Run `git pull` in the main branch to follow along today.

JavaScript

DSC 106: Data Visualization

Sam Lau

UC San Diego

Announcements

Lab 4 due on Friday

Project 2 due on Tuesday

FAQs:

1. Do I have to use the same dataset for both vis for Project 2? **Yes.**
2. How similar do the two visualizations need to be? **No requirement.**

How to do well on Project 2

Component	Excellent	Satisfactory	Poor
Data Transformations *(per visualization)*	More advanced transformations (e.g., groupings, binnings, calculated fields, etc.) extend or manipulate the dataset in interesting and/or unexpected ways. *(0.5 points)*	The raw dataset was mostly used directly, with perhaps some simple transforms (e.g., sorting, filtering) to facilitate communicating the visualization's message. *(0 points)*	-
Marks, Encodings, and Visual Design *(per visualization)*	Visual design persuasively argues the visualization's stance (for/against the proposition), and facilitates effortless reading even when used deceptively. Any deceptive visual design choices are very subtle—even seasoned readers can only identify them on close study. *(2.5 points)*	Visual design is largely persuasive, but some issues hinder comprehension. Any deceptive visual design cannot be detected at first glance, but are identifiable on a second look. *(2 points)*	Visual design is distracting or makes the visualization unnecessarily or unintentionally difficult to read. Any deceptive design can be immediately identified. *(1.5 points)*
Titles, Labels, and Annotations *(per visualization)*	Titles, labels, and annotations persuasively describe, contextualize or frame the depicted data. Any slants that may be considered deceptive are imperceptible to the reader. *(2 points)*	Necessary titles and labels are present, but annotations could be better used to persuasively narrate the visualization's stance. Any deceptively slanted content is more easily detectable by readers. *(1.5 points)*	Several titles or labels are missing, or do not provide human-understandable information. Annotations are rarely used. Strong, charged, or colorful language makes it easy to detect deceptive content. *(1 point)*
Design Rationale and Reflection	Well crafted write-up provides reasoned justification for all design choices with a thoughtful reflection on their ethical implications. *(4 points)*	Most design decisions are described, but rationale or ethical reflections could be explained at a greater level of detail. *(3 points)*	Missing or incomplete. Several design choices are left unexplained, and/or ethical reflection is relatively shallow. *(2 points)*
Creativity and Originality	You exceeded the parameters of the assignment, with creative, original or a particularly engaging designs. *(up to +1 bonus point)*	You met all the parameters of the assignment. *(0 points)*	–

Component	Excellent	Satisfactory	Poor
Data Transformations *(per visualization)*	More advanced transformations (e.g., groupings, binnings, calculated fields, etc.) extend or manipulate the dataset in interesting and/or unexpected ways. *(0.5 points)*	The raw dataset was mostly used directly, with perhaps some simple transforms (e.g., sorting, filtering) to facilitate communicating the visualization's message. *(0 points)*	-
Mark Encoding Visual *(per visualization)*	seasoned readers can only identify them on close study. *(2.5 points)*	look. *(2 points)*	immediately identified. *(1.5 points)*
Titles, Labels, and Annotations *(per visualization)*	Titles, labels, and annotations persuasively describe, contextualize or frame the depicted data. Any slants that may be considered deceptive are imperceptible to the reader. *(2 points)*	Necessary titles and labels are present, but annotations could be better used to persuasively narrate the visualization's stance. Any deceptively slanted content is more easily detectable by readers. *(1.5 points)*	Several titles or labels are missing, or do not provide human-understandable information. Annotations are rarely used. Strong, charged, or colorful language makes it easy to detect deceptive content. *(1 point)*
Design Rationale and Reflection	Well crafted write-up provides reasoned justification for all design choices with a thoughtful reflection on their ethical implications. *(4 points)*	Most design decisions are described, but rationale or ethical reflections could be explained at a greater level of detail. *(3 points)*	Missing or incomplete. Several design choices are left unexplained, and/or ethical reflection is relatively shallow. *(2 points)*
Creativity and Originality	You exceeded the parameters of the assignment, with creative, original or a particularly engaging designs. *(up to +1 bonus point)*	You met all the parameters of the assignment. *(0 points)*	–

Keyword here is **interesting**
(beyond just a groupby + mean)

Can I cherry-pick data?
Yes, if you come up with a valid-sounding reason for doing so. Remember that you must describe the transformations in the plot itself (not just in the writeup).

Component	Excellent	Satisfactory	Poor
Data Transformations *(per visualization)*	More advanced transformations (e.g., groupings, binnings, calculated fields, etc.) extend or manipulate the dataset in interesting and/or unexpected ways. *(0.5 points)*	The raw dataset was mostly used directly, with perhaps some simple transforms (e.g., sorting, filtering) to facilitate communicating the visualization's message. *(0 points)*	-
Marks, Encodings, and Visual Design *(per visualization)*	Visual design persuasively argues the visualization's stance (for/against the proposition), and facilitates effortless reading even when used deceptively. Any deceptive visual design choices are very subtle—even seasoned readers can only identify them on close study. *(2.5 points)*	Visual design is largely persuasive, but some issues hinder comprehension. Any deceptive visual design cannot be detected at first glance, but are identifiable on a second look. *(2 points)*	Visual design is distracting or makes the visualization unnecessarily or unintentionally difficult to read. Any deceptive design can be immediately identified. *(1.5 points)*
Titles, Labels, and Annotations *(per visualization)*			Several titles or labels are
Design Rationale and Reflection			
Creativity and Originality	You exceeded the parameters of the assignment, with creative, original or a particularly engaging designs. *(up to +1 bonus point)*	You met all the parameters of the assignment. *(0 points)*	–

Visualizations will be graded on **effectiveness**. E.g. avoid overplotting, avoid pie charts, use perceptually uniform color scales, etc.

Key word is **subtle.**

- Only relying on axis manipulations (e.g. dual axes) caps you at Satisfactory
- Try counting things differently!

If you use a map, we expect to see some **geospatial interpretation**. "Could this data be better shown as a bar plot?" If yes, caps score at Satisfactory.

Component	Excellent	Satisfactory	Poor
Data Transformations *(per visualization)*		The raw dataset was mostly	
Marks, Encodings, and Visual Design *(per visualization)*			
	(2.5 points)		
Titles, Labels, and Annotations *(per visualization)*	Titles, labels, and annotations persuasively describe, contextualize or frame the depicted data. Any slants that may be considered deceptive are imperceptible to the reader. *(2 points)*	Necessary titles and labels are present, but annotations could be better used to persuasively narrate the visualization's stance. Any deceptively slanted content is more easily detectable by readers. *(1.5 points)*	Several titles or labels are missing, or do not provide human-understandable information. Annotations are rarely used. Strong, charged, or colorful language makes it easy to detect deceptive content. *(1 point)*
Design Rationale and Reflection	Well crafted write-up provides reasoned justification for all design choices with a thoughtful reflection on their ethical implications. *(4 points)*	Most design decisions are described, but rationale or ethical reflections could be explained at a greater level of detail. *(3 points)*	Missing or incomplete. Several design choices are left unexplained, and/or ethical reflection is relatively shallow. *(2 points)*
Creativity and Originality	You exceeded the parameters of the assignment, with creative, original or a particularly engaging designs. *(up to +1 bonus point)*	You met all the parameters of the assignment. *(0 points)*	–

- Titles and annotations are graded similarly to Project 1. Avoid "X vs. Y" titles!
- Annotations should guide readers to an interpretation

Component	Excellent	Satisfactory	Poor
Data Transformations *(per visualization)*	More advanced transformations (e.g., groupings, binnings, calculated fields, etc.) extend or manipulate the dataset in interesting and/or unexpected ways. *(0.5 points)*	The raw dataset was mostly used directly, with perhaps some simple transforms (e.g., sorting, filtering) to facilitate communicating the visualization's message. *(0 points)*	-
Marks, Encodings, and Visual Design *(per visualization)*			
Titles, Labels, and Annotations *(per visualization)*	*(2 points)*	*(1.5 points)*	content. *(1 point)*
Design Rationale and Reflection	Well crafted write-up provides reasoned justification for all design choices with a thoughtful reflection on their ethical implications. *(4 points)*	Most design decisions are described, but rationale or ethical reflections could be explained at a greater level of detail. *(3 points)*	Missing or incomplete. Several design choices are left unexplained, and/or ethical reflection is relatively shallow. *(2 points)*
Creativity and Originality	You exceeded the parameters of the assignment, with creative, original or a particularly engaging designs. *(up to +1 bonus point)*	You met all the parameters of the assignment. *(0 points)*	–

From the project 2 page, for each design decision:

"How does this decision help make your visualization persuasive? What worked well, and what didn't? **What other alternatives did you consider, and why did you settle on this one?**"

Looking for a personal reflection on ethics. How did doing this project affect **your views**?

Component	Excellent	Satisfactory	Poor
Data Transformations *(per visualization)*	More advanced transformations (e.g., groupings, binnings, calculated fields, etc.) extend or manipulate the dataset in interesting and/or unexpected ways. *(0.5 points)*	The raw dataset was mostly used directly, with perhaps some simple transforms (e.g., sorting, filtering) to facilitate communicating the visualization's message. *(0 points)*	-
Marks, Encodings, and Visual Design *(per visualization)*	Visual design persuasively argues the visualization's stance (for/against the proposition), and facilitates effortless reading even when used deceptively. Any deceptive visual design choices are very subtle—even seasoned readers can only identify them on close study. *(2.5 points)*	Visual design is largely persuasive, but some issues hinder comprehension. Any deceptive visual design cannot be detected at first glance, but are identifiable on a second look. *(2 points)*	Visual design is distracting or makes the visualization unnecessarily or unintentionally difficult to read. Any deceptive design can be immediately identified. *(1.5 points)*
Titles, Labels, and Annotations *(per visualization)*	Titles, labels, and annotations persuasively describe, contextualize or frame the depicted data. Any slants that may be considered deceptive are imperceptible to the reader. *(2 points)*	Necessary titles and labels are present, but annotations could be better used to persuasively narrate the visualization's stance. Any deceptively slanted content is more easily detectable by readers. *(1.5 points)*	Several titles or labels are missing, or do not provide human-understandable information. Annotations are rarely used. Strong, charged, or colorful language makes it easy to detect deceptive content. *(1 point)*
Design Rationale and Reflection	Well crafted write-up provides reasoned justification for all design choices with a thoughtful reflection on their ethical implications. *(4 points)*	Most design decisions are described, but rationale or ethical reflections could be explained at a greater level of detail. *(3 points)*	Missing or incomplete. Several design choices are left unexplained, and/or ethical reflection is relatively shallow. *(2 points)*
Creativity and Originality	You exceeded the parameters of the assignment, with creative, original or a particularly engaging designs. *(up to +1 bonus point)*	You met all the parameters of the assignment. *(0 points)*	–

If we can copy-paste your reflection into someone else's Project 2, it's not specific enough!

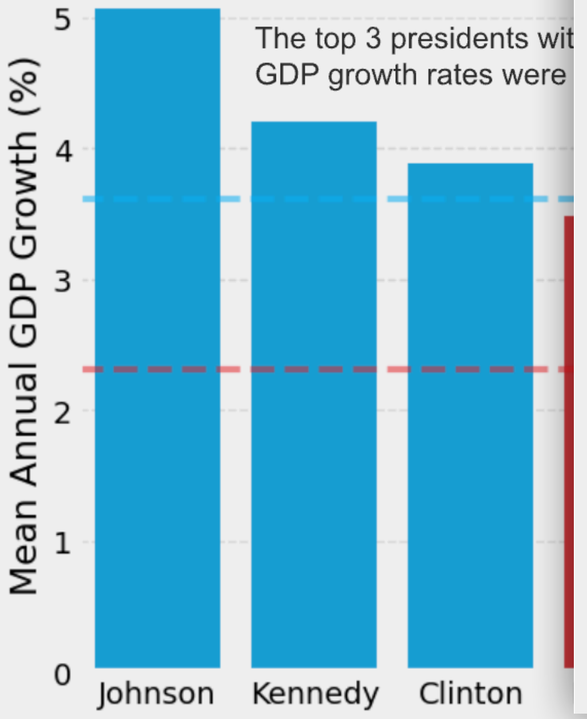
Component	Excellent	Satisfactory	Poor
Data Transformations *(per visualization)*	More advanced transformations (e.g., groupings, binnings, calculated fields, etc.) extend or manipulate the dataset in interesting and/or unexpected ways. *(0.5 points)*	The raw dataset was mostly used directly, with perhaps some simple transforms (e.g., sorting, filtering) to facilitate communicating the visualization's message. *(0 points)*	-
Marks, Encodings, and Visual Design *(per visualization)*	Visual design persuasively argues the visualization's stance (for/against the proposition), and facilitates effortless reading even when used deceptively. Any deceptive visual design choices are very subtle—even seasoned readers can only identify them on close study. *(2.5 points)*	Visual design is largely persuasive, but some issues hinder comprehension. Any deceptive visual design cannot be detected at first glance, but are identifiable on a second look. *(2 points)*	Visual design is distracting or makes the visualization unnecessarily or unintentionally difficult to read. Any deceptive design can be immediately identified. *(1.5 points)*
Titles, Labels, and Annotations *(per visualization)*	Titles, labels, and annotations persuasively describe, contextualize or frame the depicted data. Any slants that may be considered	Necessary titles and labels are present, but annotations could be better used to persuasively narrate the visualization's	Several titles or labels are missing, or do not provide human-understandable information. Annotations are rarely used. Strong, charged
Design Rationale and Reflection	*(4 points)*	detail. *(3 points)*	*(2 points)*
Creativity and Originality	You exceeded the parameters of the assignment, with creative, original or a particularly engaging designs. *(up to +1 bonus point)*	You met all the parameters of the assignment. *(0 points)*	–

Creative encodings, engaging design, interesting external data, etc.

Nifty Project 2 Submissions (from the past)

Who's the Fisca
Democrats Lead

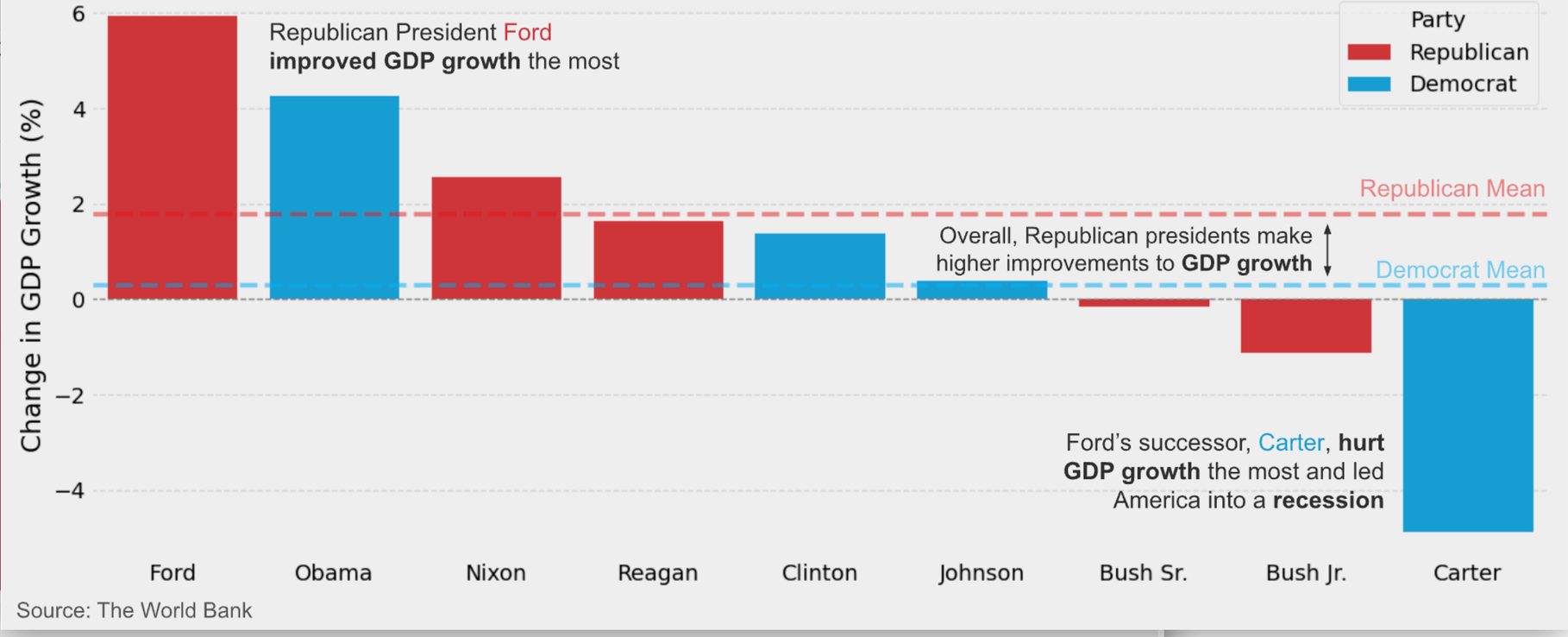
Mean G



Source: The World Bank

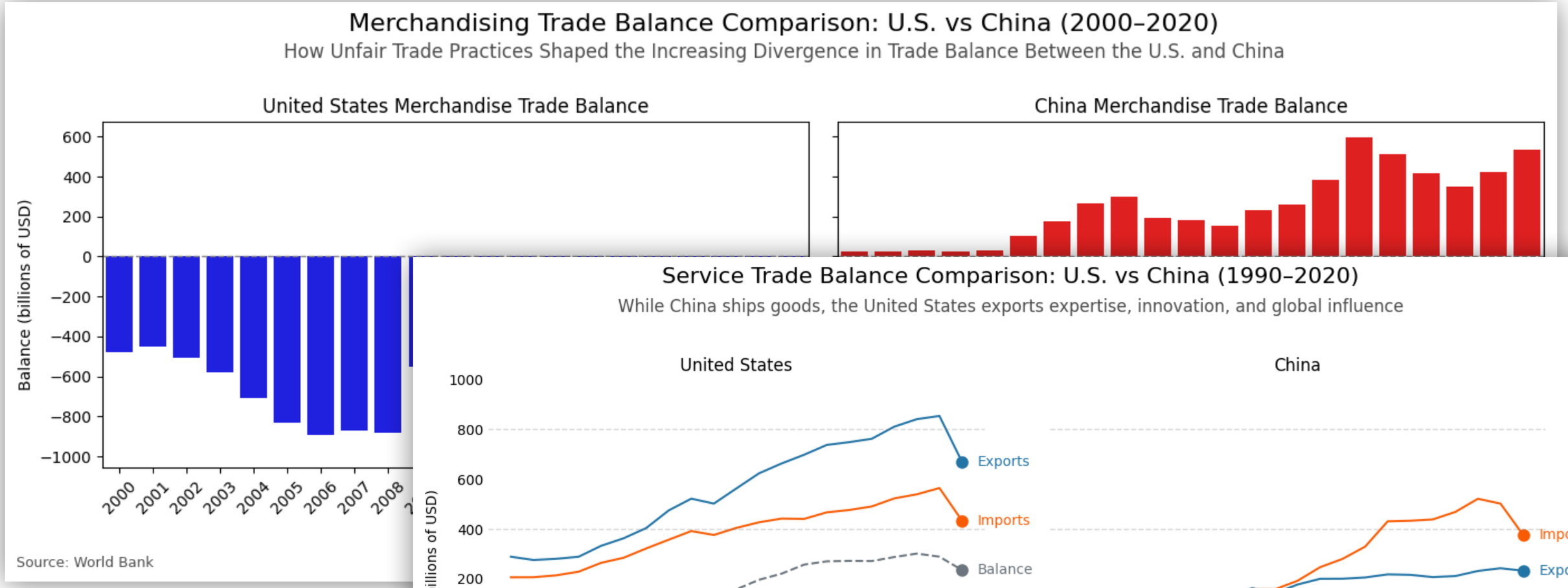
Common Sense Fiscal Policy Works.
Republicans Lead to Higher GDP Growth

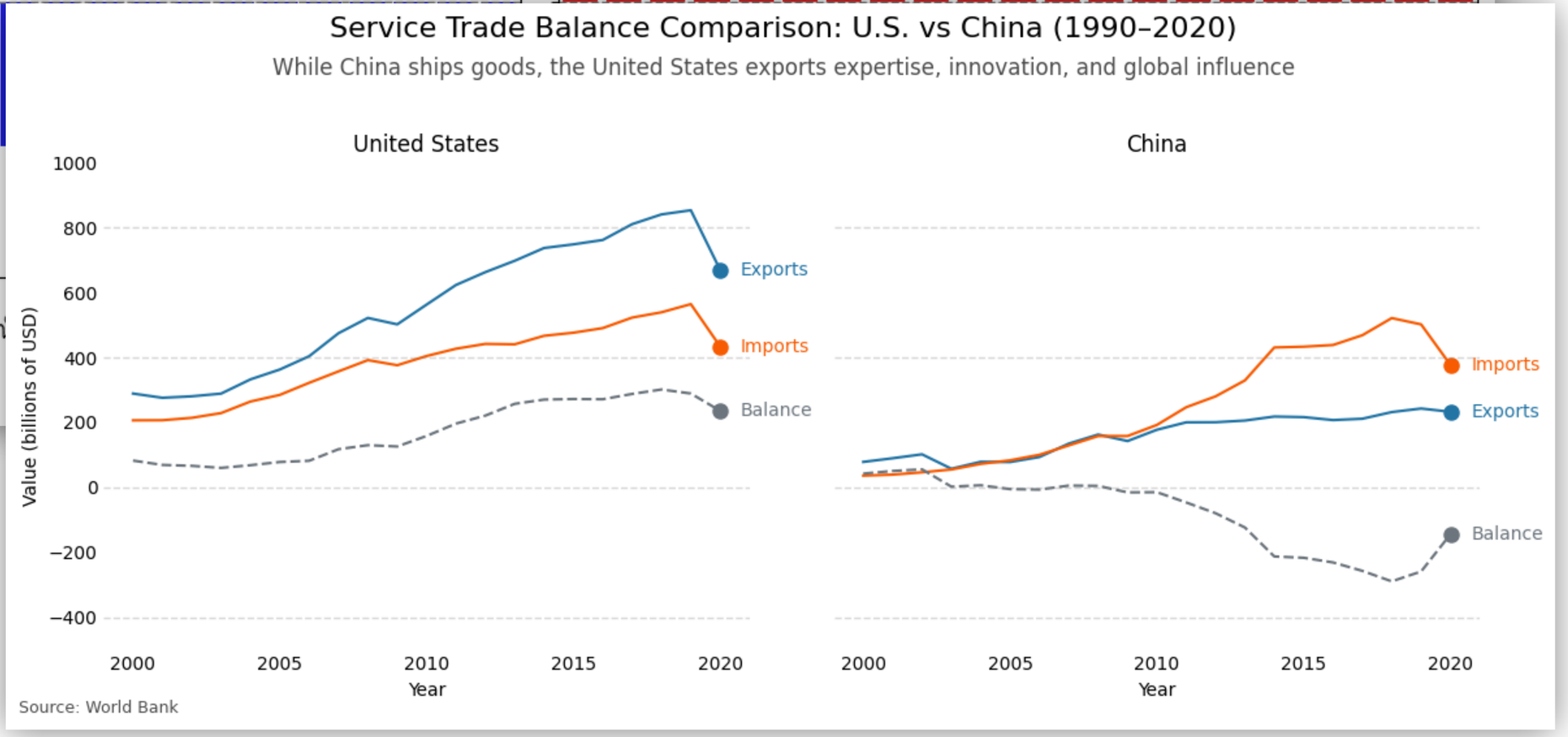
Change in GDP Growth (%) between Start and End of Terms of each President (1963-2016)



Source: The World Bank

https://vdanielb.github.io/sample-visualizations/





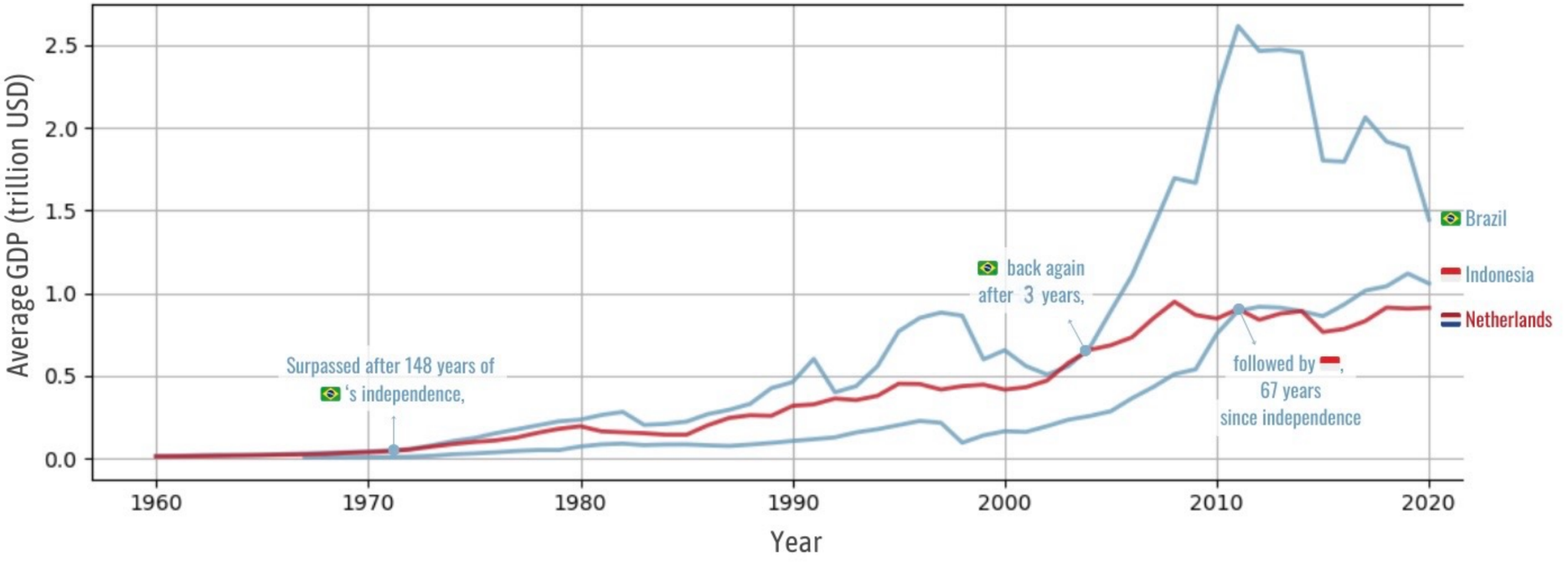
https://dariefmaes.github.io/dsc106-proj2/project2_report

Beyond Independence: A Growing Divide

Indonesia and Brazil's GDP per capita remains far below that of their former colonizer, the Netherlands

COLONIZED, NOW CAPITALIZED

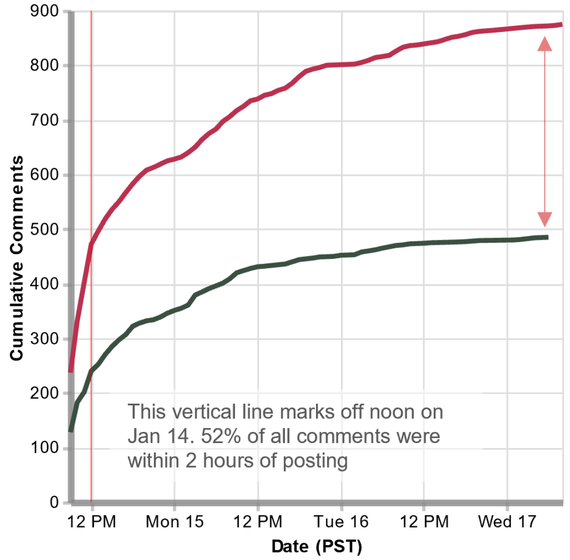
Indonesia & Brazil – previously Netherlands' colonies – now outpace economically



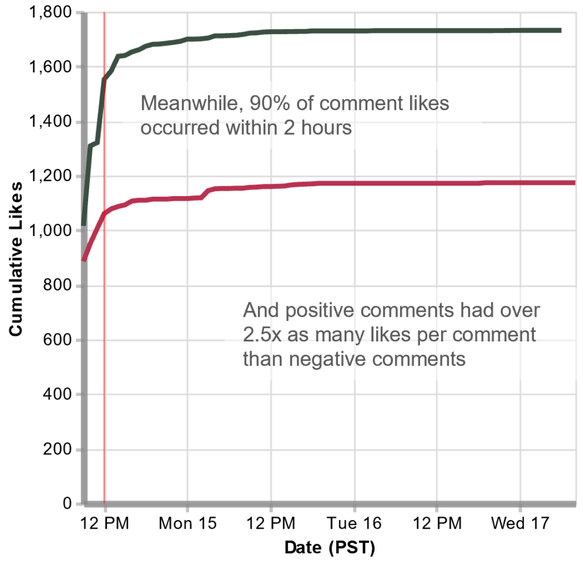
https://flaviagt.github.io/Persuasive-Deceptive-Vis/

Digital Engagement in Politics: Diving into a Facebook Comment Section

Early January, President Biden announced he created 14 million new jobs while in office. 1,360 Facebook comments on the POTUS Facebook post were analyzed with Hugging Face sentiment analysis

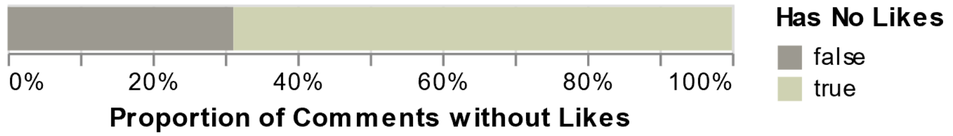


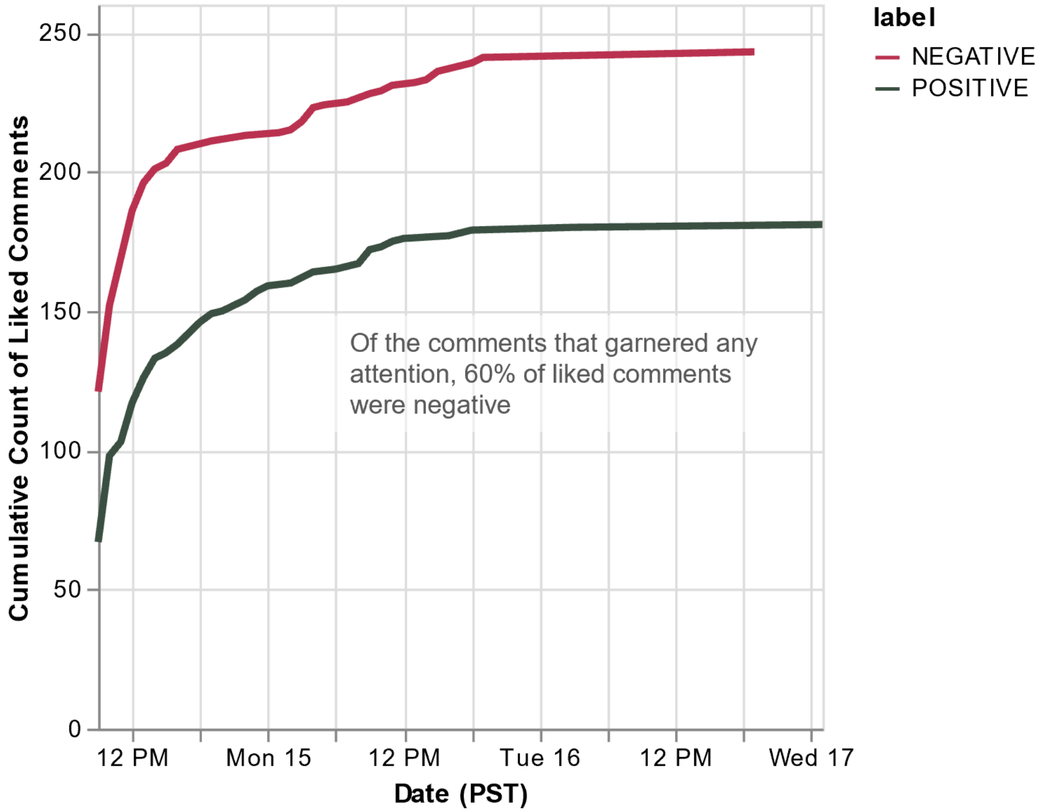
Negative comments flooded the post, reaching a peak 386 more negative comments than positive



No One Likes Biden: Diving into a Facebook comment section

Early January, President Biden announced he created 14 million new jobs while in office. 1,360 Facebook comments on the POTUS Facebook post were analyzed with Hugging Face sentiment analysis. Most comments weren't liked. Let's look at what was.

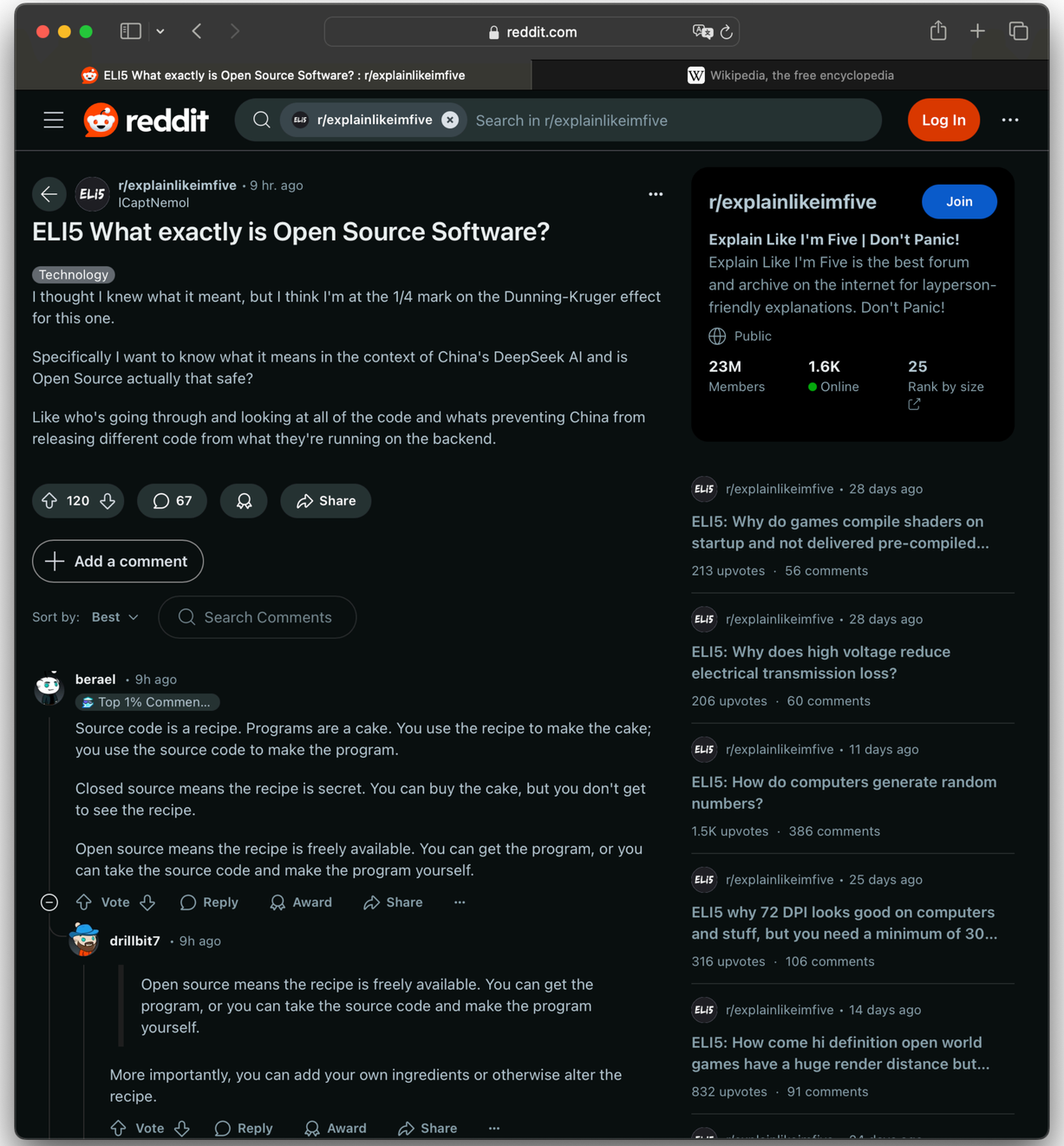




JavaScript

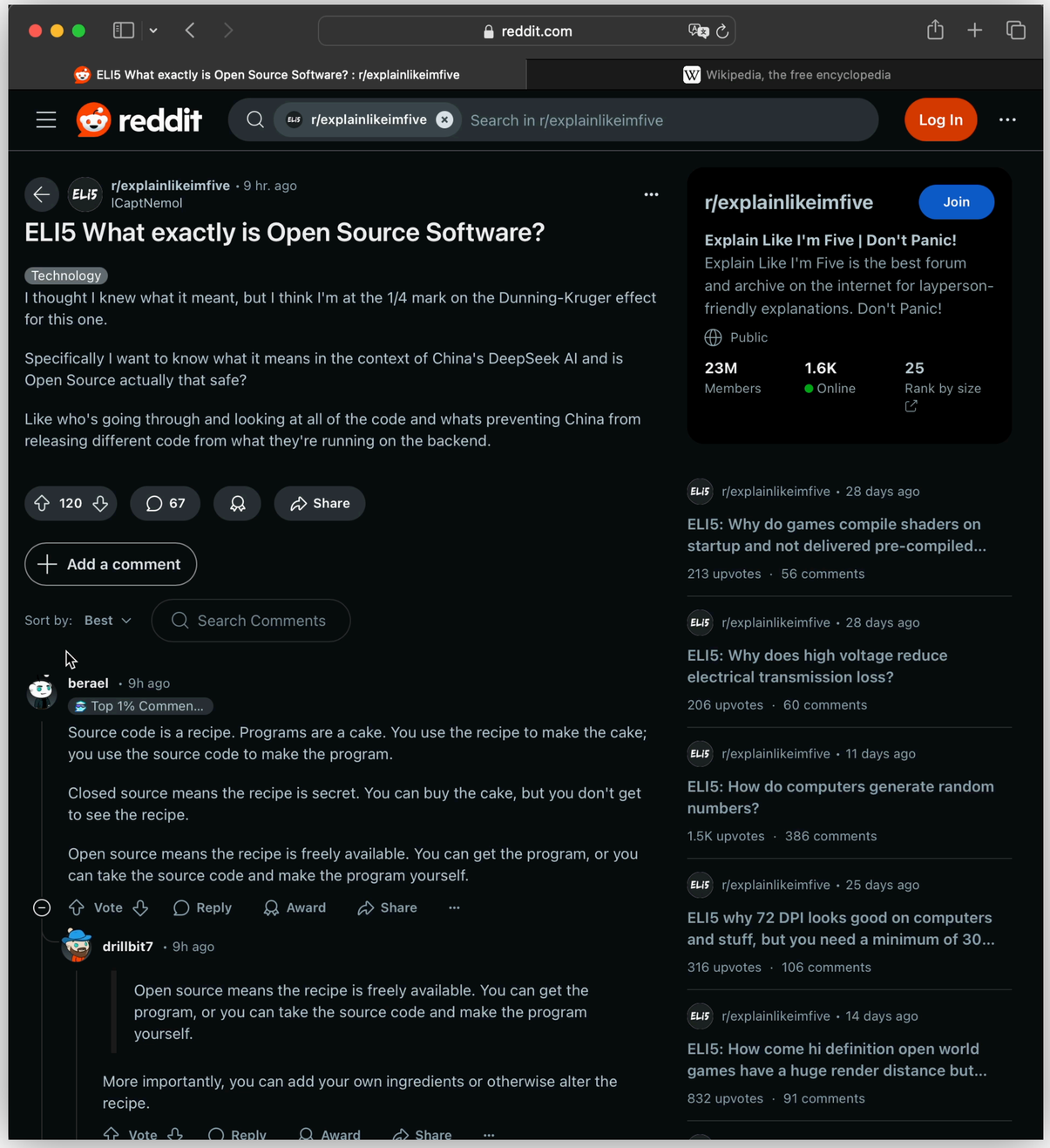
What do you find most confusing about JavaScript so far?

tryclassbuzz.com
Code: **what-is-js**



HTML defines content
(text, images, links)

CSS defines style
(layout, colors, font)



HTML defines content
(text, images, links)

CSS defines style
(layout, colors, font)

One simple mental model:
JS manipulates HTML and CSS

pandas code executes from top to bottom

Selecting columns

Selecting columns in `babypandas` 👶🐼

- In `babypandas`, you selected columns using the `.get` method.
- `.get` also works in `pandas`, but it is not **idiomatic** – people don't usually use it.

In [26]:
```
dogs
```

Out[26]:

breed	kind	lifetime_cost	longevity	size	weight	height
Brittany	sporting	22589.0	12.92	medium	35.0	19.0
Cairn Terrier	terrier	21992.0	13.84	small	14.0	10.0
English Cocker Spaniel	sporting	18993.0	11.66	medium	30.0	16.0
...	...	...	...	...	...	...
Bullmastiff	working	13936.0	7.57	large	115.0	25.5
Mastiff	working	13581.0	6.50	large	175.0	30.0
Saint Bernard	working	20022.0	7.78	large	155.0	26.5

43 rows × 6 columns

In [27]:
```
dogs.get('size')
```

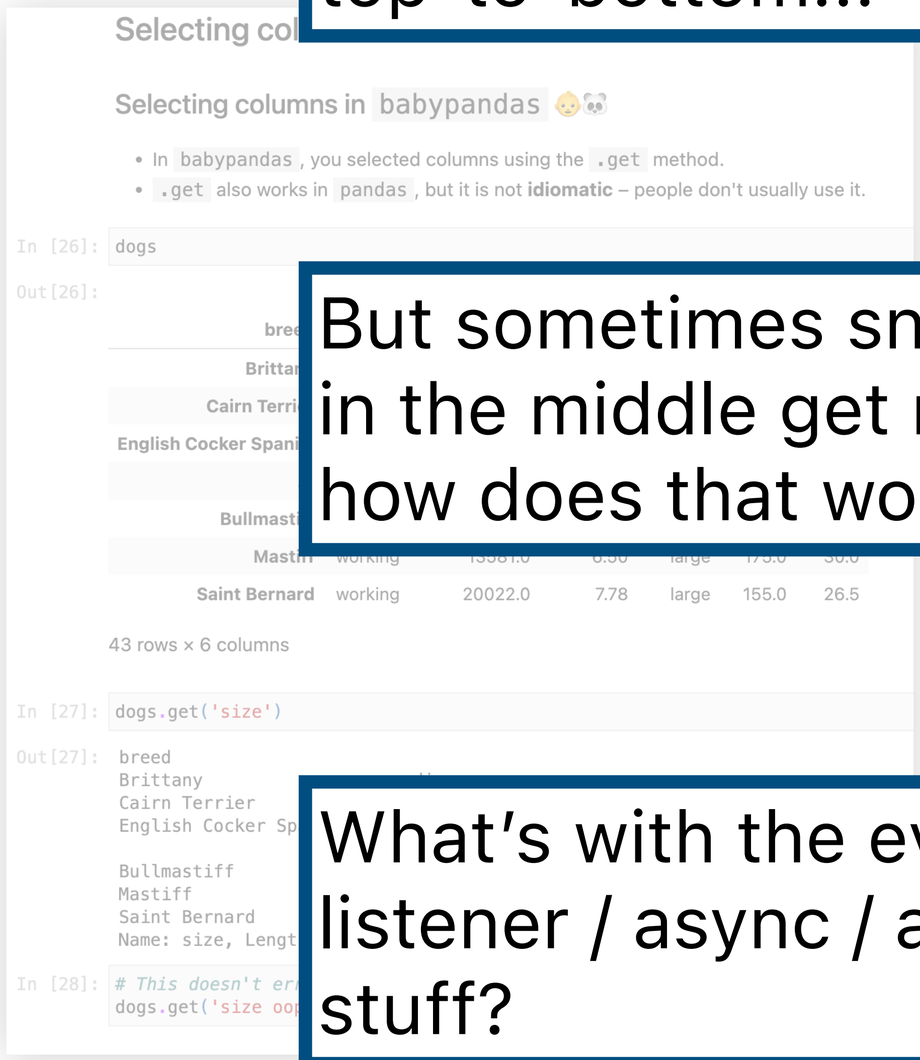
Out[27]:
```
breed
Brittany                 medium
Cairn Terrier             small
English Cocker Spaniel    medium
                           ...
Bullmastiff               large
Mastiff                   large
Saint Bernard             large
Name: size, Length: 43, dtype: object
```

In [28]:
```
# This doesn't error, but sometimes we'd like it to.
dogs.get('size oops!')
```

JS code runs once from top-to-bottom...

But sometimes snippets in the middle get re-run, how does that work?

What's with the event listener / async / await stuff?



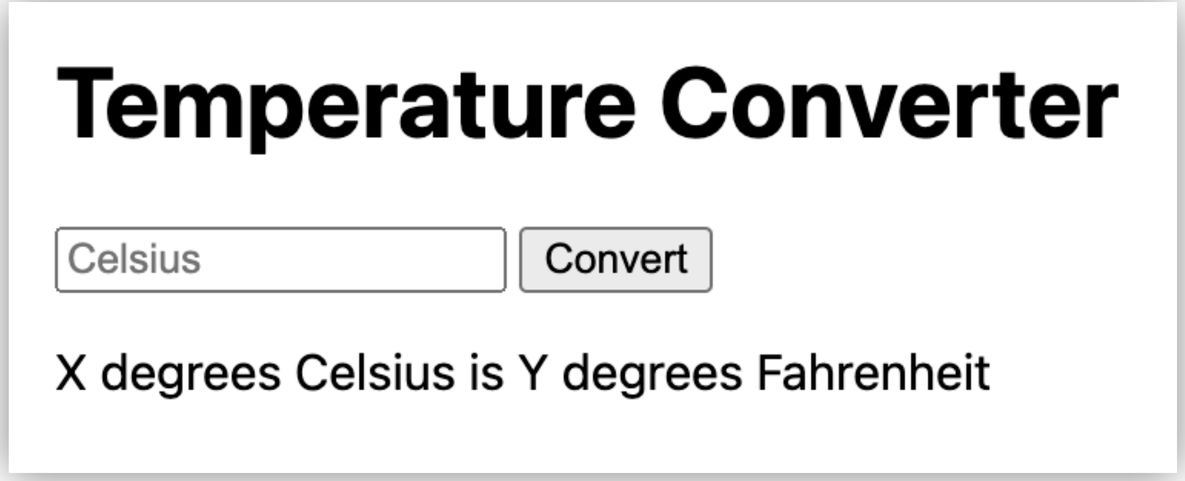

```
document.body.insertAdjacentHTML(
  'afterbegin',
  `
  <label class="color-scheme">
    Theme:
    <select>
      <option value="light dark">Automatic</option>
      <option value="light">Light</option>
      <option value="dark">Dark</option>
    </select>
  </label>`,
);

let select = document.querySelector('.color-scheme select');

if (localStorage.getItem('colorScheme')) {
  document.documentElement.style.setProperty(
    'color-scheme',
    localStorage.getItem('colorScheme'),
  );
  select.value = localStorage.getItem('colorScheme');
}

select.addEventListener('input', (event) => {
  localStorage.setItem('colorScheme', event.target.value);
  document.documentElement.style.setProperty(
    'color-scheme',
    event.target.value,
  );
});
```

Example:
Temperature Converter



Pseudocode:

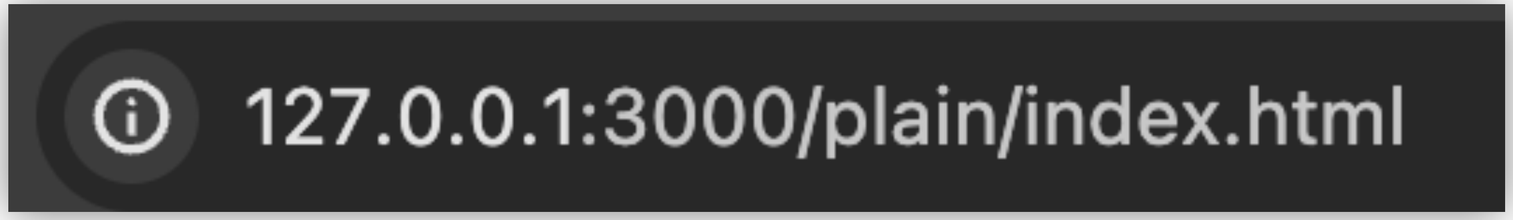
1. When we click "Convert", read value in Celsius box.
2. Convert value to F
3. Update text below box

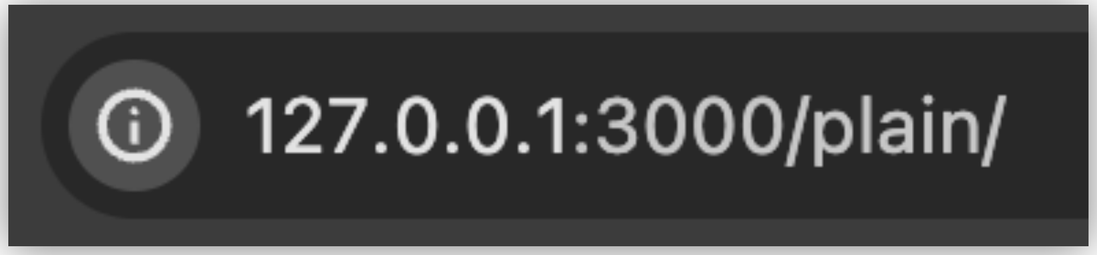
You try first, I'll walk around!

JS approach:

1. Attach event listener to Convert button. Event handler reads value from Celsius box.
2. Convert value to F
3. Replace text of the element below.

Typing a URL into address bar only asks for one HTML file (index.html in this case):





index.html is appended if it isn't in URL, so this is the same.

Download, then render index.html

Download, then render index.html

HTML is also "executed" top-to-bottom:

```html
<html>
  <head>
    <title>Temperature Converter</title>

    <link rel="stylesheet" href="main.css" />
    <script src="main.js"></script>
  </head>
  <body>
    <h1>Temperature Converter</h1>

    <div id="converter">
      <input type="text" id="celsius" placeholder="Celsius" />
      <button id="submit" type="submit">Convert</button>
    </div>

    <div id="result">X degrees Celsius is Y degrees Fahrenheit</div>
  </body>
</html>
```

Download and run the main.css file

Download and run the main.js file

Render HTML to screen

Download, then render index.html

HTML is also "executed" top-to-bottom:

```
<html>
  <head>
    <title>Temperature Converter</title>

    <link rel="stylesheet" href="main.css" />
    <script src="main.js"></script>
  </head>
  <body>
    </div>

    <div id="result">X degrees Celsius is Y degrees Fahrenheit</div>
  </body>
</html>
```

Download and run the main.css file

Download and run the main.js file

What if these files are really big, or JS has an infinite loop??

Browser waits!

Download, then render index.html

HTML is also "executed" top-to-bottom:

```html
<html>
  <head>
    <title>Temperature Converter</title>

    <link rel="stylesheet" href="main.css" />
    <script src="main.js"></script>
  </head>
  <body>
    <h1>Temperature Converter</h1>

    <div id="converter">
      <input type="text" id="celsius" placeholder="Celsius" />
      <button id="submit" type="submit">Convert</button>
    </div>

    <div id="result">X degrees Celsius is Y degrees Fahrenheit</div>
  </body>
</html>
```

Download and run the main.css file

Download and run the main.js file

Render HTML to screen

`js-lecture/plain01/`

(demo)

```html
<html>
  <head>
    <title>Temperature Converter</title>

    <link rel="stylesheet" href="main.css" />
    <script src="main.js"></script>
  </head>
  <body>
    <h1>Temperature Converter</h1>

    <div id="converter">
      <input type="text" id="celsius"
      <button id="submit" type="submi
    </div>

    <div id="result">X degrees Celsius is Y degrees Fahrenheit</div>
  </body>
</html>
```

What happened?

```js
const button = document.getElementById('submit');

button.addEventListener('click', (event) => {
  event.preventDefault();
  console.log(event);
});
```

Runs before rest of HTML loads. There are no HTML elements in document!

`js-lecture/plain02/`

(demo)

JS Modules

```
<link rel="stylesheet" href="main.css" />
<script src="main.js"></script>
<script src="main2.js"></script>
<script src="main3.js"></script>
<script src="main4.js"></script>
<script src="main5.js"></script>
```

Loading multiple JS files was a pain back in the day!

No import/export syntax = all global variables

JS Modules

```
<link rel="stylesheet" href="main.css" />
<script src="main.js" type="module"></script>
<script src="main2.js" type="module"></script>
<script src="main3.js" type="module"></script>
<script src="main4.js" type="module"></script>
<script src="main5.js" type="module"></script>
```

Now, files run AFTER HTML loads

Import/export syntax = no global variables

Don’t need to worry about order

`js-lecture/plain02/`

(demo)

Let's walk through the code line by line

```
const button = document.querySelector('#submit');

button.addEventListener('click', (event) => {
  const celsiusTag = document.querySelector('#celsius');
  const celsius = celsiusTag.value;
  const fah = (celsius * 9) / 5 + 32;

  const result = document.querySelector('#result');
  result.innerText = `${celsius} degrees Celsius is ${fah} degrees Fahrenheit.`;
});
```

```
const button = document.querySelector('#submit');

button.addEventListener('click', (event) => {
  const celsius
  const celsius
  const fah = (

  const result = document.querySelector('#result');
  result.innerText = `${celsius} degrees Celsius is ${fah} degrees Fahrenheit.`;
});
```

Now we have an HTML element.

```
const button = document.querySelector('#submit');

button.addEventListener('click', (event) => {
  const celsiusTag = document.querySelector('#celsius');
  const celsius = celsiusTag.value;
  const fah = (celsiu

  const result = docu
  result.innerText = `${celsius} degrees Celsius is ${fah} degrees Fahrenheit.`;
});
```

When button is clicked, called this function

```
const button = document.querySelector('#submit');

button.addEventListener('click', (event) => {
  const celsiusTag = document.querySelector('#celsius');
  const celsius = celsiusTag.value;
  const fah = (celsius * 9) / 5 + 32;

  const result = document
  result.innerText = `${celsius} degrees Celsius is ${fah} degrees Fahrenheit.`;
});
```

Find the celsius HTML element

Get its value, then convert to F

```
const button = document.querySelector('#submit');

button.addEventListener('click', (event) => {
  const celsiusTag = document.querySelector('#celsius');
  const celsius = celsiusTag.value;
  const fah = (celsius * 9) / 5 + 32;

  const result = document.querySelector('#result');
  result.innerText = `${celsius} degrees Celsius is ${fah} degrees Fahrenheit.`;
});
```

Get the result HTML element

And set its text.

```
const button = document.querySelector('#submit');

button.addEventListener('click', (event) => {
  const celsiusTag = document.querySelector('#celsius');
  const celsius = celsiusTag.value;
  const fah = (celsius * 9) / 5 + 32;

  const result = document.querySelector('#result');
  result.innerText = `${celsius} degrees Celsius is ${fah}`;
});
```

JS code always runs top-to-bottom, but we use event handlers to **delay execution**...

And to **rerun** code in response to user input.

```
const button = document.querySelector('#submit');

button.addEventListener('click', (event) => {
  con                                          ius');
  con
  con

  con                                          ;
  result.innerText = `${celsius} degrees Celsius is ${fah} degrees Fahrenheit.`;
});
```

This event fires every time a user clicks the button, so this function can get called many times.

Enough JS to be dangerous

Querying HTML

`document.querySelector()`	Returns first element that match
`document.querySelectorAll()`	Returns list of elements that match

Mutating HTML

`el.innerText = 'hello'`	Changes text of element
`el.innerHTML = '<p>hello</p>'`	Changes HTML of element

Enough JS to be dangerous

Event listeners

`el.addEventListener('click', fn)`	Runs fn when element is clicked
`el.addEventListener('keydown', fn)`	Runs fn when a keyboard key is pressed
`el.addEventListener('input', fn)`	Runs fn when input changes

Example: Collapsing comments

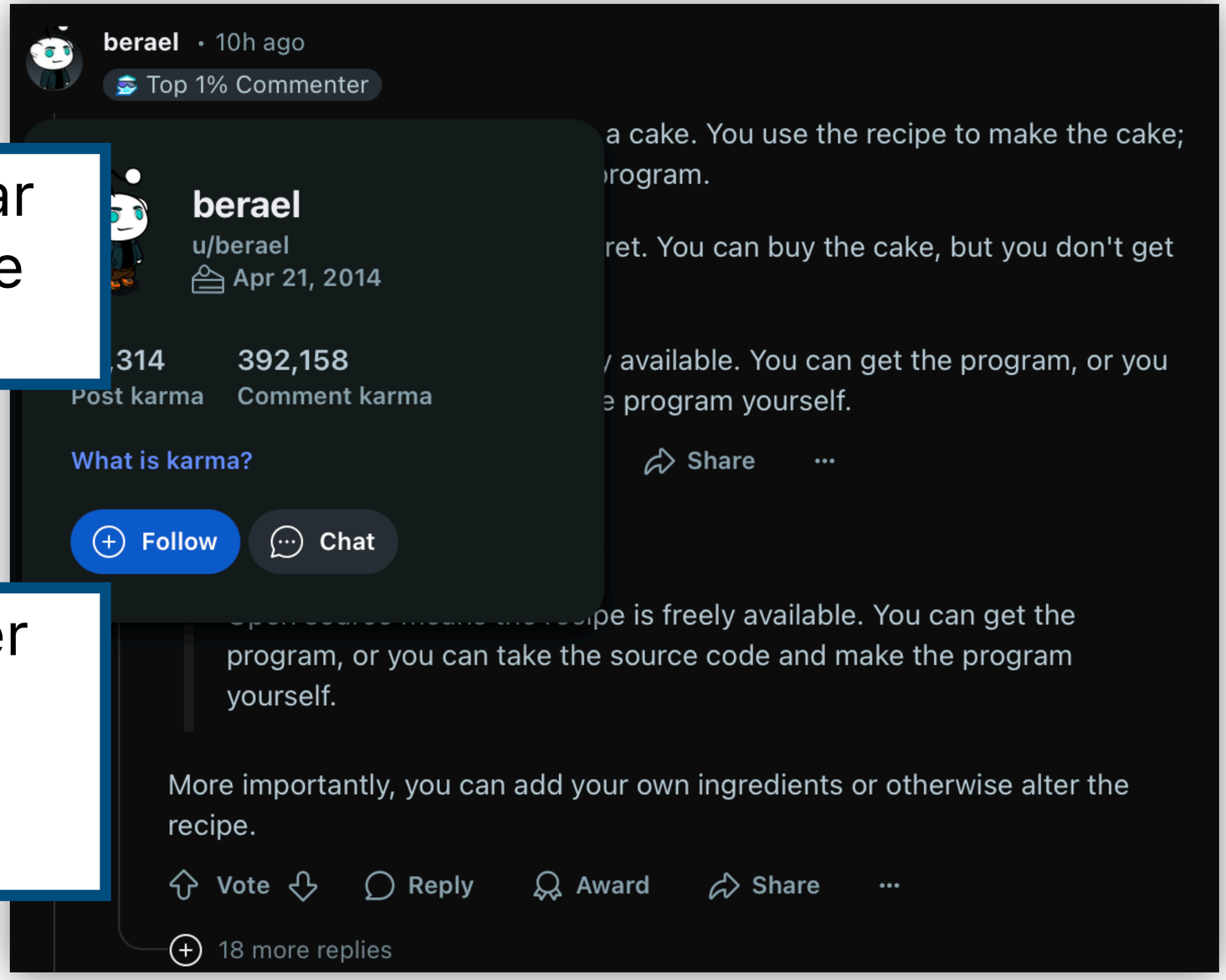
Add event listener to minus sign element for click events.

In event handler, hide HTML by adding CSS, or by adding an HTML class like “hidden” which CSS will hide.

Example: Avatar hover

Add event listener to avatar image element for a mouse hover event.

In event handler, fetch user data from the server, then display the information in HTML.



You Try: Your favorite website interaction

Go to your favorite website, pick an interactive element, describe event listener and event handler.

tryclassbuzz.com
Code: **interaction**

Async / await

`js-lecture/weather01/`

(demo)

```
<head>
  <title>Temperature Converter</title>

  <link rel="stylesheet" href="main.css" />
  <script src="main.js" type="module"></script>
```

If main.js code is slow, then page will freeze (!) while waiting for JS to finish

What if the dataset just takes a while to download?

Idea: Allow functions to run in the background (asynchronously) so that page doesn't freeze

```
async function loadWeatherData() {
  try {
    const response = await fetch('./weather-data.json');
    const weatherData = await response.json();
    return weatherData;
  } catch (error) {
    console.error('Error loading weather data:', error);
  }
}
```

async = this function uses other async functions

```
async function loadWeatherData() {
  try {
    const response = await fetch('./weather-data.json');
    const weatherData = await response.json();
    return weatherData;
  } catch (error) {
    console.error('Error loading weather data:', error);
  }
}
```

async = this function uses other async functions

await = this function might take a while, so let the browser do other stuff while we wait

```
function loadStory() {
  return getJSON('story.json')
    .then(function (story) {
      addHtmlToPage(story.heading);

      return story.chapterURLs
        .map(getJSON)
        .reduce(function (chain, chapterPromise) {
          return chain
            .then(function () {
              return chapterPromise;
            })
            .then(function (chapter) {
              addHtmlToPage(chapter.html);
            });
        }, Promise.resolve());
    })
    .then(function () {
      addTextToPage('All done');
    })
    .catch(function (err) {
      addTextToPage('Argh, broken: ' + err.message);
    })
    .then(function () {
      document.querySelector('.spinner').style.display = 'none';
    });
}
```

Back in the day, we had to use JS Promises that had a .then() and .catch() syntax.

async/await is the modern version that makes writing this code a LOT easier

Half the code!

https://jakearchibald.com/2014/es7-async-functions/

```
async function loadStory() {
  try {
    let story = await getJSON('story.json');
    addHtmlToPage(story.heading);
    for (let chapter of story.chapterURLs.map(getJSON)) {
      addHtmlToPage((await chapter).html);
    }
    addTextToPage('All done');
  } catch (err) {
    addTextToPage('Argh, broken: ' + err.message);
  }
  document.querySelector('.spinner').style.display = 'none';
}
```

Now, let’s make our very first data visualization in JS:

`js-lecture/weather02/`

(demo)

`js-lecture/weather03/`

(demo)

How would you add an x-axis and y-axis? Gridlines?

tryclassbuzz.com
Code: **axes**