See more info and examples .

Parameters

- **color** – See description .

- **eager** – Will force the components content to render on mounted. This is useful if you have content that will not be rendered in the DOM that you want crawled for SEO.

class trame.html.vuetify.**VFileInput**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VFileInput component. See more info and examples .

Parameters

- **append_icon** – Appends an icon to the component, uses the same syntax as *v-icon*
- **append_outer_icon** – Appends an icon to the outside the component's input, uses same syntax as *v-icon*
- **autofocus** – Enables autofocus
- **background_color** – Changes the background-color of the input
- **chips** – Changes display of selections to chips
- **clear_icon** – Applied when using **clearable** and the input is dirty
- **clearable** – Add input clear functionality, default icon is Material Design Icons **mdi-clear**
- **color** – See description .
- **counter** – Creates counter for input length; if no number is specified, it defaults to 25. Does not apply any validation.
- **counter_size_string** – See description .
- **counter_string** – See description .
- **counter_value** –
- **dark** – See description .
- **dense** – Reduces the input height
- **disabled** – Disable the input
- **error** – Puts the input in a manual error state
- **error_count** – The total number of errors that should display at once
- **error_messages** – Puts the input in an error state and passes through custom error messages. Will be combined with any validations that occur from the **rules** prop. This field will not trigger validation
- **filled** – Applies the alternate filled input style
- **flat** – Removes elevation (shadow) added to element when using the **solo** or **solo-inverted** props
- **full_width** – Designates input type as full-width
- **height** – Sets the height of the input
- **hide_details** – Hides hint and validation errors. When set to *auto* messages will be rendered only if there's a message (hint, error message, counter value etc) to display
- **hide_input** – Display the icon only without the input (file names)
- **hide_spin_buttons** – Hides spin buttons on the input when type is set to *number*.
- **hint** – Hint text
- **id** – Sets the DOM id on the component

- **label** – Sets input label
- **light** – Applies the light theme variant to the component.
- **loader_height** – Specifies the height of the loader
- **loading** – Displays linear progress bar. Can either be a String which specifies which color is applied to the progress bar (any material color or theme color - **primary**, **secondary**, **success**, **info**, **warning**, **error**) or a Boolean which uses the component **color** (set by color prop - if it's supported by the component) or the primary color
- **messages** – Displays a list of messages or message if using a string
- **multiple** – Adds the **multiple** attribute to the input, allowing multiple file selections.
- **outlined** – Applies the outlined style to the input
- **persistent_hint** – Forces hint to always be visible
- **persistent_placeholder** – Forces placeholder to always be visible
- **placeholder** – Sets the input's placeholder text
- **prefix** – Displays prefix text
- **prepend_icon** – Prepends an icon to the component, uses the same syntax as *v-icon*
- **prepend_inner_icon** – Prepends an icon inside the component's input, uses the same syntax as *v-icon*
- **reverse** – Reverses the input orientation
- **rounded** – Adds a border radius to the input
- **rules** – Accepts a mixed array of types *function*, *boolean* and *string*. Functions pass an input value as an argument and must return either *true* / *false* or a *string* containing an error message. The input field will enter an error state if a function returns (or any value in the array contains) *false* or is a *string*
- **shaped** – Round if *outlined* and increase *border-radius* if *filled*. Must be used with either *outlined* or *filled*
- **show_size** – Sets the displayed size of selected file(s). When using **true** will default to _1000_ displaying (**kB, MB, GB**) while _1024_ will display (**KiB, MiB, GiB**).
- **single_line** – Label does not move on focus/dirty
- **small_chips** – Changes display of selections to chips with the **small** property
- **solo** – Changes the style of the input
- **solo_inverted** – Reduces element opacity until focused
- **success** – Puts the input in a manual success state
- **success_messages** – Puts the input in a success state and passes through custom success messages.
- **suffix** – Displays suffix text
- **truncate_length** – The length of a filename before it is truncated with ellipsis
- **type** – Sets input type
- **validate_on_blur** – Delays validation until blur event
- **value** – See description .

Events

Parameters

- **blur** – Emitted when the input is blurred
- **change** – Emitted when the input is changed by user interaction
- **click_append** – Emitted when appended icon is clicked
- **click_append_outer** – Emitted when appended outer icon is clicked
- **click_clear** – Emitted when clearable icon clicked
- **click_prepend** – Emitted when prepended icon is clicked
- **click_prepend_inner** – Emitted when prepended inner icon is clicked
- **focus** – Emitted when component is focused
- **input** – The updated bound model
- **keydown** – Emitted when **any** key is pressed
- **update_error** – The *error.sync* event

class trame.html.vuetify.**VFooter**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VFooter component. See more info and examples .

Parameters

- **absolute** – Applies **position: absolute** to the component.
- **app** – See description .
- **color** – See description .
- **dark** – See description .
- **elevation** – See description .
- **fixed** – Applies **position: fixed** to the component.
- **height** – Sets the height for the component.
- **inset** – Positions the toolbar offset from an application *v-navigation-drawer*
- **light** – Applies the light theme variant to the component.
- **max_height** – Sets the maximum height for the component.
- **max_width** – Sets the maximum width for the component.
- **min_height** – Sets the minimum height for the component.
- **min_width** – Sets the minimum width for the component.
- **outlined** – Removes elevation (box-shadow) and adds a *thin* border.
- **padless** – Remove all padding from the footer
- **rounded** – See description .
- **shaped** – Applies a large border radius on the top left and bottom right of the card.
- **tag** – Specify a custom tag used on the root element.
- **tile** – Removes the component's **border-radius**.

- **width** – Sets the width for the component.

class trame.html.vuetify.**VForm**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VForm component. See more info and examples .

Parameters

- **disabled** – Puts all children inputs into a disabled state.
- **lazy_validation** – If enabled, **value** will always be _true_ unless there are visible validation errors. You can still call *validate()* to manually trigger validation
- **readonly** – Puts all children inputs into a readonly state.
- **value** – A boolean value representing the validity of the form.

Events

Parameters

- **input** – The updated bound model
- **submit** – Emitted when form is submitted

class trame.html.vuetify.**VContainer**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VContainer component. See more info and examples .

Parameters

- **fluid** – Removes viewport maximum-width size breakpoints
- **id** – Sets the DOM id on the component
- **tag** – Specify a custom tag used on the root element.

class trame.html.vuetify.**VCol**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VCol component. See more info and examples .

Parameters

- **align_self** – See description .
- **cols** – Sets the default number of columns the component extends. Available options are **1 -> 12** and **auto**.
- **lg** – Changes the number of columns on large and greater breakpoints.
- **md** – Changes the number of columns on medium and greater breakpoints.
- **offset** – Sets the default offset for the column.
- **offset_lg** – Changes the offset of the component on large and greater breakpoints.
- **offset_md** – Changes the offset of the component on medium and greater breakpoints.
- **offset_sm** – Changes the offset of the component on small and greater breakpoints.
- **offset_xl** – Changes the offset of the component on extra large and greater breakpoints.
- **order** – See description .
- **order_lg** – Changes the order of the component on large and greater breakpoints.
- **order_md** – Changes the order of the component on medium and greater breakpoints.

- **order_sm** – Changes the order of the component on small and greater breakpoints.
- **order_xl** – Changes the order of the component on extra large and greater breakpoints.
- **sm** – Changes the number of columns on small and greater breakpoints.
- **tag** – Specify a custom tag used on the root element.
- **xl** – Changes the number of columns on extra large and greater breakpoints.

class trame.html.vuetify.**VRow**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VRow component. See more info and examples .

Parameters

- **align** – See description .
- **align_content** – See description .
- **align_content_lg** – Changes the **align-content** property on large and greater breakpoints.
- **align_content_md** – Changes the **align-content** property on medium and greater breakpoints.
- **align_content_sm** – Changes the **align-content** property on small and greater breakpoints.
- **align_content_xl** – Changes the **align-content** property on extra large and greater breakpoints.
- **align_lg** – Changes the **align-items** property on large and greater breakpoints.
- **align_md** – Changes the **align-items** property on medium and greater breakpoints.
- **align_sm** – Changes the **align-items** property on small and greater breakpoints.
- **align_xl** – Changes the **align-items** property on extra large and greater breakpoints.
- **dense** – Reduces the gutter between `v-col`s.
- **justify** – See description .
- **justify_lg** – Changes the **justify-content** property on large and greater breakpoints.
- **justify_md** – Changes the **justify-content** property on medium and greater breakpoints.
- **justify_sm** – Changes the **justify-content** property on small and greater breakpoints.
- **justify_xl** – Changes the **justify-content** property on extra large and greater breakpoints.
- **no_gutters** – Removes the gutter between `v-col`s.
- **tag** – Specify a custom tag used on the root element.

class trame.html.vuetify.**VSpacer**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VSpacer component. See more info and examples .

Parameters **tag** – Specify a custom tag used on the root element.

class trame.html.vuetify.**VLayout**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VLayout component. See more info and examples .

Parameters

- **align_baseline** –
- **align_center** –
- **align_content_center** –
- **align_content_end** –
- **align_content_space_around** –
- **align_content_space_between** –
- **align_content_start** –
- **align_end** –
- **align_start** –
- **column** –
- **d_{type}** –
- **fill_height** –
- **id** – Sets the DOM id on the component
- **justify_center** –
- **justify_end** –
- **justify_space_around** –
- **justify_space_between** –
- **justify_start** –
- **reverse** –
- **row** –
- **tag** – Specify a custom tag used on the root element.
- **wrap** –

class trame.html.vuetify.**VFlex**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VFlex component. See more info and examples .

Parameters

- **(size)(1_12)** –
- **align_self_baseline** –
- **align_self_center** –
- **align_self_end** –
- **align_self_start** –
- **grow** –
- **id** – Sets the DOM id on the component
- **offset_(size)(0_12)** –
- **order_(size)(1_12)** –
- **shrink** –

- **tag** – Specify a custom tag used on the root element.

class trame.html.vuetify.**VHover**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VHover component. See more info and examples .

Parameters

- **close_delay** – Milliseconds to wait before closing component.
- **disabled** – Turns off hover functionality
- **open_delay** – Milliseconds to wait before opening component.
- **value** – Controls whether the component is visible or hidden.

class trame.html.vuetify.**VIcon**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VIcon component. See more info and examples .

Parameters

- **color** – See description .
- **dark** – See description .
- **dense** – Makes icon smaller (20px)
- **disabled** – Disable the input
- **large** – Makes the component large.
- **left** – Applies appropriate margins to the icon inside of a button when placed to the **left** of another element or text
- **light** – Applies the light theme variant to the component.
- **right** – Applies appropriate margins to the icon inside of a button when placed to the **right** of another element or text
- **size** – Specifies a custom font size for the icon
- **small** – Makes the component small.
- **tag** – Specifies a custom tag to be used
- **x_large** – Makes the component extra large.
- **x_small** – Makes the component extra small.

class trame.html.vuetify.**VImg**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VImg component. See more info and examples .

Parameters

- **alt** – Alternate text for screen readers. Leave empty for decorative images
- **aspect_ratio** – Calculated as *width/height*, so for a 1920x1080px image this will be *1.7778*. Will be calculated automatically if omitted
- **contain** – Prevents the image from being cropped if it doesn't fit
- **content_class** – Apply a custom class to the responsive content div.
- **dark** – See description .

- **eager** – Will force the components content to render on mounted. This is useful if you have content that will not be rendered in the DOM that you want crawled for SEO.
- **gradient** – See description .
- **height** – Sets the height for the component.
- **lazy_src** – See description .
- **light** – Applies the light theme variant to the component.
- **max_height** – Sets the maximum height for the component.
- **max_width** – Sets the maximum width for the component.
- **min_height** – Sets the minimum height for the component.
- **min_width** – Sets the minimum width for the component.
- **options** – See description .
- **position** – See description .
- **sizes** – See description .
- **src** – The image URL. This prop is mandatory
- **srcset** – See description .
- **transition** – The transition to use when switching from *lazy-src* to *src*
- **width** – Sets the width for the component.

Events

Parameters

- **error** – Emitted when there is an error
- **load** – Emitted when image is loaded
- **loadstart** – Emitted when the image starts to load

class trame.html.vuetify.**VInput**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VInput component. See more info and examples .

Parameters

- **append_icon** – Appends an icon to the component, uses the same syntax as *v-icon*
- **background_color** – Changes the background-color of the input
- **color** – See description .
- **dark** – See description .
- **dense** – Reduces the input height
- **disabled** – Disable the input
- **error** – Puts the input in a manual error state
- **error_count** – The total number of errors that should display at once
- **error_messages** – Puts the input in an error state and passes through custom error messages. Will be combined with any validations that occur from the **rules** prop. This field will not trigger validation

- **height** – Sets the height of the input
- **hide_details** – Hides hint and validation errors. When set to *auto* messages will be rendered only if there's a message (hint, error message, counter value etc) to display
- **hide_spin_buttons** – Hides spin buttons on the input when type is set to *number*.
- **hint** – Hint text
- **id** – Sets the DOM id on the component
- **label** – Sets input label
- **light** – Applies the light theme variant to the component.
- **loading** – Displays linear progress bar. Can either be a String which specifies which color is applied to the progress bar (any material color or theme color - **primary**, **secondary**, **success**, **info**, **warning**, **error**) or a Boolean which uses the component **color** (set by color prop - if it's supported by the component) or the primary color
- **messages** – Displays a list of messages or message if using a string
- **persistent_hint** – Forces hint to always be visible
- **prepend_icon** – Prepends an icon to the component, uses the same syntax as *v-icon*
- **readonly** – Puts input in readonly state
- **rules** – Accepts a mixed array of types *function*, *boolean* and *string*. Functions pass an input value as an argument and must return either *true* / *false* or a *string* containing an error message. The input field will enter an error state if a function returns (or any value in the array contains) *false* or is a *string*
- **success** – Puts the input in a manual success state
- **success_messages** – Puts the input in a success state and passes through custom success messages.
- **validate_on_blur** – Delays validation until blur event
- **value** – The input's value

Events

Parameters

- **change** – Emitted when the input is changed by user interaction
- **click_append** – Emitted when appended icon is clicked
- **click_prepend** – Emitted when prepended icon is clicked
- **update_error** – The *error.sync* event

class trame.html.vuetify.**VItem**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VItem component. See more info and examples .

Parameters

- **active_class** – See description .
- **disabled** – Removes the ability to click or target the component.
- **value** – The value used when the component is selected in a group. If not provided, the index will be used.

class trame.html.vuetify.**VItemGroup**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VItemGroup component. See more info and examples .

Parameters

- **active_class** – See description .
- **dark** – See description .
- **light** – Applies the light theme variant to the component.
- **mandatory** – Forces a value to always be selected (if available).
- **max** – Sets a maximum number of selections that can be made.
- **multiple** – Allow multiple selections. The **value** prop must be an _array_.
- **tag** – Specify a custom tag used on the root element.
- **value** – The designated model value for the component.
- **value_comparator** – Apply a custom value comparator function

Events

Parameters **change** – Emitted when the component value is changed by user interaction

class trame.html.vuetify.**VLazy**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VLazy component. See more info and examples .

Parameters

- **height** – Sets the height for the component.
- **max_height** – Sets the maximum height for the component.
- **max_width** – Sets the maximum width for the component.
- **min_height** – Sets the minimum height for the component.
- **min_width** – Sets the minimum width for the component.
- **options** – See description .
- **tag** – Specify a custom tag used on the root element.
- **transition** – See description .
- **value** – Controls whether the component is visible or hidden.
- **width** – Sets the width for the component.

class trame.html.vuetify.**VListItemActionText**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VListItemActionText component. See more info and examples .

Parameters **tag** – Specify a custom tag used on the root element.

class trame.html.vuetify.**VListItemContent**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VListItemContent component. See more info and examples .

Parameters **tag** – Specify a custom tag used on the root element.

class trame.html.vuetify.**VListItemTitle**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VListItemTitle component. See more info and examples .

Parameters **tag** – Specify a custom tag used on the root element.

class trame.html.vuetify.**VListItemSubtitle**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VListItemSubtitle component. See more info and examples .

Parameters **tag** – Specify a custom tag used on the root element.

class trame.html.vuetify.**VList**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VList component. See more info and examples .

Parameters

- **color** – See description .
- **dark** – See description .
- **dense** – Lowers max height of list tiles
- **disabled** – Disables all children *v-list-item* components
- **elevation** – See description .
- **expand** – Will only collapse when explicitly closed
- **flat** – Remove the highlighted background on active `v-list-item`s
- **height** – Sets the height for the component.
- **light** – Applies the light theme variant to the component.
- **max_height** – Sets the maximum height for the component.
- **max_width** – Sets the maximum width for the component.
- **min_height** – Sets the minimum height for the component.
- **min_width** – Sets the minimum width for the component.
- **nav** – See description .
- **outlined** – Removes elevation (box-shadow) and adds a *thin* border.
- **rounded** – Rounds the *v-list-item* edges
- **shaped** – Provides an alternative active style for *v-list-item*.
- **subheader** – Removes top padding. Used when previous sibling is a header
- **tag** – Specify a custom tag used on the root element.
- **three_line** – See description .
- **tile** – Removes the component's **border-radius**.
- **two_line** – See description .
- **width** – Sets the width for the component.

class trame.html.vuetify.**VListGroup**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VListGroup component. See more info and examples .

Parameters

- **active_class** – See description .
- **append_icon** – Appends an icon to the component, uses the same syntax as *v-icon*
- **color** – See description .
- **disabled** – Disables all children *v-list-item* components
- **eager** – Will force the components content to render on mounted. This is useful if you have content that will not be rendered in the DOM that you want crawled for SEO.
- **group** – Assign a route namespace. Accepts a string or regexp for determining active state
- **no_action** – Removes left padding assigned for action icons from group items
- **prepend_icon** – Prepends an icon to the component, uses the same syntax as *v-icon*
- **ripple** – See description .
- **sub_group** – Designate the component as nested list group
- **value** – Expands / Collapse the list-group

class trame.html.vuetify.**VListItem**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VListItem component. See more info and examples .

Parameters

- **active_class** – See description .
- **append** – See description .
- **color** – Applies specified color to the control when in an **active** state or **input-value** is **true** - it can be the name of material color (for example *success* or *purple*) or css color (*#033* or *rgba(255, 0, 0, 0.5)*)
- **dark** – See description .
- **dense** – Lowers max height of list tiles
- **disabled** – Disables the component
- **exact** – See description .
- **exact_active_class** – See description .
- **exact_path** – See description .
- **href** – Designates the component as anchor and applies the **href** attribute.
- **inactive** – If set, the list tile will not be rendered as a link even if it has to/href prop or @click handler
- **input_value** – Controls the **active** state of the item. This is typically used to highlight the component
- **light** – Applies the light theme variant to the component.
- **link** – Designates that the component is a link. This is automatic when using the **href** or **to** prop.

- **nuxt** – See description .
- **replace** – See description .
- **ripple** – See description .
- **selectable** – See description .
- **tag** – Specify a custom tag used on the root element.
- **target** – Designates the target attribute. This should only be applied when using the **href** prop.
- **three_line** – See description .
- **to** – See description .
- **two_line** – See description .
- **value** – See description .

Events

Parameters **keydown** –

class trame.html.vuetify.**VListItemAction**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VListItemAction component. See more info and examples .

class trame.html.vuetify.**VListItemAvatar**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VListItemAvatar component. See more info and examples .

Parameters

- **color** – See description .
- **height** – Sets the height for the component.
- **horizontal** – Uses an alternative horizontal style.
- **left** – See description .
- **max_height** – Sets the maximum height for the component.
- **max_width** – Sets the maximum width for the component.
- **min_height** – Sets the minimum height for the component.
- **min_width** – Sets the minimum width for the component.
- **right** – See description .
- **rounded** – See description .
- **size** – Sets the height and width of the component.
- **tile** – Removes the component's **border-radius**.
- **width** – Sets the width for the component.

class trame.html.vuetify.**VListItemIcon**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VListItemIcon component. See more info and examples .

class trame.html.vuetify.**VListItemGroup**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VListItemGroup component. See more info and examples .

Parameters

- **active_class** – The **active-class** applied to children when they are activated.
- **color** – See description .
- **dark** – See description .
- **light** – Applies the light theme variant to the component.
- **mandatory** – Forces a value to always be selected (if available).
- **max** – Sets a maximum number of selections that can be made.
- **multiple** – Allow multiple selections. The **value** prop must be an _array_.
- **tag** – Specify a custom tag used on the root element.
- **value** – Sets the active list-item inside the list-group
- **value_comparator** – Apply a custom value comparator function

Events

Parameters **change** – Emitted when the component value is changed by user interaction

class trame.html.vuetify.**VMain**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VMain component. See more info and examples .

Parameters **tag** – Specify a custom tag used on the root element.

class trame.html.vuetify.**VMenu**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VMenu component. See more info and examples .

Parameters

- **absolute** – Applies **position: absolute** to the component.
- **activator** – Designate a custom activator when the *activator* slot is not used. String can be any valid querySelector and Object can be any valid Node.
- **allow_overflow** – Removes overflow re-positioning for the content
- **attach** – Specifies which DOM element that this component should detach to. String can be any valid querySelector and Object can be any valid Node. This will attach to the root *v-app* component by default.
- **auto** – Centers list on selected element
- **bottom** – Aligns the component towards the bottom.
- **close_delay** – Milliseconds to wait before closing component. Only works with the **open-on-hover** prop
- **close_on_click** – Designates if menu should close on outside-activator click
- **close_on_content_click** – Designates if menu should close when its content is clicked

- **content_class** – Applies a custom class to the detached element. This is useful because the content is moved to the beginning of the *v-app* component (unless the **attach** prop is provided) and is not targetable by classes passed directly on the component.
- **dark** – See description .
- **disable_keys** – Removes all keyboard interaction
- **disabled** – Disables the menu
- **eager** – Will force the components content to render on mounted. This is useful if you have content that will not be rendered in the DOM that you want crawled for SEO.
- **internal_activator** – Detaches the menu content inside of the component as opposed to the document.
- **left** – Aligns the component towards the left.
- **light** – Applies the light theme variant to the component.
- **max_height** – Sets the max height of the menu content
- **max_width** – Sets the maximum width for the content
- **min_width** – Sets the minimum width for the content
- **nudge_bottom** – Nudge the content to the bottom
- **nudge_left** – Nudge the content to the left
- **nudge_right** – Nudge the content to the right
- **nudge_top** – Nudge the content to the top
- **nudge_width** – Nudge the content width
- **offset_overflow** – Causes the component to flip to the opposite side when repositioned due to overflow
- **offset_x** – Offset the menu on the x-axis. Works in conjunction with direction left/right
- **offset_y** – Offset the menu on the y-axis. Works in conjunction with direction top/bottom
- **open_delay** – Milliseconds to wait before opening component. Only works with the **open-on-hover** prop
- **open_on_click** – Designates whether menu should open on activator click
- **open_on_focus** –
- **open_on_hover** – Designates whether menu should open on activator hover
- **origin** – See description .
- **position_x** – Used to position the content when not using an activator slot
- **position_y** – Used to position the content when not using an activator slot
- **return_value** – The value that is updated when the menu is closed - must be primitive. Dot notation is supported
- **right** – Aligns the component towards the right.
- **rounded** – See description .
- **tile** – Removes the component's **border-radius**.
- **top** – Aligns the content towards the top.

- **transition** – See description .
- **value** – Controls whether the component is visible or hidden.
- **z_index** – The z-index used for the component

Events

Parameters **input** – The updated bound model

class trame.html.vuetify.**VNavigationDrawer**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VNavigationDrawer component. See more info and examples .

Parameters

- **absolute** – Applies **position: absolute** to the component.
- **app** – See description .
- **bottom** – Expands from the bottom of the screen on mobile devices
- **clipped** – A clipped drawer rests under the application toolbar. **Note:** requires the **clipped-left** or **clipped-right** prop on *v-app-bar* to work as intended
- **color** – See description .
- **dark** – See description .
- **disable_resize_watcher** – Will automatically open/close drawer when resized depending if mobile or desktop.
- **disable_route_watcher** – Disables opening of navigation drawer when route changes
- **expand_on_hover** – Collapses the drawer to a **mini-variant** until hovering with the mouse
- **fixed** – Applies **position: fixed** to the component.
- **floating** – A floating drawer has no visible container (no border-right)
- **height** – Sets the height of the navigation drawer
- **hide_overlay** – Hides the display of the overlay.
- **light** – Applies the light theme variant to the component.
- **mini_variant** – Condenses navigation drawer width, also accepts the **.sync** modifier. With this, the drawer will re-open when clicking it
- **mini_variant_width** – Designates the width assigned when the *mini* prop is turned on
- **mobile_breakpoint** – Sets the designated mobile breakpoint for the component. This will apply alternate styles for mobile devices such as the *temporary* prop, or activate the *bottom* prop when the breakpoint value is met. Setting the value to *0* will disable this functionality.
- **overlay_color** – Sets the overlay color.
- **overlay_opacity** – Sets the overlay opacity.
- **permanent** – The drawer remains visible regardless of screen size
- **right** – Places the navigation drawer on the right
- **src** – See description .
- **stateless** – Remove all automated state functionality (resize, mobile, route) and manually control the drawer state

- **tag** – Specify a custom tag used on the root element.
- **temporary** – A temporary drawer sits above its application and uses a scrim (overlay) to darken the background
- **touchless** – Disable mobile touch functionality
- **value** – Controls whether the component is visible or hidden.
- **width** – Sets the width for the component.

Events

Parameters

- **input** – The updated bound model
- **transitionend** – Emits event object when transition is complete.
- **update_mini_variant** – The *mini-variant.sync* event

class trame.html.vuetify.**VOtpInput**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VOtpInput component. See more info and examples .

Parameters

- **dark** – See description .
- **disabled** – Disable the input
- **hide_spin_buttons** – Hides spin buttons on the input when type is set to *number*.
- **id** – Sets the DOM id on the component
- **length** – The OTP field's length
- **plain** – Outlined style applied by default to the input, set to *true* to apply plain style
- **readonly** – Puts input in readonly state
- **type** – Supported types: *text*, *password*, *number*
- **value** – The input's value

Events

Parameters

- **change** – Emitted when the input is changed by user interaction
- **finish** – Emitted when the input is filled completely and cursor is blurred
- **input** – The updated bound model

class trame.html.vuetify.**VOverflowBtn**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VOverflowBtn component. See more info and examples .

Parameters

- **allow_overflow** – Allow the menu to overflow off the screen
- **append_icon** – Appends an icon to the component, uses the same syntax as *v-icon*
- **append_outer_icon** – Appends an icon to the outside the component's input, uses same syntax as *v-icon*

- **attach** – Specifies which DOM element that this component should detach to. String can be any valid querySelector and Object can be any valid Node. This will attach to the root *v-app* component by default.
- **auto_select_first** – When searching, will always highlight the first option
- **autofocus** – Enables autofocus
- **background_color** – Changes the background-color of the input
- **cache_items** – Keeps a local _unique_ copy of all items that have been passed through the **items** prop.
- **chips** – Changes display of selections to chips
- **clear_icon** – Applied when using **clearable** and the input is dirty
- **clearable** – Add input clear functionality, default icon is Material Design Icons **mdi-clear**
- **color** – See description .
- **counter** – Creates counter for input length; if no number is specified, it defaults to 25. Does not apply any validation.
- **counter_value** –
- **dark** – See description .
- **deletable_chips** – Adds a remove icon to selected chips
- **dense** – Reduces the input height
- **disable_lookup** – Disables keyboard lookup
- **disabled** – Disables the input
- **eager** – Will force the components content to render on mounted. This is useful if you have content that will not be rendered in the DOM that you want crawled for SEO.
- **editable** – Creates an editable button
- **error** – Puts the input in a manual error state
- **error_count** – The total number of errors that should display at once
- **error_messages** – Puts the input in an error state and passes through custom error messages. Will be combined with any validations that occur from the **rules** prop. This field will not trigger validation
- **filled** – Applies the alternate filled input style
- **filter** – See description .
- **flat** – Removes elevation (shadow) added to element when using the **solo** or **solo-inverted** props
- **full_width** – Designates input type as full-width
- **height** – Sets the height of the input
- **hide_details** – Hides hint and validation errors. When set to *auto* messages will be rendered only if there's a message (hint, error message, counter value etc) to display
- **hide_no_data** – Hides the menu when there are no options to show. Useful for preventing the menu from opening before results are fetched asynchronously. Also has the effect of opening the menu when the *items* array changes if not already open.
- **hide_selected** – Do not display in the select menu items that are already selected

- **hide_spin_buttons** – Hides spin buttons on the input when type is set to *number*.
- **hint** – Hint text
- **id** – Sets the DOM id on the component
- **item_color** – Sets color of selected items
- **item_disabled** – Set property of **items**'s disabled value
- **item_text** – Set property of **items**'s text value
- **item_value** – See description .
- **items** – Can be an array of objects or array of strings. When using objects, will look for a text, value and disabled keys. This can be changed using the **item-text**, **item-value** and **item-disabled** props. Objects that have a **header** or **divider** property are considered special cases and generate a list header or divider; these items are not selectable.
- **label** – Sets input label
- **light** – Applies the light theme variant to the component.
- **loader_height** – Specifies the height of the loader
- **loading** – Displays linear progress bar. Can either be a String which specifies which color is applied to the progress bar (any material color or theme color - **primary**, **secondary**, **success**, **info**, **warning**, **error**) or a Boolean which uses the component **color** (set by color prop - if it's supported by the component) or the primary color
- **menu_props** – Pass props through to the *v-menu* component. Accepts either a string for boolean props *menu-props="auto, overflowY"*, or an object *:menu-props="{ auto: true, overflowY: true }"*
- **messages** – Displays a list of messages or message if using a string
- **multiple** – Changes select to multiple. Accepts array for value
- **no_data_text** – Display text when there is no data
- **no_filter** – Do not apply filtering when searching. Useful when data is being filtered server side
- **open_on_clear** – When using the **clearable** prop, once cleared, the select menu will either open or stay open, depending on the current state
- **outlined** – Applies the outlined style to the input
- **persistent_hint** – Forces hint to always be visible
- **persistent_placeholder** – Forces placeholder to always be visible
- **placeholder** – Sets the input's placeholder text
- **prefix** – Displays prefix text
- **prepend_icon** – Prepends an icon to the component, uses the same syntax as *v-icon*
- **prepend_inner_icon** – Prepends an icon inside the component's input, uses the same syntax as *v-icon*
- **readonly** – Puts input in readonly state
- **return_object** – Changes the selection behavior to return the object directly rather than the value specified with **item-value**
- **reverse** – Reverses the input orientation

- **rounded** – Adds a border radius to the input
- **rules** – Accepts a mixed array of types *function*, *boolean* and *string*. Functions pass an input value as an argument and must return either *true* / *false* or a *string* containing an error message. The input field will enter an error state if a function returns (or any value in the array contains) *false* or is a *string*
- **search_input** – Search value. Can be used with *.sync* modifier.
- **segmented** – Creates a segmented button
- **shaped** – Round if *outlined* and increase *border-radius* if *filled*. Must be used with either *outlined* or *filled*
- **single_line** – Label does not move on focus/dirty
- **small_chips** – Changes display of selections to chips with the **small** property
- **solo** – Changes the style of the input
- **solo_inverted** – Reduces element opacity until focused
- **success** – Puts the input in a manual success state
- **success_messages** – Puts the input in a success state and passes through custom success messages.
- **suffix** – Displays suffix text
- **type** – Sets input type
- **validate_on_blur** – Delays validation until blur event
- **value** – The input's value
- **value_comparator** – See description .

Events

Parameters

- **blur** – Emitted when the input is blurred
- **change** – Emitted when the input is changed by user interaction
- **click_append** – Emitted when appended icon is clicked
- **click_append_outer** – Emitted when appended outer icon is clicked
- **click_clear** – Emitted when clearable icon clicked
- **click_prepend** – Emitted when prepended icon is clicked
- **click_prepend_inner** – Emitted when prepended inner icon is clicked
- **focus** – Emitted when component is focused
- **input** – The updated bound model
- **keydown** – Emitted when **any** key is pressed
- **update_error** – The *error.sync* event
- **update_list_index** – Emitted when menu item is selected using keyboard arrows
- **update_search_input** – The *search-input.sync* event

class trame.html.vuetify.**VOverlay**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VOverlay component. See more info and examples .

Parameters

- **absolute** – Applies **position: absolute** to the component.
- **color** – See description .
- **dark** – See description .
- **light** – Applies the light theme variant to the component.
- **opacity** – Sets the overlay opacity
- **value** – Controls whether the component is visible or hidden.
- **z_index** – The z-index used for the component

class trame.html.vuetify.**VPagination**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VPagination component. See more info and examples .

Parameters

- **circle** – Shape pagination elements as circles
- **color** – See description .
- **current_page_aria_label** –
- **dark** – See description .
- **disabled** – Disables component
- **length** – The length of the pagination component
- **light** – Applies the light theme variant to the component.
- **next_aria_label** –
- **next_icon** – Specify the icon to use for the next icon
- **page_aria_label** –
- **prev_icon** – Specify the icon to use for the prev icon
- **previous_aria_label** –
- **total_visible** – Specify the max total visible pagination numbers
- **value** – Current selected page
- **wrapper_aria_label** –

Events

Parameters

- **input** – The updated bound model
- **next** – Emitted when going to next item
- **previous** – Emitted when going to previous item

class trame.html.vuetify.**VSheet**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VSheet component. See more info and examples .

Parameters

- **color** – See description .
- **dark** – See description .
- **elevation** – See description .
- **height** – Sets the height for the component.
- **light** – Applies the light theme variant to the component.
- **max_height** – Sets the maximum height for the component.
- **max_width** – Sets the maximum width for the component.
- **min_height** – Sets the minimum height for the component.
- **min_width** – Sets the minimum width for the component.
- **outlined** – Removes elevation (box-shadow) and adds a *thin* border.
- **rounded** – See description .
- **shaped** – Applies a large border radius on the top left and bottom right of the card.
- **tag** – Specify a custom tag used on the root element.
- **tile** – Removes the component's **border-radius**.
- **width** – Sets the width for the component.

class trame.html.vuetify.**VParallax**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VParallax component. See more info and examples .

Parameters

- **alt** – Attaches an alt property to the parallax image
- **height** – Sets the height for the component
- **src** – The image to parallax
- **srcset** – See description .

class trame.html.vuetify.**VProgressCircular**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VProgressCircular component. See more info and examples .

Parameters

- **button** – Deprecated - Pending removal
- **color** – See description .
- **indeterminate** – Constantly animates, use when loading progress is unknown.
- **rotate** – Rotates the circle start point in deg
- **size** – Sets the diameter of the circle in pixels
- **value** – The percentage value for current progress

- **width** – Sets the stroke of the circle in pixels

class trame.html.vuetify.**VProgressLinear**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VProgressLinear component. See more info and examples .

Parameters

- **absolute** – Applies **position: absolute** to the component.
- **active** – Reduce the height to 0, hiding component
- **background_color** – Background color, set to component's color if null
- **background_opacity** – Background opacity, if null it defaults to 0.3 if background color is not specified or 1 otherwise
- **bottom** – Aligns the component towards the bottom.
- **buffer_value** – The percentage value for the buffer
- **color** – See description .
- **dark** – See description .
- **fixed** – Applies **position: fixed** to the component.
- **height** – Sets the height for the component
- **indeterminate** – Constantly animates, use when loading progress is unknown.
- **light** – Applies the light theme variant to the component.
- **query** – Animates like **indeterminate** prop but inverse
- **reverse** – Displays reversed progress (right to left in LTR mode and left to right in RTL)
- **rounded** – Adds a border radius to the progress component
- **stream** – An alternative style for portraying loading that works in tandem with **buffer-value**
- **striped** – Adds a stripe background to the filled portion of the progress component
- **top** – Aligns the content towards the top.
- **value** – The designated model value for the component.

Events

Parameters **change** – Emitted when the component value is changed by user interaction

class trame.html.vuetify.**VRadioGroup**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VRadioGroup component. See more info and examples .

Parameters

- **active_class** – The **active-class** applied to children when they are activated.
- **append_icon** – Appends an icon to the component, uses the same syntax as *v-icon*
- **background_color** – Changes the background-color of the input
- **column** – Displays radio buttons in column
- **dark** – See description .
- **dense** – Reduces the input height

- **disabled** – Disable the input
- **error** – Puts the input in a manual error state
- **error_count** – The total number of errors that should display at once
- **error_messages** – Puts the input in an error state and passes through custom error messages. Will be combined with any validations that occur from the **rules** prop. This field will not trigger validation
- **hide_details** – Hides hint and validation errors. When set to *auto* messages will be rendered only if there's a message (hint, error message, counter value etc) to display
- **hide_spin_buttons** – Hides spin buttons on the input when type is set to *number*.
- **hint** – Hint text
- **id** – Sets the DOM id on the component
- **label** – Sets input label
- **light** – Applies the light theme variant to the component.
- **mandatory** – Forces a value to always be selected (if available).
- **max** – Sets a maximum number of selections that can be made.
- **messages** – Displays a list of messages or message if using a string
- **multiple** – Allow multiple selections. The **value** prop must be an _array_.
- **name** – Sets the component's name attribute
- **persistent_hint** – Forces hint to always be visible
- **prepend_icon** – Prepends an icon to the component, uses the same syntax as *v-icon*
- **readonly** – Puts input in readonly state
- **row** – Displays radio buttons in row
- **rules** – Accepts a mixed array of types *function*, *boolean* and *string*. Functions pass an input value as an argument and must return either *true* / *false* or a *string* containing an error message. The input field will enter an error state if a function returns (or any value in the array contains) *false* or is a *string*
- **success** – Puts the input in a manual success state
- **success_messages** – Puts the input in a success state and passes through custom success messages.
- **tag** – Specify a custom tag used on the root element.
- **validate_on_blur** – Delays validation until blur event
- **value** – The input's value
- **value_comparator** – Apply a custom value comparator function

Events

Parameters

- **change** – Emitted when the input is changed by user interaction
- **click_append** – Emitted when appended icon is clicked
- **click_prepend** – Emitted when prepended icon is clicked

- **update_error** – The *error.sync* event

class trame.html.vuetify.**VRadio**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VRadio component. See more info and examples .

Parameters

- **active_class** – See description .
- **color** – See description .
- **dark** – See description .
- **disabled** – Removes the ability to click or target the component.
- **id** – Sets the DOM id on the component
- **label** – Sets input label
- **light** – Applies the light theme variant to the component.
- **name** – Sets the component's name attribute
- **off_icon** – The icon used when inactive
- **on_icon** – The icon used when active
- **readonly** – Puts input in readonly state
- **ripple** – See description .
- **value** – The value used when the component is selected in a group. If not provided, the index will be used.

Events

Parameters

- **change** – Emitted when the input is changed by user interaction
- **click_append** – Emitted when appended icon is clicked
- **click_prepend** – Emitted when prepended icon is clicked
- **update_error** – The *error.sync* event

class trame.html.vuetify.**VRangeSlider**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VRangeSlider component. See more info and examples .

Parameters

- **append_icon** – Appends an icon to the component, uses the same syntax as *v-icon*
- **background_color** – Changes the background-color of the input
- **color** – See description .
- **dark** – See description .
- **dense** – Reduces the input height
- **disabled** – Disable the input
- **error** – Puts the input in a manual error state
- **error_count** – The total number of errors that should display at once

- **error_messages** – Puts the input in an error state and passes through custom error messages. Will be combined with any validations that occur from the **rules** prop. This field will not trigger validation
- **height** – Sets the height of the input
- **hide_details** – Hides hint and validation errors. When set to *auto* messages will be rendered only if there's a message (hint, error message, counter value etc) to display
- **hide_spin_buttons** – Hides spin buttons on the input when type is set to *number*.
- **hint** – Hint text
- **id** – Sets the DOM id on the component
- **inverse_label** – Reverse the label position. Works with **rtl**.
- **label** – Sets input label
- **light** – Applies the light theme variant to the component.
- **loader_height** – Specifies the height of the loader
- **loading** – Displays linear progress bar. Can either be a String which specifies which color is applied to the progress bar (any material color or theme color - **primary**, **secondary**, **success**, **info**, **warning**, **error**) or a Boolean which uses the component **color** (set by color prop - if it's supported by the component) or the primary color
- **max** – Sets the maximum allowed value
- **messages** – Displays a list of messages or message if using a string
- **min** – Sets the minimum allowed value
- **persistent_hint** – Forces hint to always be visible
- **prepend_icon** – Prepends an icon to the component, uses the same syntax as *v-icon*
- **readonly** – Puts input in readonly state
- **rules** – Accepts a mixed array of types *function*, *boolean* and *string*. Functions pass an input value as an argument and must return either *true* / *false* or a *string* containing an error message. The input field will enter an error state if a function returns (or any value in the array contains) *false* or is a *string*
- **step** – If greater than 0, sets step interval for ticks
- **success** – Puts the input in a manual success state
- **success_messages** – Puts the input in a success state and passes through custom success messages.
- **thumb_color** – Sets the thumb and thumb label color
- **thumb_label** – Show thumb label. If *true* it shows label when using slider. If set to *'always'* it always shows label.
- **thumb_size** – Controls the size of the thumb label.
- **tick_labels** – When provided with Array<string>, will attempt to map the labels to each step in index order
- **tick_size** – Controls the size of **ticks**
- **ticks** – Show track ticks. If *true* it shows ticks when using slider. If set to *'always'* it always shows ticks.

- **track_color** – Sets the track's color
- **track_fill_color** – Sets the track's fill color
- **validate_on_blur** – Delays validation until blur event
- **value** – The input's value
- **vertical** – Changes slider direction to vertical

Events

Parameters

- **change** – Emitted when the input is changed by user interaction
- **click_append** – Emitted when appended icon is clicked
- **click_prepend** – Emitted when prepended icon is clicked
- **end** – Slider value emitted at the end of slider movement
- **input** – The updated bound model
- **start** – Slider value emitted at start of slider movement
- **update_error** – The *error.sync* event

class trame.html.vuetify.**VRating**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VRating component. See more info and examples .

Parameters

- **background_color** – The color used empty icons
- **clearable** – Allows for the component to be cleared. Triggers when the icon containing the current value is clicked.
- **close_delay** – Milliseconds to wait before closing component.
- **color** – See description .
- **dark** – See description .
- **dense** – Icons have a smaller size
- **empty_icon** – The icon displayed when empty
- **full_icon** – The icon displayed when full
- **half_icon** – The icon displayed when half (requires **half-increments** prop)
- **half_increments** – Allows the selection of half increments
- **hover** – Provides visual feedback when hovering over icons
- **icon_label** – The **aria-label** used for icons
- **large** – Makes the component large.
- **length** – The amount of ratings to show
- **light** – Applies the light theme variant to the component.
- **open_delay** – Milliseconds to wait before opening component.
- **readonly** – Removes all hover effects and pointer events
- **ripple** – See description .

- **size** – Sets the height and width of the component.
- **small** – Makes the component small.
- **value** – The rating value
- **x_large** – Makes the component extra large.
- **x_small** – Makes the component extra small.

Events

Parameters **input** – Emits the rating number when this value changes

class trame.html.vuetify.**VResponsive**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VResponsive component. See more info and examples .

Parameters

- **aspect_ratio** – Sets a base aspect ratio, calculated as width/height. This will only set a **minimum** height, the component can still grow if it has a lot of content.
- **content_class** – Apply a custom class to the responsive content div.
- **height** – Sets the height for the component.
- **max_height** – Sets the maximum height for the component.
- **max_width** – Sets the maximum width for the component.
- **min_height** – Sets the minimum height for the component.
- **min_width** – Sets the minimum width for the component.
- **width** – Sets the width for the component.

class trame.html.vuetify.**VSelect**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VSelect component. See more info and examples .

Parameters

- **append_icon** – Appends an icon to the component, uses the same syntax as *v-icon*
- **append_outer_icon** – Appends an icon to the outside the component's input, uses same syntax as *v-icon*
- **attach** – Specifies which DOM element that this component should detach to. String can be any valid querySelector and Object can be any valid Node. This will attach to the root *v-app* component by default.
- **autofocus** – Enables autofocus
- **background_color** – Changes the background-color of the input
- **cache_items** – Keeps a local _unique_ copy of all items that have been passed through the **items** prop.
- **chips** – Changes display of selections to chips
- **clear_icon** – Applied when using **clearable** and the input is dirty
- **clearable** – Add input clear functionality, default icon is Material Design Icons **mdi-clear**
- **color** – See description .

- **counter** – Creates counter for input length; if no number is specified, it defaults to 25. Does not apply any validation.
- **counter_value** –
- **dark** – See description .
- **deletable_chips** – Adds a remove icon to selected chips
- **dense** – Reduces the input height
- **disable_lookup** – Disables keyboard lookup
- **disabled** – Disables the input
- **eager** – Will force the components content to render on mounted. This is useful if you have content that will not be rendered in the DOM that you want crawled for SEO.
- **error** – Puts the input in a manual error state
- **error_count** – The total number of errors that should display at once
- **error_messages** – Puts the input in an error state and passes through custom error messages. Will be combined with any validations that occur from the **rules** prop. This field will not trigger validation
- **filled** – Applies the alternate filled input style
- **flat** – Removes elevation (shadow) added to element when using the **solo** or **solo-inverted** props
- **full_width** – Designates input type as full-width
- **height** – Sets the height of the input
- **hide_details** – Hides hint and validation errors. When set to *auto* messages will be rendered only if there's a message (hint, error message, counter value etc) to display
- **hide_selected** – Do not display in the select menu items that are already selected
- **hide_spin_buttons** – Hides spin buttons on the input when type is set to *number*.
- **hint** – Hint text
- **id** – Sets the DOM id on the component
- **item_color** – Sets color of selected items
- **item_disabled** – Set property of **items**'s disabled value
- **item_text** – Set property of **items**'s text value
- **item_value** – See description .
- **items** – Can be an array of objects or array of strings. When using objects, will look for a text, value and disabled keys. This can be changed using the **item-text**, **item-value** and **item-disabled** props. Objects that have a **header** or **divider** property are considered special cases and generate a list header or divider; these items are not selectable.
- **label** – Sets input label
- **light** – Applies the light theme variant to the component.
- **loader_height** – Specifies the height of the loader
- **loading** – Displays linear progress bar. Can either be a String which specifies which color is applied to the progress bar (any material color or theme color - **primary**, **secondary**,

success, **info**, **warning**, **error**) or a Boolean which uses the component **color** (set by color prop - if it's supported by the component) or the primary color

- `menu_props` – Pass props through to the *v-menu* component. Accepts either a string for boolean props *menu-props="auto, overflowY"*, or an object *:menu-props="{ auto: true, overflowY: true }"*
- `messages` – Displays a list of messages or message if using a string
- `multiple` – Changes select to multiple. Accepts array for value
- `no_data_text` – Display text when there is no data
- `open_on_clear` – When using the **clearable** prop, once cleared, the select menu will either open or stay open, depending on the current state
- `outlined` – Applies the outlined style to the input
- `persistent_hint` – Forces hint to always be visible
- `persistent_placeholder` – Forces placeholder to always be visible
- `placeholder` – Sets the input's placeholder text
- `prefix` – Displays prefix text
- `prepend_icon` – Prepends an icon to the component, uses the same syntax as *v-icon*
- `prepend_inner_icon` – Prepends an icon inside the component's input, uses the same syntax as *v-icon*
- `readonly` – Puts input in readonly state
- `return_object` – Changes the selection behavior to return the object directly rather than the value specified with **item-value**
- `reverse` – Reverses the input orientation
- `rounded` – Adds a border radius to the input
- `rules` – Accepts a mixed array of types *function*, *boolean* and *string*. Functions pass an input value as an argument and must return either *true* / *false* or a *string* containing an error message. The input field will enter an error state if a function returns (or any value in the array contains) *false* or is a *string*
- `shaped` – Round if *outlined* and increase *border-radius* if *filled*. Must be used with either *outlined* or *filled*
- `single_line` – Label does not move on focus/dirty
- `small_chips` – Changes display of selections to chips with the **small** property
- `solo` – Changes the style of the input
- `solo_inverted` – Reduces element opacity until focused
- `success` – Puts the input in a manual success state
- `success_messages` – Puts the input in a success state and passes through custom success messages.
- `suffix` – Displays suffix text
- `type` – Sets input type
- `validate_on_blur` – Delays validation until blur event
- `value` – The input's value

- **value_comparator** – See description .

Events

Parameters

- **blur** – Emitted when the input is blurred
- **change** – Emitted when the input is changed by user interaction
- **click_append** – Emitted when appended icon is clicked
- **click_append_outer** – Emitted when appended outer icon is clicked
- **click_clear** – Emitted when clearable icon clicked
- **click_prepend** – Emitted when prepended icon is clicked
- **click_prepend_inner** – Emitted when prepended inner icon is clicked
- **focus** – Emitted when component is focused
- **input** – The updated bound model
- **keydown** – Emitted when **any** key is pressed
- **update_error** – The *error.sync* event
- **update_list_index** – Emitted when menu item is selected using keyboard arrows
- **update_search_input** – The *search-input.sync* event

class trame.html.vuetify.**VSkeletonLoader**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VSkeletonLoader component. See more info and examples .

Parameters

- **boilerplate** – Remove the loading animation from the skeleton
- **dark** – See description .
- **elevation** – See description .
- **height** – Sets the height for the component.
- **light** – Applies the light theme variant to the component.
- **loading** – Applies a loading animation with a on-hover loading cursor. A value of **false** will only work when there is content in the *default* slot.
- **max_height** – Sets the maximum height for the component.
- **max_width** – Sets the maximum width for the component.
- **min_height** – Sets the minimum height for the component.
- **min_width** – Sets the minimum width for the component.
- **tile** – Removes the component's border-radius
- **transition** – See description .
- **type** – A string delimited list of skeleton components to create such as *type="text@3"* or *type="card, list-item"*. Will recursively generate a corresponding skeleton from the provided string. Also supports short-hand for multiple elements such as **article@3** and **paragraph@2** which will generate 3 _article_ skeletons and 2 _paragraph_ skeletons. Please see below for a list of available pre-defined options.

- **types** – A custom types object that will be combined with the pre-defined options. For a list of available pre-defined options, see the **type** prop.
- **width** – Sets the width for the component.

class trame.html.vuetify.**VSlider**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VSlider component. See more info and examples .

Parameters

- **append_icon** – Appends an icon to the component, uses the same syntax as *v-icon*
- **background_color** – Changes the background-color of the input
- **color** – See description .
- **dark** – See description .
- **dense** – Reduces the input height
- **disabled** – Disable the input
- **error** – Puts the input in a manual error state
- **error_count** – The total number of errors that should display at once
- **error_messages** – Puts the input in an error state and passes through custom error messages. Will be combined with any validations that occur from the **rules** prop. This field will not trigger validation
- **height** – Sets the height of the input
- **hide_details** – Hides hint and validation errors. When set to *auto* messages will be rendered only if there's a message (hint, error message, counter value etc) to display
- **hide_spin_buttons** – Hides spin buttons on the input when type is set to *number*.
- **hint** – Hint text
- **id** – Sets the DOM id on the component
- **inverse_label** – Reverse the label position. Works with **rtl**.
- **label** – Sets input label
- **light** – Applies the light theme variant to the component.
- **loader_height** – Specifies the height of the loader
- **loading** – Displays linear progress bar. Can either be a String which specifies which color is applied to the progress bar (any material color or theme color - **primary**, **secondary**, **success**, **info**, **warning**, **error**) or a Boolean which uses the component **color** (set by color prop - if it's supported by the component) or the primary color
- **max** – Sets the maximum allowed value
- **messages** – Displays a list of messages or message if using a string
- **min** – Sets the minimum allowed value
- **persistent_hint** – Forces hint to always be visible
- **prepend_icon** – Prepends an icon to the component, uses the same syntax as *v-icon*
- **readonly** – Puts input in readonly state

- **rules** – Accepts a mixed array of types *function*, *boolean* and *string*. Functions pass an input value as an argument and must return either *true* / *false* or a *string* containing an error message. The input field will enter an error state if a function returns (or any value in the array contains) *false* or is a *string*
- **step** – If greater than 0, sets step interval for ticks
- **success** – Puts the input in a manual success state
- **success_messages** – Puts the input in a success state and passes through custom success messages.
- **thumb_color** – Sets the thumb and thumb label color
- **thumb_label** – Show thumb label. If *true* it shows label when using slider. If set to *'always'* it always shows label.
- **thumb_size** – Controls the size of the thumb label.
- **tick_labels** – When provided with Array<string>, will attempt to map the labels to each step in index order
- **tick_size** – Controls the size of **ticks**
- **ticks** – Show track ticks. If *true* it shows ticks when using slider. If set to *'always'* it always shows ticks.
- **track_color** – Sets the track's color
- **track_fill_color** – Sets the track's fill color
- **validate_on_blur** – Delays validation until blur event
- **value** – The input's value
- **vertical** – Changes slider direction to vertical

Events

Parameters

- **change** – Emitted when the input is changed by user interaction
- **click_append** – Emitted when appended icon is clicked
- **click_prepend** – Emitted when prepended icon is clicked
- **end** – Slider value emitted at the end of slider movement
- **input** – The updated bound model
- **start** – Slider value emitted at start of slider movement
- **update_error** – The *error.sync* event

class trame.html.vuetify.**VSlideGroup**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VSlideGroup component. See more info and examples .

Parameters

- **active_class** – The **active-class** applied to children when they are activated.
- **center_active** – Forces the selected component to be centered
- **dark** – See description .
- **light** – Applies the light theme variant to the component.

- **mandatory** – Forces a value to always be selected (if available).
- **max** – Sets a maximum number of selections that can be made.
- **mobile_breakpoint** – Sets the designated mobile breakpoint for the component.
- **multiple** – Allow multiple selections. The **value** prop must be an _array_.
- **next_icon** – The appended slot when arrows are shown
- **prev_icon** – The prepended slot when arrows are shown
- **show_arrows** – See description .
- **tag** – Specify a custom tag used on the root element.
- **value** – The designated model value for the component.
- **value_comparator** – Apply a custom value comparator function

Events

Parameters

- **change** – Emitted when the component value is changed by user interaction
- **click_next** – Emitted when the next is clicked
- **click_prev** – Emitted when the prev is clicked

class trame.html.vuetify.**VSlideItem**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VSlideItem component. See more info and examples .

Parameters

- **active_class** – See description .
- **disabled** – Removes the ability to click or target the component.
- **value** – The value used when the component is selected in a group. If not provided, the index will be used.

class trame.html.vuetify.**VSnackbar**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VSnackbar component. See more info and examples .

Parameters

- **absolute** – Applies **position: absolute** to the component.
- **app** – Respects boundaries of—and will not overlap with—other *app* components like *v-app-bar*, *v-navigation-drawer*, and *v-footer*.
- **bottom** – Aligns the component towards the bottom.
- **centered** – Positions the snackbar in the center of the screen, (x and y axis).
- **color** – See description .
- **content_class** – Apply a custom class to the snackbar content
- **dark** – See description .
- **elevation** – See description .
- **height** – Sets the height for the component.

- **left** – Aligns the component towards the left.
- **light** – Applies the light theme variant to the component.
- **max_height** – Sets the maximum height for the component.
- **max_width** – Sets the maximum width for the component.
- **min_height** – Sets the minimum height for the component.
- **min_width** – Sets the minimum width for the component.
- **multi_line** – Gives the snackbar a larger minimum height.
- **outlined** – Removes elevation (box-shadow) and adds a *thin* border.
- **right** – Aligns the component towards the right.
- **rounded** – See description .
- **shaped** – Applies a large border radius on the top left and bottom right of the card.
- **tag** – Specify a custom tag used on the root element.
- **text** – Applies the defined **color** to text and a low opacity background of the same.
- **tile** – Removes the component's **border-radius**.
- **timeout** – Time (in milliseconds) to wait until snackbar is automatically hidden. Use *-1* to keep open indefinitely (*0* in version < 2.3). It is recommended for this number to be between *4000* and *10000*. Changes to this property will reset the timeout.
- **top** – Aligns the content towards the top.
- **transition** – See description .
- **value** – Controls whether the component is visible or hidden.
- **vertical** – Stacks snackbar content on top of the actions (button).
- **width** – Sets the width for the component.

Events

Parameters **input** – The updated bound model

class trame.html.vuetify.**VSparkline**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VSparkline component. See more info and examples .

Parameters

- **auto_draw** – Trace the length of the line when first rendered
- **auto_draw_duration** – Amount of time (in ms) to run the trace animation
- **auto_draw_easing** – The easing function to use for the trace animation
- **auto_line_width** – Automatically expand bars to use space efficiently
- **color** – See description .
- **fill** – Using the **fill** property allows you to better customize the look and feel of your sparkline.
- **gradient** – An array of colors to use as a linear-gradient
- **gradient_direction** – The direction the gradient should run

- **height** – Height of the SVG trendline or bars
- **label_size** – The label font size
- **labels** – An array of string labels that correspond to the same index as its data counterpart
- **line_width** – The thickness of the line, in px
- **padding** – Low *smooth* or high *line-width* values may result in cropping, increase padding to compensate
- **show_labels** – Show labels below each data point
- **smooth** – Number of px to use as a corner radius. *true* defaults to 8, *false* is 0
- **type** – Choose between a trendline or bars
- **value** – An array of numbers.
- **width** – Width of the SVG trendline or bars

class trame.html.vuetify.**VSpeedDial**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VSpeedDial component. See more info and examples .

Parameters

- **absolute** – Applies **position: absolute** to the component.
- **bottom** – Aligns the component towards the bottom.
- **direction** – Direction in which speed-dial content will show. Possible values are *top*, *bottom*, *left*, *right*.
- **fixed** – Applies **position: fixed** to the component.
- **left** – Aligns the component towards the left.
- **mode** – See description .
- **open_on_hover** – Opens speed-dial on hover
- **origin** – See description .
- **right** – Aligns the component towards the right.
- **top** – Aligns the content towards the top.
- **transition** – See description .
- **value** – Controls whether the component is visible or hidden.

class trame.html.vuetify.**VStepper**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VStepper component. See more info and examples .

Parameters

- **alt_labels** – Places the labels beneath the step
- **color** – See description .
- **dark** – See description .
- **elevation** – See description .
- **flat** – Removes the stepper's elevation.

- **height** – Sets the height for the component.
- **light** – Applies the light theme variant to the component.
- **max_height** – Sets the maximum height for the component.
- **max_width** – Sets the maximum width for the component.
- **min_height** – Sets the minimum height for the component.
- **min_width** – Sets the minimum width for the component.
- **non_linear** – Allow user to jump to any step
- **outlined** – Removes elevation (box-shadow) and adds a *thin* border.
- **rounded** – See description .
- **shaped** – Applies a large border radius on the top left and bottom right of the card.
- **tag** – Specify a custom tag used on the root element.
- **tile** – Removes the component's **border-radius**.
- **value** – The designated model value for the component.
- **vertical** – Display steps vertically
- **width** – Sets the width for the component.

Events

Parameters **change** – Emitted when step is changed by user interaction

class trame.html.vuetify.**VStepperContent**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VStepperContent component. See more info and examples .

Parameters **step** – Sets step to associate the content to

class trame.html.vuetify.**VStepperStep**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VStepperStep component. See more info and examples .

Parameters

- **color** – See description .
- **complete** – Marks step as complete
- **complete_icon** – Icon to display when step is marked as completed
- **edit_icon** – Icon to display when step is editable
- **editable** – Marks step as editable
- **error_icon** – Icon to display when step has an error
- **rules** – Accepts a mixed array of types *function*, *boolean* and *string*. Functions pass an input value as an argument and must return either *true* / *false* or a *string* containing an error message. The input field will enter an error state if a function returns (or any value in the array contains) *false* or is a *string*
- **step** – Content to display inside step circle

class trame.html.vuetify.**VStepperHeader**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VStepperHeader component. See more info and examples .

Parameters **tag** – Specify a custom tag used on the root element.

class trame.html.vuetify.**VStepperItems**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VStepperItems component. See more info and examples .

Parameters **tag** – Specify a custom tag used on the root element.

class trame.html.vuetify.**VSubheader**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VSubheader component. See more info and examples .

Parameters

- **dark** – See description .
- **inset** – Adds indentation (72px)
- **light** – Applies the light theme variant to the component.

class trame.html.vuetify.**VSwitch**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VSwitch component. See more info and examples .

Parameters

- **append_icon** – Appends an icon to the component, uses the same syntax as *v-icon*
- **background_color** – Changes the background-color of the input
- **color** – See description .
- **dark** – See description .
- **dense** – Reduces the input height
- **disabled** – Disable the input
- **error** – Puts the input in a manual error state
- **error_count** – The total number of errors that should display at once
- **error_messages** – Puts the input in an error state and passes through custom error messages. Will be combined with any validations that occur from the **rules** prop. This field will not trigger validation
- **false_value** – Sets value for falsy state
- **flat** – Display component without elevation. Default elevation for thumb is 4dp, *flat* resets it
- **hide_details** – Hides hint and validation errors. When set to *auto* messages will be rendered only if there's a message (hint, error message, counter value etc) to display
- **hide_spin_buttons** – Hides spin buttons on the input when type is set to *number*.
- **hint** – Hint text
- **id** – Sets the DOM id on the component
- **input_value** – The **v-model** bound value

- **inset** – Enlarge the *v-switch* track to encompass the thumb
- **label** – Sets input label
- **light** – Applies the light theme variant to the component.
- **loading** – Displays circular progress bar. Can either be a String which specifies which color is applied to the progress bar (any material color or theme color - primary, secondary, success, info, warning, error) or a Boolean which uses the component color (set by color prop - if it's supported by the component) or the primary color
- **messages** – Displays a list of messages or message if using a string
- **multiple** – Changes expected model to an array
- **persistent_hint** – Forces hint to always be visible
- **prepend_icon** – Prepends an icon to the component, uses the same syntax as *v-icon*
- **readonly** – Puts input in readonly state
- **ripple** – See description .
- **rules** – Accepts a mixed array of types *function*, *boolean* and *string*. Functions pass an input value as an argument and must return either *true / false* or a *string* containing an error message. The input field will enter an error state if a function returns (or any value in the array contains) *false* or is a *string*
- **success** – Puts the input in a manual success state
- **success_messages** – Puts the input in a success state and passes through custom success messages.
- **true_value** – Sets value for truthy state
- **validate_on_blur** – Delays validation until blur event
- **value** – The input's value
- **value_comparator** – Apply a custom value comparator function

Events

Parameters

- **change** – Emitted when the input is changed by user interaction
- **click_append** – Emitted when appended icon is clicked
- **click_prepend** – Emitted when prepended icon is clicked
- **update_error** – The *error.sync* event

class trame.html.vuetify.**VSystemBar**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VSystemBar component. See more info and examples .

Parameters

- **absolute** – Applies **position: absolute** to the component.
- **app** – See description .
- **color** – See description .
- **dark** – See description .
- **fixed** – Applies **position: fixed** to the component.

- **height** – Sets the height for the component.
- **light** – Applies the light theme variant to the component.
- **lights_out** – Reduces the system bar opacity.
- **window** – Increases the system bar height to 32px (24px default).

class trame.html.vuetify.**VTabs**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VTabs component. See more info and examples .

Parameters

- **active_class** – The **active-class** applied to children when they are activated.
- **align_with_title** – Make *v-tabs* lined up with the toolbar title
- **background_color** – Changes the background color of the component.
- **center_active** – Forces the selected tab to be centered
- **centered** – Centers the tabs
- **color** – See description .
- **dark** – See description .
- **fixed_tabs** – *v-tabs-item* min-width 160px, max-width 360px
- **grow** – Force *v-tab*'s to take up all available space
- **height** – Sets the height of the tabs bar
- **hide_slider** – Hide's the generated *v-tabs-slider*
- **icons_and_text** – Will stack icon and text vertically
- **light** – Applies the light theme variant to the component.
- **mobile_breakpoint** – Sets the designated mobile breakpoint for the component.
- **next_icon** – Right pagination icon
- **optional** – Does not require an active item. Useful when using *v-tab* as a *router-link*
- **prev_icon** – Left pagination icon
- **right** – Aligns tabs to the right
- **show_arrows** – Show pagination arrows if the tab items overflow their container. For mobile devices, arrows will only display when using this prop.
- **slider_color** – Changes the background color of an auto-generated *v-tabs-slider*
- **slider_size** – Changes the size of the slider, **height** for horizontal, **width** for vertical.
- **value** – The designated model value for the component.
- **vertical** – Stacks tabs on top of each other vertically.

Events

Parameters **change** – Emitted when tab is changed by user interaction. Returns a string if **href** attribute is set and number if it is not.

class trame.html.vuetify.**VTab**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VTab component. See more info and examples .

Parameters

- **active_class** – See description .
- **append** – See description .
- **dark** – See description .
- **disabled** – Removes the ability to click or target the component.
- **exact** – See description .
- **exact_active_class** – See description .
- **exact_path** – See description .
- **href** – Designates the component as anchor and applies the **href** attribute.
- **light** – Applies the light theme variant to the component.
- **link** – Designates that the component is a link. This is automatic when using the **href** or **to** prop.
- **nuxt** – See description .
- **replace** – See description .
- **ripple** – See description .
- **tag** – Specify a custom tag used on the root element.
- **target** – Designates the target attribute. This should only be applied when using the **href** prop.
- **to** – See description .

Events

Parameters

- **change** – Emitted when tab becomes active
- **keydown** – Emitted when **enter** key is pressed

class trame.html.vuetify.**VTabItem**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VTabItem component. See more info and examples .

Parameters

- **active_class** – See description .
- **disabled** – Removes the ability to click or target the component.
- **eager** – Will force the components content to render on mounted. This is useful if you have content that will not be rendered in the DOM that you want crawled for SEO.
- **id** – Sets the DOM id on the component
- **reverse_transition** – Sets the reverse transition
- **transition** – See description .
- **value** – Sets the value of the tab. If not provided, the index will be used.

class trame.html.vuetify.**VTabsItems**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VTabsItems component. See more info and examples .

Parameters

- **active_class** – The **active-class** applied to children when they are activated.
- **continuous** – If *true*, window will "wrap around" from the last item to the first, and from the first item to the last
- **dark** – See description .
- **light** – Applies the light theme variant to the component.
- **mandatory** – Forces a value to always be selected (if available).
- **max** – Sets a maximum number of selections that can be made.
- **multiple** – Allow multiple selections. The **value** prop must be an _array_.
- **next_icon** – Icon used for the "next" button if *show-arrows* is *true*
- **prev_icon** – Icon used for the "prev" button if *show-arrows* is *true*
- **reverse** – Reverse the normal transition direction.
- **show_arrows** – Display the "next" and "prev" buttons
- **show_arrows_on_hover** – Display the "next" and "prev" buttons on hover. *show-arrows* MUST ALSO be set.
- **tag** – Specify a custom tag used on the root element.
- **touch** – Provide a custom **left** and **right** function when swiped left or right.
- **touchless** – Disable touch support.
- **value** – The designated model value for the component.
- **value_comparator** – Apply a custom value comparator function
- **vertical** – Uses a vertical transition when changing windows.

Events

Parameters **change** – Emitted when user swipes between tabs.

class trame.html.vuetify.**VTabsSlider**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VTabsSlider component. See more info and examples .

Parameters **color** – See description .

class trame.html.vuetify.**VTextarea**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VTextarea component. See more info and examples .

Parameters

- **append_icon** – Appends an icon to the component, uses the same syntax as *v-icon*
- **append_outer_icon** – Appends an icon to the outside the component's input, uses same syntax as *v-icon*
- **auto_grow** – Automatically grow the textarea depending on amount of text

- **autofocus** – Enables autofocus
- **background_color** – Changes the background-color of the input
- **clear_icon** – Applied when using **clearable** and the input is dirty
- **clearable** – Add input clear functionality, default icon is Material Design Icons **mdi-clear**
- **color** – See description .
- **counter** – Creates counter for input length; if no number is specified, it defaults to 25. Does not apply any validation.
- **counter_value** –
- **dark** – See description .
- **dense** – Reduces the input height
- **disabled** – Disable the input
- **error** – Puts the input in a manual error state
- **error_count** – The total number of errors that should display at once
- **error_messages** – Puts the input in an error state and passes through custom error messages. Will be combined with any validations that occur from the **rules** prop. This field will not trigger validation
- **filled** – Applies the alternate filled input style
- **flat** – Removes elevation (shadow) added to element when using the **solo** or **solo-inverted** props
- **full_width** – Designates input type as full-width
- **height** – Sets the height of the input
- **hide_details** – Hides hint and validation errors. When set to *auto* messages will be rendered only if there's a message (hint, error message, counter value etc) to display
- **hide_spin_buttons** – Hides spin buttons on the input when type is set to *number*.
- **hint** – Hint text
- **id** – Sets the DOM id on the component
- **label** – Sets input label
- **light** – Applies the light theme variant to the component.
- **loader_height** – Specifies the height of the loader
- **loading** – Displays linear progress bar. Can either be a String which specifies which color is applied to the progress bar (any material color or theme color - **primary**, **secondary**, **success**, **info**, **warning**, **error**) or a Boolean which uses the component **color** (set by color prop - if it's supported by the component) or the primary color
- **messages** – Displays a list of messages or message if using a string
- **no_resize** – Remove resize handle
- **outlined** – Applies the outlined style to the input
- **persistent_hint** – Forces hint to always be visible
- **persistent_placeholder** – Forces placeholder to always be visible
- **placeholder** – Sets the input's placeholder text

- **prefix** – Displays prefix text
- **prepend_icon** – Prepends an icon to the component, uses the same syntax as *v-icon*
- **prepend_inner_icon** – Prepends an icon inside the component's input, uses the same syntax as *v-icon*
- **readonly** – Puts input in readonly state
- **reverse** – Reverses the input orientation
- **rounded** – Adds a border radius to the input
- **row_height** – Height value for each row. Requires the use of the **auto-grow** prop.
- **rows** – Default row count
- **rules** – Accepts a mixed array of types *function*, *boolean* and *string*. Functions pass an input value as an argument and must return either *true* / *false* or a *string* containing an error message. The input field will enter an error state if a function returns (or any value in the array contains) *false* or is a *string*
- **shaped** – Round if *outlined* and increase *border-radius* if *filled*. Must be used with either *outlined* or *filled*
- **single_line** – Label does not move on focus/dirty
- **solo** – Changes the style of the input
- **solo_inverted** – Reduces element opacity until focused
- **success** – Puts the input in a manual success state
- **success_messages** – Puts the input in a success state and passes through custom success messages.
- **suffix** – Displays suffix text
- **type** – Sets input type
- **validate_on_blur** – Delays validation until blur event
- **value** – The input's value

Events

Parameters

- **blur** – Emitted when the input is blurred
- **change** – Emitted when the input is changed by user interaction
- **click_append** – Emitted when appended icon is clicked
- **click_append_outer** – Emitted when appended outer icon is clicked
- **click_clear** – Emitted when clearable icon clicked
- **click_prepend** – Emitted when prepended icon is clicked
- **click_prepend_inner** – Emitted when prepended inner icon is clicked
- **focus** – Emitted when component is focused
- **input** – The updated bound model
- **keydown** – Emitted when **any** key is pressed
- **update_error** – The *error.sync* event

class trame.html.vuetify.**VTextField**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VTextField component. See more info and examples .

Parameters

- **append_icon** – Appends an icon to the component, uses the same syntax as *v-icon*
- **append_outer_icon** – Appends an icon to the outside the component's input, uses same syntax as *v-icon*
- **autofocus** – Enables autofocus
- **background_color** – Changes the background-color of the input
- **clear_icon** – Applied when using **clearable** and the input is dirty
- **clearable** – Add input clear functionality, default icon is Material Design Icons **mdi-clear**
- **color** – See description .
- **counter** – Creates counter for input length; if no number is specified, it defaults to 25. Does not apply any validation.
- **counter_value** –
- **dark** – See description .
- **dense** – Reduces the input height
- **disabled** – Disable the input
- **error** – Puts the input in a manual error state
- **error_count** – The total number of errors that should display at once
- **error_messages** – Puts the input in an error state and passes through custom error messages. Will be combined with any validations that occur from the **rules** prop. This field will not trigger validation
- **filled** – Applies the alternate filled input style
- **flat** – Removes elevation (shadow) added to element when using the **solo** or **solo-inverted** props
- **full_width** – Designates input type as full-width
- **height** – Sets the height of the input
- **hide_details** – Hides hint and validation errors. When set to *auto* messages will be rendered only if there's a message (hint, error message, counter value etc) to display
- **hide_spin_buttons** – Hides spin buttons on the input when type is set to *number*.
- **hint** – Hint text
- **id** – Sets the DOM id on the component
- **label** – Sets input label
- **light** – Applies the light theme variant to the component.
- **loader_height** – Specifies the height of the loader
- **loading** – Displays linear progress bar. Can either be a String which specifies which color is applied to the progress bar (any material color or theme color - **primary**, **secondary**,

success, **info**, **warning**, **error**) or a Boolean which uses the component **color** (set by color prop - if it's supported by the component) or the primary color

- **messages** – Displays a list of messages or message if using a string
- **outlined** – Applies the outlined style to the input
- **persistent_hint** – Forces hint to always be visible
- **persistent_placeholder** – Forces placeholder to always be visible
- **placeholder** – Sets the input's placeholder text
- **prefix** – Displays prefix text
- **prepend_icon** – Prepends an icon to the component, uses the same syntax as *v-icon*
- **prepend_inner_icon** – Prepends an icon inside the component's input, uses the same syntax as *v-icon*
- **readonly** – Puts input in readonly state
- **reverse** – Reverses the input orientation
- **rounded** – Adds a border radius to the input
- **rules** – Accepts a mixed array of types *function*, *boolean* and *string*. Functions pass an input value as an argument and must return either *true* / *false* or a *string* containing an error message. The input field will enter an error state if a function returns (or any value in the array contains) *false* or is a *string*
- **shaped** – Round if *outlined* and increase *border-radius* if *filled*. Must be used with either *outlined* or *filled*
- **single_line** – Label does not move on focus/dirty
- **solo** – Changes the style of the input
- **solo_inverted** – Reduces element opacity until focused
- **success** – Puts the input in a manual success state
- **success_messages** – Puts the input in a success state and passes through custom success messages.
- **suffix** – Displays suffix text
- **type** – Sets input type
- **validate_on_blur** – Delays validation until blur event
- **value** – The input's value

Events

Parameters

- **blur** – Emitted when the input is blurred
- **change** – Emitted when the input is changed by user interaction
- **click_append** – Emitted when appended icon is clicked
- **click_append_outer** – Emitted when appended outer icon is clicked
- **click_clear** – Emitted when clearable icon clicked
- **click_prepend** – Emitted when prepended icon is clicked

- **click_prepend_inner** – Emitted when prepended inner icon is clicked
- **focus** – Emitted when component is focused
- **input** – The updated bound model
- **keydown** – Emitted when **any** key is pressed
- **update_error** – The *error.sync* event

class trame.html.vuetify.**VThemeProvider**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VThemeProvider component. See more info and examples .

Parameters

- **dark** – See description .
- **light** – Applies the light theme variant to the component.
- **root** – Use the current value of *$vuetify.theme.dark* as opposed to the provided one.

class trame.html.vuetify.**VTimeline**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VTimeline component. See more info and examples .

Parameters

- **align_top** – Align caret and dot of timeline items to the top
- **dark** – See description .
- **dense** – Hide opposite slot content, and position all items to one side of timeline
- **light** – Applies the light theme variant to the component.
- **reverse** – Reverse direction of timeline items

class trame.html.vuetify.**VTimelineItem**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VTimelineItem component. See more info and examples .

Parameters

- **color** – See description .
- **dark** – See description .
- **fill_dot** – Remove padding from dot container
- **hide_dot** – Hide display of timeline dot
- **icon** – Specify icon for dot container
- **icon_color** – See description .
- **large** – Large size dot
- **left** – Explicitly set the item to a left orientation
- **light** – Applies the light theme variant to the component.
- **right** – Explicitly set the item to a right orientation
- **small** – Small size dot

class trame.html.vuetify.**VTimePicker**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VTimePicker component. See more info and examples .

Parameters

- **allowed_hours** – Restricts which hours can be selected
- **allowed_minutes** – Restricts which minutes can be selected
- **allowed_seconds** – Restricts which seconds can be selected
- **ampm_in_title** – Place AM/PM switch in title, not near the clock.
- **color** – See description .
- **dark** – See description .
- **disabled** – disables picker
- **elevation** – See description .
- **flat** – Removes elevation
- **format** – Defines the format of a time displayed in picker. Available options are *ampm* and *24hr*.
- **full_width** – Forces 100% width
- **header_color** – Defines the header color. If not specified it will use the color defined by <code>color</code> prop or the default picker color
- **landscape** – Orients picker horizontal
- **light** – Applies the light theme variant to the component.
- **max** – Maximum allowed time
- **min** – Minimum allowed time
- **no_title** – Hide the picker title
- **readonly** – Puts picker in readonly state
- **scrollable** – Allows changing hour/minute with mouse scroll
- **use_seconds** – Toggles the use of seconds in picker
- **value** – Time picker model (ISO 8601 format, 24hr hh:mm)
- **width** – Width of the picker

Events

Parameters

- **change** – Emitted when the time selection is done (when user changes the minute for HH:MM picker and the second for HH:MM:SS picker
- **click_hour** – Emitted when user selects the hour
- **click_minute** – Emitted when user selects the minute
- **click_second** – Emitted when user selects the second
- **input** – The updated bound model
- **update_period** – Emitted when user clicks the AM/PM button

class trame.html.vuetify.**VToolbar**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VToolbar component. See more info and examples .

Parameters

- **absolute** – Applies position: absolute to the component.
- **bottom** – Aligns the component towards the bottom.
- **collapse** – Puts the toolbar into a collapsed state reducing its maximum width.
- **color** – See description .
- **dark** – See description .
- **dense** – Reduces the height of the toolbar content to 48px (96px when using the **prominent** prop).
- **elevation** – See description .
- **extended** – Use this prop to increase the height of the toolbar _without_ using the *extension* slot for adding content. May be used in conjunction with the **extension-height** prop, and any of the other props that affect the height of the toolbar, e.g. **prominent**, **dense**, etc., **WITH THE EXCEPTION** of **height**.
- **extension_height** – Specify an explicit height for the *extension* slot.
- **flat** – Removes the toolbar's box-shadow.
- **floating** – Applies **display: inline-flex** to the component.
- **height** – Designates a specific height for the toolbar. Overrides the heights imposed by other props, e.g. **prominent**, **dense**, **extended**, etc.
- **light** – Applies the light theme variant to the component.
- **max_height** – Sets the maximum height for the component.
- **max_width** – Sets the maximum width for the component.
- **min_height** – Sets the minimum height for the component.
- **min_width** – Sets the minimum width for the component.
- **outlined** – Removes elevation (box-shadow) and adds a *thin* border.
- **prominent** – Increases the height of the toolbar content to 128px.
- **rounded** – See description .
- **shaped** – Applies a large border radius on the top left and bottom right of the card.
- **short** – Reduce the height of the toolbar content to 56px (112px when using the **prominent** prop).
- **src** – See description .
- **tag** – Specify a custom tag used on the root element.
- **tile** – Removes the component's **border-radius**.
- **width** – Sets the width for the component.

class trame.html.vuetify.**VToolbarItems**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VToolbarItems component. See more info and examples .

Parameters **tag** – Specify a custom tag used on the root element.

class trame.html.vuetify.**VToolbarTitle**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VToolbarTitle component. See more info and examples .

Parameters **tag** – Specify a custom tag used on the root element.

class trame.html.vuetify.**VTooltip**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VTooltip component. See more info and examples .

Parameters

- **absolute** – Applies **position: absolute** to the component.
- **activator** – Designate a custom activator when the *activator* slot is not used. String can be any valid querySelector and Object can be any valid Node.
- **allow_overflow** – Removes overflow re-positioning for the content
- **attach** – Specifies which DOM element that this component should detach to. String can be any valid querySelector and Object can be any valid Node. This will attach to the root *v-app* component by default.
- **bottom** – Aligns the component towards the bottom.
- **close_delay** – Delay (in ms) after which menu closes (when open-on-hover prop is set to true)
- **color** – See description .
- **content_class** – Applies a custom class to the detached element. This is useful because the content is moved to the beginning of the *v-app* component (unless the **attach** prop is provided) and is not targetable by classes passed directly on the component.
- **disabled** – Disables the tooltip
- **eager** – Will force the components content to render on mounted. This is useful if you have content that will not be rendered in the DOM that you want crawled for SEO.
- **internal_activator** – Designates whether to use an internal activator
- **left** – Aligns the component towards the left.
- **max_width** – Sets the maximum width for the content
- **min_width** – Sets the minimum width for the content
- **nudge_bottom** – Nudge the content to the bottom
- **nudge_left** – Nudge the content to the left
- **nudge_right** – Nudge the content to the right
- **nudge_top** – Nudge the content to the top
- **nudge_width** – Nudge the content width
- **offset_overflow** – Causes the component to flip to the opposite side when repositioned due to overflow
- **open_delay** – Delay (in ms) after which tooltip opens (when *open-on-hover* prop is set to **true**)
- **open_on_click** – Designates whether the tooltip should open on activator click

- **open_on_focus** –
- **open_on_hover** – Designates whether the tooltip should open on activator hover
- **position_x** – Used to position the content when not using an activator slot
- **position_y** – Used to position the content when not using an activator slot
- **right** – Aligns the component towards the right.
- **tag** – Specifies a custom tag for the activator wrapper
- **top** – Aligns the content towards the top.
- **transition** – See description .
- **value** – Controls whether the component is visible or hidden.
- **z_index** – The z-index used for the component

class trame.html.vuetify.**VTreeview**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VTreeview component. See more info and examples .

Parameters

- **activatable** – Allows user to mark a node as active by clicking on it
- **active** – Syncable prop that allows one to control which nodes are active. The array consists of the *item-key* of each active item.
- **active_class** – The class applied to the node when active
- **color** – Sets the color of the active node
- **dark** – See description .
- **dense** – Decreases the height of the items
- **disable_per_node** – Prevents disabling children nodes
- **disabled** – Disables selection for all nodes
- **expand_icon** – Icon used to indicate that a node can be expanded
- **filter** – Custom item filtering function. By default it will use case-insensitive search in item's label.
- **hoverable** – Applies a hover class when mousing over nodes
- **indeterminate_icon** – Icon used when node is in an indeterminate state. Only visible when *selectable* is *true*.
- **item_children** – Property on supplied *items* that contains its children
- **item_disabled** – Property on supplied *items* that contains the disabled state of the item
- **item_key** – Property on supplied *items* used to keep track of node state. The value of this property has to be unique among all items.
- **item_text** – Property on supplied *items* that contains its label text
- **items** – An array of items used to build the treeview
- **light** – Applies the light theme variant to the component.

- **load_children** – A function used when dynamically loading children. If this prop is set, then the supplied function will be run if expanding an item that has a *item-children* property that is an empty array. Supports returning a Promise.
- **loading_icon** – Icon used when node is in a loading state
- **multiple_active** – When *true*, allows user to have multiple active nodes at the same time
- **off_icon** – Icon used when node is not selected. Only visible when *selectable* is *true*.
- **on_icon** – Icon used when leaf node is selected or when a branch node is fully selected. Only visible when *selectable* is *true*.
- **open** – Syncable prop that allows one to control which nodes are open. The array consists of the *item-key* of each open item.
- **open_all** – When *true* will cause all branch nodes to be opened when component is mounted
- **open_on_click** – When *true* will cause nodes to be opened by clicking anywhere on it, instead of only opening by clicking on expand icon. When using this prop with *activatable* you will be unable to mark nodes with children as active.
- **return_object** – When *true* will make *v-model*, *active.sync* and *open.sync* return the complete object instead of just the key
- **rounded** – Provides an alternative active style for *v-treeview* node. Only visible when *activatable* is *true* and should not be used in conjunction with the *shaped* prop.
- **search** – The search model for filtering results
- **selectable** – Will render a checkbox next to each node allowing them to be selected
- **selected_color** – The color of the selection checkbox
- **selection_type** – Controls how the treeview selects nodes. There are two modes available: 'leaf' and 'independent'
- **shaped** – Provides an alternative active style for *v-treeview* node. Only visible when *activatable* is *true* and should not be used in conjunction with the *rounded* prop.
- **transition** – Applies a transition when nodes are opened and closed
- **value** – Allows one to control which nodes are selected. The array consists of the *item-key* of each selected item. Is used with *@input* event to allow for *v-model* binding.

Events

Parameters

- **input** – Emits the array of selected items when this value changes
- **update_active** – Emits the array of active items when this value changes
- **update_open** – Emits the array of open items when this value changes

class trame.html.vuetify.**VVirtualScroll**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VVirtualScroll component. See more info and examples .

Parameters

- **bench** – The number of items **outside** the user view that are rendered (even if they are **not** viewable); to help prevent empty white space when scrolling *fast*.
- **height** – Height of the component as a css value

- **item_height** – Height in pixels of the items to display
- **items** – The array of items to display
- **max_height** – Sets the maximum height for the component.
- **max_width** – Sets the maximum width for the component.
- **min_height** – Sets the minimum height for the component.
- **min_width** – Sets the minimum width for the component.
- **width** – Sets the width for the component.

class trame.html.vuetify.**VWindow**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VWindow component. See more info and examples .

Parameters

- **active_class** – The **active-class** applied to children when they are activated.
- **continuous** – If *true*, window will "wrap around" from the last item to the first, and from the first item to the last
- **dark** – See description .
- **light** – Applies the light theme variant to the component.
- **next_icon** – Icon used for the "next" button if *show-arrows* is *true*
- **prev_icon** – Icon used for the "prev" button if *show-arrows* is *true*
- **reverse** – Reverse the normal transition direction.
- **show_arrows** – Display the "next" and "prev" buttons
- **show_arrows_on_hover** – Display the "next" and "prev" buttons on hover. *show-arrows* MUST ALSO be set.
- **tag** – Specify a custom tag used on the root element.
- **touch** – Provide a custom **left** and **right** function when swiped left or right.
- **touchless** – Disable touch support.
- **value** – The designated model value for the component.
- **value_comparator** – Apply a custom value comparator function
- **vertical** – Uses a vertical transition when changing windows.

Events

Parameters **change** – Emitted when the component value is changed by user interaction

class trame.html.vuetify.**VWindowItem**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

Vuetify's VWindowItem component. See more info and examples .

Parameters

- **active_class** – See description .
- **disabled** – Prevents the item from becoming active when using the "next" and "prev" buttons or the *toggle* method

- **eager** – Will force the components content to render on mounted. This is useful if you have content that will not be rendered in the DOM that you want crawled for SEO.
- **reverse_transition** – Sets the reverse transition
- **transition** – See description .
- **value** – The value used when the component is selected in a group. If not provided, the index will be used.

class trame.html.vuetify.**VCarouselTransition**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VCarouselTransition component. See more info and examples .

Parameters

- **group** – See description .
- **hide_on_leave** – Hides the leaving element (no exit animation)
- **leave_absolute** – See description .
- **mode** – See description .
- **origin** – See description .

class trame.html.vuetify.**VCarouselReverseTransition**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VCarouselReverseTransition component. See more info and examples .

Parameters

- **group** – See description .
- **hide_on_leave** – Hides the leaving element (no exit animation)
- **leave_absolute** – See description .
- **mode** – See description .
- **origin** – See description .

class trame.html.vuetify.**VTabTransition**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VTabTransition component. See more info and examples .

Parameters

- **group** – See description .
- **hide_on_leave** – Hides the leaving element (no exit animation)
- **leave_absolute** – See description .
- **mode** – See description .
- **origin** – See description .

class trame.html.vuetify.**VTabReverseTransition**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VTabReverseTransition component. See more info and examples .

Parameters

- **group** – See description .

- **hide_on_leave** – Hides the leaving element (no exit animation)
- **leave_absolute** – See description .
- **mode** – See description .
- **origin** – See description .

class trame.html.vuetify.**VMenuTransition**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VMenuTransition component. See more info and examples .

Parameters

- **group** – See description .
- **hide_on_leave** – Hides the leaving element (no exit animation)
- **leave_absolute** – See description .
- **mode** – See description .
- **origin** – See description .

class trame.html.vuetify.**VFabTransition**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VFabTransition component. See more info and examples .

Parameters

- **group** – See description .
- **hide_on_leave** – Hides the leaving element (no exit animation)
- **leave_absolute** – See description .
- **mode** – See description .
- **origin** – See description .

class trame.html.vuetify.**VDialogTransition**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VDialogTransition component. See more info and examples .

Parameters

- **group** – See description .
- **hide_on_leave** – Hides the leaving element (no exit animation)
- **leave_absolute** – See description .
- **mode** – See description .
- **origin** – See description .

class trame.html.vuetify.**VDialogBottomTransition**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VDialogBottomTransition component. See more info and examples .

Parameters

- **group** – See description .
- **hide_on_leave** – Hides the leaving element (no exit animation)

- **leave_absolute** – See description .
- **mode** – See description .
- **origin** – See description .

class trame.html.vuetify.**VDialogTopTransition**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VDialogTopTransition component. See more info and examples .

Parameters

- **group** – See description .
- **hide_on_leave** – Hides the leaving element (no exit animation)
- **leave_absolute** – See description .
- **mode** – See description .
- **origin** – See description .

class trame.html.vuetify.**VFadeTransition**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VFadeTransition component. See more info and examples .

Parameters

- **group** – See description .
- **hide_on_leave** – Hides the leaving element (no exit animation)
- **leave_absolute** – See description .
- **mode** – See description .
- **origin** – See description .

class trame.html.vuetify.**VScaleTransition**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VScaleTransition component. See more info and examples .

Parameters

- **group** – See description .
- **hide_on_leave** – Hides the leaving element (no exit animation)
- **leave_absolute** – See description .
- **mode** – See description .
- **origin** – See description .

class trame.html.vuetify.**VScrollXTransition**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VScrollXTransition component. See more info and examples .

Parameters

- **group** – See description .
- **hide_on_leave** – Hides the leaving element (no exit animation)
- **leave_absolute** – See description .

- **mode** – See description .
- **origin** – See description .

class trame.html.vuetify.**VScrollXReverseTransition**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VScrollXReverseTransition component. See more info and examples .

Parameters

- **group** – See description .
- **hide_on_leave** – Hides the leaving element (no exit animation)
- **leave_absolute** – See description .
- **mode** – See description .
- **origin** – See description .

class trame.html.vuetify.**VScrollYTransition**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VScrollYTransition component. See more info and examples .

Parameters

- **group** – See description .
- **hide_on_leave** – Hides the leaving element (no exit animation)
- **leave_absolute** – See description .
- **mode** – See description .
- **origin** – See description .

class trame.html.vuetify.**VScrollYReverseTransition**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VScrollYReverseTransition component. See more info and examples .

Parameters

- **group** – See description .
- **hide_on_leave** – Hides the leaving element (no exit animation)
- **leave_absolute** – See description .
- **mode** – See description .
- **origin** – See description .

class trame.html.vuetify.**VSlideXTransition**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VSlideXTransition component. See more info and examples .

Parameters

- **group** – See description .
- **hide_on_leave** – Hides the leaving element (no exit animation)
- **leave_absolute** – See description .
- **mode** – See description .

- **origin** – See description .

class trame.html.vuetify.**VSlideXReverseTransition**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VSlideXReverseTransition component. See more info and examples .

Parameters

- **group** – See description .
- **hide_on_leave** – Hides the leaving element (no exit animation)
- **leave_absolute** – See description .
- **mode** – See description .
- **origin** – See description .

class trame.html.vuetify.**VSlideYTransition**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VSlideYTransition component. See more info and examples .

Parameters

- **group** – See description .
- **hide_on_leave** – Hides the leaving element (no exit animation)
- **leave_absolute** – See description .
- **mode** – See description .
- **origin** – See description .

class trame.html.vuetify.**VSlideYReverseTransition**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VSlideYReverseTransition component. See more info and examples .

Parameters

- **group** – See description .
- **hide_on_leave** – Hides the leaving element (no exit animation)
- **leave_absolute** – See description .
- **mode** – See description .
- **origin** – See description .

class trame.html.vuetify.**VExpandTransition**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VExpandTransition component. See more info and examples .

Parameters **mode** – See description .

class trame.html.vuetify.**VExpandXTransition**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Vuetify's VExpandXTransition component. See more info and examples .

Parameters **mode** – See description .

2.8 trame.html.widgets

These auto-generated docs only show this module's objects, which rely on keyword arguments (***kwargs*) for configuration. You can find more information in the modules section .

class trame.html.widgets.**FloatCard**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

A which floats above the application and can be moved freely from a handle

Parameters

- **handle_color** –
- **handle_position** –
- **handle_size** –
- **location** –

Vuetify VCard attributes

Parameters

- **color** –
- **dark** –
- **flat** –
- **height** –
- **elevation** –
- **hover** –
- **img** –
- **light** –
- **loader_height** –
- **loading** –
- **max_height** –
- **max_width** –
- **min_height** –
- **min_width** –
- **outlined** –
- **raised** –
- **rounded** –
- **shaped** –
- **tile** –
- **width** –

class trame.html.widgets.**ListBrowser**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

A component that list items that be used for browsing directories or simple item picking

Parameters

- **list** – List stored in state
- **filter** – Function to filter list
- **path_icon** –
- **path_selected_icon** –
- **filter_icon** –
- **path** –

class trame.html.widgets.**GitTree**(*children=None*, ***kwargs*)

Bases: *trame.html.AbstractElement*

A component to present a Tree the same way Git does it (Like a subway map)

Parameters

- **sources** – All of the elements of the tree
- **actives** – Any active elements of the tree

Vuetify styling attributes

Parameters

- **active_background** –
- **delta_x** –
- **delta_y** –
- **font_size** –
- **margin** –
- **multiselect** –
- **offset** –
- **palette** –
- **radius** –
- **root_id** –
- **stroke** –
- **width** –
- **active_circle_stroke_color** –
- **not_visible_circle_fill_color** –
- **text_color** –
- **text_weight** –
- **action_map** –
- **action_size** –

Events

Parameters

- **actives_change** –

- **visibility_change** –
- **action** –

2.9 __init__.py

trame.html.**js2py_key**(*key*)

trame.html.**build_attr_names**(*name_prefix*, *key_names*, *kwargs*)
Used to generate a list of attr_names with a common name_prefix.

class trame.html.**AbstractElement**(*_elem_name*, *children=None*, ***kwargs*)
Bases: object

A Vue component which can integrate with the rest of trame

See Vue docs for more info

Parameters

- **name** (*str*) – The name of the element, like 'div' for a element
- **children** (*str | list[trame.html.*] | trame.html.* | None*) – The children nested within this element
- **__properties** – Provide more attribute names that should be handle
- **__events** – Provide more event names that should be handle

Html attributes - See for more info

Parameters

- **id** – See for more info
- **classes** – Match the HTML *class* attribute. See for more info
- **style** – See for more info

Vue attributes - See for more info

Parameters

- **ref** – See for more info
- **v_model** – See for more info
- **v_if** – See for more info
- **v_show** – See for more info
- **v_for** – See for more info
- **v_on** – See for more info
- **v_bind** – See for more info
- **key** – See for more info

Events - See for more info

Parameters

- **click** – See for more info
- **mousedown** – See for more info

- **mouseup** – See for more info
- **mouseenter** – See for more info
- **mouseleave** – See for more info
- **contextmenu** – See for more info

ttsSensitive()
Calling this function on an element will make it fully recreate itself every time the layout update. Internally it is managed by adding a *key*= attribute which use a layout timestamp.

This is especially useful for component that manage other elements outside of themself like VSelect in Vuetify.

attrs(**names*)
Calling this function will process the provided attribute names and configure its internal so the macthing HTML string could easily be generated later on.

Parameters **names** (*str) – The names attribute to process

events(**names*)
Calling this function will process the provided event names and configure its internal so the macthing HTML string could easily be generated later on.

Parameters **names** (*str) – The names events to process

clear()
Remove all children

hide()
Hide element while keeping it in the DOM. (display: none)

add_child(*child*)
Add a component to this component's children

Parameters **child** (*str | AbstractElement*) – The component to add as a child

add_children(*children*)
Add components to this component's children. The provided children is expected to be a list.

Parameters **children** (*list*) – The list of components to add to the children

property children
Children components

set_text(*value*)
Replace children with a single text child element

Parameters **value** (*str*) – The text for the new text child element

property html
Return a string representation of the HTML component

class trame.html.**Element**(*_elem_name*, *children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Any html element you would like to use in trame

Parameters

- **_elem_name** (*str*) – The name of the element, like 'div' for a element
- **children** (*str | list[trame.html.*] | trame.html.* | None*) – The children nested within this element

class trame.html.**Div**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

The standard html content div element

Parameters **children** (*str | list[trame.html.*] | trame.html.* | None*) – The children nested within this element

class trame.html.**Span**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

The standard html content span element

Parameters **children** (*str | list[trame.html.*] | trame.html.* | None*) – The children nested within this element

class trame.html.**Form**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

The standard html form element

Parameters **children** (*str | list[trame.html.*] | trame.html.* | None*) – The children nested within this element

class trame.html.**Label**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

The standard html input label element

Parameters **children** (*str | list[trame.html.*] | trame.html.* | None*) – The children nested within this element

class trame.html.**Input**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

The standard html input (form input) element

Parameters **children** (*str | list[trame.html.*] | trame.html.* | None*) – The children nested within this element

class trame.html.**Template**(*children=None*, ***kwargs*)
Bases: *trame.html.AbstractElement*

The standard html content template element. This is mostly used by .

Parameters

- **children** (*str | list[trame.html.*] | trame.html.* | None*) – The children nested within this element
- **v_slot** – The slot this template corresponds to

slot_names = {'action', 'actions', 'activator', 'append', 'append-item', 'append-outer', 'appendIcon', 'badge', 'body', 'body.append', 'body.prepend', 'category', 'close', 'counter', 'day', 'day-body', 'day-header', 'day-label', 'day-label-header', 'day-month', 'default', 'divider', 'event', 'expanded-item', 'extension', 'foot', 'footer', 'footer.page-text', 'footer.prepend', 'group', 'group.header', 'group.summary', 'header', 'header.<name>', 'header.data-table-select', 'icon', 'img', 'input', 'interval', 'item', 'item.<name>', 'item.data-table-expand', 'item.data-table-select', 'label', 'loader', 'loading', 'message', 'next', 'no-data', 'no-results', 'opposite', 'page-text', 'placeholder', 'prepend', 'prepend-inner', 'prepend-item', 'prependIcon', 'prev', 'progress', 'selection', 'thumb-label', 'top'}

class trame.html.**StateChange**(*name*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Component to react when a state entry change so an event can be triggered

Parameters **name** (*str*) – Which part of the state to listen to

Events

Parameters **change** (*function*) – Function to run if state changes

class trame.html.**Triggers**(*ref*, *triggers={}*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Component to trigger JS actions from Python

Parameters

- **ref** (*str*) – Name for Vue reference to this object
- **triggers** (*dict[str, str]*) – Mapping from names of triggers to expressions or methods in JS which they will call

```
>>> triggers = trame.html.Triggers(ref="all_triggers", triggers={ "reset_camera": "
↪$refs.view.resetCamera()" })
```

add(*name*, *call*)
Add a trigger which can call JS from Python

Parameters

- **name** (*str*) – Reference for this JS method or expression trigger
- **call** (*str*) – JS method or expression to call when triggered

```
>>> triggers.add("created", "console.log('UI is created')")
>>> triggers.add("mounted", "console.log('UI is mounted')")
>>> triggers.add("beforeDestroy", "console.log('UI is going away')")
```

call(*name*, **args*)
Trigger JS code previously added to this object

Parameters

- **name** (*str*) – Reference for this JS method or expression trigger
- **args** – Parameters passed to JS method

```
>>> triggers.call("reset_camera")
```

class trame.html.**VTKLoading**(*message=''*, ***kwargs*)
Bases: *trame.html.AbstractElement*

Component to show the 3 spinning partial circles using the ParaView Red/Green/Yellow colors.

Parameters **message** (*str*) – Message to put below the spinning circles

CHAPTER

THREE

TRAME.LAYOUTS

class trame.layouts.**AbstractLayout**(*_root_elem*, *name*, *favicon=None*, *on_ready=None*)

Bases: object

property root

Top level Vue component. Useful for providing / injecting into children components. Setting makes old root child of new root.

property html

Compute corresponding layout String which represent the html part.

property state

Return App state as a dictionary or extend it when setting. This is a safe way to build the state incrementaly.

```
>>> layout.state = { "a": 1, "b": 2 }
>>> print(layout.state)
... {"a": 1, "b": 2}
>>> layout.state = { "c": 3, "d": 4 }
>>> print(layout.state)
... {"a": 1, "b": 2, "c": 3, "d": 4}
```

flush_content()

Push new content to client

start(*port=None*, *debug=None*, ***kwargs*)

Start the application server.

Parameters

- **port** – Which port to run the server on
- **debug** (*bool or None*) – Whether to enable debugging tools. Defaults to None, in which case it is set to True if the –dev flag was passed as a command line argument.
- **kwargs** – arguments to forward to run_server()

Some of the kwargs that may be forwarded to run_server() include:

- **exec_mode (str): “main” (default) or “task” for running in an** environment that already has an event loop, such as a Jupyter notebook.

The kwargs will also be forwarded to print_server_info(), so that the *server* kwarg may be used to indicate whether a new window should be opened.

start_thread(*port=None*, *print_server_info=False*, *on_server_listening=None*, ***kwargs*)

start_desktop_window(*on_msg=None*, ***kwargs*)

add_route(*name*, *path*, *template*)

with_route(*name*, *path*, *root*)

class trame.layouts.**FullScreenPage**(*name*, *favicon=None*, *on_ready=None*)

Bases: *trame.layouts.core.AbstractLayout*

A layout that takes the whole screen.

Parameters

- **name** (*str*) – Text for this page's browser tab (required)
- **favicon** (*str*) – Filename of image for this page's browser tab
- **on_ready** (*function*) – Function to run on startup

```
>>> FullScreenPage("Simple Page").start()
```

class trame.layouts.**SinglePage**(*name*, *favicon=None*, *on_ready=None*)

Bases: *trame.layouts.core.FullScreenPage*

A layout that takes the whole screen, adding a for a *toolbar*, a VMain as *content* and a VFooter as a *footer*.

Parameters **name** (*str*) – Text for this page's browser tab (required)

```
>>> layout = SinglePage("Page with header / app bar")
```

The toolbar starts with 2 children, a *logo* and a *title* which are accessible at the root of the layout object.

```
>>> layout.toolbar.children += ["More stuff to the toolbar"]
>>> layout.logo.children = [VIcon("mdi-menu")]
>>> layout.title.set_text("My Super App")
```

Then we have *content* and *footer*. Content is by default empty but the footer has the default trame information regarding its versions and feature feedback on when the server is busy with a spining progress.

You can quickly hide the footer by calling the following.

```
>>> layout.footer.hide()
```

class trame.layouts.**SinglePageWithDrawer**(*name*, *favicon=None*, *on_ready=None*, *show_drawer=True*, *width=200*, *show_drawer_name='drawerOpen'*)

Bases: *trame.layouts.core.SinglePage*

A layout that takes the whole screen, adding a for a toolbar, a content, a drawer, and a footer.

Parameters

- **name** (*str*) – Text for this page's browser tab (required)
- **show_drawer** (*bool*) – Whether the drawer is open. Default True
- **width** (*Number*) – How many pixels wide the drawer should be
- **show_drawer_name** (*str*) – The name referencing the drawer's state. Default "drawerOpen".

```
>>> SinglePageWithDrawer("Page with drawer").start()
```

trame.layouts.**update_layout**(*layout*)

Flush layout to the client

Parameters **layout** (*str | trame.layouts.**) – UI content for your application

```
>>> layout.title.set_text("Workload finished!")
>>> update_layout(layout)
```

CHAPTER

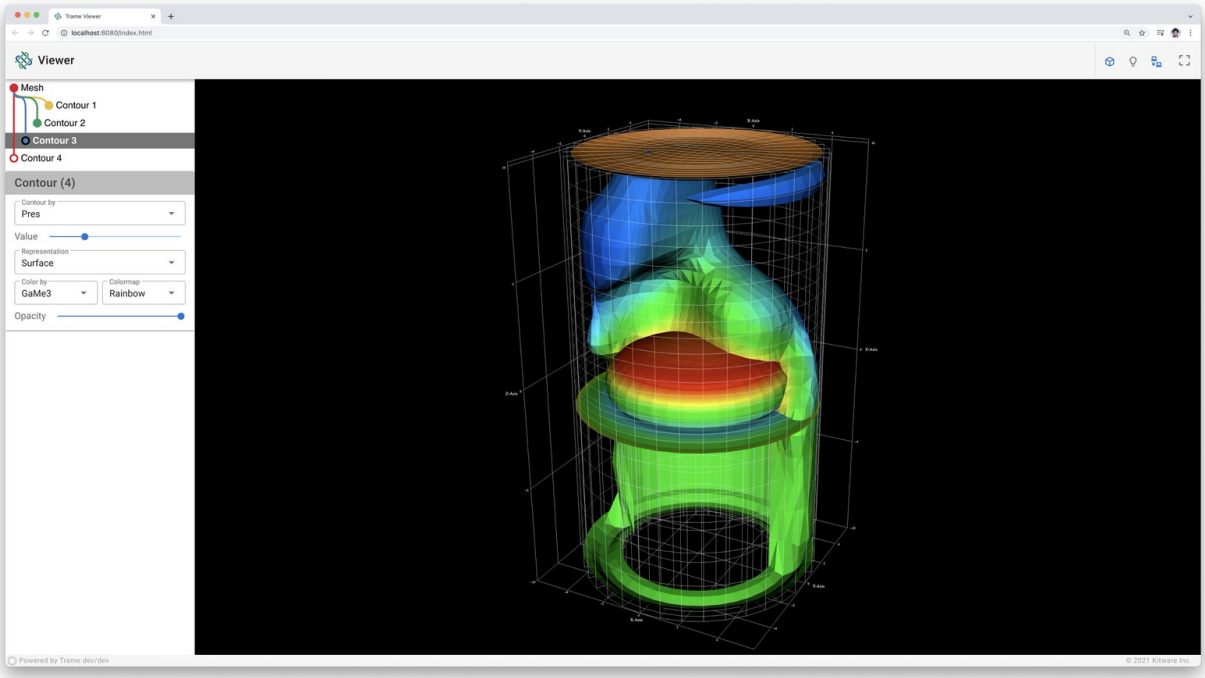
FOUR

TRAME: SIMPLE, POWERFUL, INNOVATIVE

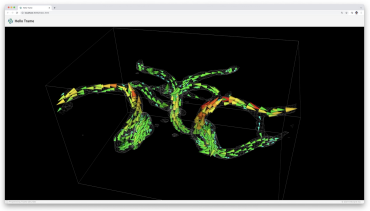
trame - a web framework that weaves together open source components into customized visual analytics easily.

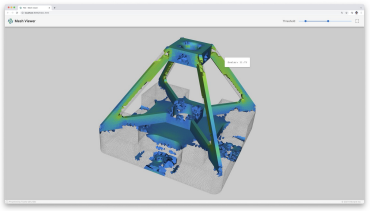
trame is French for

- the core that ties things together
- a guide providing the essence of a task



With **trame**, create stunning, interactive web applications compactly and intuitively.





4.1 3D Visualization

VTK and ParaView are first-class citizens at the core of **trame**, providing complete control of 3D visualizations and data movement. The developers enjoy a write-once environment while trame simply exposes both local and remote rendering through a single method.

With best-in-class VTK and ParaView platforms at its core, **trame** provides complete control of 3D visualizations and data movements. Developers benefit from a write-once environment while **trame** simply exposes both local and remote rendering through a single method.

4.2 Rich Features

trame leverages existing libraries and tools such as Vuetify, Altair, Vega, deck.gl, VTK, ParaView, and more, to create vivid content for visual analytics applications.

4.3 Problem Focused

By relying simply on Python and HTML, **trame** focuses on one's data and associated analysis and visualizations while hiding the complications of web development.

4.4 Desktop to cloud

The resulting **trame** applications act as local desktop applications or remote cloud applications both accessed through the browser.

4.5 Community

- WebSite
- Discussions
- Issues
- RoadMap
- Contact Us
-

4.5.1 Enjoying trame?

Share your experience with a testimonial or with a brand approval.

PYTHON MODULE INDEX

t

INDEX

P

R

S

T

U

V

W