plotly

Star 23,447

Dash Python > ***Loading States***

Plotly Studio: Transform any dataset into an interactive data application in minutes with AI. **Sign up for early access now.**

Loading States

Loading Component

The `dcc.Loading` component displays a spinner when the components it wraps are in a loading state. Internally, Dash updates the `dcc.Loading` component when the loading state of the components it wraps changes.

```
from dash import Dash, dcc, html, Input, Output, callback
import time

app = Dash()

app.layout = html.Div(
    children=[
        html.H3("Edit text input to see loading state"),
        dcc.Input(id="ls-input-1", value='Input triggers local spinner'),
        dcc.Loading(id="ls-loading-1", children=[html.Div(id="ls-loading-output-1")], type="de
        html.Div(
            [
                dcc.Input(id="ls-input-2", value='Input triggers nested spinner'),
                dcc.Loading(
                    id="ls-loading-2",
                    children=[html.Div([html.Div(id="ls-loading-output-2")])],
                    type="circle",
                )
            ]
        ),
    ],
)

@callback(Output("ls-loading-output-1", "children"), Input("ls-input-1", "value"))
def input_triggers_spinner(value):
    time.sleep(1)
    return value


@callback(Output("ls-loading-output-2", "children"), Input("ls-input-2", "value"))
def input_triggers_nested(value):
    time.sleep(1)
    return value


if __name__ == "__main__":
    app.run(debug=False)
```

Edit text input to see loading state

Input triggers local spi...

Input triggers local spinner

Input triggers nested s...

Input triggers nested spinner

Please also check out the docs for the **Loading component** for more information on how to use the Loading component.

Check Loading States From Components

Aside from using the **Loading component**, you can check if a certain component (either from `dash_core_components` or `dash_html_components`) is loading by checking the `data-dash-is-loading` attribute set on that component's HTML output. This means that you can target those components yourself with CSS, and create your own custom loading for them. Here's an example of what that could look like:

```
from dash import Dash, dcc, html, Input, Output, callback
import time

app = Dash()

app.layout = html.Div(
    children=[
        html.Div(id="cls-output-1", className='output-example-loading'),
        dcc.Input(id="cls-input-1", value="Input triggers local spinner"),
        html.Div(
            [
                html.Div(id="cls-output-2", className='output-example-loading'),
                dcc.Input(id="cls-input-2", value="Input triggers nested spinner"),
            ]
        ),
    ]
)


@callback(Output("cls-output-1", "children"), Input("cls-input-1", "value"))
def input_triggers_spinner(value):
    time.sleep(1)
    return value


@callback(Output("cls-output-2", "children"), Input("cls-input-2", "value"))
def input_triggers_nested(value):
    time.sleep(1)
    return value


if __name__ == "__main__":
    app.run(debug=False)
```

Input triggers local spinner

Input triggers local spi...

Input triggers nested spinner

Input triggers nested s...

You could target all components in the layout above that are loading using the following CSS:

```
*[data-dash-is-loading="true"]{
    visibility: hidden;
}
*[data-dash-is-loading="true"]::before{
    content: "Loading...";
    display: inline-block;
    color: magenta;
    visibility: visible;
}
```

You can also target any other HTML attributes using CSS. To target a specific class use the following CSS:

```
*[class="output-example-loading"][data-dash-is-loading="true"]{
    visibility: hidden;
}
*[class="output-example-loading"][data-dash-is-loading="true"]::before{
    content: "Loading...";
    display: inline-block;
    color: magenta;
    visibility: visible;
}
```

Dash Python ***> Loading States***