Part 3 Conventions & Common Patterns

Naming Conventions

Using a consistent naming convention across the community makes the developer experience more predictable and enjoyable. Here are our conventions:

Module Name

kebab case, prefixed with `dash-`

e.g. `pip install dash-table`, `pip install dash-vtk`

Component Library

1. Prefix with `dash_`
2. Use `snake_case`

e.g. `dash_table` or `dash_vtk`

Component Names

`UpperCamelCase` - `DataTable` not `dataTable` or `data_table`. The convention for components is to use `UpperCamelCase` that is similar to - `DataTable` not `data_table`

Property Names

`snake_case` almost everywhere

`snake_case` is the Python convention and is what should be used in most cases. If a user sees `camelCase` then they feel like "this wasn't written for Python". `snake_case` is also generally acceptable in R & Julia.

Exception being sophisticated JavaScript libraries

Exception being component libraries where we *rely* on the JavaScript component's documentation for their nested properties. For example, Cytoscape (uses css kebab-case e.g. `background-color`), vtk (uses camelCase e.g. `edgeVisibility`), plotly.js (uses flatcase, e.g. `bgcolor`). In those cases, stick to the underlying JS library's convention and point users to their docs. The users will know about the underlying library

Convert `camelCase` to `snake_case` within React

If the component library uses `camelCase` but does not have an insurmountable API, then use convert `camelCase` to `snake_case` within the component itself and redocument all of the properties in the Dash docs. Example being `dcc.DatePicker` . The user does not need to know that we’re using `react-dates` under the hood and we don’t need to refer to `react-dates` ’s documentation. It’s easy for us to re-document.

Considerations for Dash .NET

`snake_case` will go against .NET convention and we’ll live with it for now. Maybe in the future we’ll have some automagic converter in Dash's frontend that will automatically convert property names but that's down the line.

Mention the exception

If components use `camelCase` , we’ll mention this convention in the component docs.

e.g. “ `dash-cytoscape` uses `kebab-case` properties instead of dash’s conventional `snake_case` . This is because `dash-cytoscape` is a simple wrapper around the powerful `cytoscape.js` library. As you use `dash-cytoscape` , you will likely need to refer to `cytoscape.js` ’s documentation. We’ve chosen to make it easier for you to navigate cytoscape.js’s documentation by keeping the same naming convention.“.

Utility Functions

The conventions mentioned above are for the component properties themselves.

If a package has a utility library that goes along with the component, it will use `snake_case` .

For example, `dash-cytoscape` has `utils.get_children` methods (https://)dash.plotly.com/cytoscape/reference even though the cytoscape properties use `kebab-case`

Use `value` where appropriate

`value` is a common property name representing the main user-editable input in a component. If your component has a single user input, consider using the property name `value` .

However, avoid `value` if it would be ambiguous. For example, in `dcc.DatePickerRange` , there are two input boxes - start date and end date. We used `start_date` and `end_date` instead of e.g. `value` and `end_date` .

Props for Different UX Updates

- Separate Props - n_clicks vs n_blur
- Or "update_mode" that determines when e.g. `value` is updated

Up Next Part 4 Examples