 Plotly Studio: the AI-native desktop app for creating data apps and visualizing data. [Sign up for early access!](#)

Dash 2.9.2 Released - Partial Property Updates with Patch(), Duplicate Outputs, dcc.Geolocation, Scatter Group Attributes and More

[announcements](#)

adamschroeder 1 April 4, 2023, 4:41pm

Update : **version 2.11.0 has been released** since this was posted.

We're excited to announce that Dash 2.9.2 has been released 

Register for the **upcoming webinar** to learn how to use the open-source feature.

```
pip install dash==2.9.2
```

Official Changelog  **[Dash v2.9.2](#)**

Highlights

Partial Property Updates with Patch()

Partial Property Updates can improve the performance of your Dash apps by preventing unnecessary computation, data fetching, and reducing the amount of data that travels over the network between the browser and the server.

Consider a simple example with a dropdown and a graph: Currently, Dash callbacks update the entire figure object and send that data over the network. However in many cases, only certain parts of the graph change. For example, the x-axis data might stay the same while the y-axis data changes or the color of the graph will change but the data will remain the same.

With Partial Property Updates, you can update your callbacks to send over the new y-axis data on its own without the rest of the figure. As a result, it reduces the size of the network payload by 50% (1 array of **x** data instead of 2 arrays of **x** and **y** data), and there can be a noticeable performance improvement to charts or tables with >10K points.

The code change is minimal. In previous Dash versions (previous to Dash v2.9.0), you would have:

```
app.layout = html.Div([
    dcc.Dropdown(df.columns, id="column-selected"),
    dcc.Graph(id="my-graph")
])
@app.callback(Output("my-graph", "figure"), Input("column-selected", "value"))
def update(value):
    return px.scatter(df, x=df.Index, y=df.columns[1])
```

Notice how the whole figure is recreated inside the callback and sent back to the browser.

With Partial Property Updates (Dash v.2.9 or higher), you can return a Patch() object that tells Dash's frontend to only change the "y" data in the first trace instead of returning the entire figure.

```
app.layout = html.Div([
    dcc.Dropdown(df.columns, id="column-selected"),
    dcc.Graph(id="my-graph", figure=px.scatter(df, x=df.index, y=df.columns[1]))
])
@app.callback(Output("my-graph", "figure"), Input("column-selected", "value"),
    prevent_initial_call=True)
def update(value):
    patch_figure = Patch()
    patch_figure["data"][0]["y"] = df[value]
    return patch_figure
```

Extending, Appending, Merging, and More

Patch() defines operations that can be applied to the property in the browser client. In the example above, we did a nested assignment to the data in the figure located at `["data"][0]["y"]`.

In addition to assignment, Patch() supports extending, appending, merging, and more. Here's the full list. The "Result" assumes that `["data"][0]["x"]` is initialized to `["B", "C", "D"]`.

Method	Works On	Example	Result
1.prepend	Lists	Patch()["data"][0]["x"].prepend("A")	<code>["A", "B", "C", "D"]</code>
2.append	Lists	Patch()["data"][0]["x"].append("E")	<code>["B", "C", "D", "E"]</code>
3.extend	Lists	Patch()["data"][0]["x"].extend(["E","F"])	<code>["B", "C", "D", "E", "F"]</code>
4.reverse	Lists	Patch()["data"][0]["x"].reverse()	<code>["D", "C", "B"]</code>
5.insert	Lists	Patch()["data"][0]["x"].insert(2, "X")	<code>["B", "C", "X", "D"]</code>
6.clear	Lists	Patch()["data"][0]["x"].clear()	<code>[]</code>
7.remove	Lists	Patch()["data"][0]["x"].remove("C")	<code>["B", "D"]</code>
8.del	Lists	del Patch()["data"][0]["x"][0]	<code>["C", "D"]</code>
9.update	Dictionaries	Patch()["data"][0]["marker"].update({"color": "red"})	<code>{ "color": "red" }</code> instead of <code>{ "color": "#636efa" }</code>
10. =	List, Dict, Str, Int	Patch()["layout"]["title"] = "New Title of App"	New Title of App

The most important thing to keep in mind is the type of objects you're attempting to update.

`["data"][0]["x"]` is of type list, which is why we could use methods 1 through 8. The **update** method will only work on objects that are of dictionary type, such as `["data"][0]["marker"]`.

The new **[Dash Docs chapter on the Partial Property Updates](#)** is very informative and full of helpful examples; we highly recommend you check it out. That said, below we'll highlight a few of our favorite examples.

Adding Data

Adding data to an existing component is a really common use case for Patch(). Here are 3 examples of adding data to graphs, tables, and network graphs.

Graph:

```
from dash import Dash, html, dcc, Input, Output, Patch
import plotly.graph_objects as go
```

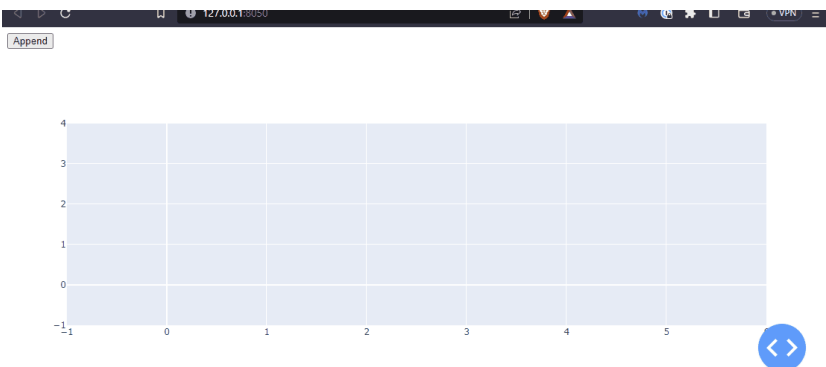
```
import datetime
import random

app = Dash(__name__)

fig = go.Figure()

app.layout = html.Div([
    html.Button("Append", id="append-new-val"),
    dcc.Graph(figure=fig, id="append-example-graph"),
])

@app.callback(
    Output("append-example-graph", "figure"),
    Input("append-new-val", "n_clicks"),
    prevent_initial_call=True,
)
def add_data_to_fig(n_clicks):
    current_time = datetime.datetime.now()
    random_value = random.randrange(1, 30, 1)
    patched_figure = Patch()
    patched_figure["data"][0]["x"].append(current_time)
    patched_figure["data"][0]["y"].append(random_value)
    return patched_figure
```



AG Grid Table

```
from dash import Dash, html, Input, Output, Patch, no_update
import plotly.express as px
import dash_ag_grid as dag

app = Dash(__name__)

df = px.data.iris()

app.layout = html.Div([
    html.Button("Add Rows", id="add-data-rows"),
    dag.AgGrid(
        id="ag-table",
        className="ag-theme-alpine-dark",
        columnDefs=[{"headerName": x, "field": x} for x in df.columns],
        rowData=df.to_dict("records"),
        columnSize="sizeToFit",
        dashGridOptions={"pagination": True},
    )
])

@app.callback(
    Output("ag-table", "rowData"),
    Input("add-data-rows", "n_clicks"),
    prevent_initial_call=True,
)
def add_data_to_fig(n_clicks):
    patched_table = Patch()
```

Add Rows					
sepal_length	sepal_width	petal_length	petal_width	species	species_id
5.1	3.5	1.4	0.2	setosa	1
4.9	3	1.4	0.2	setosa	1
4.7	3.2	1.3	0.2	setosa	1
4.6	3.1	1.5	0.2	setosa	1
5	3.6	1.4	0.2	setosa	1
5.4	3.9	1.7	0.4	setosa	1
4.6	3.4	1.4	0.3	setosa	1

Network Graph (Cytoscape):

```
from dash import Dash, html, Output, Input, Patch, State
import dash_cytoscape as cyto

app = Dash(__name__)
original_elements = [
```

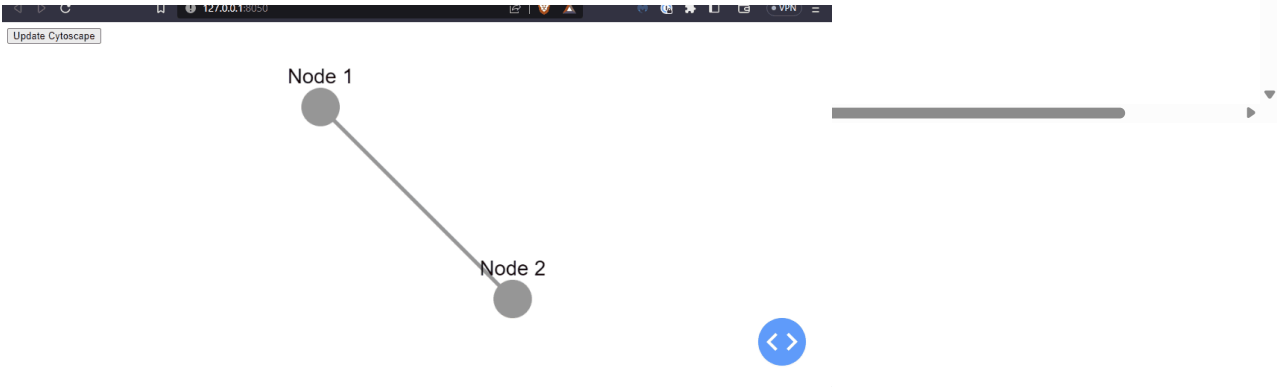
```

# The nodes elements
{'data': {'id': 'id_1', 'label': 'Node 1'},
 'position': {'x': 50, 'y': 50}},
{'data': {'id': 'id_2', 'label': 'Node 2'},
 'position': {'x': 200, 'y': 200}},

# The edge elements
{'data': {'source': 'id_1', 'target': 'id_2', 'label': 'Node 1 to 2'}},
]
app.layout = html.Div([
    html.Button("Update Cytoscape", id="add-data", n_clicks=2),
    cyto.Cytoscape(
        id='my-cytoscape',
        layout={'name': 'preset'},
        style={'width': '100%', 'height': '400px'},
        elements=original_elements
    )
])

@app.callback(
    Output("my-cytoscape", "elements"),
    Input("add-data", "n_clicks"),

```



Updating Charts

Another common class of use cases is updating a chart without changing the data.

This example demonstrates how to highlight scatter points within a graph:

```

from dash import Dash, dcc, html, Input, Output, Patch
import plotly.express as px

app = Dash(__name__)

# Getting our data
df = px.data.gapminder()
df = df.loc[df.year == 2002].reset_index()
df = df.loc[:10,:]

# Creating our figure
fig = px.scatter(x=df.lifeExp, y=df.gdpPercap, hover_name=df.country)
fig.update_traces(marker=dict(color="blue"))

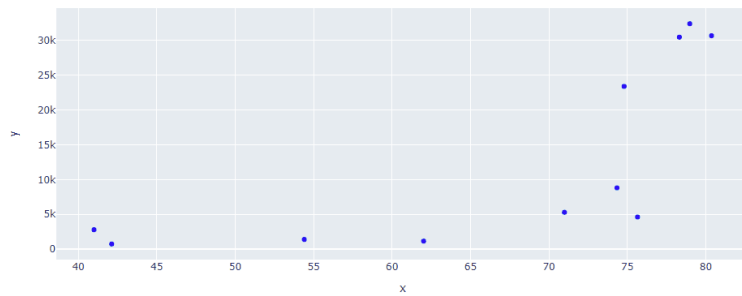
app.layout = html.Div([
    html.H4("Updating Point Colors"),
    dcc.Dropdown(id="dropdown", options=df.country.unique(), multi=True),
    dcc.Graph(id="graph-update-example", figure=fig),
])

@app.callback(
    Output("graph-update-example", "figure"), Input("dropdown", "value"), prevent_initial_call=True
)
def update_markers(countries):
    country_count = list(df[df.country.isin(countries)].index)
    patched_figure = Patch()

```

Updating Point Colors

Select...



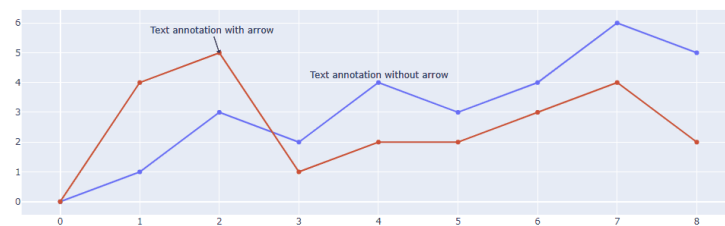
Or adding annotations:

```
from dash import Dash, html, dcc, Input, Output, Patch
import plotly.graph_objects as go

app = Dash(__name__)

fig = go.Figure(
    [
        go.Scatter(x=[0, 1, 2, 3, 4, 5, 6, 7, 8], y=[0, 1, 3, 2, 4, 3, 4, 6, 5]),
        go.Scatter(x=[0, 1, 2, 3, 4, 5, 6, 7, 8], y=[0, 4, 5, 1, 2, 2, 3, 4, 2]),
    ],
    go.Layout(
        dict(
            annotations=[
                dict(
                    x=2,
                    y=5,
                    text="Text annotation with arrow",
                    showarrow=True,
                    arrowhead=1,
                ),
                dict(
                    x=4,
                    y=4,
                    text="Text annotation without arrow",
                    showarrow=False,
                    yshift=10,
                ),
            ],
        )
    )
)
```

Show/Clear Annotations



Pattern Matching Callback Support

This even works with appending components with the children component. Callbacks can be defined on dynamic components with pattern matching callbacks.

```
from dash import Dash, dcc, html, Input, Output, ALL, Patch

app = Dash(__name__, suppress_callback_exceptions=True)

app.layout = html.Div([
    html.Button("Add Filter", id="add-filter", n_clicks=0),
    html.Div(id="dropdown-container", children=[]),
    html.Div(id="dropdown-container-output"),
])

@app.callback(
    Output("dropdown-container", "children"),
    Input("add-filter", "n_clicks"),
)
def display_dropdowns(n_clicks):
    patched_children = Patch()
    new_dropdown = dcc.Dropdown(
        ["NYC", "MTL", "LA", "TOKYO"],
        id={"type": "filter-dropdown", "index": n_clicks},
    )
    patched_children.append(new_dropdown)
    return patched_children
```


```
patched_children.append(new_dropdown)
return patched_children

@app.callback(
    Output("dropdown-container-output", "children"),
    Input("dropdown", "value"),
    prevent_initial_call=True
)
def update_dropdown(selected_value):
    # Logic to update children based on selected value
    return [new_dropdown]
```

Add Filter

Select...

Dropdown 1 = None



Allowing Duplicate Callback Outputs

Multiple callbacks can now target the same output! Previously, you had to use `ctx.triggered` within a single callback. Now you can just write separate callback functions. This feature was discussed in length by the community in the topic "[Duplicate Callback Outputs - Solution & API Discussion](#)" - Many thanks to everyone who has contributed to the discussion over the years 😊

To target the same component-property from multiple callbacks, set `allow_duplicate=True` on the duplicate callback Output:

```
@app.callback(
    Output("id_name", "compnt_prop", allow_duplicate=True),
    Input("id_name", "compnt_prop"),
    prevent_initial_call=True
)
```

Duplicate callback outputs appear frequently with Patch() & partial property outputs where different callbacks will perform different operations. Here is the earlier example rewritten to allow duplicates. In this example clicking one button adds rows to AgGrid and clicking the second button refreshes the entire dataset.

```
from dash import Dash, html, Input, Output, Patch, no_update
import plotly.express as px
import dash_ag_grid as dag

app = Dash(__name__)

df = px.data.iris()

app.layout = html.Div(
    [
        html.Button("Add Rows", id="add-data-rows"),
        html.Button("Reload data", id="reload-button"),
        dag.AgGrid(
            id="ag-table",
            className="ag-theme-alpine-dark",
            columnDefs=[{"headerName": x, "field": x} for x in df.columns],
            rowData=df.to_dict("records"),
            columnSize="sizeToFit",
            dashGridOptions={"pagination": True}
        )
    ]
)

@app.callback(
    Output("ag-table", "rowData", allow_duplicate=True),
    Input("add-data-rows", "n_clicks"),
    prevent_initial_call=True,
)
```

Add Rows | Reload data

sepal_length	sepal_width	petal_length	petal_width	species	species_id
5.1	3.5	1.4	0.2	setosa	1
4.9	3	1.4	0.2	setosa	1
4.7	3.2	1.3	0.2	setosa	1
4.6	3.1	1.5	0.2	setosa	1
5	3.6	1.4	0.2	setosa	1
5.4	3.9	1.7	0.4	setosa	1
4.6	3.4	1.4	0.3	setosa	1

1 to 15 of 150 | Page 1 of 10

When setting `allow_duplicate=True` on a callback output, you'll need to either set `prevent_initial_call=True` on the callback, or set `app = Dash(prevent_initial_callback="initial_duplicate")` on your app. This prevents callbacks that target the same output running at the same time when the page initially loads.

Curious why we didn't make `allow_duplicate=True` the default? It's because of this ambiguity around initial calls when the page loads. The order is not guaranteed if multiple callbacks target the same output and are fired at the same time without `prevent_initial_call=True`. This can be the source of subtle bugs! So we decided that it would be better if we could warn users that they need to set `prevent_initial_call=True` in addition to `allow_duplicate=True` when using multiple outputs.

We encourage you to read further about the duplicate callback feature in [its new documentation page](#) or in the [partial property updates chapter](#).

New dcc.Geolocation component

The dcc.Geolocation component can be used to access location data from viewers of the Dash app (with their permission!). It uses the [browser's built-in Geolocation API](#). Thank you [@AnnMarieW](#) for this contribution.

In the example below, the first callback updates the location data when the button is selected. The second callback then triggers with the `local_date` and `position` data and outputs it to a div.

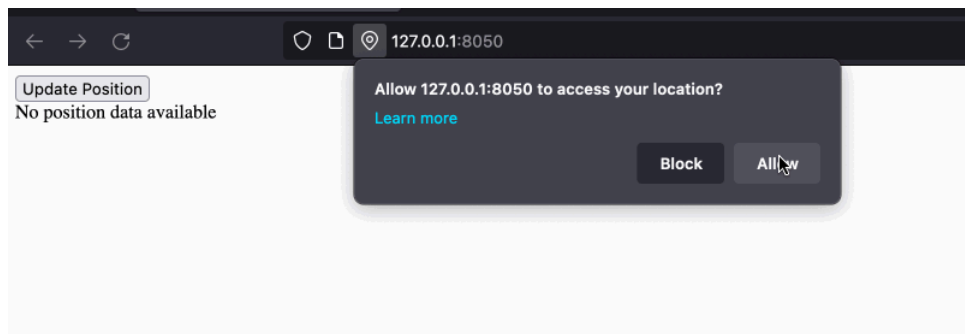
```
from dash import Dash, dcc, html, Input, Output

app = Dash(__name__)

app.layout = html.Div(
    [
        html.Button("Update Position", id="update_btn"),
        dcc.Geolocation(id="geolocation"),
        html.Div(id="text_position"),
    ]
)

@app.callback(Output("geolocation", "update_now"), Input("update_btn", "n_clicks"))
def update_now(click):
    return True if click and click > 0 else False

@app.callback(
    Output("text_position", "children"),
    Input("geolocation", "local_date"),
    Input("geolocation", "position"),
)
def display_output(date, pos):
    if pos:
        return html.P(
            f"As of {date} your location was: lat {pos['lat']}, lon {pos['lon']}, accuracy {pos['accuracy']} meters",
        )
```



Another example where we use the dcc.Geolocation component to access location and display it on a map:

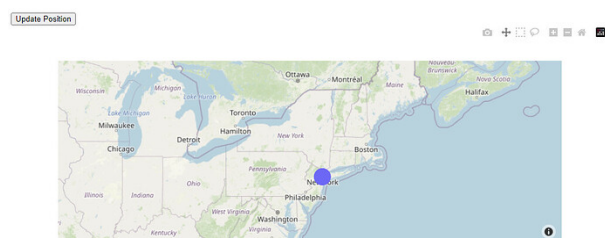
```
from dash import Dash, dcc, html, Input, Output
import plotly.express as px

app = Dash(__name__)

app.layout = html.Div(
    [
        html.Button("Update Position", id="update_btn"),
        dcc.Geolocation(id="geolocation"),
        html.Div(id="map-location"),
    ]
)

@app.callback(Output("geolocation", "update_now"), Input("update_btn", "n_clicks"))
def update_now(click):
    return True if click and click > 0 else False

@app.callback(
    Output("map-location", "children"),
    Input("geolocation", "position"),
)
def display_output(pos):
    if pos:
        fig = px.scatter_mapbox(lat=[pos['lat']], lon=[pos['lon']])
        fig.update_layout(mapbox_style="open-street-map")
        fig.update_traces(marker_size=30)
```



You can read more about the [Geolocation properties](#) in the docs.

Plotly.py & dcc.Graph Updates

Updated Plotly.js to from version 2.18.2 to version 2.20.0.

The version of Plotly.js that is built in here is the same one that is bundled with the recently released Plotly.py 5.14.0, so we recommend that you upgrade to Plotly 5.14.0 to get the full benefit of all of these libraries working together.

```
pip install plotly==5.14.0
```

Official Changelog Plotly v5.14.0

There have been many developments in the world of Plotly.py and dcc.Graph recently. To highlight just a few:

dcc.Graph - Adding Text Labels to Shapes

You can now add **labels directly to shapes** (lines, rectangles, and circles) within a dcc.Graph using the new shape label property. Previously, adding text to shapes required adding a custom annotation or a scatter plot with text and both options were a huge pain 😞. Special thanks to our customer at the Volkswagen Center of Excellence for Battery Systems for sponsoring development.

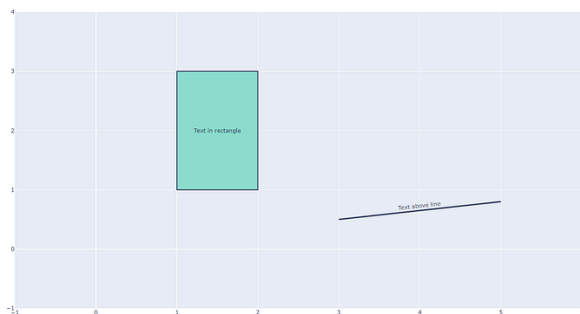
```
import plotly.graph_objects as go

fig = go.Figure()

fig.add_shape(
    type="rect",
    fillcolor='turquoise',
    x0=1,
    y0=1,
    x1=2,
    y1=3,
    label=dict(text="Text in rectangle")
)

fig.add_shape(
    type="line",
    x0=3,
    y0=0.5,
    x1=5,
    y1=0.8,
    line_width=3,
    label=dict(text="Text above line")
)

fig.show()
```



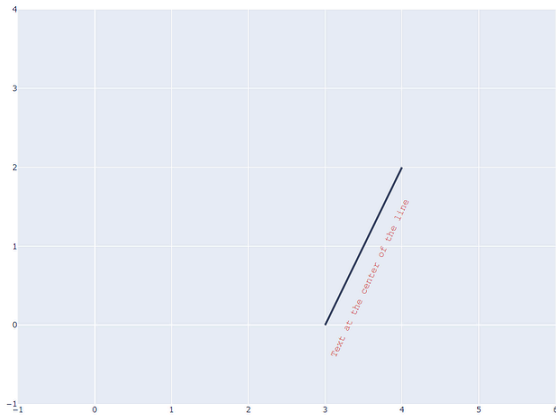
You can use other dictionary keys inside the `label` property to further customize the label. For example:

```
import plotly.graph_objects as go

fig = go.Figure()

fig.add_shape(
    type="line",
    x0=3,
    y0=0,
    x1=4,
    y1=2,
    line_width=3,
    label=dict(
        text="Text at the center of the line",
        textposition='middle', # for lines: [start | middle | end], for other shapes: [ top | middle | bottom ] [ left | right ]
        textangle=-65, # Number between -180 and 180
        padding=45,
        yanchor="top", # [top | middle | bottom]
        xanchor="auto", # [ auto | left | center | right ]
        font=dict(family="Courier New, monospace", size=15, color='red')
    )
)

fig.show()
```



For a few more examples see the section on [Adding Text Labels to Shapes](#). Or, for greater detail, see [the Figure Reference docs](#) on layout shapes.

Specifying Label Aliases

With `labelalias`, you can specify replacement text for specific tick and hover labels without needing to change the labels in your underlying data.

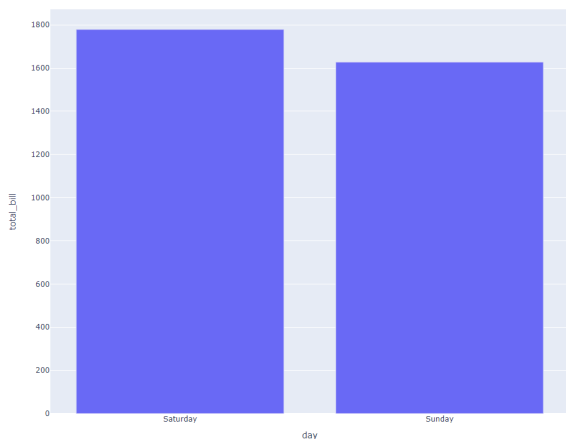
In this example, the dataset has the values of "Sat" and "Sun" in the day column. By setting `labelalias=dict(Sat="Saturday", Sun="Sunday")`, we swap these out for "Saturday" and "Sunday".

```
import plotly.express as px
import pandas as pd

df = px.data.tips()
df = df[df.day.isin(['Sat', 'Sun'])].groupby(by='day', as_index=False).sum(numeric_only=True)

fig = px.bar(df, x="day", y="total_bill")
fig.update_xaxes(labelalias=dict(Sat="Saturday", Sun="Sunday"))

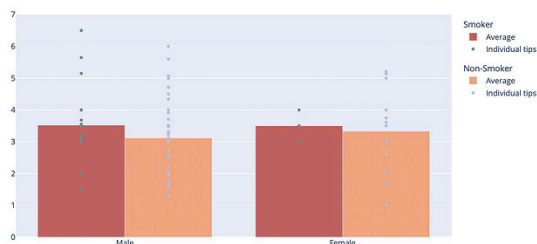
fig.show()
```



Grouped Scatter Support

You can now overlay scatter points over grouped bars. In the following example, we use the new attributes `scattermode` and `offsetgroup` on scatter to create a figure that shows individual points on a grouped scatter and averages with grouped bars. Previously, all of the scatter points would be positioned over the Male and Female label instead of over their respective bars.

For the full code for this example, see the [multiple chart types page](#) in the Plotly.py docs.



There's also a new `scattergap` attribute that you can use to adjust the gap between the different scatter groups. These examples show grouped scatter points with 1) a default gap and with 2) a scattergap of 0.75.

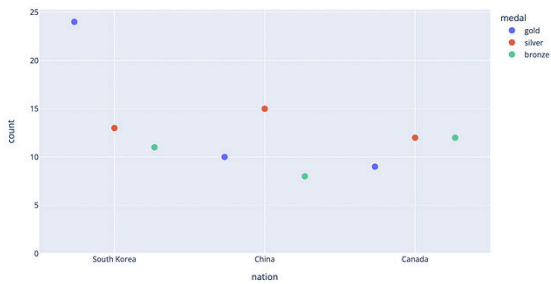
```
import plotly.express as px

df = px.data.medals_long()

fig = px.scatter(df, y="count", x="nation", color="medal")
fig.update_traces(marker_size=10)
```



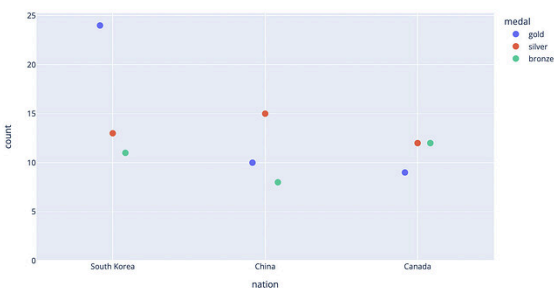
```
fig.update_layout(scattermode="group")
fig.show()
```



```
import plotly.express as px

df = px.data.medals_long()

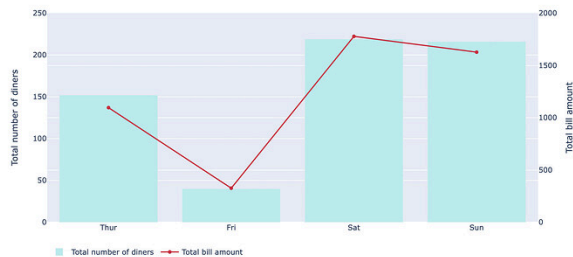
fig = px.scatter(df, y="count", x="nation", color="medal")
fig.update_traces(marker_size=10)
fig.update_layout(scattermode="group", scattergap=0.75)
fig.show()
```



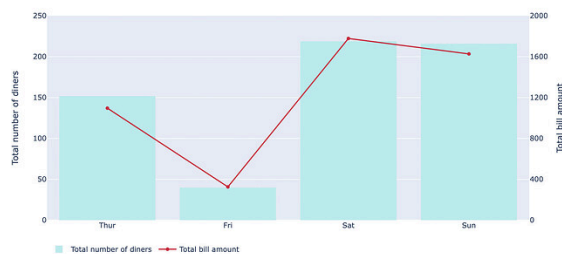
This feature was anonymously sponsored. Thank you to our sponsor! ❤️

Syncing Axes Ticks

With overlaid axes, each axis by default has its own number of ticks. This can lead to graphs like this with odd spacing on ticks and gridlines (see how the axes on both sides don't align)



You can now sync the number of ticks on overlaid axes, by setting `"tickmode=sync"` on an axis that is overlaying another:



The full code for this example is available on the [multiple axes](#) page.

Notable Bug Fixes, Additions & Minor Changes

Dash:

- [#2479](#) Fix `KeyError` "Callback function not found for output [...], perhaps you forgot to prepend the '@?' issue when using duplicate callbacks targeting the same output. This issue would occur when the app is restarted or when running with multiple `unicorn` workers.
- [#2471](#) Fix [#2467](#). `allow_duplicate` output with clientside callback.
- [#2473](#) Fix [#2221](#). Fix background callbacks with different outputs but same function.
- [#2461](#) Fix [#2460](#). Fixes Pytest plugin make report when testing not installed.
- [#2450](#) Fix [#2063](#). Set label style `display: block` if `inline` is false in RadioItems & Checklist components. To keep previous behavior, set `inline=True`.
- [#2400](#) Added `disable_n_clicks=True` to the `html.Div` components in `page_container`. You can [read more about it](#) in the docs. Thank you [@AnnMarieW](#) 🙏
- [#2388](#) Fix [#2368](#) ordering of Pattern Matching ALL after update to the subtree.
- [#2367](#) Updated the default `favicon.ico` to the current Plotly logo
- [#2068](#) Added `refresh="callback-nav"` in `dcc.Location`. This allows for navigation without refreshing the page when url is updated in a callback. With thanks to [@AnnMarieW](#)
- [#2417](#) Add `wait_timeout` property to customize the behavior of the default wait timeout used for by `wait_for_page`, fix [#1595](#)

- [#2417](#) Add the element target text for wait_for_text* error message, fix [#945](#)
- [#2425](#) Add `add_log_handler=True` to Dash init, if you don't want a log stream handler at all.
- [#2260](#) Experimental support for React 18. The default is still React v16.14.0, but to use React 18 you can either set the environment variable `REACT_VERSION=18.2.0` before running your app, or inside the app call `dash._dash_renderer._set_react_version("18.2.0")`. THIS FEATURE IS EXPERIMENTAL. It has not been tested with component suites outside the Dash core, and we may add or remove available React versions in any future release.
- [#2429](#) Fix side effect on updating possible array children triggering callbacks, fix [#2411](#).
- [#2415](#) Fix background callbacks progress not deleted after fetch.
- [#2426](#) Set default interval to 1 second for app.long_callback, restoring the behavior it had before v2.6.0 when we introduced `background=True` callbacks.

Plotly:

- Fixed an issue with characters displaying incorrectly, by adding `charset="utf-8"` to scripts in `to_html` [[#4114](#)]
- Added `packaging` to install requirements, fixing a `No module named 'packaging'` error on Python 3.6 [[#4113](#)]
- Ensure slider range stays in bounds during the drag [[#4448](#)], with thanks to @jay-bis for the contribution!
- `write_html()` now explicitly encodes output as UTF-8 because Plotly.js' bundle contains such characters [[#4021](#)] and [[#4022](#)]
- fixed `iframe` renderer regression from 5.12 and also fixed error when this renderer was used in the very first cell in a notebook [[#4036](#)]
- Fixed bug for treelines with datetime axes [[#3683](#)]
- Disable slider interactions when `staticPlot` is set to true [[#6393](#)]
- Fix auto `backoff` when marker symbols and sizes are arrays [[#6414](#)]
- Avoid displaying resize cursor on static sliders [[#6397](#)]
- Change bundler from browserify to webpack [[#6355](#)]
- Add `shift` and `autoshift` to cartesian y axes to help avoid overlapping of multiple axes [[#6334](#)], with thanks to [Gamma Technologies](#) for sponsoring the related development!
- Introduce group attributes for `scatter` trace i.e. `alignmentgroup`, `offsetgroup`, `scattermode` and `scattergap` [[#6381](#)], this feature was anonymously sponsored: thank you to our sponsor!
- Add `marker.cornerradius` attribute to `treemap` trace [[#6351](#)]
- Fix line redraw (regression introduced in 2.15.0) [[#6429](#)]
- Fix library's imported name using `requirejs` AMD loader (regression introduced in 2.17.0) [[#6440](#)]
- Improve detection of mobile & tablet devices for WebGL rendering by upgrading `is-mobile` [[#6432](#)]
- Add `sync` tickmode option [[#6356](#), [#6443](#)], with thanks to @filipesantiagoAM and @VictorBezák for the contribution!

Previous Releases

- 🔊 **Dash 2.7 Released** - Directional Arrows Feature, Map Bounds, and DataTable Filter Text
- 🔊 **Dash 2.6 Released** - Background Callbacks, Unit Testing, Persistent Selections, Dropdown Features
- 🔊 **Dash 2.5 Released** - Easier Multi-Page Apps, Component Properties
- 🔊 **Dash 2.4 Released** - Improved Callback Context, Clientside Promises, Typescript Components, Minor Ticks
- 🔊 **Dash 2.3.0 Release** - MathJax and fillpattern option in scatter trace
- 🔊 **Dash 2.2.0 Release** - Adds `ticklabelstep` to axes, and added `dash.get_asset_url`
- 🔊 **Dash 2.1.0 Release** - Autogenerated IDs and rearranged keyword arguments in Dash components

34 Likes

:mega: Dash 2.11.0 Released - Dash in Jupyter, Locked Flask versions, and dcc.Graph Updates

Categorical x axis in Scatter plots

Duplicate callback outputs when restarting the dash app.

Sharing examples of partial update & allow duplicate=True from Dash 2.9.2!

Fig.add_shape() implemented for partial updates?

dcc.Graph: possible to add text boxes?

Show name of shape on hover (or always visible)

mega: Dash 2.17.0 Released - Callback updates with set_props, No Output Callbacks, Layout as List, dcc.Loading, Trace Zorder

:mega: Dash 2.15.0 Released - Slider Tooltip Styling, Setting Allowed Axes Range, More Page Routing Inputs

Partial Property Updates - Patch Class

Update url pathname without using dcc.Link (just callback)

Infinite Callback Recursion

mega: Dash 2.13.0 Released - Coupling with Plotly.py, Shapes in Legends, and Broader DataFrame Support

Fig.add_shape() implemented for partial updates?

filiron 2 April 4, 2023, 5:45pm

Wow Duplicate Callback Outputs is potentially a game changer! Will test this out soon. Prior to this, handling everything within a single callback was very clunky.

7 Likes

Ajay.cse2004 3 April 5, 2023, 5:52am

Thanks
Very nice improvement
Please keep sharing and motivating us

4 Likes

martin20974 April 5, 2023, 7:47am

Huge update! Congratulations, IMHO thi is the biggest game changer since introduction of pattern matching callbacks.

Maybe a stupid question regarding initial update. Wouldn't it be possible to set something like order for initial loading instead of just preventing it?

2 Likes

dash-beginner 5 April 5, 2023, 12:30pm

+1 - Being able to set up an ordering / priority hierarchy would be awesome if possible.

1 Like

David22 6 April 5, 2023, 12:58pm

Thanks for these features!

2 Likes

adamschroeder 7 April 5, 2023, 1:39pm

@martin2097

👋 welcome to the community.

What do you mean? can you clarify or give an example of what you'd like to do?

1 Like

siner308 8 April 5, 2023, 2:25pm

Duplicate callback output that's what we want!!

1 Like

dash-beginner 9 April 5, 2023, 5:54pm

So right now, when you use `allow_duplicate`, you need to set `prevent_initial_call = True` or `prevent_initial_call = "initial_duplicate"` (I believe this is because of the possibility of race conditions that would result in the output of one callback being overwritten by another one that was fired at the same time and updating the same output. Instead, if it were somehow possible to store the responses and then apply the one with the highest priority when multiple are targeting the same output, that would be awesome (but admittedly probably wildly difficult to actually do).

1 Like

hypervalent 10 April 5, 2023, 8:04pm

The duplicate callback output is such a gamechanger!

I had to use the `MultiplexerTransform()` extension from `dash_extensions.enrich` for the longest time and it produced so many roadblocks with later versions of Dash. This is exactly what I needed. Thank you so much for offering it!

3 Likes

allsyntax 11 April 6, 2023, 4:13am

can you use patch in plotly-js, or is this already a feature that is just new to Dash? Ummm, asking for a friend.

1 Like

Anfas 13 April 6, 2023, 9:23am

Hi Adam,
Thanks a lot. It is a huge improvement! Expecting a video tutorial on your youtube channel soon:).

1 Like

jinnyzor 14 April 6, 2023, 2:20pm

For plotly.js, you have direct access to the data, so, yes, you can patch directly. And then pass the new elements to the plotly functions.

The benefit of the patch for Dash is the decrease of network traffic. 😊

1 Like

adamschroeder 15 April 6, 2023, 3:08pm

Hey **@allsyntax**

Patch() is a dash feature, so you can use it in `dcc.graph` to modify a plotly.js figure, but you can't use it directly in plotly.js.

That said, there are lots of ways to incrementally modify a plotly.js figure.

If you're in javascript, you can use `Plotly.restyle` and `Plotly.relayout` or you can directly modify the figure object and use `Plotly.react`.

if you're in jupyter you can use the `FigureWidget`, that allows you to modify the figure object in Python and it'll send only the necessary edits to the browser.

1 Like

aschutte 16 April 6, 2023, 3:17pm

Great updates and can't wait to use.

I ran into an issue with a clientside callback that works in Dash 2.9.0 but not 2.9.2. I'm getting an error that says cannot read undefined properties 'apply'. I can create a post on this, but not sure if there were breaking changes in 2.9.2.

jinnvzor 17 April 6, 2023, 3:19pm

Hello [@aschutte](#),

Yes, there is a patch already in place for this, it should be included in the next release. 😊

1 Like

jokin 18 April 6, 2023, 3:49pm

Thanks for the update! Not sure if we should reopen a bug report, but despite this **Fix side effect on updating possible array children triggering callbacks that were not updated. by Tarkin · Pull Request #2429 · plotly/dash · GitHub**, the bug still exists in 2.9.2. I have tested multiple versions and the latest version to work was 2.7.0. Is this a known issue?

Updating dropdown options triggers another callback that not supposed to trigger

AnnMarieW 19 April 6, 2023, 5:56pm

[@jokin](#) That fix will be released in 2.9.3

2 Likes

rictuar 20 April 7, 2023, 1:01am

n callbacks → 1 output is so major, thanks

2 Likes

martin2097 21 April 7, 2023, 7:58am

Thanks for welcome Adam!

What I meant is that now when I have 2 callbacks targeting same output i.e. one of them handling data and one of them handling style I have to prevent initial call at one of them because there is no way to set order in which they fire on initial load. My idea was that instead prevent initial call one would set initial order for each callback (i.e. 1 for data 2 for style) so there would be clear order in which the callbacks fire on page load. I hope it is clear what I mean from this description 😊

adamschroeder 23 April 7, 2023, 2:35pm

martin2097:

My idea was that instead prevent initial call one would set initial order for each callback (i.e. 1 for data 2 for style) so there would be clear order in which the callbacks fire on page load.

hi [@martin2097](#)

We actually talked about that possibility, but we decided not to move forward with that for the time being, because the solution was not really straightforward. It would require modifying the rendered in a specific way which might cause other issues.

1 Like

leosmi 24 April 7, 2023, 3:42pm

Thank you so much Dash team. I'm so proud of you guys.

4 Likes

martin2097 25 April 7, 2023, 6:43pm

Thanks for explanation. Nevertheless it is fantastic update and it will make dash life much easier!

2 Likes

rictuar 26 April 7, 2023, 11:52pm

Callback Gotchas | Dash for Python Documentation | Plotly

Dash callbacks have some idiosyncrasies that should be taken into consideration when building a Dash app. If you're running into unexpected callback behavior, and the rest of the documentation hasn't shed any light on the situation, try taking a look...

I guess the gotcha associated with (n callbacks → 1 output) can be updated?

3 Likes

luggie 27 April 8, 2023, 2:47pm

Very hyped about **duplicate callback outputs** and **partially update figures**!! Thanks heaps 😊

4 Likes

adamschroeder 28 April 10, 2023, 1:38pm

Good point, [@rictuar](#). Thank you for the tip.



tysonwu 29 April 10, 2023, 8:41pm

Patch() looks really nice! Makes data-related callbacks look a lot more neat.

1 Like

mc2pezwb 30 April 12, 2023, 9:36am

Fantastic! The multiple output will make code much cleaner.

Regarding the partial updates, how would one update only a column of data in dash datatable? The data is a list of dictionaries, that does not seem obvious to me
Thanks

1 Like

adamschroeder 31 April 12, 2023, 12:47pm

hi [@mc2pezwb](#)

It depends what part of the column you'd like to update.

If, for example, you'd like to update the first cell in the first column, try to figure out how to do that within a list of dictionaries. Then, you could try to implement that with Patch().

Let's take the Iris dataset.

```
import plotly.express as px
import pandas as pd

df = px.data.iris()
dff = df.to_dict("records") # this is the data format accepted by the DataTable
dff[0].update({'sepal_length': 99.1}) # update first cell of column `sepal_length` to 99.1
print(dff) # print to see result
```

Once you confirmed your result, you can try to implement the same thing with Patch().

```
from dash import Dash, html, Input, Output, Patch, dash_table, no_update
import plotly.express as px

app = Dash(__name__)

df = px.data.iris()
# dff = df.to_dict("records") # this is the data format accepted by the DataTable
# dff[0].update({'sepal_length': 99.1}) # update first cell of column `sepal_length` to 99.1
# print(dff) # print to see result
# exit()

app.layout = html.Div(
    [
        html.Button("Update first cell of first row only once", id="add-data-rows"),
        dash_table.DataTable(
            data=df.to_dict("records"),
            columns=[{"name": i, "id": i} for i in df.columns],
            page_size=10,
            id="df-table",
        ),
    ],
)

@app.callback(
    Output("df-table", "data"),
    Input("add-data-rows", "n_clicks"),
    prevent_initial_call=True,
```

1 Like

mc2pezwb 32 April 12, 2023, 1:46pm

Great, ok so basically, using iloc to get the list position. This is rather neat.
Thanks a lot for the quick answer !

hypervalent 33 April 14, 2023, 9:01pm

I just wanted to ask to clarify:

If I have a certain output that is shared by multiple callbacks, all I have to do is just add "allow_duplicate=True" to only one of them?

Does it matter which callback is given the tag?

And also does adding that tag to one of the callback sets that output to ensure duplicates are allowed for all other and future callbacks using that same output?

adamschroeder 34 April 15, 2023, 6:22pm

hi [@hypervalent](#)

if you have two callbacks that have the same output component property, only one of them needs an `allow_duplicate=True` with `prevent_initial_call=True`.

But if you add more callbacks with the same output component property, you will need to add the `allow_duplicate=True` and the `prevent_initial_call=True` to those callbacks as well.

See how in the example below the first and third callbacks have `allow_duplicate=True`. Try to remove that from the third callback and you will get an error.

```
from dash import Dash, Input, Output, html, dcc, Patch
import plotly.express as px
import plotly.graph_objects as go

app = Dash(__name__)

app.layout = html.Div([
    html.Button('Draw Graph', id='draw-2'),
    html.Button('Reset Graph', id='reset-2'),
    html.Button('Change Title', id='title-2'),
    dcc.Graph(id='duplicate-output-graph')
])

@app.callback(
    Output('duplicate-output-graph', 'figure', allow_duplicate=True),
    Input('draw-2', 'n_clicks'),
    prevent_initial_call=True
)
def draw_graph(n_clicks):
    df = px.data.iris()
    return px.scatter(df, x=df.columns[0], y=df.columns[1])

@app.callback(
    Output('duplicate-output-graph', 'figure'),
    Input('reset-2', 'n_clicks'),
)
def reset_graph(input):
    return go.Figure()
```

1 Like

hypervalent 35 April 16, 2023, 6:29amHello [@adamschroeder](#) ! Thank you for the quick response!Just to make sure then, in the case of, say, n callbacks with the same output. All callbacks except one of them (n-1 callbacks) require the `allow_duplicate=True` ?

And in the case of there being more than 2 callbacks with the same output, does it matter which callback does not get the `allow_duplicate=True` ? I know in your specific scenario it's because the second callback does not want `prevent_initial_call=True`, but in the case where all of the callbacks need `prevent_initial_call=True`, does it ultimately matter which one I remove the `allow_duplicate` tag?

Thank you!

AnnMarieW 36 April 16, 2023, 4:20pm

There are so many cool new features in 2.9.2, I think this is my favorite release to date.

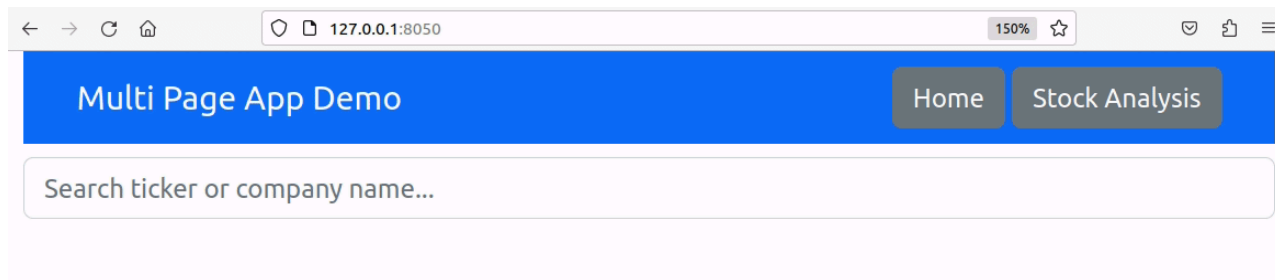
I just wanted to highlight a few posts with more great info on the new things now available.
(And be sure to update to 2.9.3 to get the patch with the minor bugs reported in 2.9.2)

Examples of multi-page app navigation without refreshing the page

Anyone making multi-page apps where you update the URL in a callback will find this new feature! Super fast navigation without the annoying page refresh. See more information and examples in this post:

Sharing examples of navigation without refreshing the page when URL is updated in a callback in Dash 2.9.2!

🎉 New in dash 2.9.2 It's now possible to update the URL in a callback to navigate to a new page of a multi-page app without refreshing the page 🤖 NEW – `dcc.Location(refresh="callback-nav")`
- No page refresh! [callback-nav] 🤖 OLD – `dcc.Location(refresh=True)` Note the screen flash: [\[image\]](#) To see these examples (and more!) check out: Background With Dash Pages, the rout...



Geolocation

When deploying an app with the the new `dcc.Geolocation` be sure to be using HTTPS, else Geolocation will not be enabled.

Here is another demo app showing more features available. This one also uses `geopy` to get the address (if the "Include Address" checkbox is selected). Try it out on a phone or some other device with GPS enabled. The accuracy is amazing

<https://dccgeolocation.pythonanywhere.com/>



4 Likes

Examples of partial update and allow duplicates

This post has some great examples from community members [@AIMPED](#) and [@jinyvzor](#) including examples with pattern matching callbacks.

Sharing examples of partial update & allow_duplicate=True from Dash 2.9.2!

Hi all, recently I started playing around with partial updates and the `allow_duplicate=True` parameter. For my app the benefit of using the new features is huge. I thought I share two MRE do demonstrate the capabilities. change the trace color of selected traces, uses partial updates and duplicated outputs add annotations to a image displayed with `px.imshow()`, here you should take a look on the data size traveling from client to server depending on the button you click. First example, change...

4 Likes

[adamschroeder](#) 37 April 18, 2023, 10:33am

hi [@hypervalent](#) ,

Just to make sure then, in the case of, say, n callbacks with the same output. All callbacks except one of them (n-1 callbacks) require the `allow_duplicate=True` ?

That is correct. They will also need the `prevent_initial_call=True` . Same output meaning the same `component_id` as well as the same `component_property` used.

Regarding your second question,

Does it matter which callback does not get the `allow_duplicate=True` ?

I don't think it matters that much, or at least I haven't encountered a case where I it made a difference. If you encounter that case, let me know.

1 Like

[hypervalent](#) 38 April 18, 2023, 4:20pm

Thank you so much! That pretty much answers all my questions


I will let you know if I encounter such a case

1 Like

[AIMPED](#) Split this topic 39 April 21, 2023, 11:15am

A post was split to a new topic: [Label Aliases wot working, plotly 5.14.1](#)

[Emil](#) 40 April 22, 2023, 5:07pm

This is an amazing release! So many great features 

1 Like

[jokin](#) 41 May 11, 2023, 7:42am

I see this is not fixed in 2.9.3, should we reopen a bug report?

Related topics

Topic	Replies	Activity
:mega: Dash 2.17.0 Released - Callback updates with set_props, No Output Callbacks, Layout as List, dcc.Loading, Trace Zorder announcements	11	June 20, 2024
Sharing examples of partial update & allow_duplicate=True from Dash 2.9.2! show-and-tell tips-and-tricks	6	May 25, 2023
:mega: Dash 2.4 Released - Improved Callback Context, Clientside Callback Promises, Typescript Components, Minor Ticks, and More! announcements	0	May 25, 2022
:mega: Dash 2.11.0 Released - Dash in Jupyter, Locked Flask versions, and dcc.Graph Updates	3	June 30, 2023
Duplicate Callback Outputs - Solution & API Discussion	22	March 16, 2023