Cytoscape Reference

Access this documentation in your Python terminal with:

```
>>> help(dash_cytoscape.Cytoscape)
```

Our recommended IDE for writing Dash apps is Dash Enterprise's **Data Science Workspaces**, which has typeahead support for Dash Component Properties. **Find out if your company is using Dash Enterprise**.

`id` (*string*; optional): The ID used to identify this component in Dash callbacks.

`autoungrabify` (*boolean*; optional): Whether nodes should be ungrabified (not grabbable by user) by default (if true, overrides individual node state).

`autolock` (*boolean*; optional): Whether nodes should be locked (not draggable at all) by default (if true, overrides individual node state).

`autounselectify` (*boolean*; optional): Whether nodes should be unselectified (immutable selection state) by default (if true, overrides individual element state).

`autoRefreshLayout` (*boolean*; optional): Whether the layout should be refreshed when elements are added or removed.

`boxSelectionEnabled` (*boolean*; optional): Whether box selection (i.e. drag a box overlay around, and release it to select) is enabled. If enabled, the user must taphold to pan the graph.

`className` (*string*; optional): Sets the class name of the element (the value of an element's html class attribute).

`elements` (*list | dict*; optional): A list of dictionaries representing the elements of the networks. 1. Each dictionary describes an element, and specifies its purpose. - `group` (string): Either 'nodes' or 'edges'. If not given, it's automatically inferred. - `data` (dictionary): Element specific data. - `id` (string): Reference to the element, useful for selectors and edges. Randomly assigned if not given. - `label` (string): Optional name for the element, useful when `data(label)` is given to a style's `content` or `label`. It is only a convention. - `parent` (string): Only for nodes. Optional reference to another node. Needed to create compound nodes. - `source` (string): Only for edges. The id of the source node, which is

where the edge starts. - `target` (string): Only for edges. The id of the target node, where the edge ends. - `position` (dictionary): Only for nodes. The position of the node. - `x` (number): The x-coordinate of the node. - `y` (number): The y-coordinate of the node. - `selected` (boolean): If the element is selected upon initialisation. - `selectable` (boolean): If the element can be selected. - `locked` (boolean): Only for nodes. If the position is immutable. - `grabbable` (boolean): Only for nodes. If the node can be grabbed and moved by the user. - `classes` (string): Space separated string of class names of the element. Those classes can be selected by a style selector. 2. The **official Cytoscape.js documentation** offers an extensive overview and examples of element declaration. Alternatively, a dictionary with the format { 'nodes':[], 'edges':[]} is allowed at initialization, but arrays remain the recommended format.

`generateImage` (*dict*; optional): Dictionary specifying options to generate an image of the current cytoscape graph. Value is cleared after data is received and image is generated. This property will be ignored on the initial creation of the cytoscape object and must be invoked through a callback after it has been rendered. The `'type'` key is required. The following keys are supported: - `type` (string): File type to ouput of 'svg, 'png', 'jpg', or 'jpeg' (alias of 'jpg') - `options` (dictionary, optional): Dictionary of options to cy.png() / cy.jpg() or cy.svg() for image generation. See **http://js.cytoscape.org/#core/export** for details. For `'output'`, only 'base64' and 'base64uri' are supported. Default: `{'output': 'base64uri'}`. - `action` (string, optional): Default: `'store'`. Must be one of the following: - `'store'`: Stores the image data (only jpg and png are supported) in `imageData` and invokes server-side Dash callbacks. - `'download'`: Downloads the image as a file with all data handling done client-side. No `imageData` callbacks are fired. - `'both'`: Stores image data and downloads image as file. - `filename` (string, optional): Name for the file to be downloaded. Default: 'cyto'. If the app does not need the image data server side and/or it will only be used to download the image, it may be prudent to invoke `'download'` for `action` instead of `'store'` to improve performance by preventing transfer of data to the server.

`imageData` (*string*; optional): String representation of the image requested with generateImage. Null if no image was requested yet or the previous request failed. Read-only.

`layout` (*dict*; optional): A dictionary specifying how to set the position of the elements in your graph. The `'name'` key is required, and indicates which layout (algorithm) to use. 1. The layouts available by default are: - `random`: Randomly assigns positions - `preset`: Assigns position based on the `position` key in element dictionaries - `circle`: Single-level circle, with optional radius - `concentric`: Multi-level circle, with optional radius - `grid`: Square grid, optionally with numbers of `rows` and `cols` - `breadthfirst`: Tree structure built using BFS, with optional `roots` - `cose`: Force-directed physics simulation 2. Some external layouts are also included. To use them, run `dash_cytoscape.load_extra_layouts()` before creating your Dash app. Be careful about using the extra layouts when not necessary, since they require supplementary bandwidth for loading, which impacts the startup time of the app. - `cose-bilkent`: **https://github.com/cytoscape/cytoscape.js-cose-bilkent** - `cola`: **https://github.com/cytoscape/cytoscape.js-cola** - `euler`: **https://github.com/cytoscape/cytoscape.js-dagre** - `spread`: **https://github.com/cytoscape/cytoscape.js-spread** - `dagre`: **https://github.com/cytoscape/cytoscape.js-dagre** - `klay`: **https://github.com/cytoscape/cytoscape.js-klay** 3. The keys accepted by `layout` vary depending on the algorithm, but some keys are accepted by all layouts: - `fit` (boolean): Whether to render the nodes in order to fit the canvas. - `padding` (number): Padding around the sides of the canvas, if fit is enabled. - `animate` (boolean): Whether to animate change in position when the layout changes. - `animationDuration` (number): Duration of animation in milliseconds, if enabled. - `boundingBox` (dictionary): How to constrain the layout in a specific area.

Keys accepted are either `x1, y1, x2, y2` or `x1, y1, w, h`, all of which receive a pixel value. 4. The complete list of layouts and their accepted options are available on the **Cytoscape.js docs**. For the external layouts, the options are listed in the "API" section of the README. Note that certain keys are not supported in Dash since the value is a JavaScript function or a callback. Please visit **this issue** for more information.

`minZoom` (*number*; optional): A minimum bound on the zoom level of the graph. The viewport can not be scaled smaller than this zoom level.

`maxZoom` (*number*; optional): A maximum bound on the zoom level of the graph. The viewport can not be scaled larger than this zoom level.

`mouseoverNodeData` (*dict*; optional): The data dictionary of a node returned when you hover over it. Read-only.

`mouseoverEdgeData` (*dict*; optional): The data dictionary of an edge returned when you hover over it. Read-only.

`pan` (*dict*; optional): Dictionary indicating the initial panning position of the graph. The following keys are accepted: - `x` (number): The x-coordinate of the position. - `y` (number): The y-coordinate of the position.

`panningEnabled` (*boolean*; optional): Whether panning the graph is enabled (i.e., the position of the graph is mutable overall).

`responsive` (*boolean*; optional): Toggles intelligent responsive resize of Cytoscape graph with viewport size change

`style` (*dict*; optional): Add inline styles to the root element.

`stylesheet` (*list*; optional): A list of dictionaries representing the styles of the elements. 1. Each dictionary requires the following keys: - `selector` (string): Which elements you are styling. Generally, you select a group of elements (node, edges, both), a class (that you declare in the element dictionary), or an element by ID. - `style` (dictionary): What aspects of the elements you want to modify. This could be the size or color of a node, the shape of an edge arrow, or many more. 2. Both **the selector string** and **the style dictionary** are exhaustively documented in the Cytoscape.js docs. Although methods such as `cy.elements(...)` and `cy.filter(...)` are not available, the selector string syntax stays the same.

`selectedNodeData` (*list*; optional): The list of data dictionaries of all selected nodes (e.g. using Shift+Click to select multiple nodes, or Shift+Drag to use box selection). Read-only.

`selectedEdgeData` (*list*; optional): The list of data dictionaries of all selected edges (e.g. using Shift+Click to select multiple nodes, or Shift+Drag to use box selection). Read-only.

`tapNode` (*dict*; optional): The complete node dictionary returned when you tap or click it. Read-only. 1. Node-specific items: - `edgesData` (dictionary) - `renderedPosition` (dictionary) - `timeStamp` (number) 2. General items (for all elements): - `classes` (string) - `data` (dictionary) - `grabbable` (boolean) - `group` (string) - `locked` (boolean) - `position` (dictionary) - `selectable` (boolean) - `selected` (boolean) - `style` (dictionary) 3. Items for compound nodes: - `ancestorsData` (dictionary) - `childrenData` (dictionary) - `descendantsData` (dictionary) - `parentData` (dictionary) - `siblingsData` (dictionary) - `isParent` (boolean) - `isChildless` (boolean) - `isChild` (boolean) - `isOrphan` (boolean) - `relativePosition` (dictionary)

`tapNodeData` (*dict*; optional): The data dictionary of a node returned when you tap or click it. Read-only.

`tapEdge` (*dict*; optional): The complete edge dictionary returned when you tap or click it. Read-only. 1. Edge-specific items: - `isLoop` (boolean) - `isSimple` (boolean) - `midpoint` (dictionary) - `sourceData` (dictionary) - `sourceEndpoint` (dictionary) - `targetData` (dictionary) - `targetEndpoint` (dictionary) - `timeStamp` (number) 2. General items (for all elements): - `classes` (string) - `data` (dictionary) - `grabbable` (boolean) - `group` (string) - `locked` (boolean) - `selectable` (boolean) - `selected` (boolean) - `style` (dictionary)

`tapEdgeData` (*dict*; optional): The data dictionary of an edge returned when you tap or click it. Read-only.

`userPanningEnabled` (*boolean*; optional): Whether user events (e.g. dragging the graph background) are allowed to pan the graph.

`userZoomingEnabled` (*boolean*; optional): Whether user events (e.g. dragging the graph background) are allowed to pan the graph.

`zoom` (*number*; optional): The initial zoom level of the graph. You can set `minZoom` and `maxZoom` to set restrictions on the zoom level.

`zoomingEnabled` (*boolean*; optional): Whether zooming the graph is enabled (i.e., the zoom level of the graph is mutable overall).

utils.Tree

A class to facilitate tree manipulation in Cytoscape.

param node_id: The ID of this tree, passed to the node data dict

param children: The children of this tree, also Tree objects

param data: Dictionary passed to this tree's node data dict

param edge_data: Dictionary passed to the data dict of the edge connecting this tree to its parent

Tree.is_leaf()

return: If the Tree is a leaf or not.

Tree.add_children(children)

Add a list of children to the current children of a Tree.

param children: List of Tree objects

Tree.get_edges()

Get all the edges of the tree in Cytoscape JSON format.

return: List of dictionaries, each specifying an edge.

Tree.get_nodes()

Get all the nodes of the tree in Cytoscape JSON format.

return: List of dictionaries, each specifying a node.

Tree.get_elements()

Get all the elements of the tree in Cytoscape JSON format.

return: List of dictionaries, each specifying an element.

Tree.find_by_id(search_id, method='bfs')

Find a Tree object by its ID.

param search_id: the queried ID

param method: Which traversal method to use. Either "bfs" or "dfs".

return: Tree object if found, None otherwise.

Tree.create_index()

Generate the index of a Tree, and set it in place. If there was a previous index, it is erased. This uses a BFS traversal. Please note that when a child is added to the tree, the index is not regenerated. Furthermore, an index assigned to a parent cannot be accessed by its children, and vice-versa.

return: Dictionary mapping node_id to Tree object.