

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

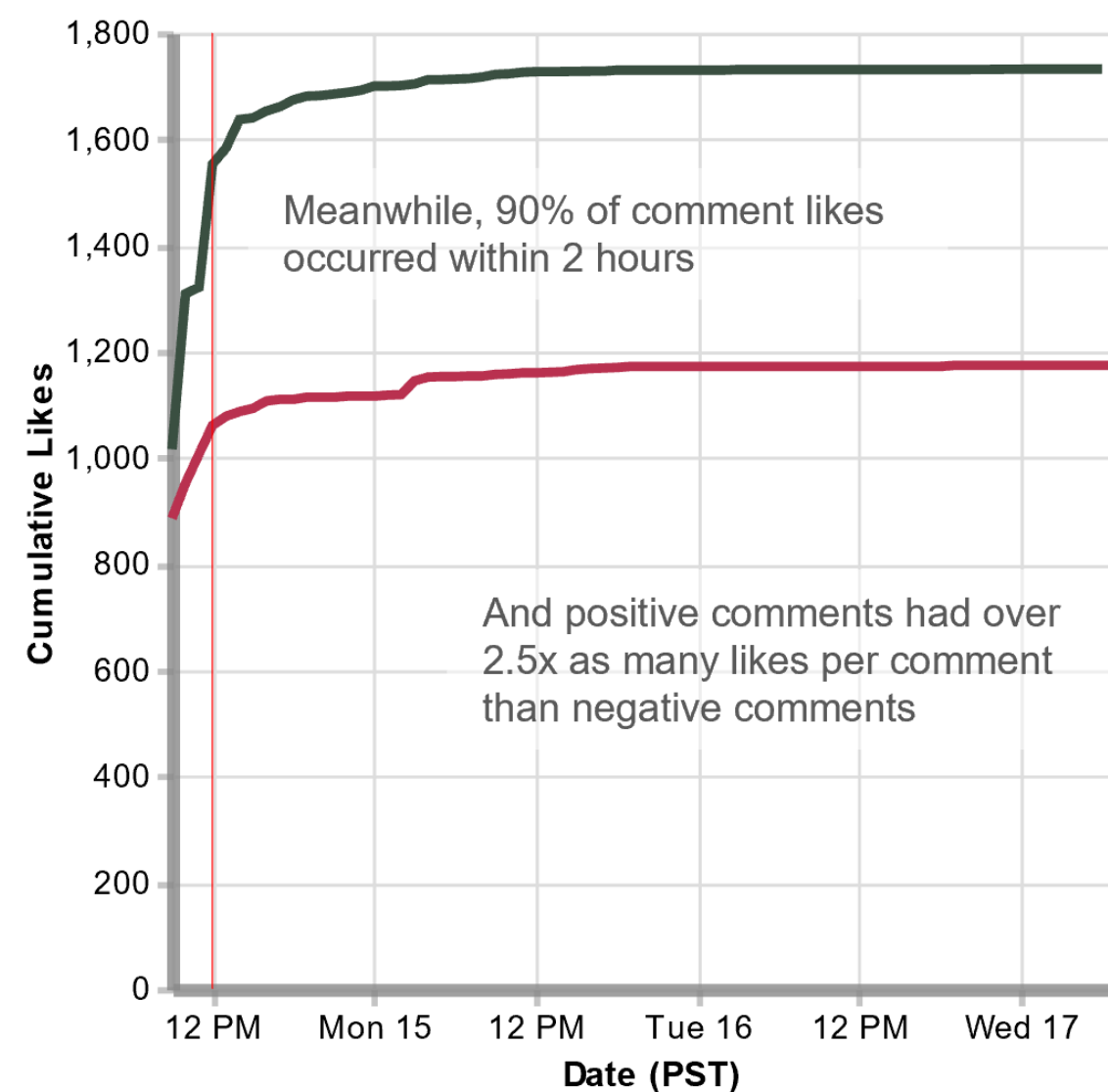
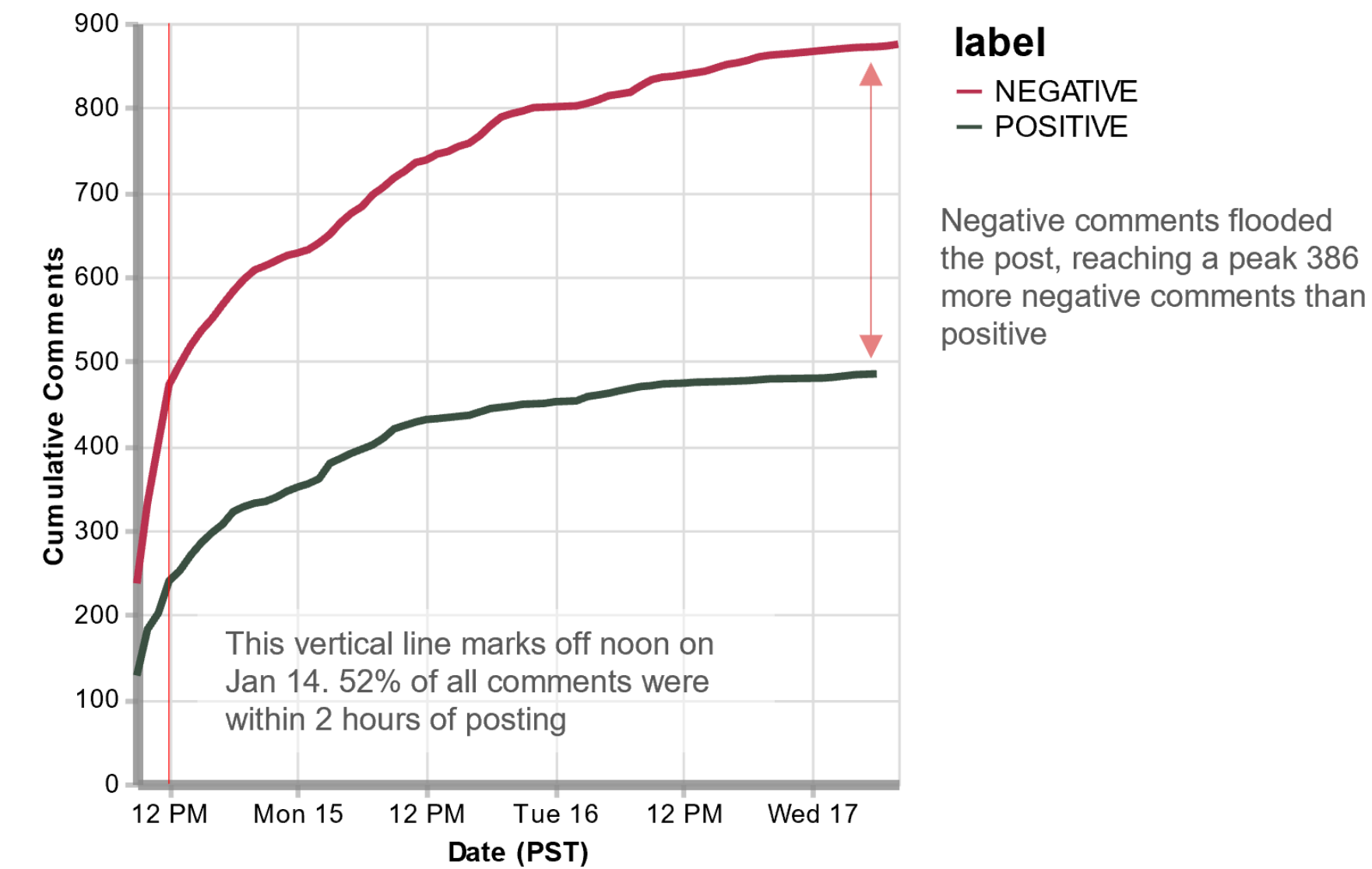
FAQs:

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, but can help.**

Nifty Project 2 Submissions (from the past)

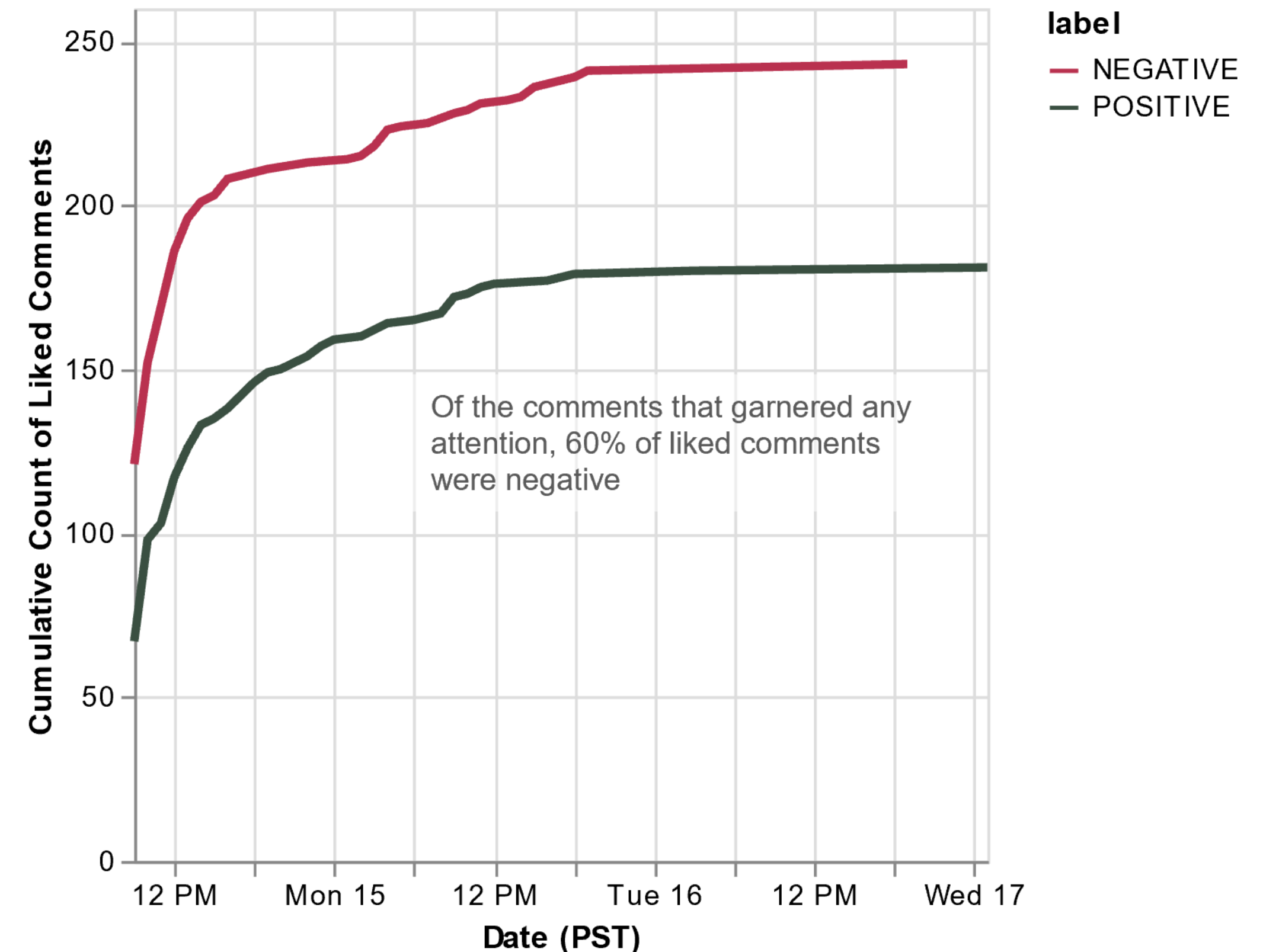
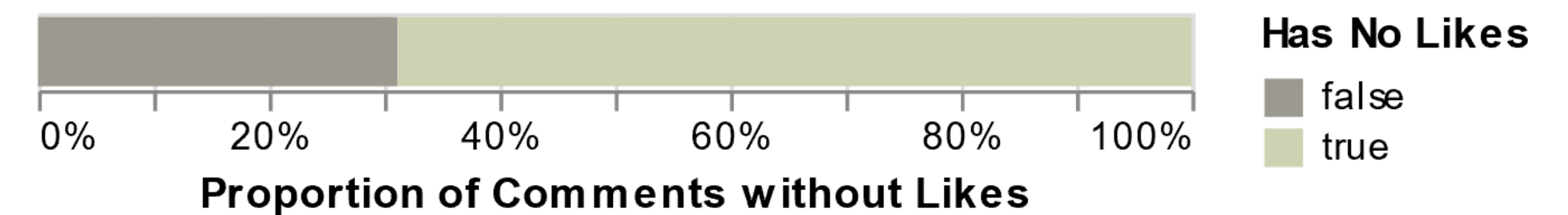
Digital Engagement in Politics: Diving into a Facebook Comment Section

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



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.

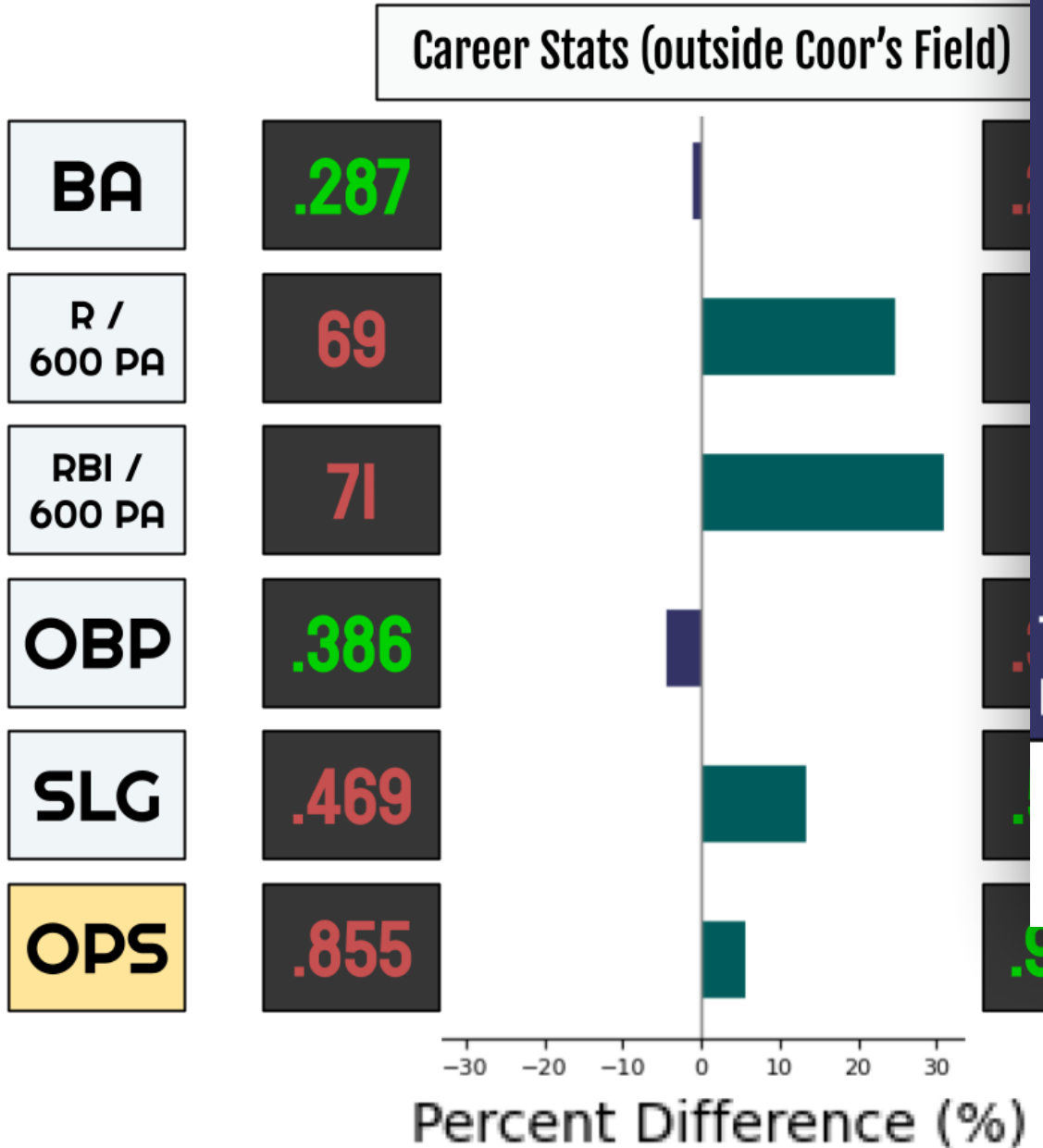
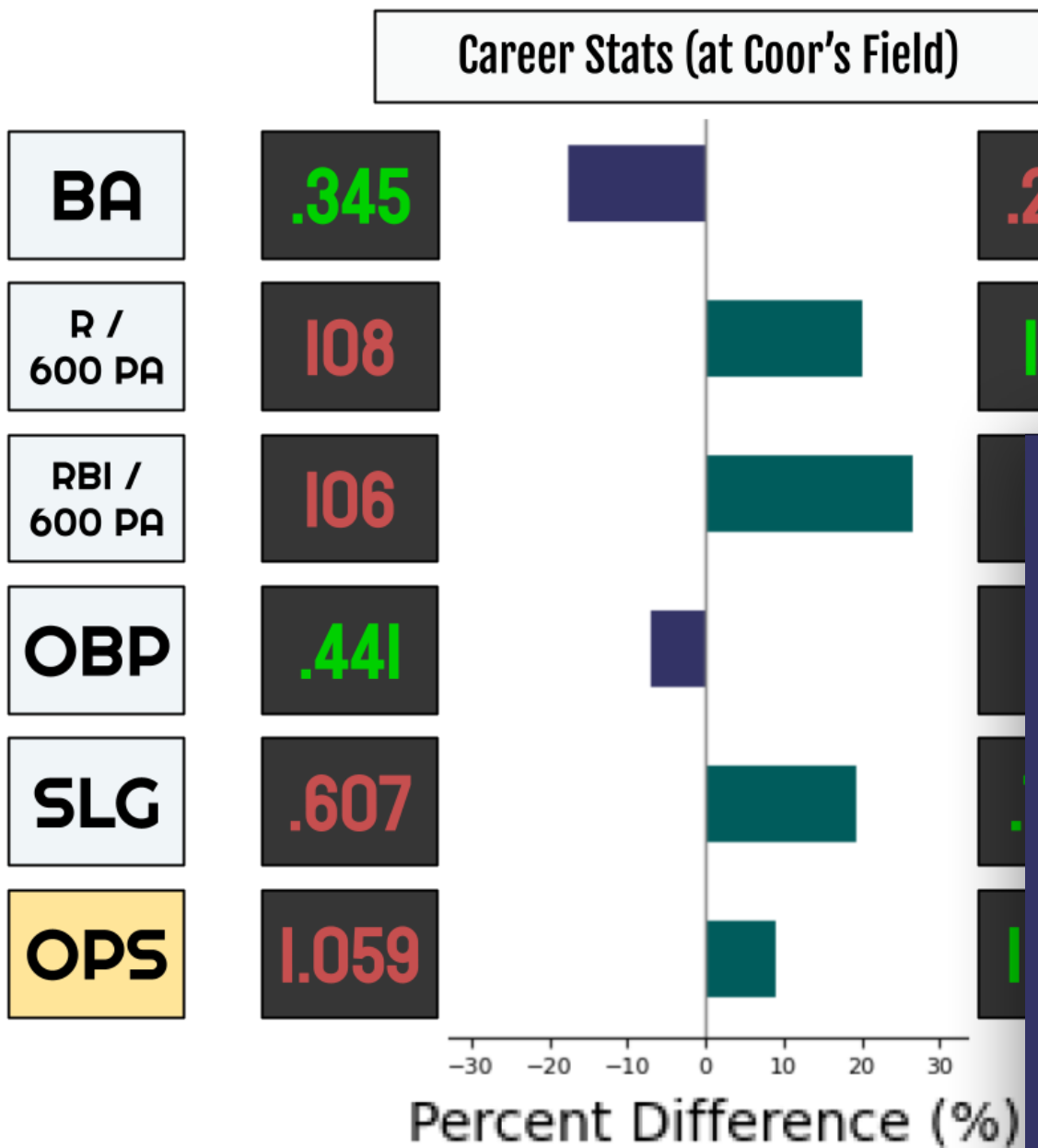


TODD HELTON VS KEN GRIFFEY JR:

WHO'S THE GREATEST LEFTY OF THE GENERATION?



TODD
HELTON



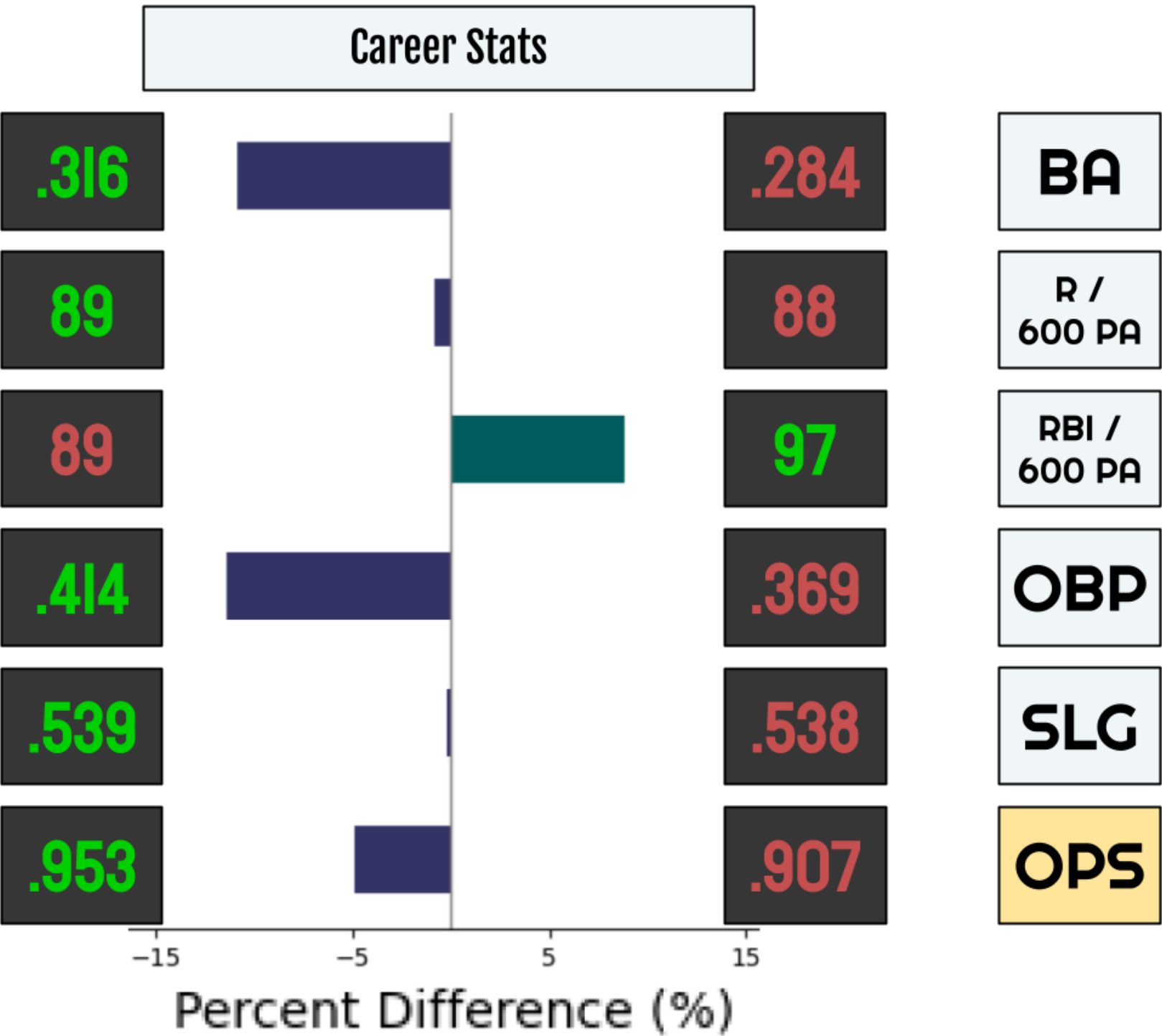
TODD
HELTON



TODD HELTON:

GREATEST LEFTY OF THE GENERATION?

TODD HELTON **WINS**
5 OF 6 MAJOR
CATEGORIES,
INCLUDING OPS,
ACROSS THEIR
CAREERS

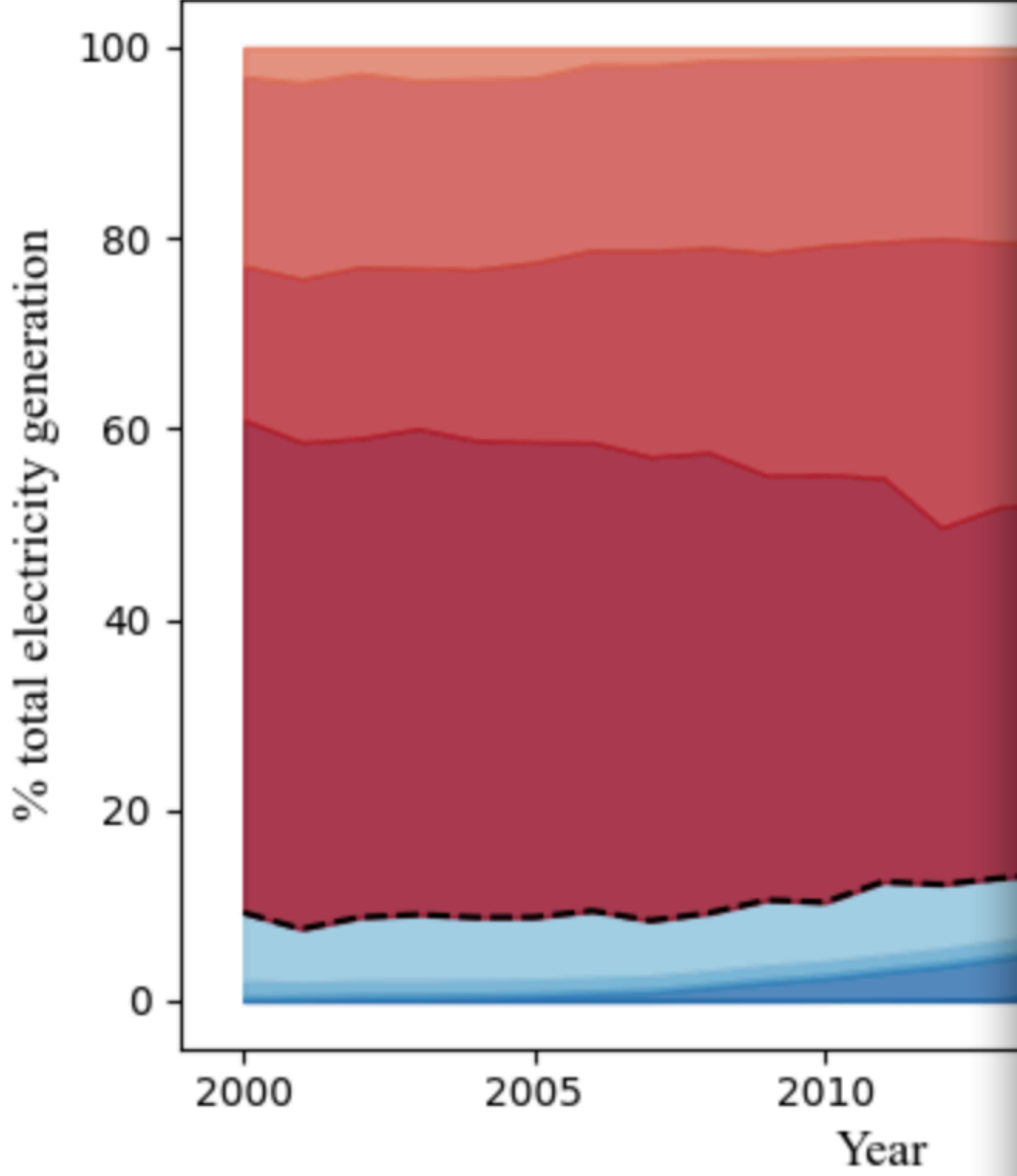


KEN
GRIFFEY JR.

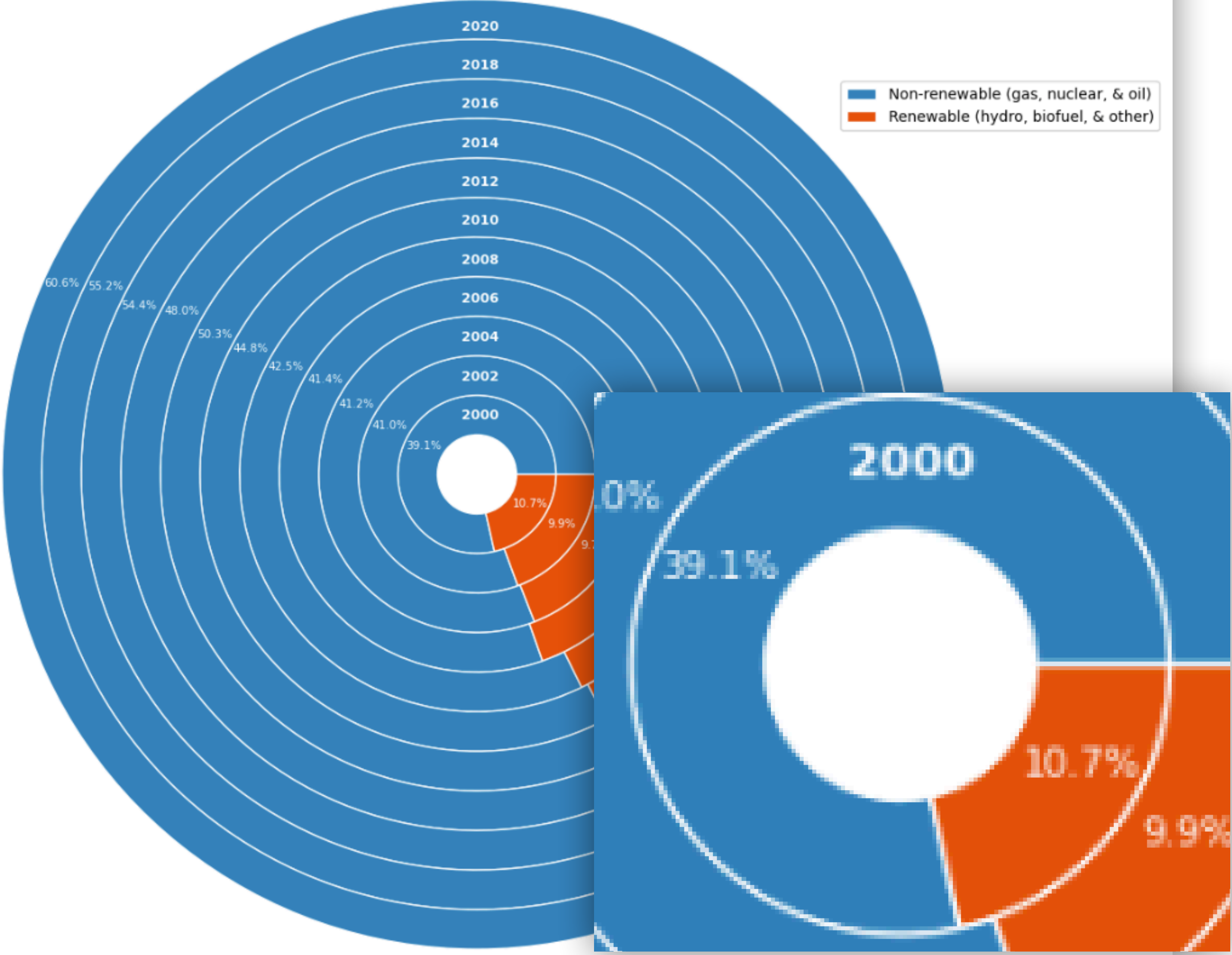
DATA FROM BASEBALLREFERENCE.COM

DATA FROM BASEBALLREFERENCE.COM

Has the proportion of total electricity in the US generated by renewable sources increased during the 21st century?



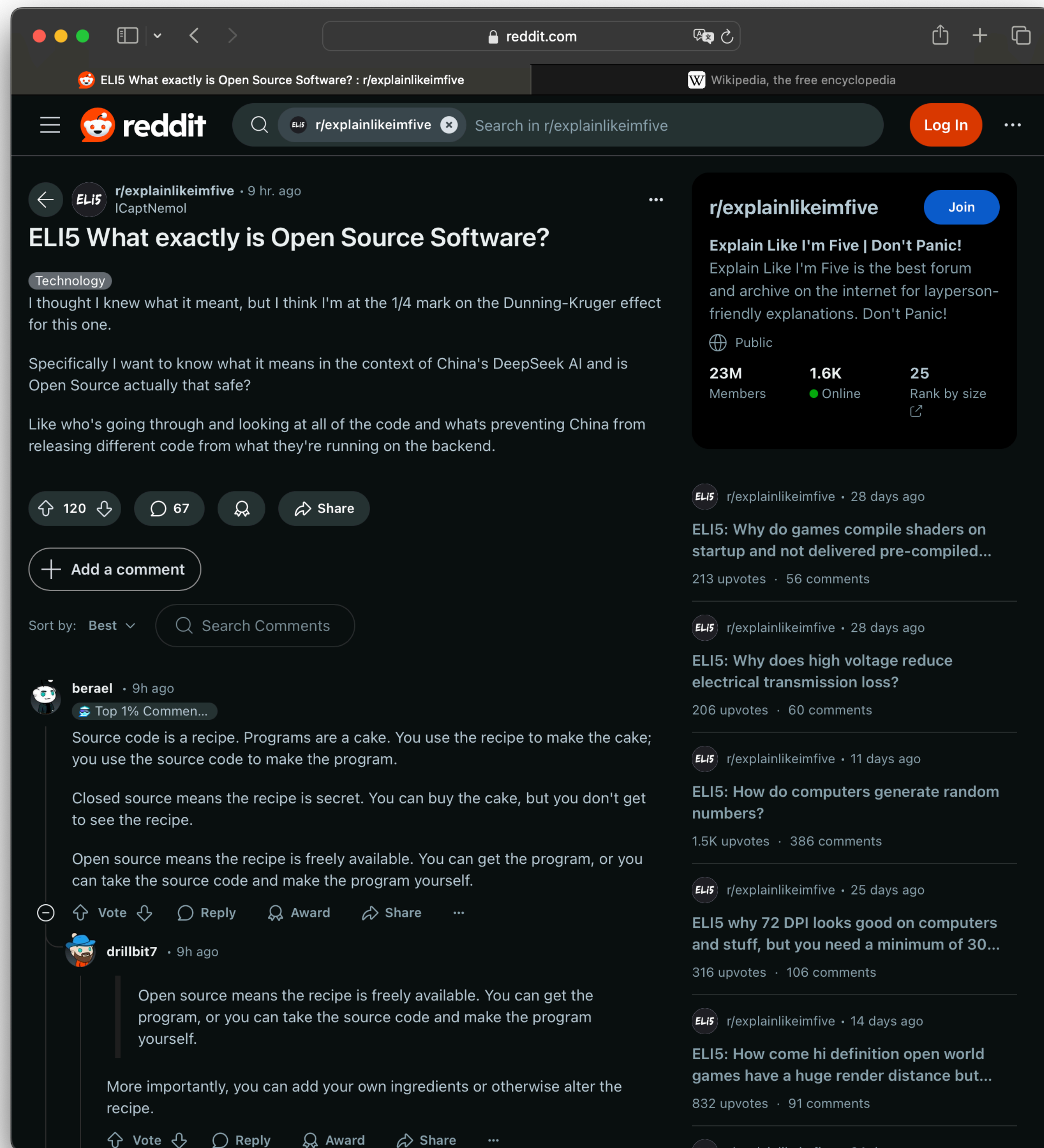
Has the proportion of total electricity in the US generated by renewable sources increased during the 21st century?



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)

reddit.com

ELI5 What exactly is Open Source Software? : r/explainlikeimfive

Wikipedia, the free encyclopedia

reddit

r/explainlikeimfive

Log In

r/explainlikeimfive • 9 hr. ago

ICaptNemol

ELI5 What exactly is Open Source Software?

Technology

I thought I knew what it meant, but I think I'm at the 1/4 mark on the Dunning-Kruger effect for this one.

Specifically I want to know what it means in the context of China's DeepSeek AI and is Open Source actually that safe?

Like who's going through and looking at all of the code and whats preventing China from releasing different code from what they're running on the backend.

120 Upvotes 67 Comments

Add a comment

Sort by: Best

Search Comments

berael • 9h ago

Top 1% Comment

Source code is a recipe. Programs are a cake. You use the recipe to make the cake; you use the source code to make the program.

Closed source means the recipe is secret. You can buy the cake, but you don't get to see the recipe.

Open source means the recipe is freely available. You can get the program, or you can take the source code and make the program yourself.

drillbit7 • 9h ago

Open source means the recipe is freely available. You can get the program, or you can take the source code and make the program yourself.

More importantly, you can add your own ingredients or otherwise alter the recipe.

r/explainlikeimfive

Join

Explain Like I'm Five | Don't Panic!

Explain Like I'm Five is the best forum and archive on the internet for layperson-friendly explanations. Don't Panic!

Public

23M Members 1.6K Online 25 Rank by size

r/explainlikeimfive • 28 days ago

ELI5: Why do games compile shaders on startup and not delivered pre-compiled...

213 upvotes • 56 comments

r/explainlikeimfive • 28 days ago

ELI5: Why does high voltage reduce electrical transmission loss?

206 upvotes • 60 comments

r/explainlikeimfive • 11 days ago

ELI5: How do computers generate random numbers?

1.5K upvotes • 386 comments

r/explainlikeimfive • 25 days ago

ELI5 why 72 DPI looks good on computers and stuff, but you need a minimum of 30...

316 upvotes • 106 comments

r/explainlikeimfive • 14 days ago

ELI5: How come hi definition open world games have a huge render distance but...

832 upvotes • 91 comments

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]:
```

	kind	lifetime_cost	longevity	size	weight	height
breed						
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 x 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...

Selecting col

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						
Brittan						
Cairn Terri						
English Cocker Spani						
Bullmast						
Mastin	working	13581.0	0.50	large	178.0	30.0
Saint Bernard	working	20022.0	7.78	large	155.0	26.5

43 rows x 6 columns

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

Out[27]: breed
Brittany
Cairn Terrier
English Cocker Sp

Bullmastiff
Mastiff
Saint Bernard
Name: size, Length

In [28]: `# This doesn't error
dogs.get('size oop`

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

Temperature Converter

X degrees Celsius is Y degrees Fahrenheit

Pseudocode:

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

(demo)

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 <div> element below.

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

127.0.0.1:3000/plain/index.html

127.0.0.1:3000/plain/

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>
  <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>
```

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!

```
    </div>

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

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>
  <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>
  <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="submit">Submit</button>
    </div>

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

```

What happened?

```

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 = (fah - 32) * 5 / 9;  
  const celsius = Math.round(celsius);  
  const fah = (celsius * 9 / 5) + 32;  
  const fah = Math.round(fah);  
  const result = document.querySelector('#result');  
  result.innerText = `${celsius} degrees Celsius is ${fah} degrees Fahrenheit.`;  
});
```

Now we have an HTML element.

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



```
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.`;
});
```

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;
```

Find the celsius HTML element

Get its value, then convert to F

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

```
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) => {
```

```
  const celsius = document.querySelector('#celsius').value;  
  const fahrenheit = (celsius * 9 / 5) + 32;  
  const result = document.querySelector('#result');  
  result.innerHTML = `${celsius} degrees Celsius is ${fahrenheit} 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

```
e1.innerText = 'hello'
```

Changes text of element

```
e1.innerHTML = '<p>hello</p>'
```

Changes HTML of element

Enough JS to be dangerous

Event listeners

`e1.addEventListener('click', fn)` Runs fn when element is clicked

`e1.addEventListener('keydown', fn)` Runs fn when a keyboard key is pressed

`e1.addEventListener('input', fn)` Runs fn when input changes

Example: Collapsing comments

Add event listener to minus sign element for click events.

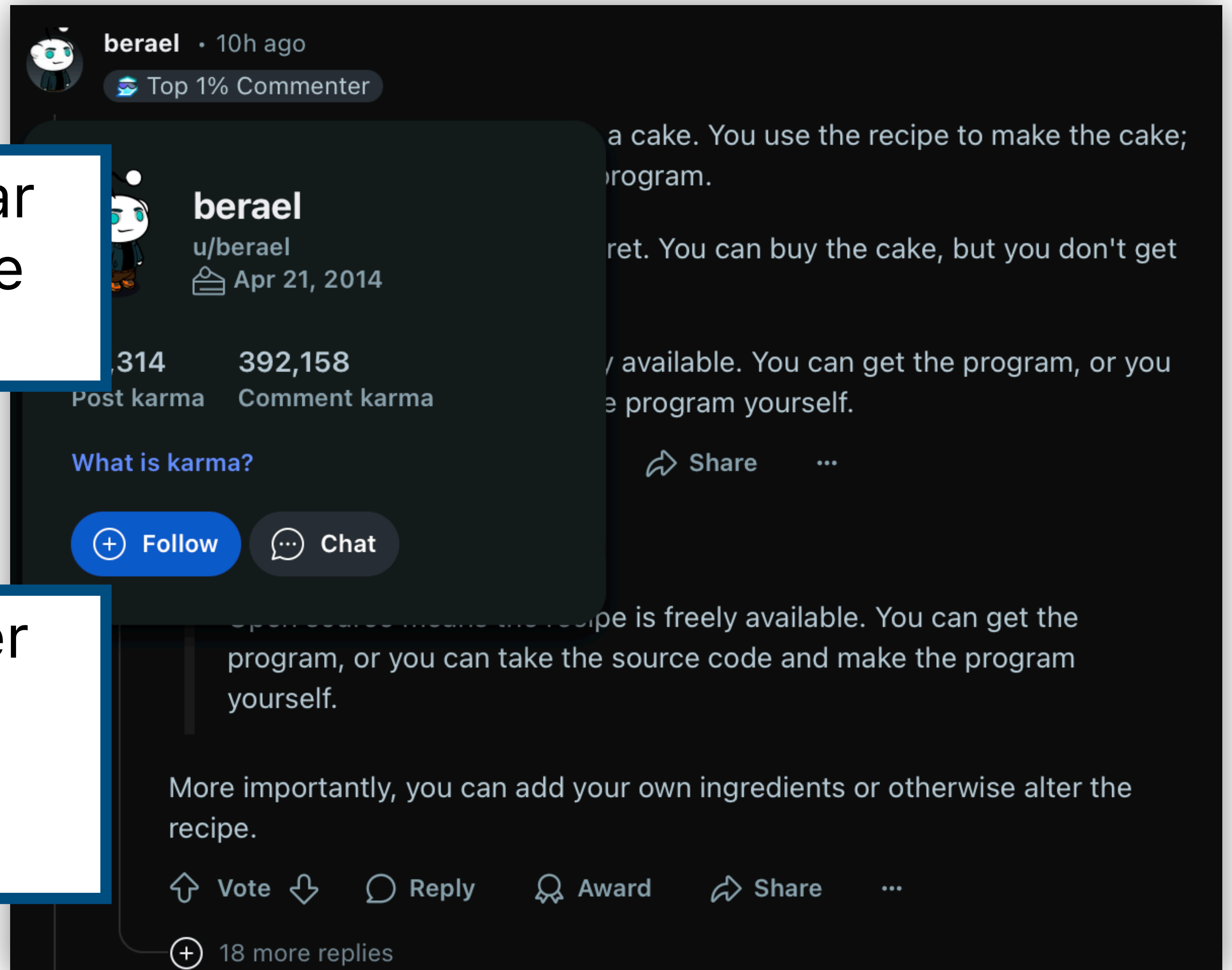
In event handler, hide HTML by adding CSS, or by adding an HTML class like "hidden" which CSS will hide.



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?

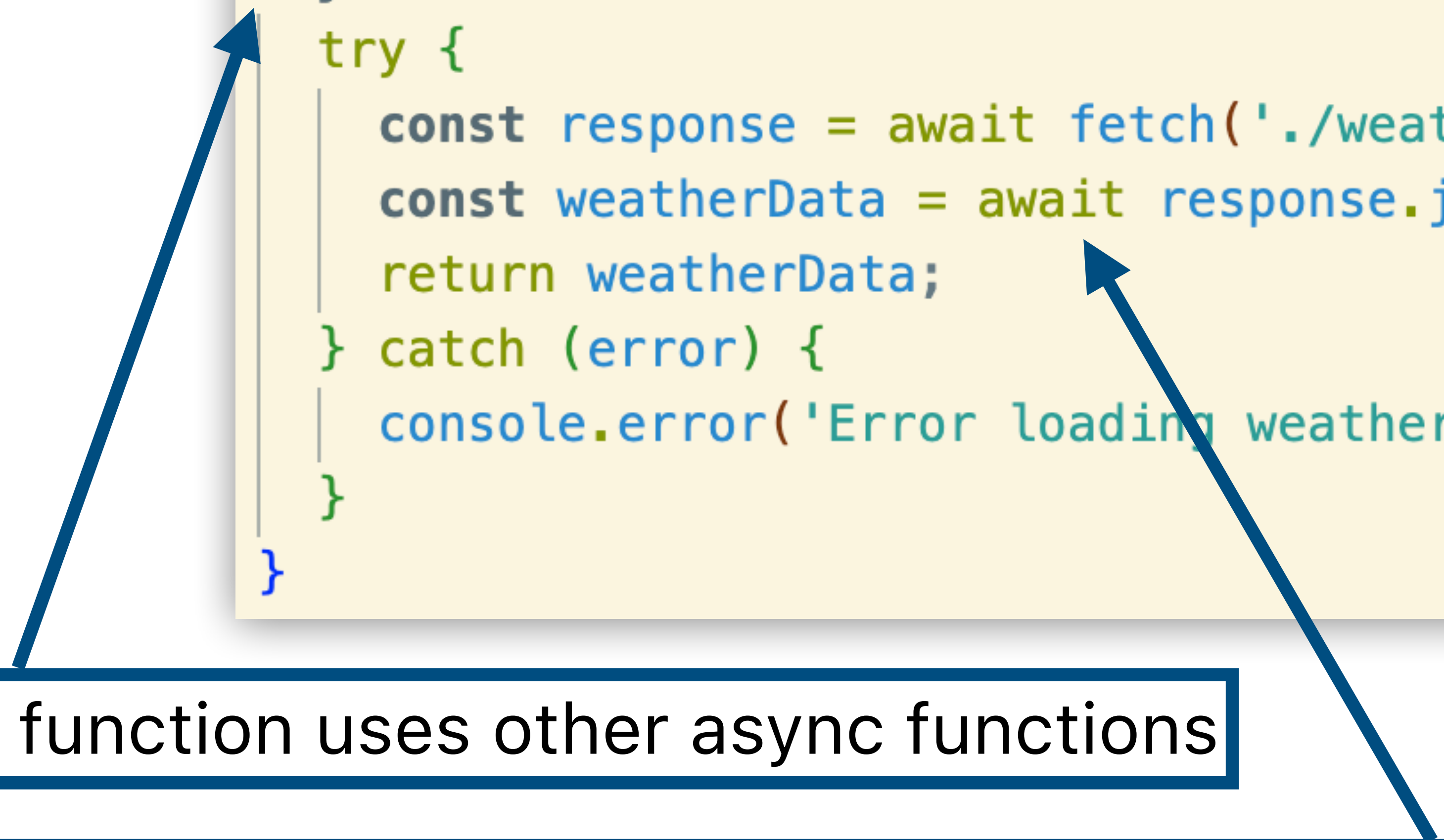
Idea: Allow functions to run in the background
(asynchronously) so that page doesn't freeze



```
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);  
  }  
}
```

Two blue arrows originate from the explanatory boxes below. One arrow points from the 'async' keyword in the function signature to the box explaining its meaning. The other arrow points from the 'await' keyword in the 'fetch' call to the box explaining its meaning.

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!

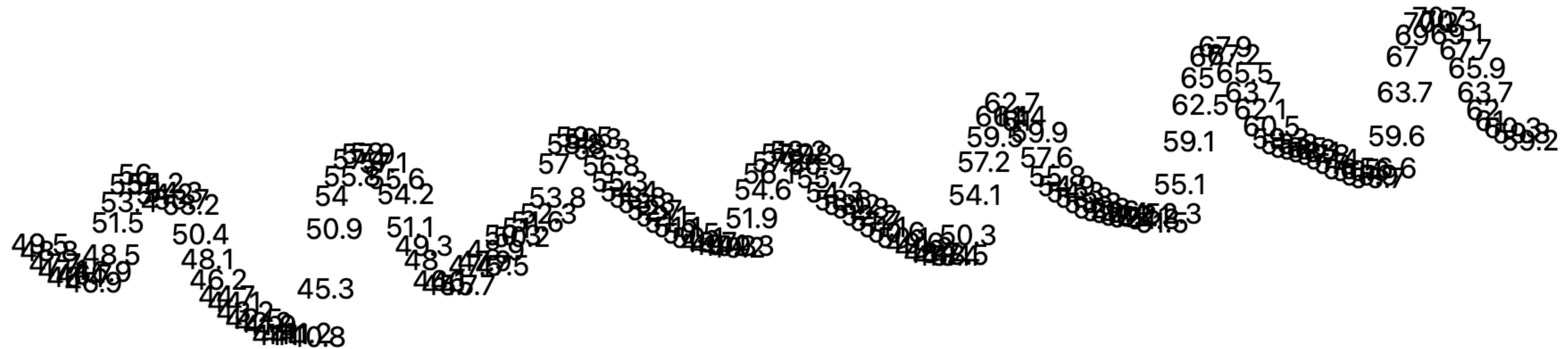
<https://jakearchibald.com/2014/es7-async-functions/>

```

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:

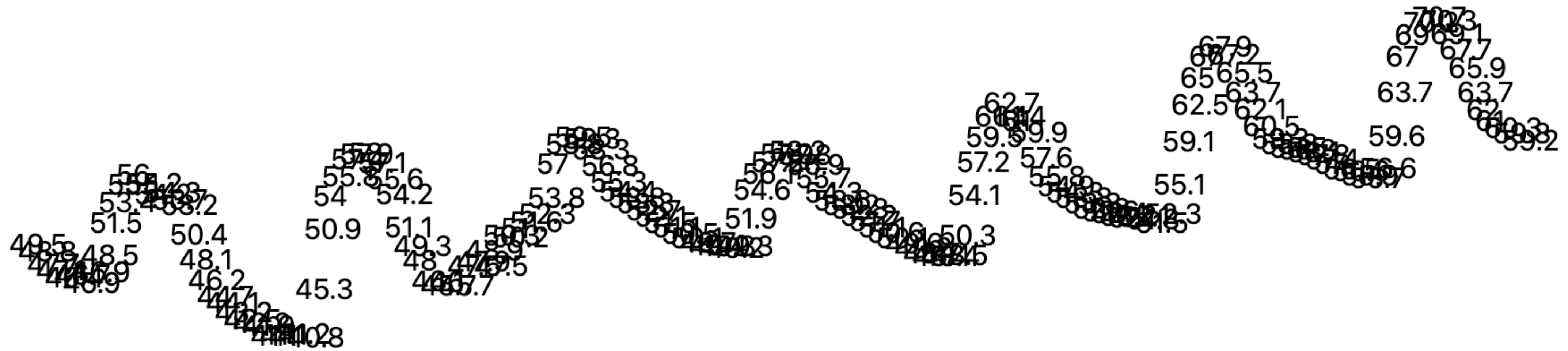


[js-lecture/weather02/](#)

(demo)

[js-lecture/weather03/](#)

(demo)



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

tryclassbuzz.com

Code: **axes**