Visualization seminar

02.05.2023, recording.

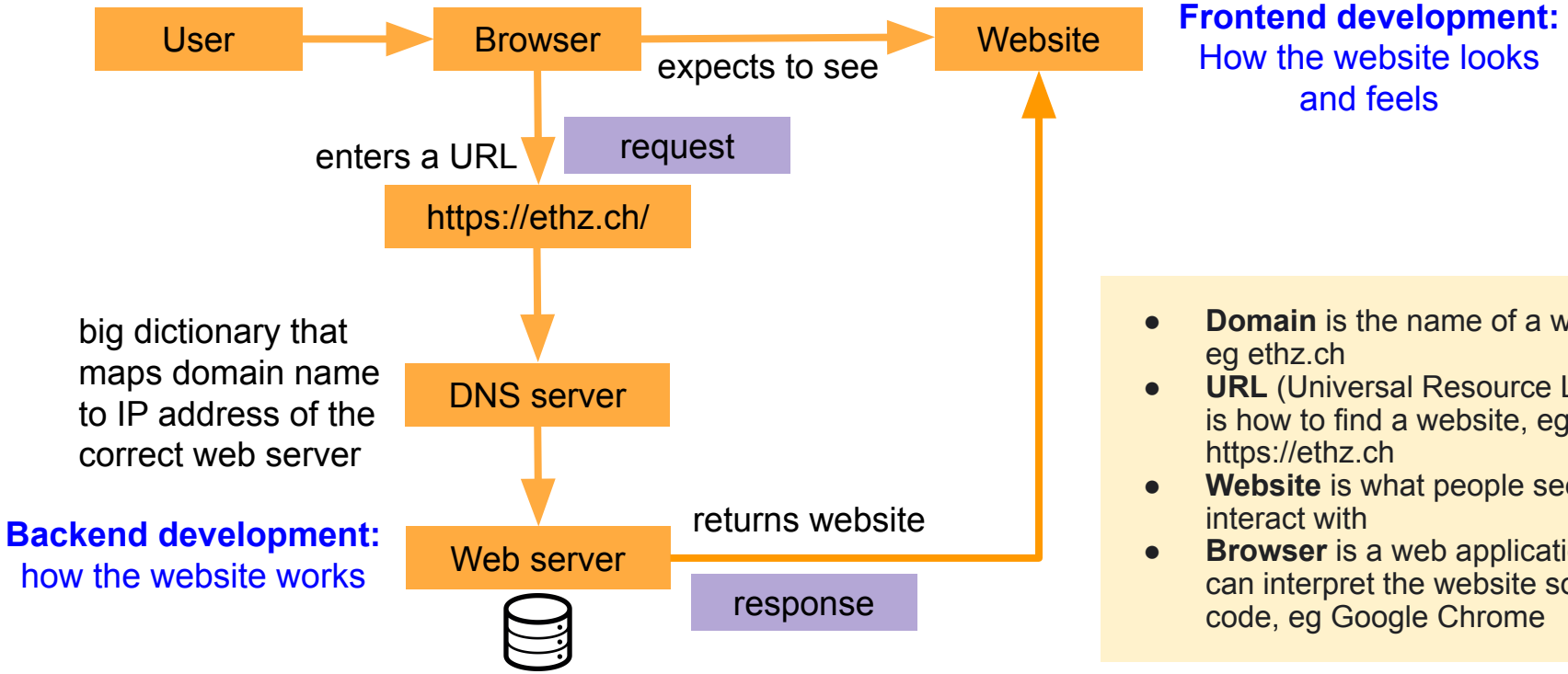
Contents

Part 1
Introduction to visualization dashboards with Dash

Goal: give you a high level overview so that you know what to google.

How the Web works

https://www.youtube.com/watch?v=hJHvdBlSxug&ab_channel=Academind



- **Domain** is the name of a website, eg ethz.ch
- **URL** (Universal Resource Locator) is how to find a website, eg https://ethz.ch
- **Website** is what people see and interact with
- **Browser** is a web application that can interpret the website source code, eg Google Chrome

Frontend development

Responsible for how the website **looks** and **feels**.

Tools:

- HTML
  - HyperText Markup Language
  - **Structure / layout** of the website
  - Where do we place headers, tabs, buttons
- CSS
  - Cascading Style Sheets
  - **Styling** your website
  - Fonts, colors, thickness of lines, shadows, etc
- Javascript
  - Programming language for the web
  - Can update and change both HTML and CSS, adds **logic** to a website
  - What happens when we open a dropdown list?

HTML, CSS, Javascript implement website design.

Frontend

HTML

CSS

CSS Frameworks		
Bootstrap	Tailwind CSS	Bulma
Foundation	Primer CSS	Spectre CSS
Materialize CSS	Onsen UI	Semantic UI
Blaze UI	Pure CSS	Tachyons

CSS Preprocessors
SASS
LESS

JavaScript

ES6

Typescript

Package Manager
npm
yarn

JS Framework

Angular JS
Angular ngx Bootstrap
Angular PrimeNG
Angular Material UI

Vue JS
Nuxt.js

script.aculo.us

JS Libraries

Jquery
jQuery UI
jQuery Mobile
jQWidgets
jQuery EasyUI

React JS
Next.js
Ant Design
React Desktop
React Rebass
React Bootstrap
React Reactstrap
Blueprint.js
React Suite
React.js Evergreen

P5.js
Fabric.js
D3.js
Collect.js
Underscore.js
Lodash
TensorFlow.js

Web components

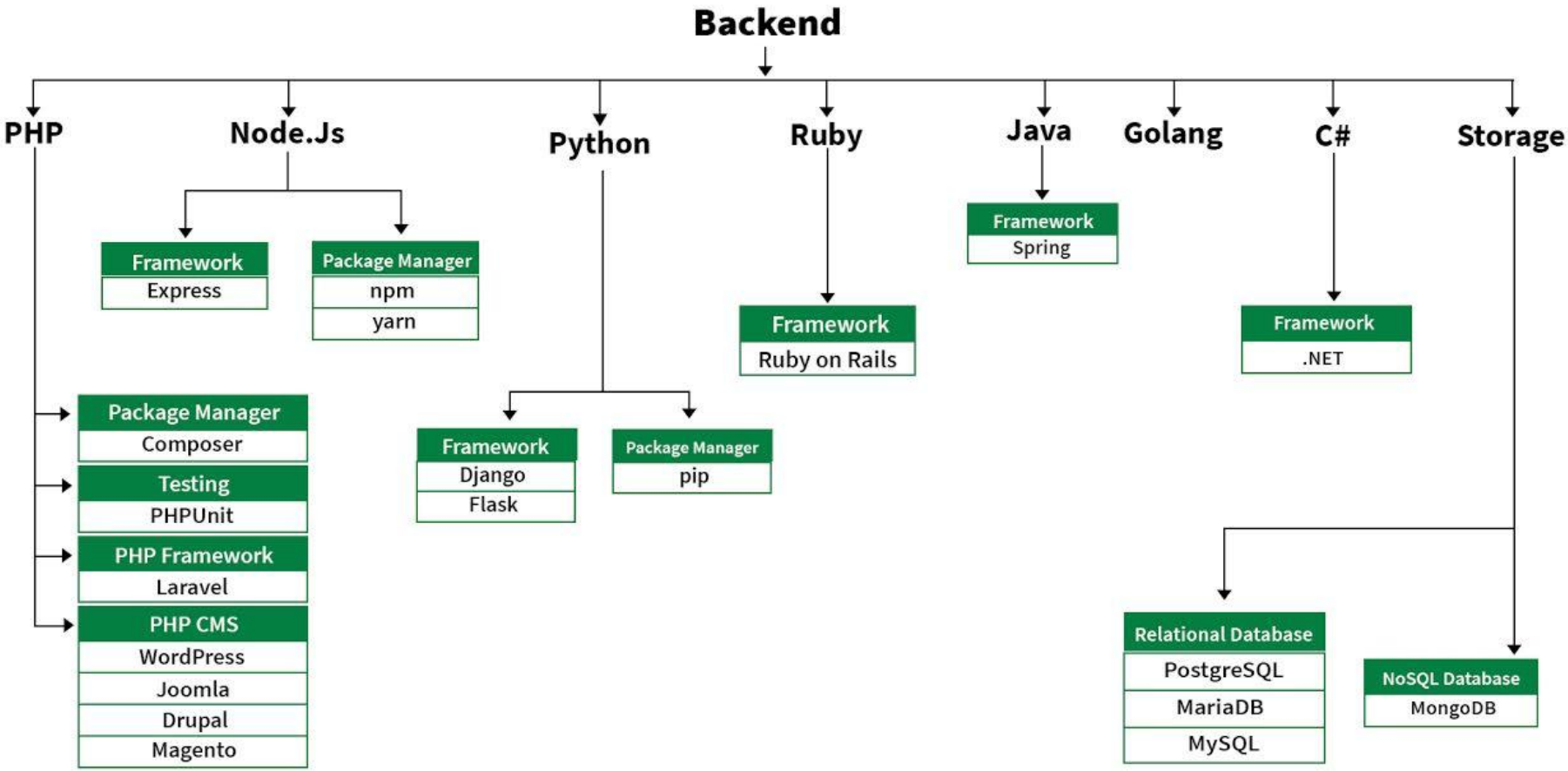
- **Components** are sets of specifications that add functionalities and features to web pages and applications.
- Examples are buttons, dropdown menus, navigation bars, etc.
- Many modern apps are built using components.
- Enable building **functions** that can be **reused**.
- Allow more **consistent** user experience.

Backend development

Responsible for:

- Interactions with a database
- User authentification
- Server and network configurations
- Website responsiveness and speed

Tools: PHP, Node.js, Python, Ruby, Java, SQL, …


Backend
PHP
Node.Js
Python
Ruby
Java
Golang
C#
Storage
Framework
Express
Package Manager
npm
yarn
Framework
Spring
Framework
Ruby on Rails
Framework
.NET
Package Manager
Composer
Testing
PHPUnit
PHP Framework
Laravel
PHP CMS
WordPress
Joomla
Drupal
Magento
Framework
Django
Flask
Package Manager
pip
Relational Database
PostgreSQL
MariaDB
MySQL
NoSQL Database
MongoDB

Frontend developers

Backend developers

Website designers

Full stack developers

ESD members

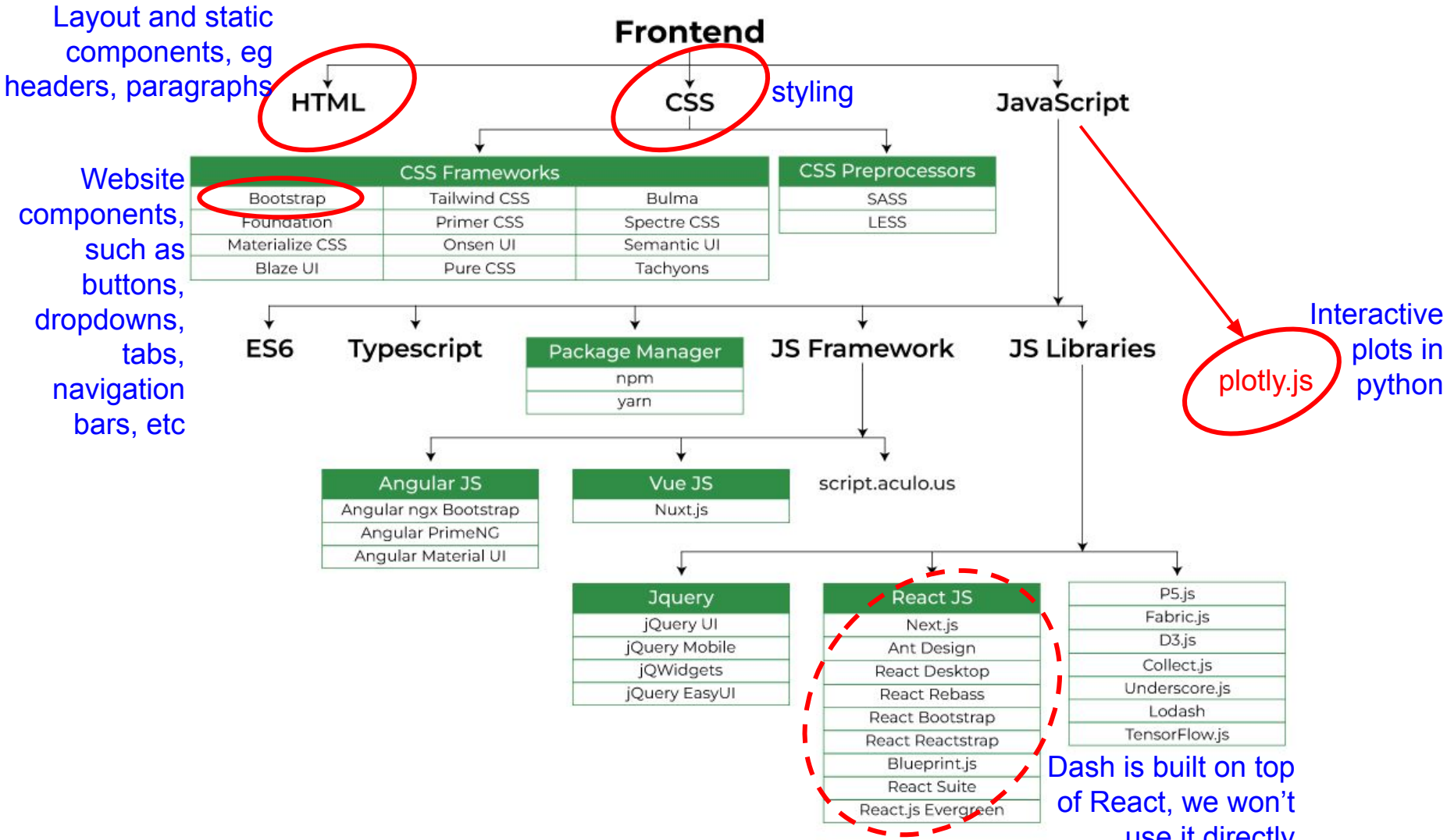
Dash

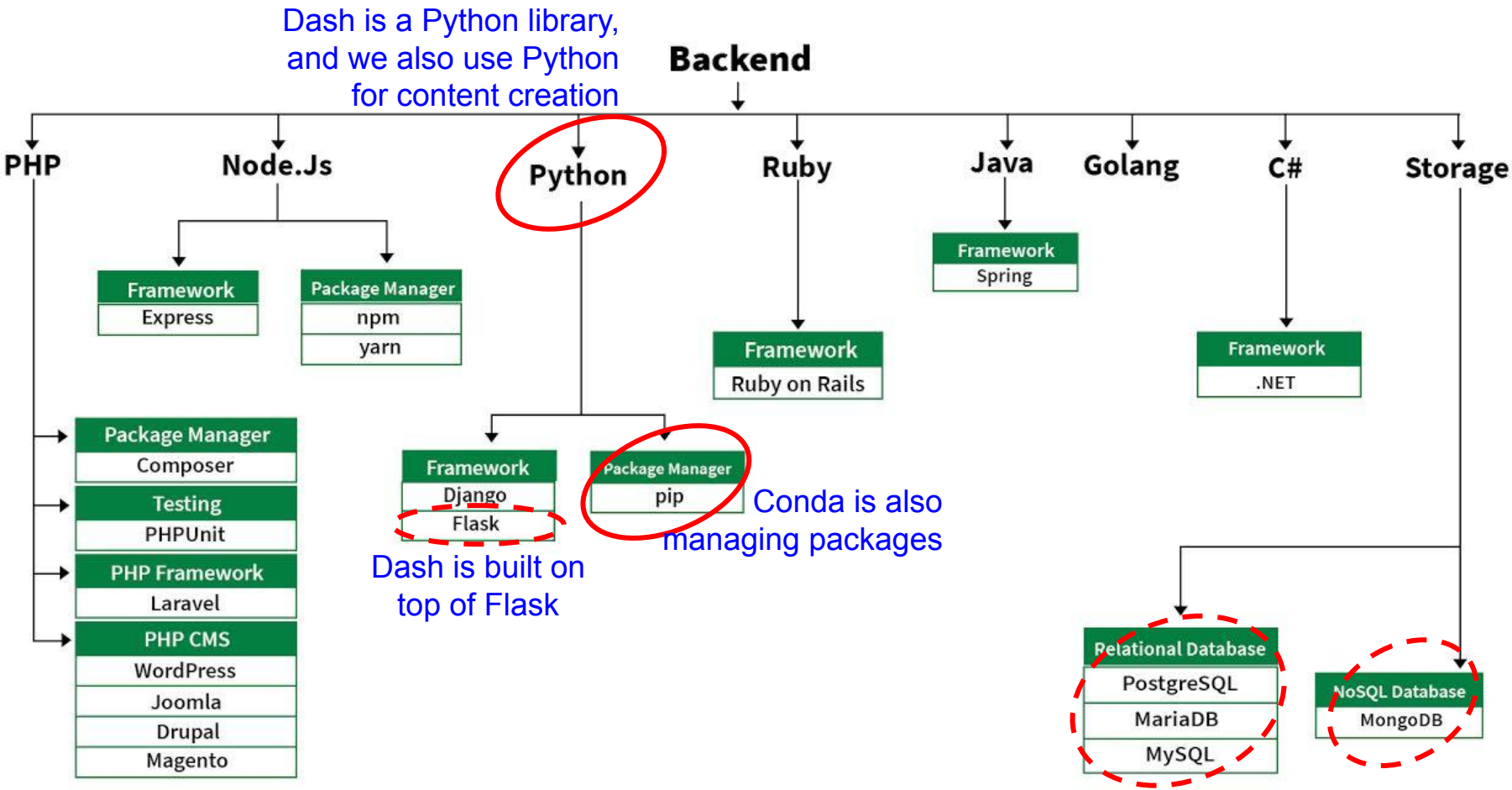
Most downloaded, trusted Python framework for building ML & data science web apps.

https://dash.plotly.com/

About Dash

- Framework to build **visualization interfaces**.
- Was released in 2017 as a Python library, now includes implementations for R, Julia and F#.
- **3 technologies** constitute the core of Dash:
  - **Flask** for web server functionality
  - **React.Js** renders user interface of the web page
  - **Plotly.Js** generates charts and plots
- Contains many web components.
- Has open source and enterprise versions. For us, open source is enough.
- R has smth similar: https://shiny.rstudio.com/


Layout and static components, eg headers, paragraphs
Frontend
HTML
CSS
styling
JavaScript
Website components, such as buttons, dropdowns, tabs, navigation bars, etc
CSS Frameworks
Bootstrap
Tailwind CSS
Bulma
Foundation
Primer CSS
Spectre CSS
Materialize CSS
Onsen UI
Semantic UI
Blaze UI
Pure CSS
Tachyons
CSS Preprocessors
SASS
LESS
ES6
Typescript
Package Manager
npm
yarn
JS Framework
JS Libraries
Interactive plots in python
plotly.js
Angular JS
Angular ngx Bootstrap
Angular PrimeNG
Angular Material UI
Vue JS
Nuxt.js
script.aculo.us
Jquery
jQuery UI
jQuery Mobile
jQWidgets
jQuery EasyUI
React JS
Next.js
Ant Design
React Desktop
React Rebass
React Bootstrap
React Reactstrap
Blueprint.js
React Suite
React.js Evergreen
P5.js
Fabric.js
D3.js
Collect.js
Underscore.js
Lodash
TensorFlow.js
Dash is built on top of React, we won't use it directly


Dash is a Python library, and we also use Python for content creation
Backend
PHP
Node.Js
Python
Ruby
Java
Golang
C#
Storage
Framework
Express
Package Manager
npm
yarn
Framework
Spring
Framework
Ruby on Rails
Framework
.NET
Package Manager
Composer
Testing
PHPUnit
PHP Framework
Laravel
PHP CMS
WordPress
Joomla
Drupal
Magento
Framework
Django
Flask
Package Manager
pip
Conda is also managing packages
Dash is built on top of Flask
Relational Database
PostgreSQL
MariaDB
MySQL
NoSQL Database
MongoDB
Should include, but we don't consider databases in this seminar

Steps for website creation

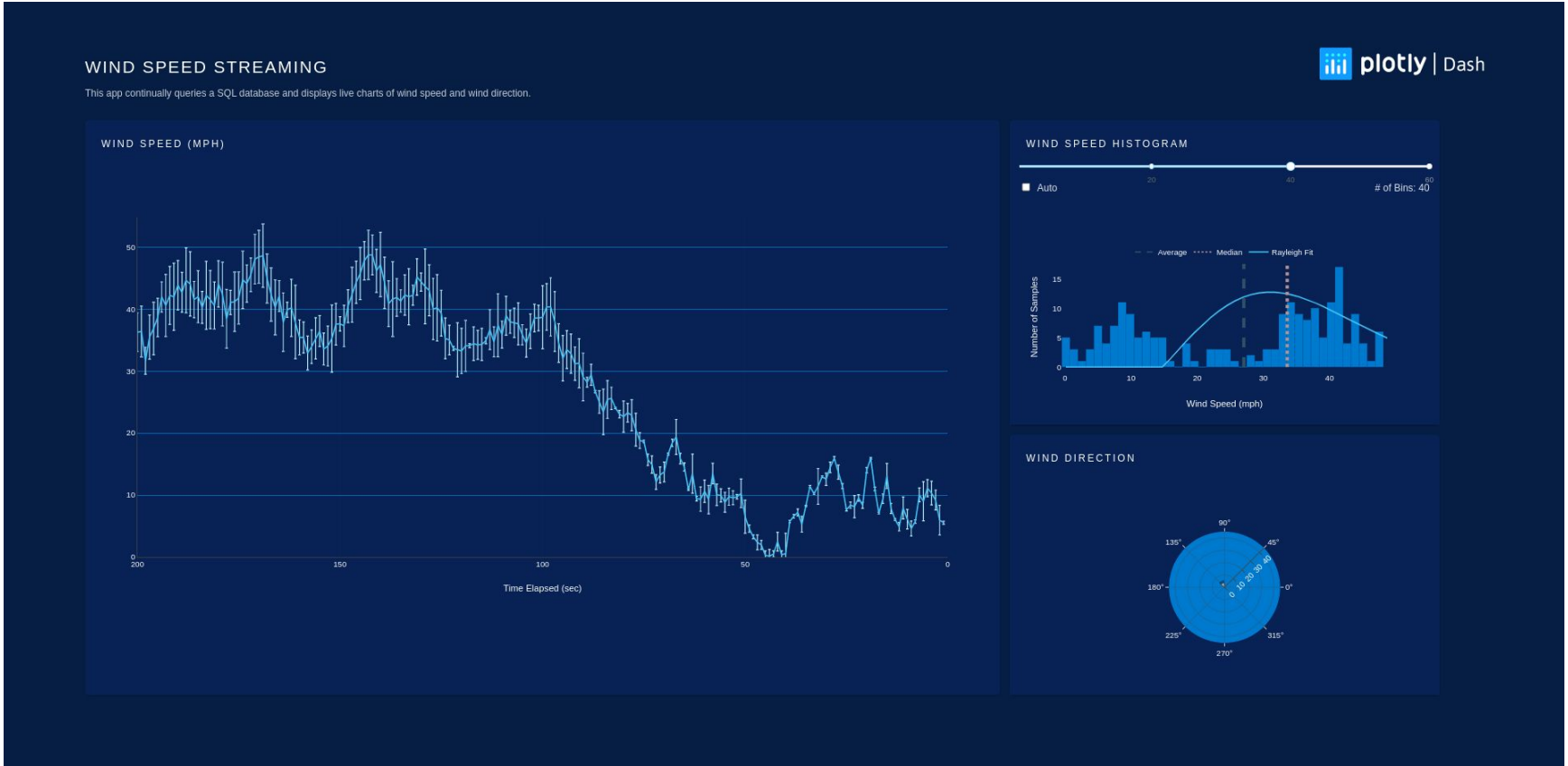
1. Create website content
   a. You are doing it on a daily basis!
2. Design and build your website
   a. For example, using Dash.
3. Publish your website
   a. Register domain name.
   b. Choose web hosting provider. The web host provides server space for the website's files. Can also be own server.
   c. Disclaimer: it is possible to publish with Github for free!

Dash Gallery

https://dash.gallery/Portal/

https://dash.gallery/dash-wind-streaming/

Streaming real-time data



https://dash.gallery/american-commute/

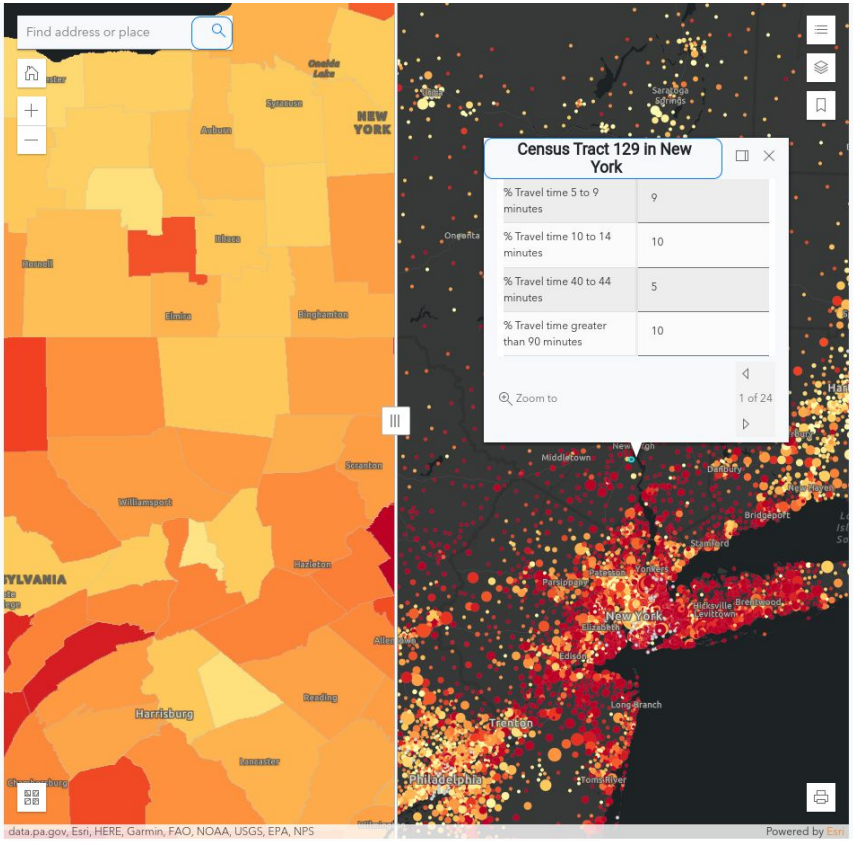
Maps

USA TRAVEL TIME TO WORK ANALYSIS

—

Overview

Detailed view



Go to a specific county

Features shown in the dropdown are the currently on screen counties. Zoom in to detect counties.

Select a county...

Currently visible counties

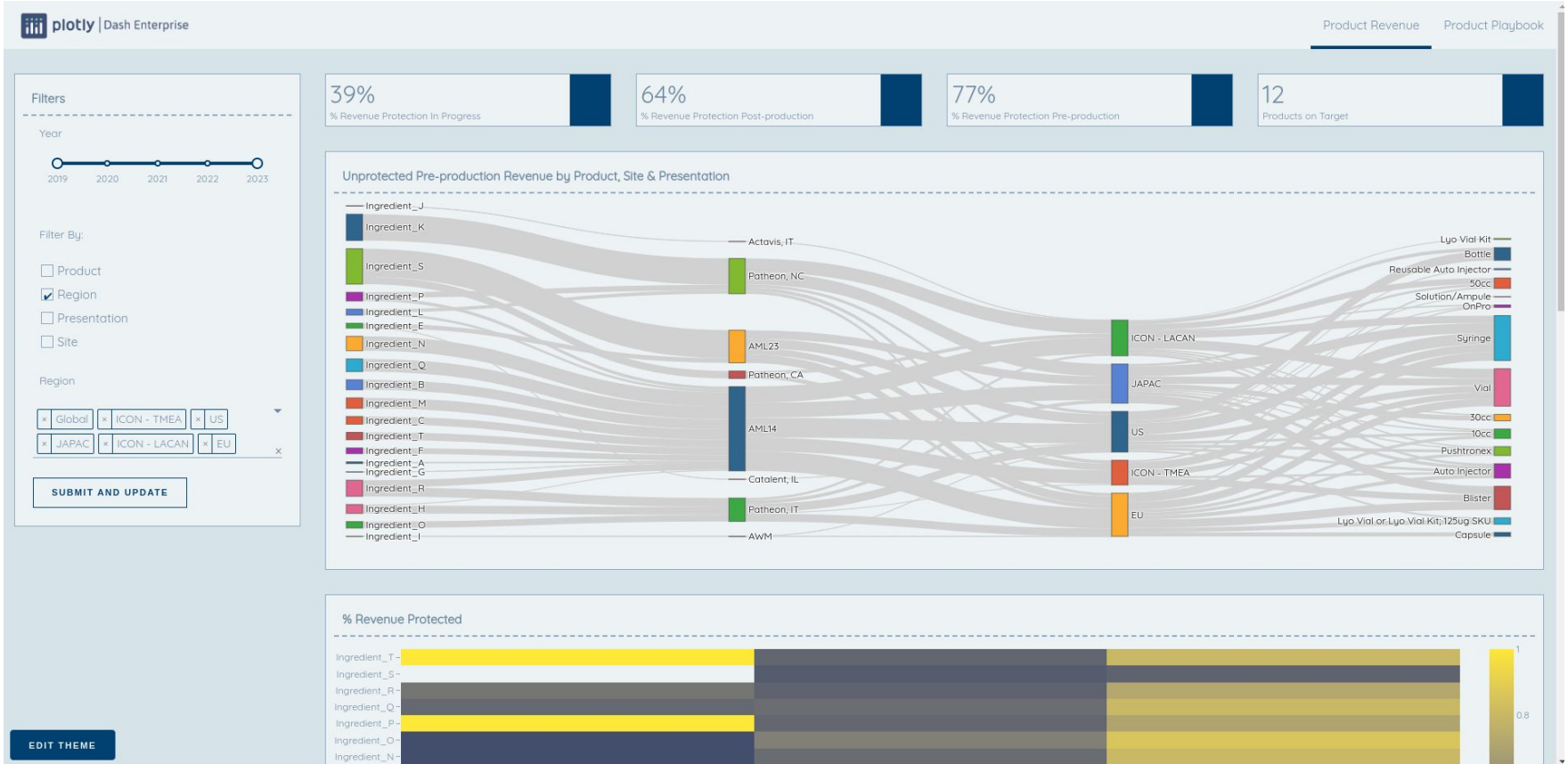
State	GEOID	Census Tract	% Workers whose travel time 5 to 9 minute	%
filter data...	filter d...	filter data...	filter data...	
New Jersey	34035	Somerset County	6.9	
New York	36005	Bronx County	2.8	
New York	36047	Kings County	2.5	
New York	36059	Nassau County	6.3	
New York	36061	New York County	4.2	
New York	36081	Queens County	2.4	
New York	36085	Richmond County	3.9	
New York	36103	Suffolk County	7.8	
New Jersey	34001	Atlantic County	10.4	
New Jersey	34033	Salem County	9.7	

« ‹ 8 / 13 › »

EDIT THEME

https://dash.gallery/product-playbook/

Sankey diagrams



- Check out existing examples to see what is **definitely** possible with dash.
- Sometimes their code is open source.

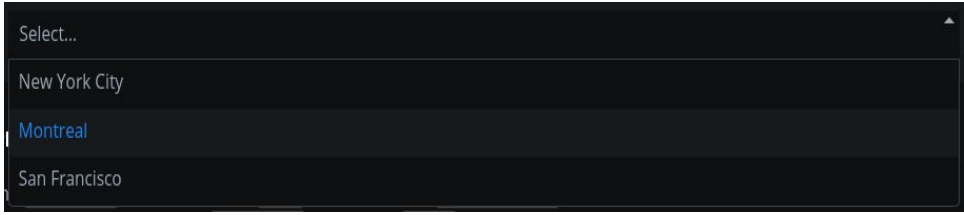
Dash Core Components

https://dash.plotly.com/dash-core-components

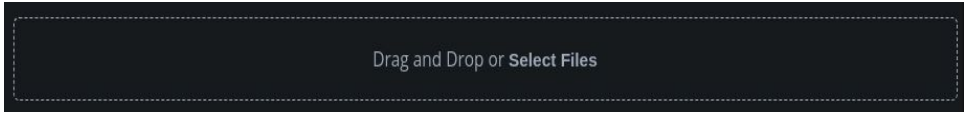
All core components

- Checklist
- Clipboard
- ConfirmDialog
- ConfirmDialogProvider
- DatePickerRange
- DatePickerSingle
- Download
- Dropdown
- Graph
- Geolocation
- Input
- Interval
- Link
- Loading
- Location
- LogoutButton
- Markdown
- RadioItems
- RangeSlider
- Slider
- Store
- Tab
- Tabs
- Textarea
- Tooltip
- Upload

Dropdown



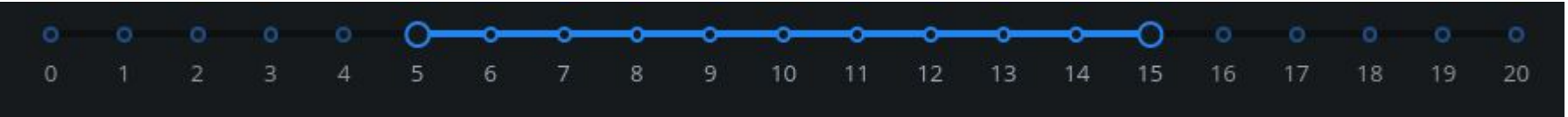
Upload



Checklist

Confirm Dialog

Range Slider



Dash datatables

- Work with pandas dataframes
- Interactive
- Multiple pages
- Adjust heights
- Style cells separately and link cell colors to a legend
- Export tables as excel / csv
- Filter and sort data

Date	Region	Temperature	Humidity	Pressure
2015-01-01	Montreal	1	10	2
2015-10-24	Toronto	-20	20	10924
2016-05-10	New York City	3.512	30	3912
2017-01-10	Miami	4	40	-10
2018-05-10	San Francisco	10423	50	3591.2
2018-08-15	London	-441.2	60	15
2015-01-01	Montreal	1	10	2
2015-10-24	Toronto	-20	20	10924
2016-05-10	New York City	3.512	30	3912
2017-01-10	Miami	4	40	-10

« ‹ 1 / 6 › »

-10 -6.6 -3.2 0.2 3.6

Firm	2017	2018	2019	2020
Acme	13	5	10	4
Olive	3	3	13	3
Barnwood	6	7	3	6
Henrietta	-3	-10	-5	-6

Date	Region	Temperature	Humidity	Pressure	Rating	Growth	Status
2015-01-01	Montreal	1	10	2		↗️	🔥
2015-10-24	Toronto	-20	20	10924	⭐	↘️	🚒
2016-05-10	New York City	3.512	30	3912	⭐⭐	↗️	🔥
2017-01-10	Miami	4	40	-10	⭐⭐⭐	↗️	🔥
2018-05-10	San Francisco	10423	50	3591.2	⭐⭐⭐	↗️	🔥
2018-08-15	London	-441.2	60	15	⭐⭐⭐	↘️	🚒

Bootstrap Components

https://dash-bootstrap-components.opensource.faculty.ai/docs/components/

All bootstrap components

- Accordion
- Alert
- Badge
- Breadcrumb
- Button
- ButtonGroup
- Card
- Carousel
- Collapse
- DropdownMenu
- Fade
- Form
- Input
- InputGroup
- Jumbotron
- Layout
- ListGroup
- Modal
- Nav
- Navbar
- Offcanvas
- Pagination
- Placeholder
- Popover
- Progress
- Spinner
- Table
- Tabs
- Toast
- Tooltip

Buttons

Primary Secondary Success Warning Danger Info Light Dark Link

Spinners

Loading...

Card

Card title

Some quick example text to build on the card title and make up the bulk of the card's content.

Go somewhere

Accordion

Item 1

Item 2

This is the content of the second section

Item 3

Progress

71 %

Exploring bootstrap themes

https://dash-bootstrap-components.opensource.faculty.ai/docs/themes/explorer/

- Can take certain theme as a base, and tweak style of some components.
- Switching between themes is very very easy.
- Switching between core and bootstrap components is also easy, but not instantaneous.

Dash and Plotly graphs

https://plotly.com/python/

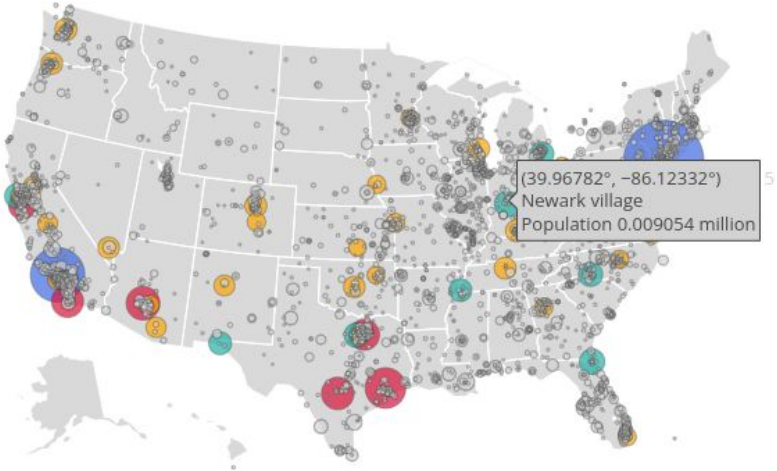
What is so special about Plotly?

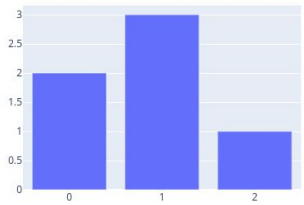
- Unlike matplotlib, it has logic :)
- So once you know it, it is easier to make plots, in my opinion.
- It is interactive: can zoom in, rotate 3D plots, see values of data points.
- You can save plots as interactive html files and share them.
- Integrating plotly charts into Dash is very easy.

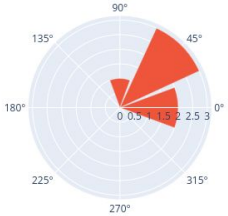
However, it does take time to learn plotly.

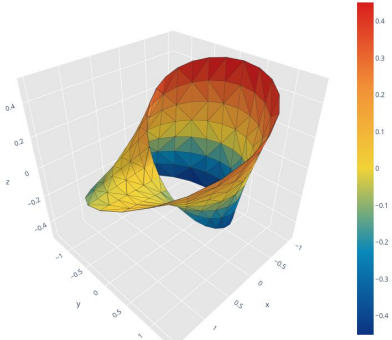
All plotly charts

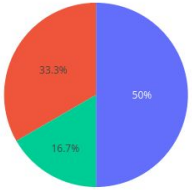
- Basic charts
  - Scatter plots
  - Line charts
  - Bar charts
  - Pie charts
  - Bubble charts
- Statistical charts
  - Error bars
  - Boxplots
  - Histograms
  - Distributions
- Maps
- Scientific charts
  - Contour plots
  - Heatmaps
  - Log plots
  - Imshow (imaginary axis)
- AI and ML
  - Classification
  - PCA
- 3D charts
- Subplots
- Animations

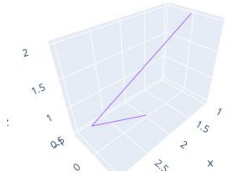










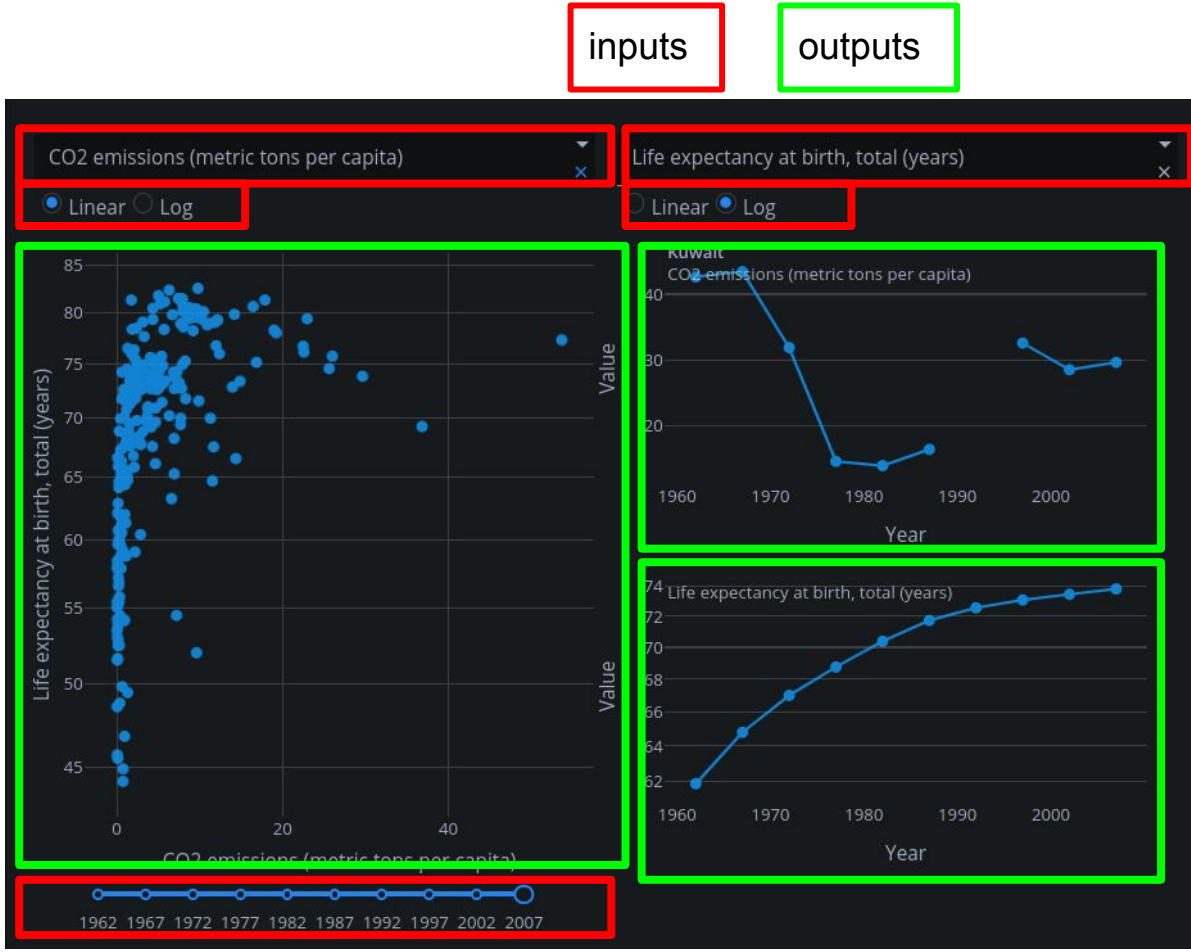


Dash Callbacks

https://dash.plotly.com/basic-callbacks

Callback functions

- Functions that are automatically called by Dash whenever an **input** component's property changes, in order to update some property in another component (the **output**).
- Needed to connect control components to eg charts.



Visualizing maps

https://plotly.com/python/maps/

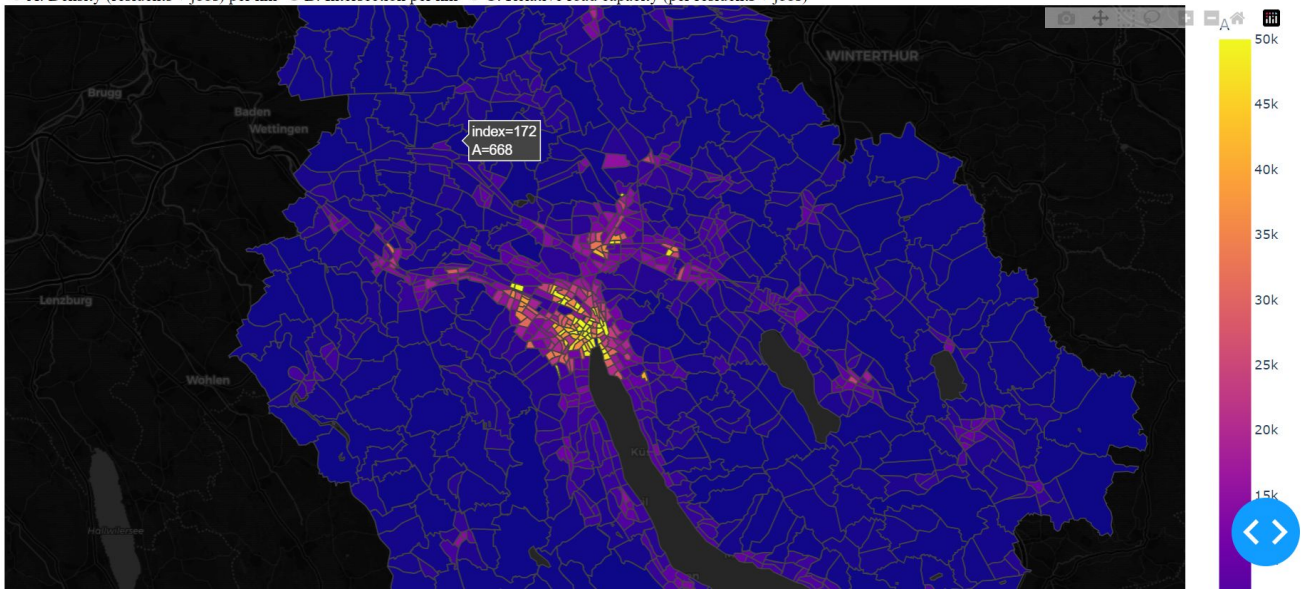
Dash + Plotly + Geospatial information

Making an interactive map in 1 hour!

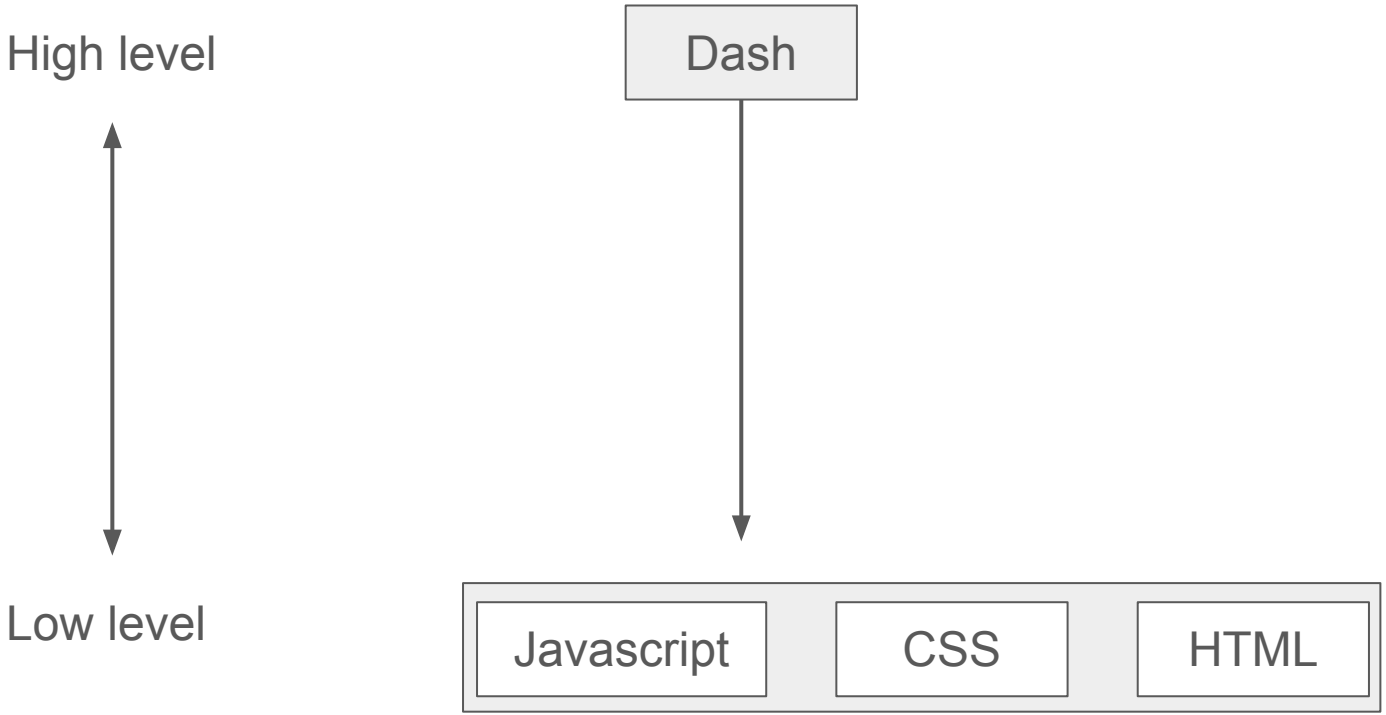
Examples of urban form metrics by traffic zone (Verkehrszone) in Zurich

Select a metric to show:

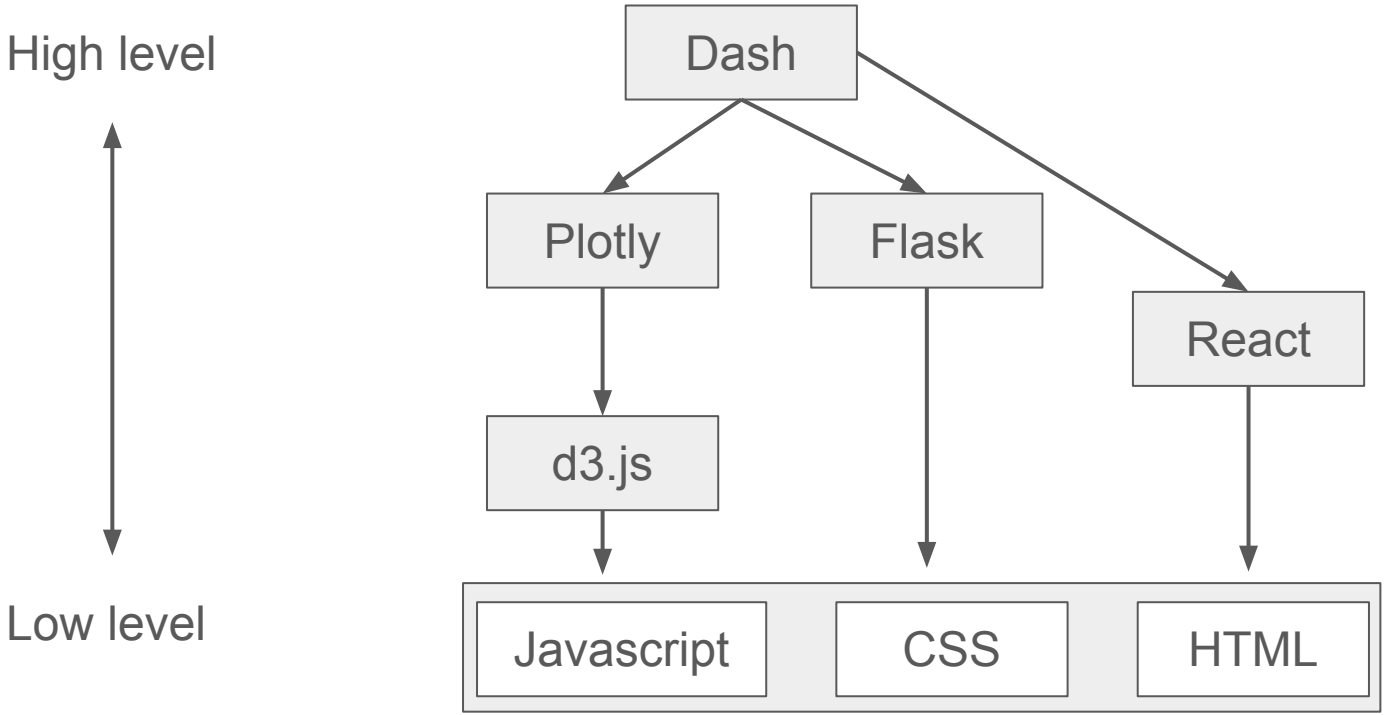
◉ A: Density (residents + jobs) per km² ○ B: Intersection per km² ○ C: Relative road capacity (per residents + jobs)



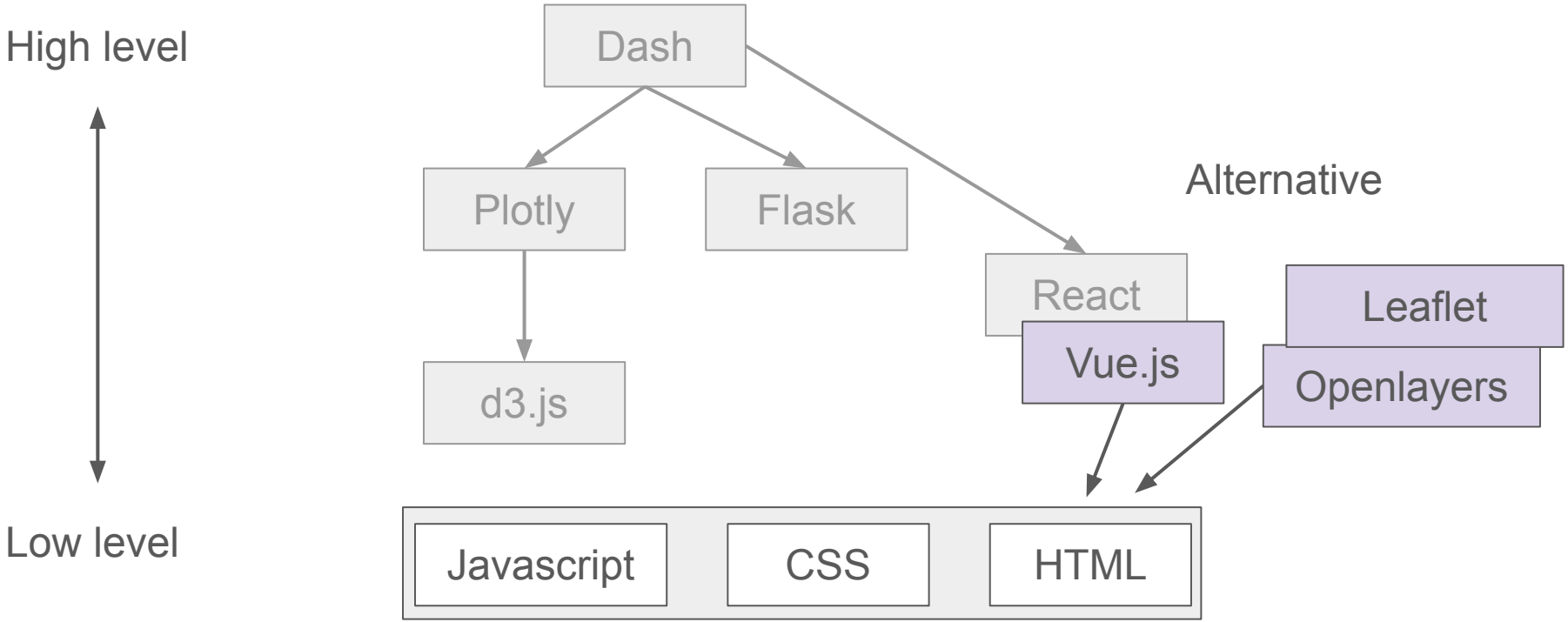
High level vs. low level for maps



High level vs. low level for maps



High level vs. low level for maps



High level vs. low level for maps

Time to get things done

High level

Low level

High level vs. low level for maps

	Time to get things done	Debugging and “safety”	Speed	Customization potential
High level	+ +	+	-	~
Low level	- -	-	+	+ +

The benefits and caveats of interactive visualizations

- Engaging: visitors/users might take away more from it than a from a static plot
- Multifaceted: can present a lot more information at once
- Good for validation: allow to scan data for consistency more effectively

The benefits and caveats of interactive visualizations

- \+ Engaging: visitors/users might take away more from it than a from a static plot
- \+ Multifaceted: can present a lot more information at once
- \+ Good for validation: allow to scan data for consistency more effectively

- \- Inherently mutable: not ideal (often not allowed) for “final” research output
- \- Misuable: especially if users are allowed to change input parameters

  → Can make it more likely that messages not consistent with research findings are attributed to authors/institution e.g. for political reasons

Opinion **The Big Read**

Electric cars' green image blackens beneath the bonnet

Research into the lifecycle of electric vehicles is a wake-up call for an industry geared up to promote 'zero emission cars'

PATRICK MCGEE + Add to myFT

Please be back at 11:10

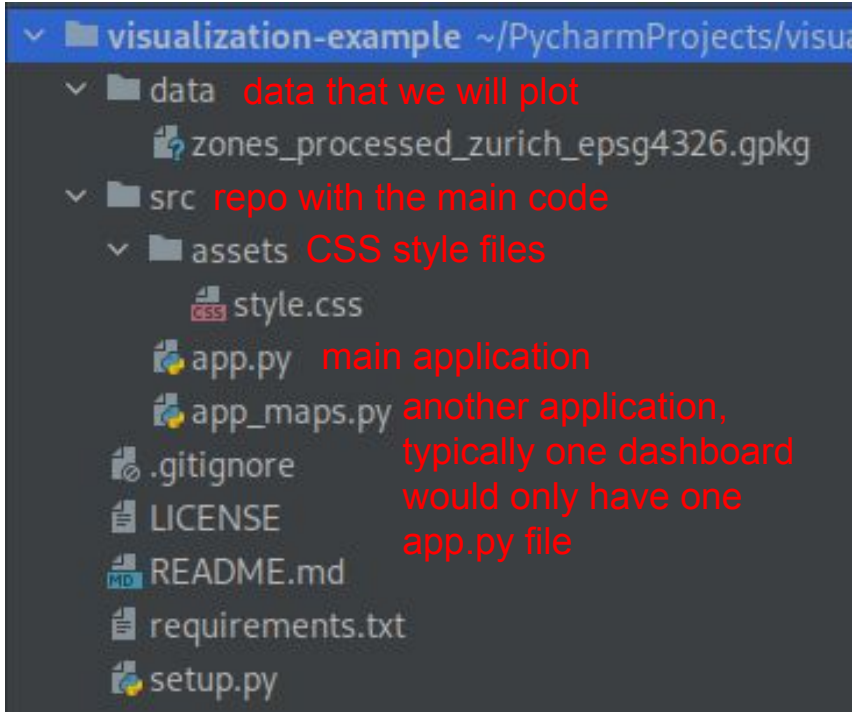
Part 2
Hands on visualization example with Dash

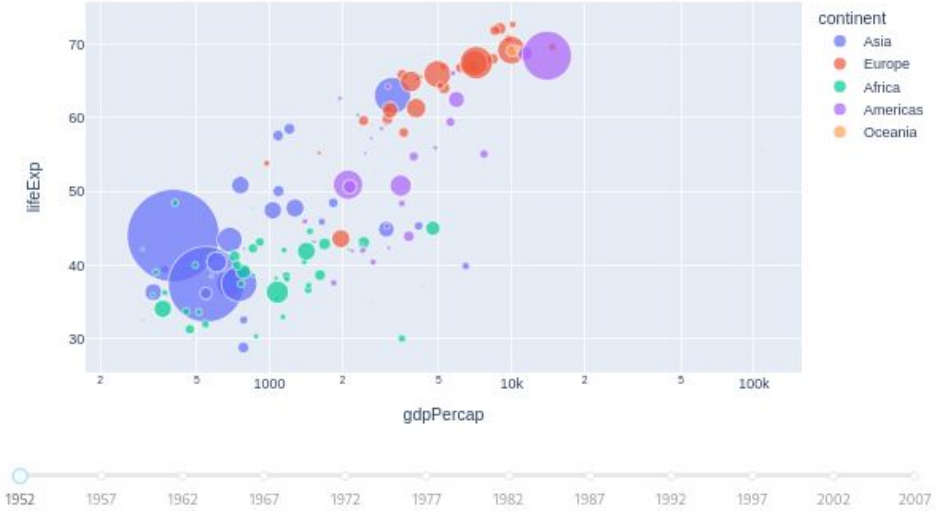
Goal: provide practical suggestions, and set up tools to get you started.

Visualization example explained

Visualization example

https://github.com/ecological-systems-design/visualization-example





app.py

- from **dash** import:
  - Dash - defines the application
  - dcc - contains dash core components
  - html - dash interface to html (static) components, such as headers
  - Input and Output - needed in callback functions
- **dash_bootstrap_components** - contains bootstrap components
- **plotly**.express - one of the modules to make plotly charts
- **pandas** - library for working with tables

Debug mode: change the code while running the app, and your dashboard would be updated on the fly.

app.py × style.css ×

```python
from dash import Dash, dcc, html, Input, Output
import dash_bootstrap_components as dbc
import plotly.express as px
import pandas as pd

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

app = Dash(
    __name__,
    external_stylesheets=[dbc.themes.PULSE]
)

app.layout = html.Div([
    dcc.Graph(id='graph-with-slider'),
    dcc.Slider(
        df['year'].min(),
        df['year'].max(),
        step=None,
        value=df['year'].min(),
        marks={str(year): str(year) for year in df['year'].unique()},
        id='year-slider'
    )
], style={"width": "800px", "height": "800px"})


@app.callback(
    Output('graph-with-slider', 'figure'),
    Input('year-slider', 'value'))
def update_figure(selected_year):
    filtered_df = df[df.year == selected_year]

    fig = px.scatter(filtered_df, x="gdpPercap", y="lifeExp",
                     size="pop", color="continent", hover_name="country",
                     log_x=True, size_max=55)

    fig.update_layout(transition_duration=500)

    return fig


# Run the app
if __name__ == '__main__':
    app.run_server(debug=True, port=8051)
```

read the data

initialize the app

bootstrap theme

layout that consists of 2 components

common style of the 2 components

decorator (starts with @) for the callback function that has input and output

figure is updated based on the slider values

running the app in the debug mode

Keyword arguments

app.py × style.css ×

```python
app.layout = html.Div([
    html.H1('Visualization example', className="my-header"),
    dcc.Graph(id='graph-with-slider'),
    dcc.Slider(
        df['year'].min(),
        df['year'].max(),
        step=None,
        value=df['year'].min(),
        marks={str(year): str(year) for year in df['year'].unique()},
        id='year-slider'
    )
], style={"width": "800px", "height": "800px"})
```

- **className** - needed for modifying component's style in the CSS file (see next slide)
- **style** - accepts a dictionary with CSS properties to modify component's style
- **id** - you can define it for each component, useful in callback functions
- There exist many other keyword arguments

Updating style

app.py × style.css ×

Option 1

```python
app.layout = html.Div([
    html.H1('Visualization example', className="my-header"),
    dcc.Graph(id='graph-with-slider'),
    dcc.Slider(
        df['year'].min(),
        df['year'].max(),
        step=None,
        value=df['year'].min(),
        marks={str(year): str(year) for year in df['year'].unique()},
        id='year-slider'
    )
], style={"width": "800px", "height": "800px"})
```

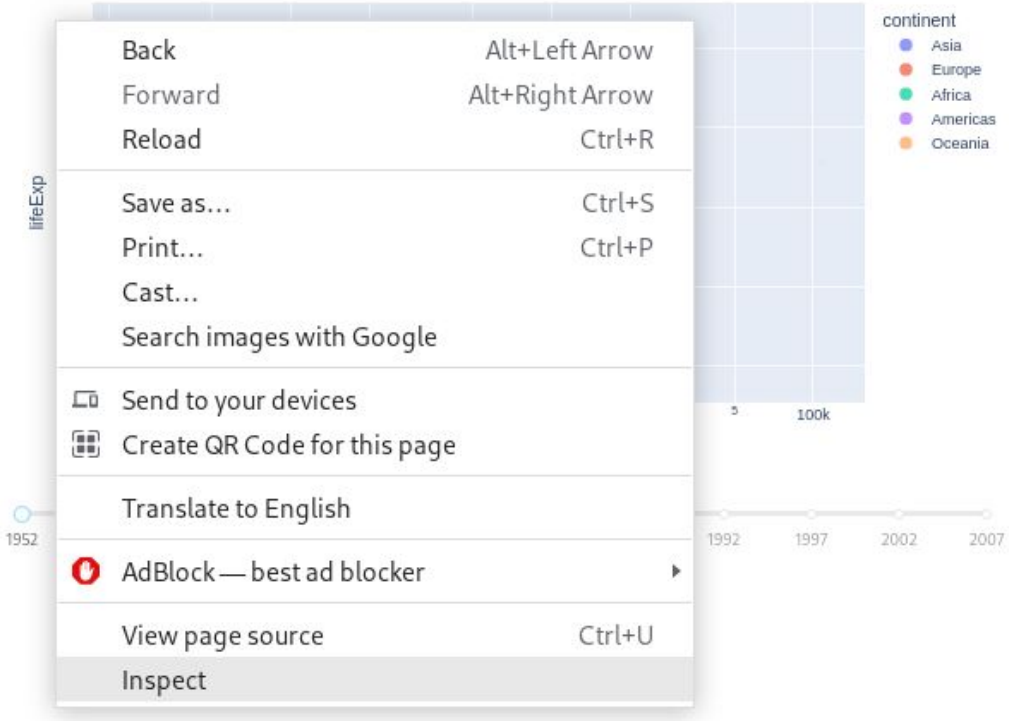
Option 2

app.py × style.css ×

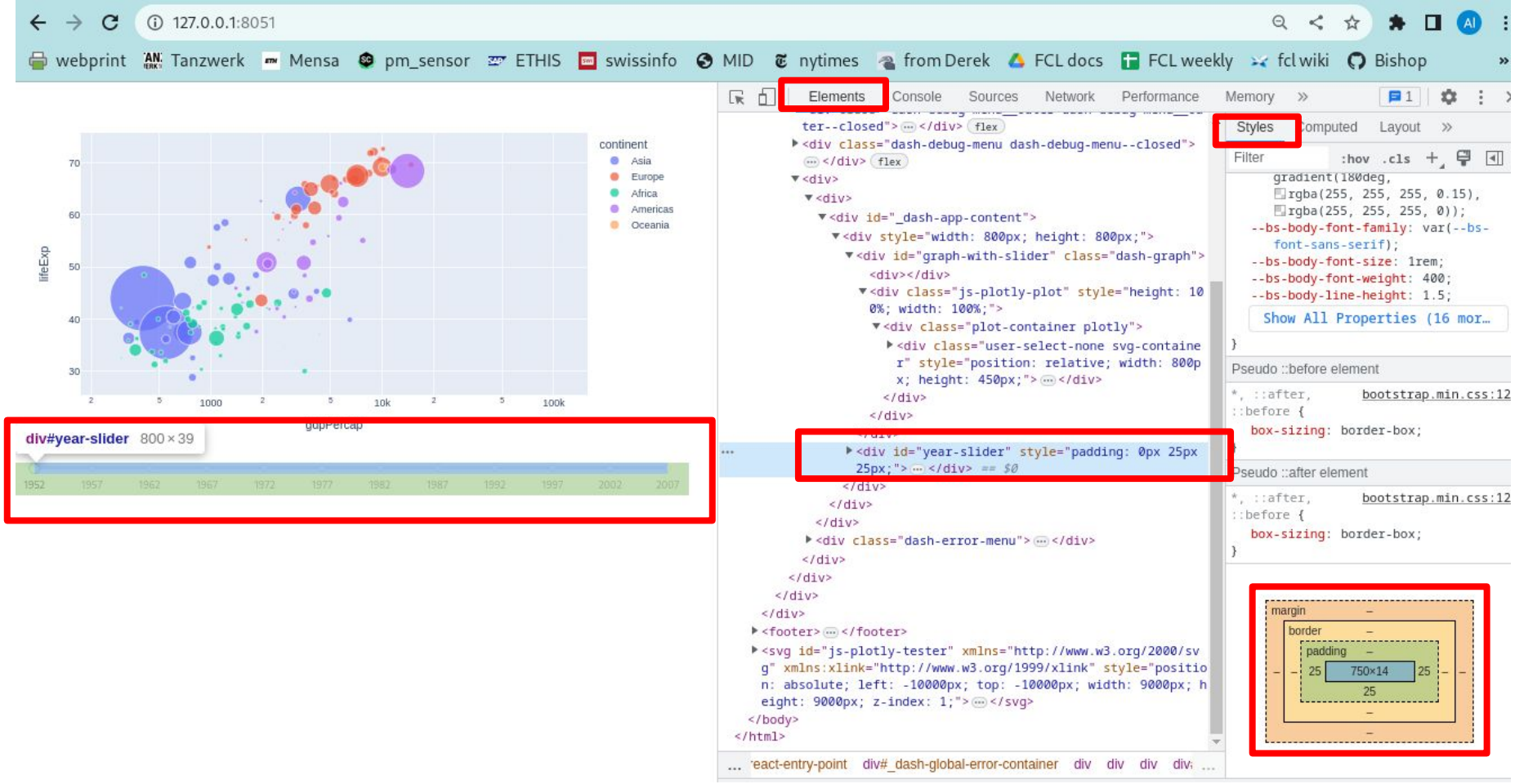
```css
body {
    background-color: white;
}

.my-header {
    color: green;
}
```

Option 1 is the CSS way, and Option 2 is the Dash way. Probably Option 1 is more flexible than Option 2.

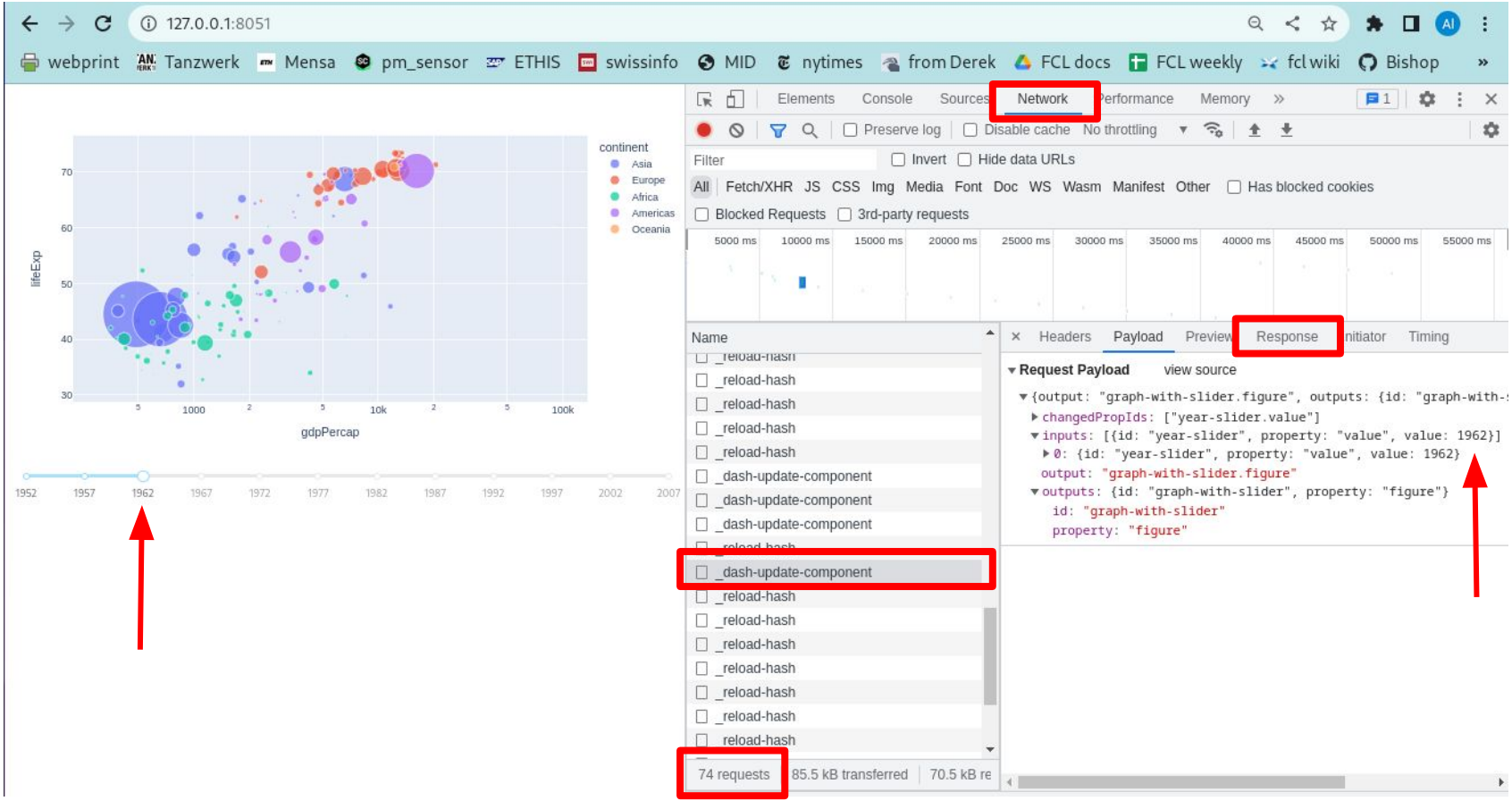
Webpage inspector



On any webpage, right click and choose Inspect.



Tab **Elements** can help in understanding the layout of a page and CSS style of each component. Also of other websites if you want to understand how the design was implemented.



Tab **Network** tracks requests to the web server and responses from the web server. This is helpful if you need to understand callback functions.

Summary of the tools

- Figma for design (allows export of CSS styles, and you can preview webpage design in real size with Figma’s presentation mode)
- Conda for virtual environments
- Pycharm (or any other IDE) for code development
- Git for version control
- Dash debug mode and website inspector for debugging

Things to consider

- Is my app only for me or do I want to eventually publish it?
- What is the underlying **content** data? How heavy is it? Do I need to set up a database? Precompute whatever you can.
- Is Dash enough? Explore existing Dash gallery and core + bootstrap components documentation!
- Layout - on which devices does it need to work? Check out flexboxes and dash bootstrap Layout component.
- Develop modularly. Understand the interfaces between the different components, create placeholder functions, and gradually fill them.
- Implement minimal version first that works from beginning till the end, then build on it. But keep the final goal in mind.

Getting started with your own dashboard

- Setup up a conda environment
- Setup Github repository
- Read Dash and Plotly docs
- Create the content you want to display
- Think of a website design
- Create layout and skeleton functions
- Implement minimal version of your dashboard
- Implement the complete design
- Publish the website, at eg github.io

Exercises

Exercise: Visualization example

https://github.com/ecological-systems-design/visualization-example

Task: Follow the instructions on the repository website, and run the visualization example we provided.

Running other exercises

- Either create new app_xxx.py files and run them.
- Or replace the code in the existing app.py file and run it.
- Browsers often cache elements of a webpage. If smth on the page that you are developing is not updating as per your expectations, use **Hard Refresh** (probably ctrl+shift+R but can be different for your laptop and browser configuration)

Exercise: Change dbc theme

https://dash-bootstrap-components.opensource.faculty.ai/docs/themes/explorer/

Task: Explore existing DBC themes and try changing it in your app.

Exercise: Layout and Styling

https://dash.plotly.com/layout

Task: Complete Part 1 to understand basics of layout and styling with Dash.

Exercise: Bootstrap component Layout

https://dash-bootstrap-components.opensource.faculty.ai/docs/components/layout/

Task: Try to add this component to understand how to implement flexible layouts.

Related material (optional):
Flexboxes for flexible layout but implemented completely in CSS
https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Exercise: Other Bootstrap components

https://dash-bootstrap-components.opensource.faculty.ai/docs/components/

Task: Explore other bootstrap components and try adding them to your dashboard.

Exercise: Callbacks

https://dash.plotly.com/basic-callbacks

Task: Complete Part 2 to understand callback functions.

Exercise (optional): Maps

https://plotly.com/python/maps/

Task: Try running app_maps.py from our visualization example, then try adding other plotly maps.