



Star 23,446



Dash Python > Dash DataTable

Plotly Studio: Transform any dataset into an interactive data application in minutes with AI. [Sign up for early access now.](#)



Dash DataTable

Dash DataTable (`dash.dash_table.DataTable`) is an interactive table component designed for viewing, editing, and exploring large datasets.

This component was written from scratch in React.js specifically for the Dash community. Its API was designed to be ergonomic and its behavior is completely customizable through its properties. DataTable is rendered with standard, semantic HTML `<table/>` markup, which makes it accessible, responsive, and easy to style.

Import DataTable with:

```
from dash import dash_table
```

Tip: In production Dash apps, we recommend using DataTable with **Python data pipelines** for ingesting the table data and **Design Kit** for DataTable styling.

For examples of minimal Dash apps that use the `dash_table`, go to the community-driven **Example Index**.

Quickstart

Dash open-source

Dash Enterprise Design Kit

```
from dash import Dash, dash_table
import pandas as pd


df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/solar.csv')

app = Dash()

app.layout = dash_table.DataTable(df.to_dict('records'), [{"name": i, "id": i} for i in df.columns])

if __name__ == '__main__':
    app.run(debug=True)
```

State	Number of Solar Plants	Installed Capacity (MW)	Average MW Per Plant	Generation (GWh)
California	289	4395	15.3	10826
Arizona	48	1078	22.5	2550
Nevada	11	238	21.6	557
New Mexico	33	261	7.9	590
Colorado	20	118	5.9	235
Texas	12	187	15.6	354
North Carolina	148	669	4.5	1162
New York	13	53	4.1	84



https://dash.plotly.com/datatable

1/4

The `data` and `columns` properties are the first two arguments of `dash_table.DataTable`. You can set these with **positional or keyword arguments**. The following examples define the same `DataTable`:

```
dash_table.DataTable(  
    df.to_dict('records'),  
    [{"name": i, "id": i} for i in df.columns]  
)
```

```
dash_table.DataTable(  
    data=df.to_dict('records'),  
    columns=[{"name": i, "id": i} for i in df.columns]  
)
```

You can also omit `columns`. When no `columns` argument is provided, columns are auto-generated based on the first row in `data`.

```
dash_table.DataTable(df.to_dict('records'))
```

Table with Click Callback

[run on repl.it](#)[Open in Colab](#)[Databricks](#)

```
from dash import Dash, Input, Output, callback, dash_table  
import pandas as pd  
import dash_bootstrap_components as dbc  
  
df = pd.read_csv('https://git.io/Juf1t')  
  
app = Dash(external_stylesheets=[dbc.themes.BOOTSTRAP])  
  
app.layout = dbc.Container([  
    dbc.Label('Click a cell in the table:'),  
    dash_table.DataTable(df.to_dict('records'), [{"name": i, "id": i} for i in df.columns], id=  
    dbc.Alert(id='tbl_out'),  
)  
  
@callback(Output('tbl_out', 'children'), Input('tbl', 'active_cell'))  
def update_graphs(active_cell):  
    return str(active_cell) if active_cell else "Click the table"  
  
if __name__ == "__main__":  
    app.run(debug=True)
```

Click a cell in the table:

State	Number of Solar Plants	Installed Capacity (MW)	Average MW Per Plant	Gene
California	289	4395	15.3	
Arizona	48	1078	22.5	
Nevada	11	238	21.6	
New Mexico	33	261	7.9	
Colorado	20	118	5.9	
Texas	12	187	15.6	
North Carolina	148	669	4.5	
New York	13	53	4.1	

Click the table



Sign up for Dash Club → Two free cheat sheets plus updates from Chris Parmer and Adam Schroeder delivered to your inbox every two months. Includes tips and tricks, community apps, and deep dives into the Dash architecture. [Join now.](#)

User Guide

Reference

A comprehensive list of all of the DataTable properties.

DataTable Height

How to set the height of the DataTable. Examples include how to set the height with vertical scroll, pagination, virtualization, and fixed headers.

DataTable Width & Column Width

How to set the width of the table and the columns. Examples include how to handle word wrapping, cell clipping, horizontal scroll, fixed columns, and more.

Styling

The style of the DataTable is highly customizable. This chapter includes examples for:

- Displaying multiple rows of headers
- Text alignment
- Styling the table as a list view
- Changing the colors (including a dark theme!)

Conditional Formatting

Several examples of how to highlight certain cells, rows, or columns based on their value or state.

Number Formatting

Several examples of how to format and localize numbers.

Sorting, Filtering, Selecting, and Paging Natively

The DataTable is interactive. This chapter demonstrates the interactive features of the table and how to wire up these interactions to Python callbacks. These actions include:

- Paging
- Selecting Rows
- Sorting Columns
- Filtering Data

DataTable Tooltips

Display tooltips on data and header rows, conditional tooltips, define tooltips for each cell, customize behavior.

Python-Driven Filtering, Paging, Sorting

In Part 3, the paging, sorting, and filtering was done entirely clientside (in the browser). This means that you need to load all of the data into the table up-front. If your data is large, then this can be prohibitively slow. In this chapter, you'll learn how to write your own filtering, sorting, and paging backends in Python with Dash. We'll do the data processing with Pandas but you could write your own routines with SQL or even generate the data on the fly!

Editable DataTable

The DataTable is editable. Like a spreadsheet, it can be used as an input for controlling models with a variable number of inputs. This chapter includes recipes for:

- Determining which cell has changed
- Filtering out null values



- Adding or removing columns
- Adding or removing rows
- Ensuring that a minimum set of rows are visible
- Running Python computations on certain columns or cells

Typing and User Input Processing

In this chapter, you'll learn how to configure the table to

- assign the column type
- change the data presentation
- change the data formatting
- validate or coerce user data input
- apply default behavior for valid and invalid data

Dropdowns Inside DataTable

Cells can be rendered as editable Dropdowns. This is our first stake in bringing a full typing system to the table. Rendering cells as dropdowns introduces some complexity in the markup and so there are a few limitations that you should be aware of.

Virtualization

Examples using DataTable virtualization.

Filtering Syntax

An explanation and examples of filtering syntax for both frontend and backend filtering in the DataTable.

Dash Python > **Dash DataTable**

Products

Dash
Consulting and Training

Pricing

Enterprise Pricing

About Us

Careers
Resources
Blog

Support

Community Support
Graphing Documentation

Join our mailing

list

Sign up to stay in the loop with all things Plotly — from Dash Club to product updates, webinars, and more!

SUBSCRIBE

Copyright © 2025 Plotly. All rights reserved.

[Terms of Service](#) [Privacy Policy](#)