SI649-W20-LAB5

Exporting Altair Charts

Link to Notebook
Link to html/javascript folder

Today

Intro to/reminder of **basic webpages**

Embedding Altair charts in webpages

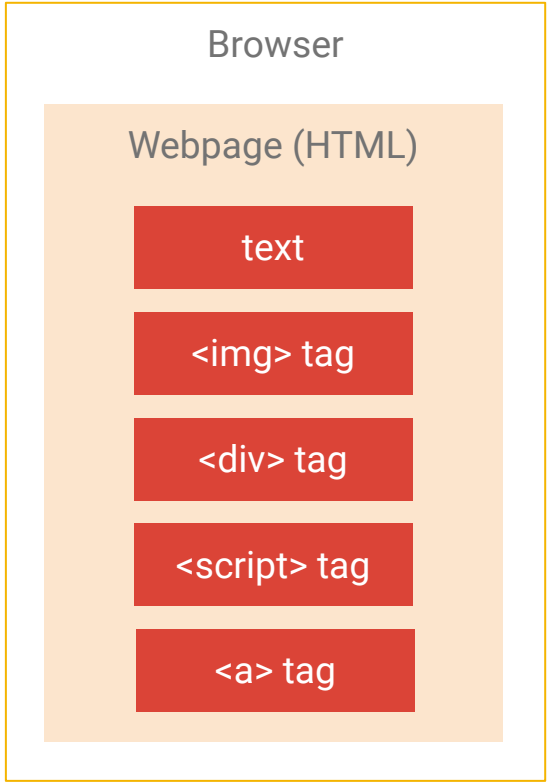
Lab (create a **web article** with interactive visualizations!)

This will be scary for some of you (sorry!)

This will be boring for some of you (sorry!)

Basic webpages

Webpage structure



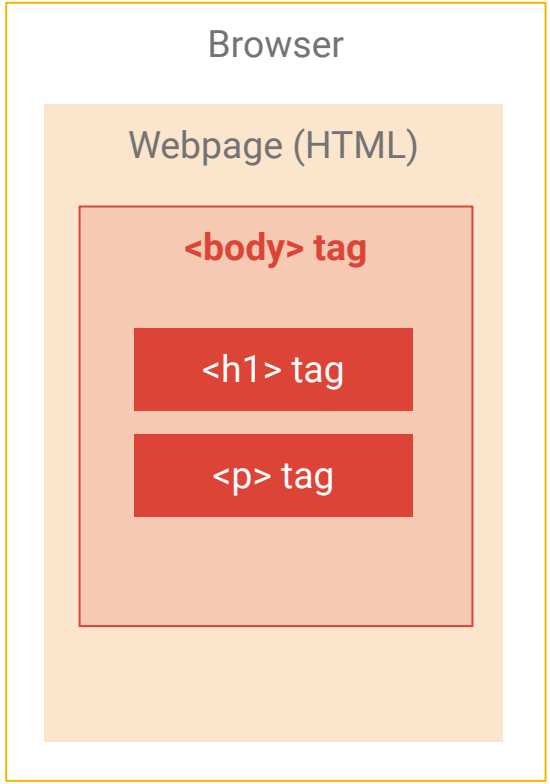
images

blocks (whatever you want)

JavaScript code

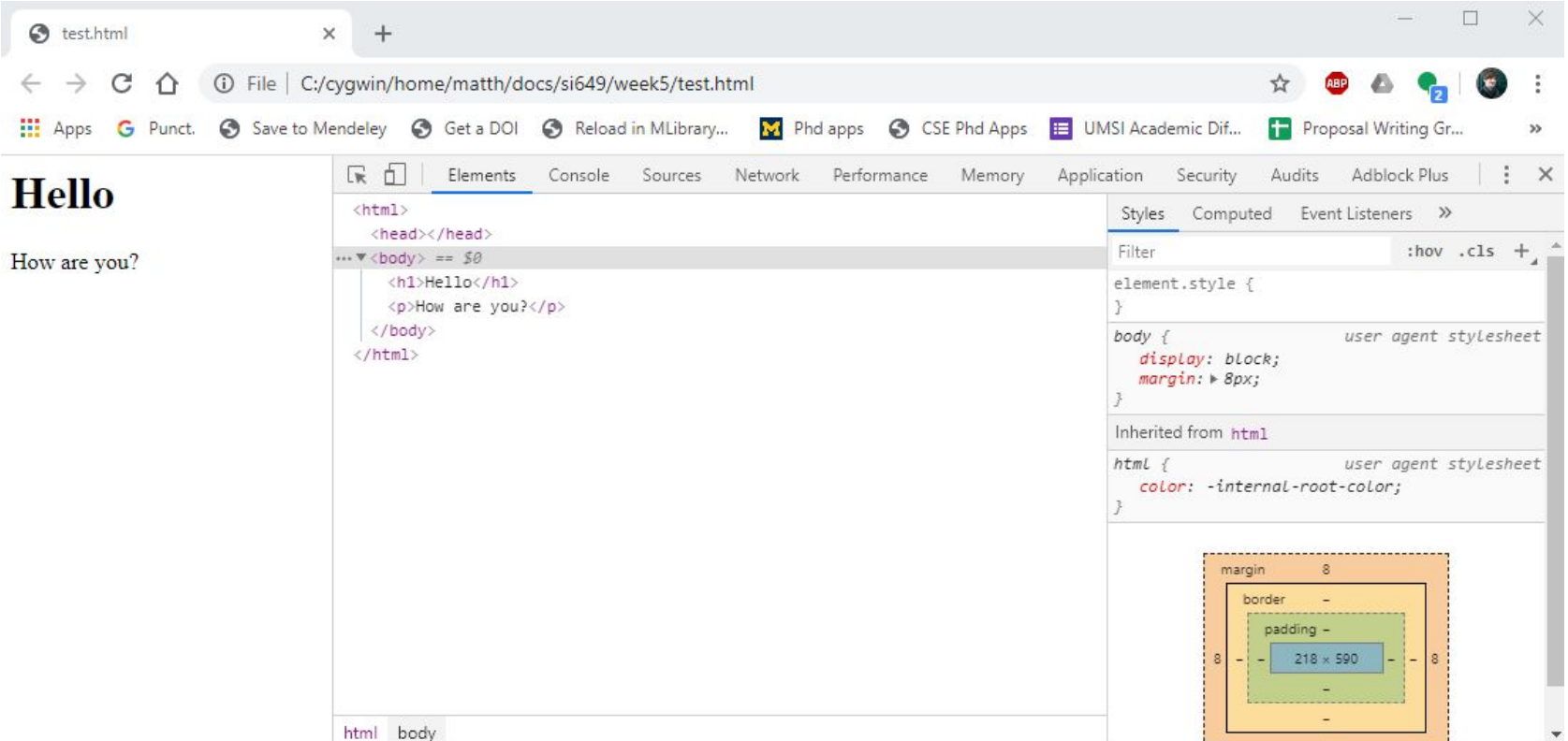
links

HTML (HyperText Markup Language)




```
<html>
  <body>
    <h1>Hello</h1>
    <p>How are you?</p>
  </body>
</html>
```

Open the Debug Window to see the tree



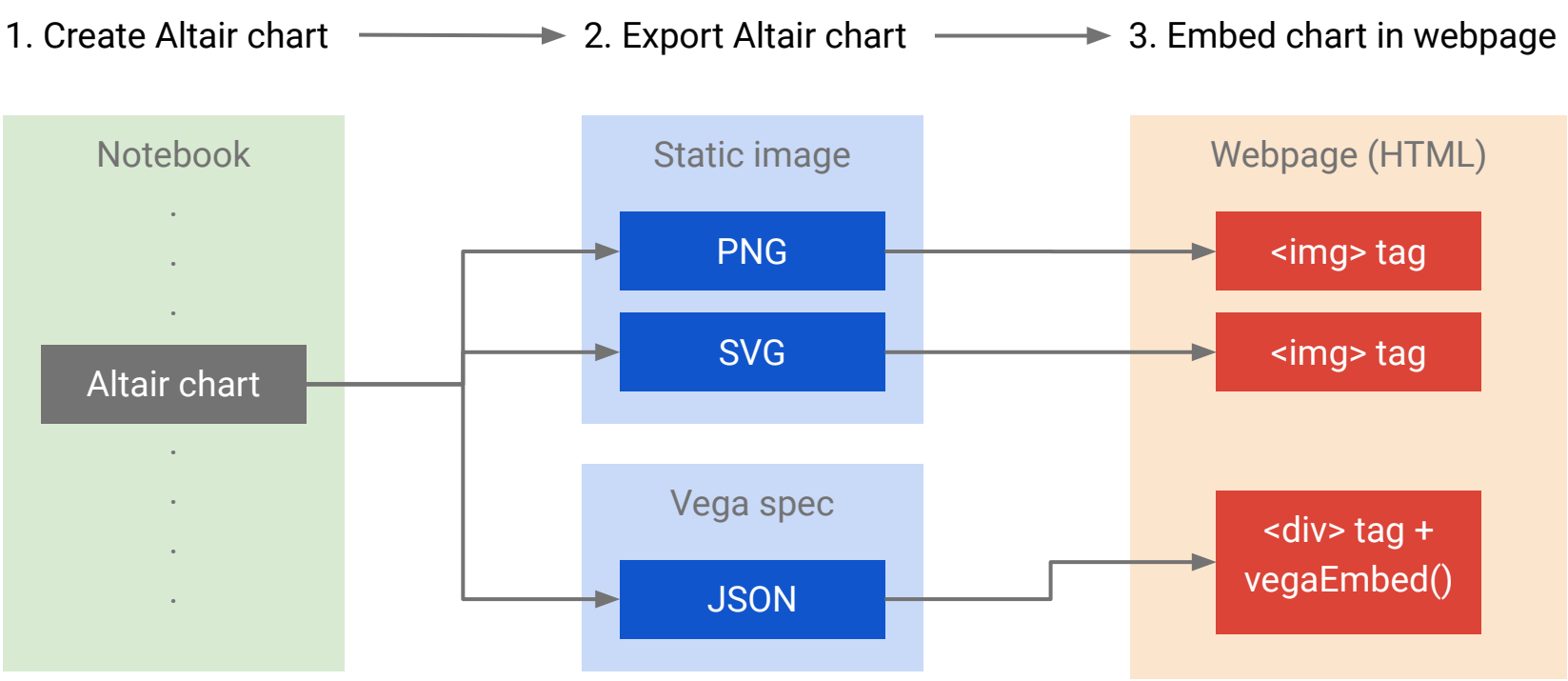
For more on web technologies

Check out the Mozilla Developer Network: https://developer.mozilla.org/en-US/

It is an excellent HTML, CSS, JavaScript resource

Embedding Altair in webpages

Embedding Altair charts in webpages



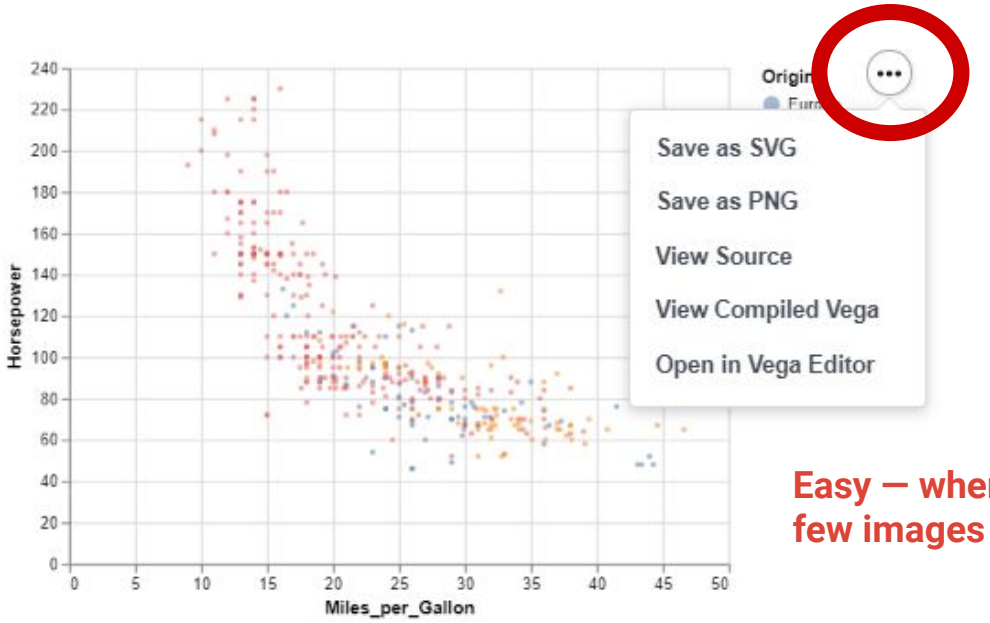
Open and copy this notebook

Static export option 1: Save As menu

Static image

PNG

SVG



Easy — when you have only a few images to export

Static export option 2: .save() method

Static image

PNG

SVG

Step1: install dependencies

```
#install
!apt-get update
!apt install chromium-chromedriver
!cp /usr/lib/chromium-browser/chromedriver
/usr/bin
!pip install selenium
```

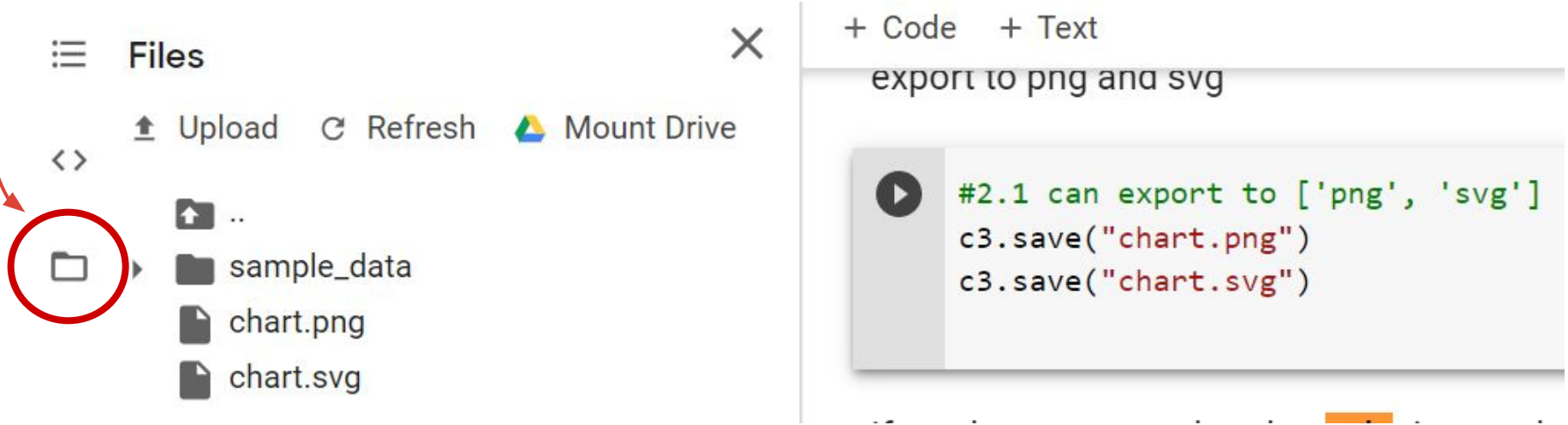
Step2: call .save() on each chart

```
c3.save("chart.png")
c3.save("chart.svg")
```

Best if you need to export the images programmatically

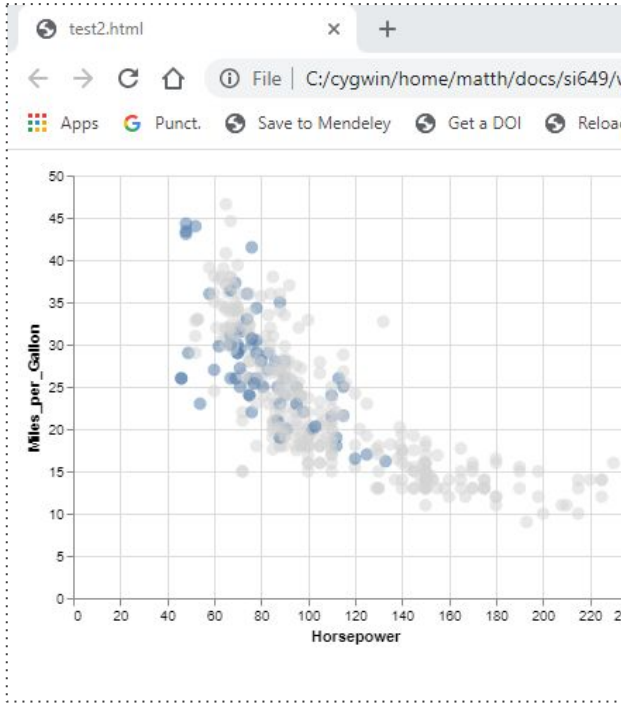
Static export option 2: .save() method

Click on the folder icon in Google Colab to access saved files



Static export option 2: .save() method


```
<html>
  <body>
    <img src="chart.png">
  </body>
</html>
```



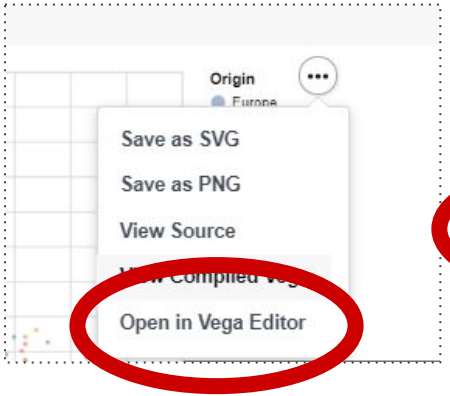
Interactive Export (Vega/JSON)

Vega spec

JSON

Option 1

Option 2



Option 3

```
#c3 is the variable
that stores your
altair chart.

c3.to_json()
```

All 3 options produce the same result.

These specs may contain the entire dataset (might be huge!)

If you pass a dataframe to alt.Chart() ...

```
c1=alt.Chart(data).mark_circle(size=80,opacity=0.5).encode(
  x='Horsepower:Q',
  y='Miles_per_Gallon:Q',
  color="Origin:N",
  size=alt.condition(
      alt.datum.Horsepower<selector.cutoff,
      alt.SizeValue(300),alt.SizeValue(10)
  )).add_selection(selector)
```

...the Vega spec will include every row of the data frame!


```
"data": [
  {
    "name": "selector001_store",
    "values": [
      {
        "unit": "\"concat_0\"",
        "fields": [{"type": "E", "field": "cutoff"}],
        "values": [230]
      }
    ]
  },
  {
    "name": "data-8e25068d8b9a578f41a59c91e539d690",
    "values": [
      {
        "Acceleration": 12,
        "Cylinders": 8,
        "Displacement": 307,
        "Horsepower": 130,
        "Miles_per_Gallon": 18,
        "Name": "chevrolet chevelle malibu",
        "Origin": "USA",
        "Weight_in_lbs": 3504,
        "Year": "1970-01-01"
      },
      {
        "Acceleration": 11.5,
        "Cylinders": 8,
        "Displacement": 350,
        "Horsepower": 165,
```

Pass a URL to make your chart spec smaller and easier to read

If you pass a URL to alt.Chart() ...

```
hp_mpg_2=alt.Chart(car_url).mark_circle(size=80,opacity=0.5).encode(
    x='Horsepower:Q',
    y='Miles_per_Gallon:Q',
    color="Origin:N"
)
hp_mpg_2
```

...the Vega spec only includes a link to the dataset.

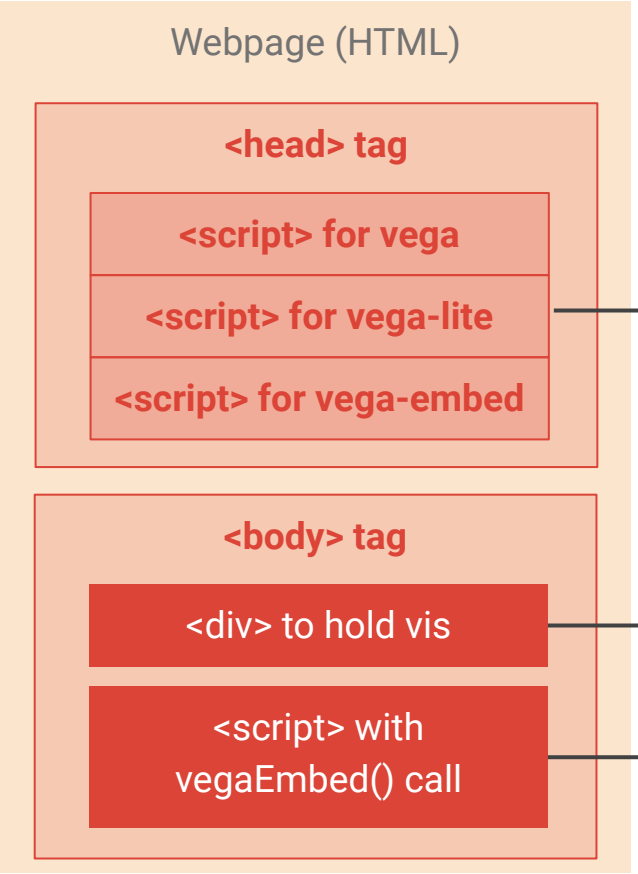

```
"$schema": "https://vega.github.io/schema/vega/v5.json",
"autosize": "pad",
"padding": 5,
"width": 400,
"height": 300,
"style": "cell",
"data": [
  {
    "name": "source_0",
    "url": "https://vega.github.io/vega-datasets/data/cars.json",
    "format": {"type": "json"},
    "transform": [
      {
        "type": "filter",
        "expr": "datum[\"Horsepower\"] !== null && !isNaN(datum[
      }
    ]
  }
],
"marks": [
  {
    "name": "marks",
    "type": "symbol",
    "style": ["circle"],
```

Putting a Vega spec in a webpage

1. Load JavaScript libraries: vega, vega-lite, vega-embed

2. Create a **placeholder** to hold the visualization

3. Call **vegaEmbed()** to create the visualization

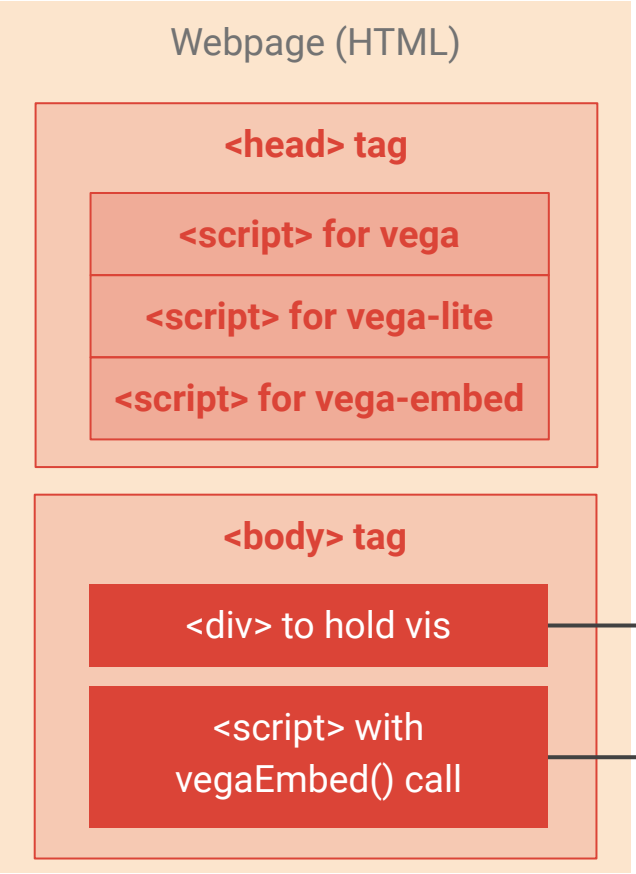



```
<!DOCTYPE html>
<html>
<head>
    <script type="text/javascript"
src="https://cdn.jsdelivr.net/npm//vega@5"></script>
    <script type="text/javascript"
src="https://cdn.jsdelivr.net/npm//vega-lite@3.4.0"></script>
    <script type="text/javascript"
src="https://cdn.jsdelivr.net/npm//vega-embed@4"></script>
</head>
<body>

<div id="vis"></div>

<script type="text/javascript">
    let jsonSpec="chart.json";
    vegaEmbed('#vis', jsonSpec,{theme:
'latimes',"renderer":"svg"}).then(function(result) {
    }).catch(console.error);
</script>
</body>
</html>
```

Load libraries: like **import** in python

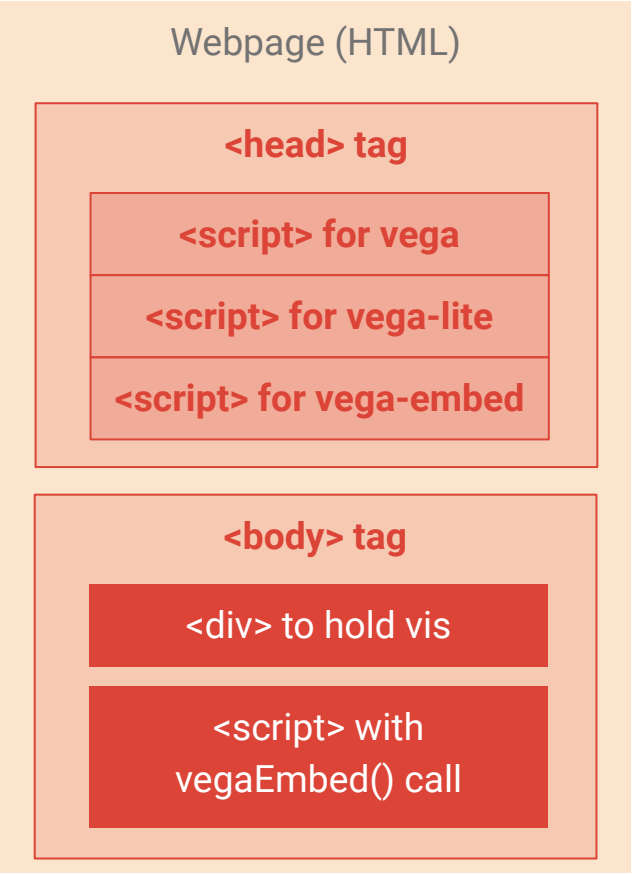
create **placeholder**

load vis with **vegaEmbed()**

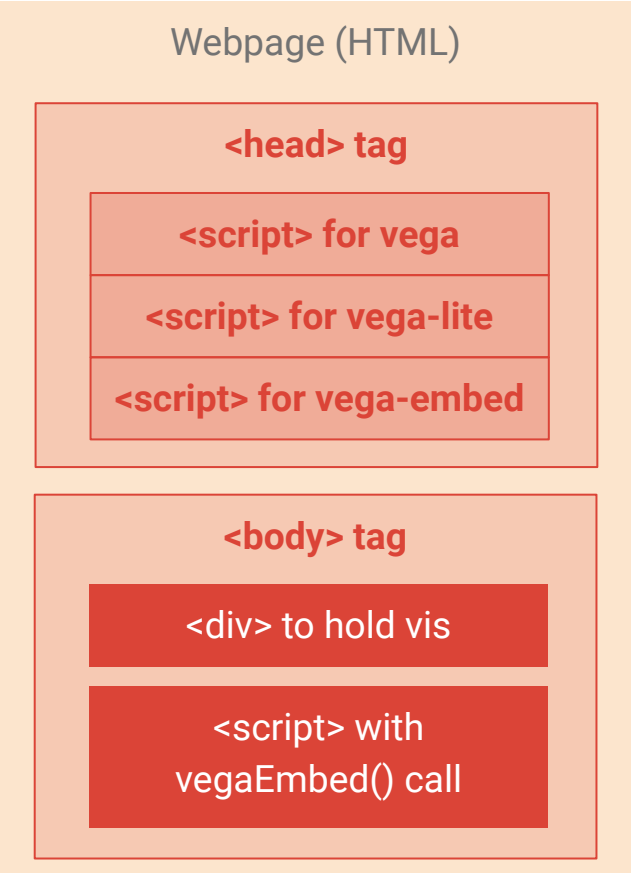



```
<!DOCTYPE html>
<html>
<head>
   <script type="text/javascript"
src="https://cdn.jsdelivr.net/npm//vega@5"></script>
   <script type="text/javascript"
src="https://cdn.jsdelivr.net/npm//vega-lite@3.4.0"></script>
   <script type="text/javascript"
src="https://cdn.jsdelivr.net/npm//vega-embed@4"></script>
</head>
<body>

<div id="vis">
</div>

<script type="text/javascript">
   let jsonSpec="chart.json";
   vegaEmbed('#vis', jsonSpec,{theme:
'latimes',"renderer":"svg"}).then(function(result) {
   }).catch(console.error);
</script>
</body>
</html>
```

use **id** attribute to link vis to tag




```
<!DOCTYPE html>
<html>
<head>
   <script type="text/javascript"
src="https://cdn.jsdelivr.net/npm//vega@5"></script>
   <script type="text/javascript"
src="https://cdn.jsdelivr.net/npm//vega-lite@3.4.0"></script>
   <script type="text/javascript"
src="https://cdn.jsdelivr.net/npm//vega-embed@4"></script>
</head>
<body>

<div id="vis"></div>

<script type="text/javascript">
   let jsonSpec=PASTE YOUR JSON HERE;
   vegaEmbed('#vis', jsonSpec,{theme:
'latimes',"renderer":"svg"}).then(function(result) {
   }).catch(console.error);
</script>
</body>
</html>
```




```
<!DOCTYPE html>
<html>
<head>
   <script type="text/javascript"
src="https://cdn.jsdelivr.net/npm//vega@5"></script>
   <script type="text/javascript"
src="https://cdn.jsdelivr.net/npm//vega-lite@3.4.0"></script>
   <script type="text/javascript"
src="https://cdn.jsdelivr.net/npm//vega-embed@4"></script>
</head>
<body>

<div id="vis"></div>

<script type="text/javascript">
   let jsonSpec=LINK TO JSON FILE;
   vegaEmbed('#vis', jsonSpec,{theme:
'latimes',"renderer":"svg"}).then(function(result) {
   }).catch(console.error);
</script>
</body>
</html>
```

Webpage (HTML)

<head> tag

- **<script> for vega**
- **<script> for vega-lite**
- **<script> for vega-embed**

<body> tag

- to hold vis
- <script> with vegaEmbed() call

```
<!DOCTYPE ht
<html>
<head>
    <script t
src="https:/
    <script t
src="https:/
    <script t
src="https:/
</head>
<body>

<div id="vis"></div>

<script type="text/javascript">
    let jsonSpec="chart.json";
    vegaEmbed('#vis', jsonSpec,{theme:
'latimes',"renderer":"svg"}).then(function(result) {
    }).catch(console.error);
</script>
</body>
</html>
```

```
▶ Fetch API cannot load fil   vega@5:1
e:///C:/licia/si649-github/week5/dem
o/vega-embed/answer/chart.json. URL
scheme must be "http" or "https" for
CORS request.

TypeError: Failed to fetch
    at Object.http (vega@5:1)
    at Object.Kt [as load] (vega@5:1)
```

Using local files

You need to have a server in order to load external content (such as .json, .csv)

Use Python:

```
cd <folder containing your .html>
Python -m http.server
```

Then go to localhost:8000 or http://0.0.0.0:8000/

```
<!DOCTYPE ht
<html>
<head>
   <script t
src="https:/
   <script t
src="https:/
   <script t
src="https:/
</head>
<body>

<div id="vis"></div>

<script type="text/javascript">
   let jsonSpec="chart.json";
   vegaEmbed('#vis', jsonSpec,{theme:
'latimes',"renderer":"svg"}).then(function(result) {
   }).catch(console.error);
</script>
</body>
</html>
```

```
▶ Fetch API cannot load fil   vega@5:1
e:///C:/licia/si649-github/week5/dem
o/vega-embed/answer/chart.json. URL
scheme must be "http" or "https" for
CORS request.

TypeError: Failed to fetch
    at Object.http (vega@5:1)
    at Object.Kt [as load] (vega@5:1)
```

Closer look at the vegaEmbed() call

Specify where to insert your chart

Give chart spec (JSON or a URL)

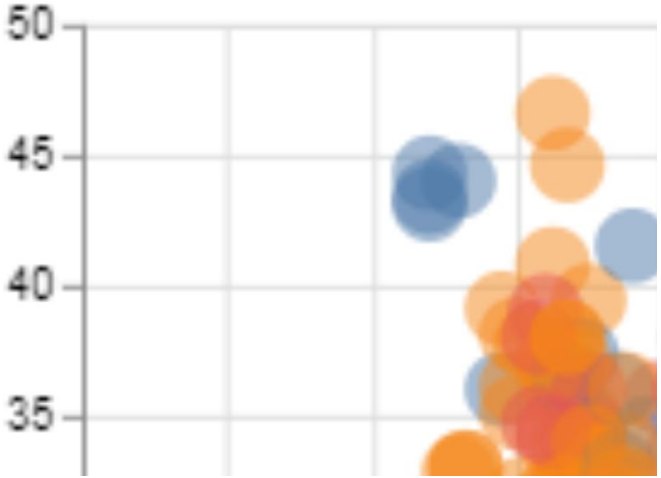
Add custom configuration here

```
let jsonSpec="chart.json";

vegaEmbed('#vis',

jsonSpec,{}).then(function(res

ult) {}).catch(console.error);
```

Embed Configurations: Renderer

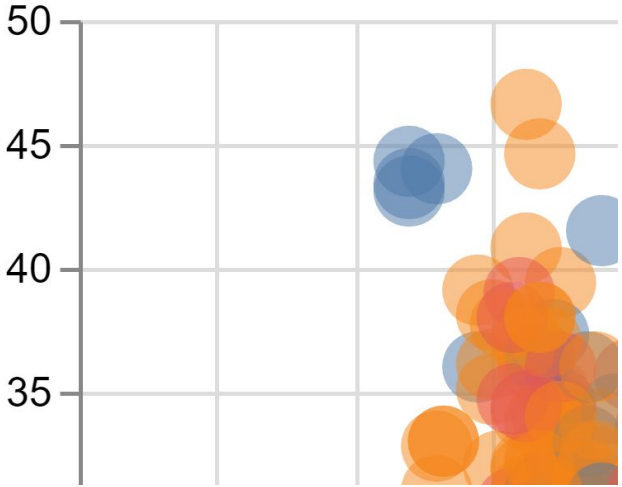
Canvas renderer:
Good if you have lots of data and need fast rendering

```
vegaEmbed('#vis',
jsonSpec,{"renderer":"canvas"}).then(function
(result) {}).catch(console.error);
```



SVG renderer:
Good if you need high-quality rendering

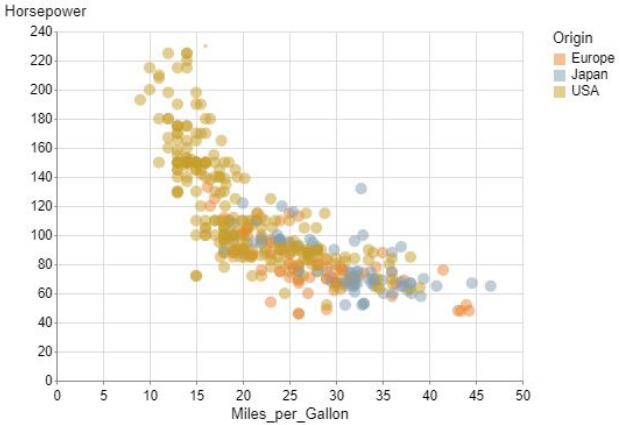
```
vegaEmbed('#vis',
jsonSpec,{"renderer":"svg"}).then(function(result)
{}).catch(console.error);
```

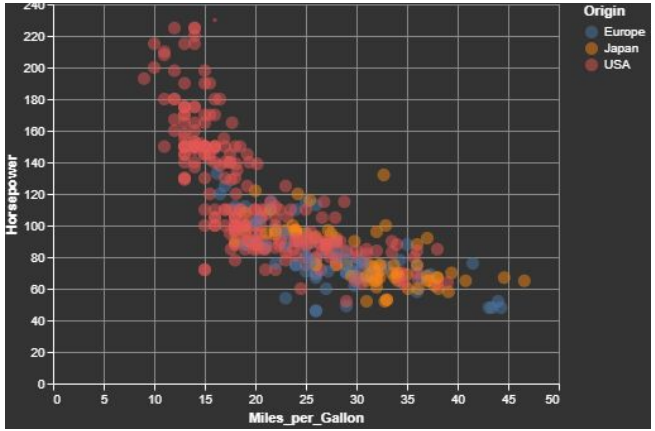


Embed Configurations: Theme

```
vegaEmbed('#vis', jsonSpec,
{theme: 'latimes'}).then(function(result)
{}).catch(console.error);
```

```
vegaEmbed('#vis', jsonSpec,
{theme: 'dark'}).then(function(result)
{}).catch(console.error);
```





Using HTML templates

Example

1. Download example, read to understand the components you need
2. Make sure all links (to js, css, image) are correct
3. Add vega embed to the appropriate location

Additional Customization

- Add more HTML elements (img, a, h1, p, ...)
- Use CSS to set visual style of elements
- Other javascript, external libraries

HW

Lab 5 HW: Write an article using the Presidential General Election Polls

Author a data-driven article that reveals something interesting about U.S. Presidential General Election Polls.

You can find the data here: (https://projects.fivethirtyeight.com/polls-page/president_polls.csv)

More data description

Checkpoints

1. Load the data into Tableau/R/python/a tool of your choice and identify **at least 5 insights**

    - Insights should **not** be at the scale of a single data point (e.g., “candidate A has x% approval in year y” is not an exciting insight)
    - You can use articles on 538 or any news source as a starting point

2. Pick **3 insights** and sketch at least 2 alternative visualizations for each insight

    - Both paper/digital sketches are fine.

3. Generate **3 interactive visualizations** to communicate these 3 insights using Altair

Checkpoints

4. **[article.html]**
   Create a webpage (article) embedding the visualizations and sufficient text for reader to understand the insights and their implications.

   - Your webpage should be professional quality

5. **[documentation.html]**
   Create a webpage with all of your insights analysis, sketches, and any other experiments. Show us your process. Caption everything so we know what you did and why you did it.

HW notes

1. You can clean/organize your data with any tool you want (e.g., python, excel, Tableau, R, etc.)
2. You can build your website with any tool you want (e.g., plain html, js+css+html, other libraries, Idyll, etc.).
3. If you use Idyll, please DO NOT submit your node module folder. Build your Idyll project and only submit files under the “build” folder.
4. Make sure all of your links are relative. E.g., if you html’s path is “usr/lab4/project/index.html”, your image is stored as “usr/lab4/project/imgs/img.png”, your url to image should be “imgs/img.png” instead of “usr/lab4/project/imgs/img.png”.

UPLOADING DETAILS

You are likely to have additional images and json. Make sure all the links in your html are relative. Zip the entire folder and upload it.

Main files that we are looking for:

- article.html
- documentation.html