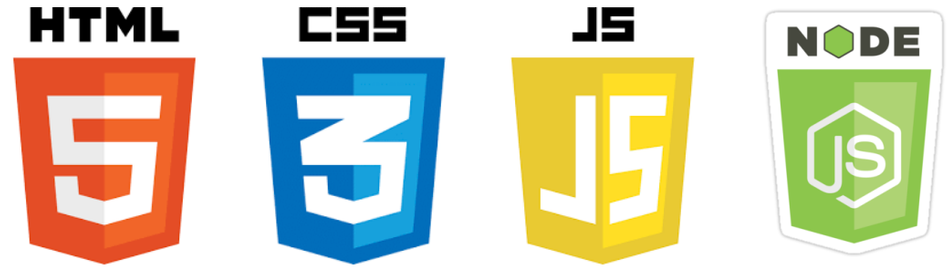
App Dev

Stefano Balietti

Center for European Social Science Research at Mannheim University (MZES)
Alfred-Weber Institute of Economics at Heidelberg University

@balietti | stefanobalietti.com | @nodegameorg | nodegame.org

Building Digital Skills: 5-14 May 2021, University of Luzern



What Is This Course About?

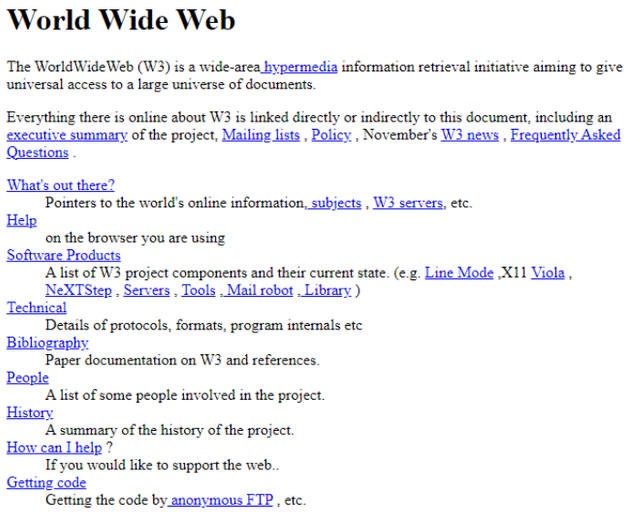
- Leverage knowledge acquired in course **"Intro to Programming for App Development"**
- The course expects some **knowledge of computer programming** and a basic understanding of **front-end web development** …
- *but we will recap the core concepts.*
- The goal is to gain a fine understanding of the **full web stack** to create **web-based apps** that can run on **multiple devices**
- The **learning speed and materials will be adapted** to the level of the participants

What is Web Development?

1. At the beginning, it was almost synonymous with "creating web sites"

2. However, it's meaning has evolved and it includes a broader definition of creating "web services" or "web apps"

From the first web site ever to…



World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

http://info.cern.ch/hypertext/WWW/TheProject.html

From the first web site ever to…

World Wide Web

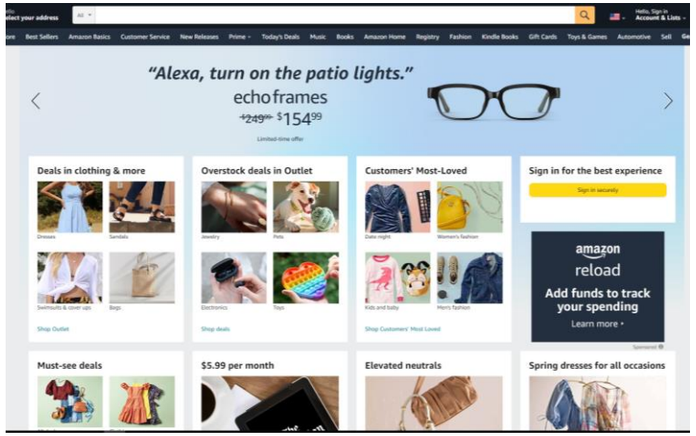
The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

http://info.cern.ch/hypertext/WWW/TheProject.html

Complex E-commerce



https://www.amazon.com/

From the first web site ever to…

World Wide Web

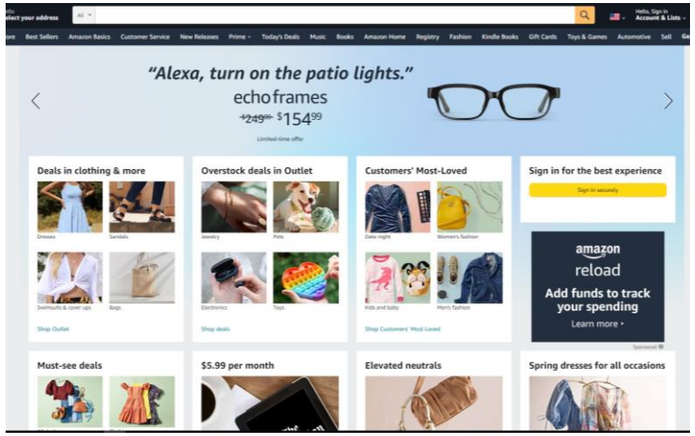
The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

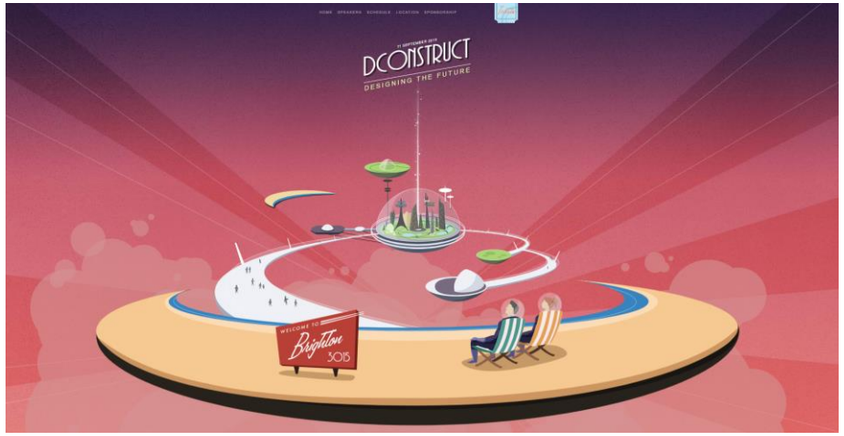
http://info.cern.ch/hypertext/WWW/TheProject.html

Complex E-commerce



https://www.amazon.com/

Beautiful Interfaces



http://2015.dconstruct.org/

From the first web site ever to…

World Wide Web

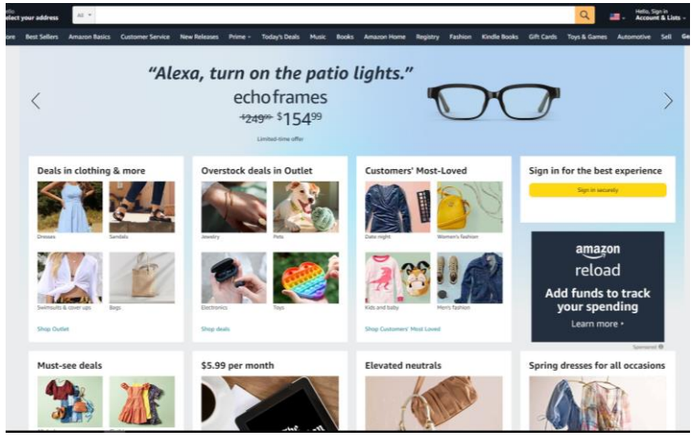
The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

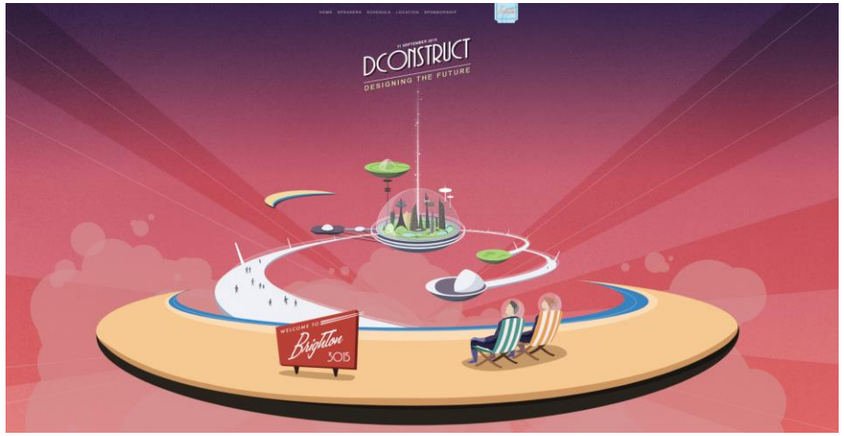
http://info.cern.ch/hypertext/WWW/TheProject.html

Complex E-commerce



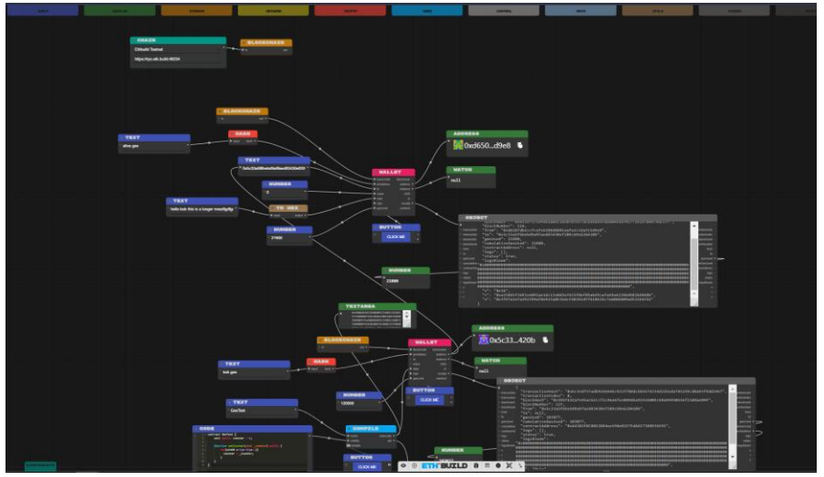
https://www.amazon.com/

Beautiful Interfaces



http://2015.dconstruct.org/

Complex interactions



https://eth.build/

From the first web site ever to…

World Wide Web

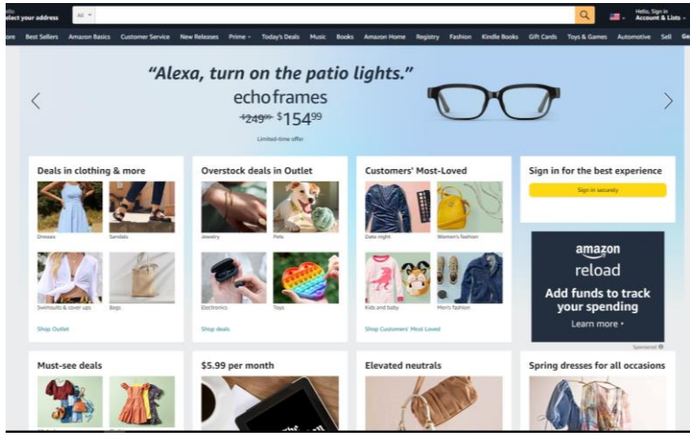
The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

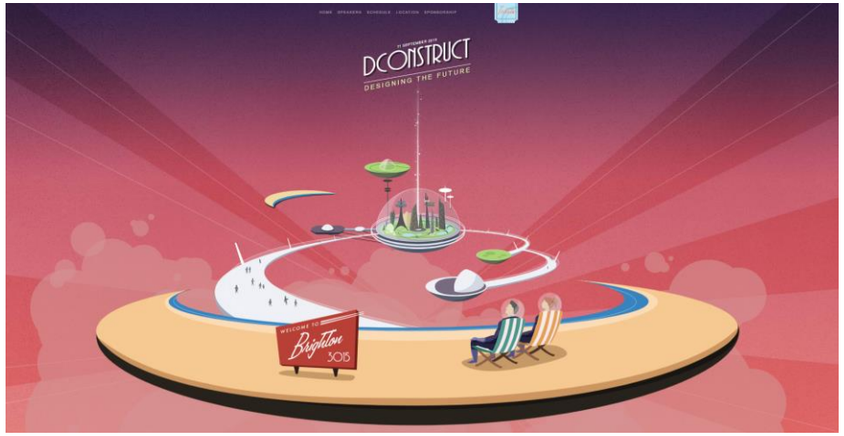
http://info.cern.ch/hypertext/WWW/TheProject.html

Complex E-commerce



https://www.amazon.com/

Beautiful Interfaces



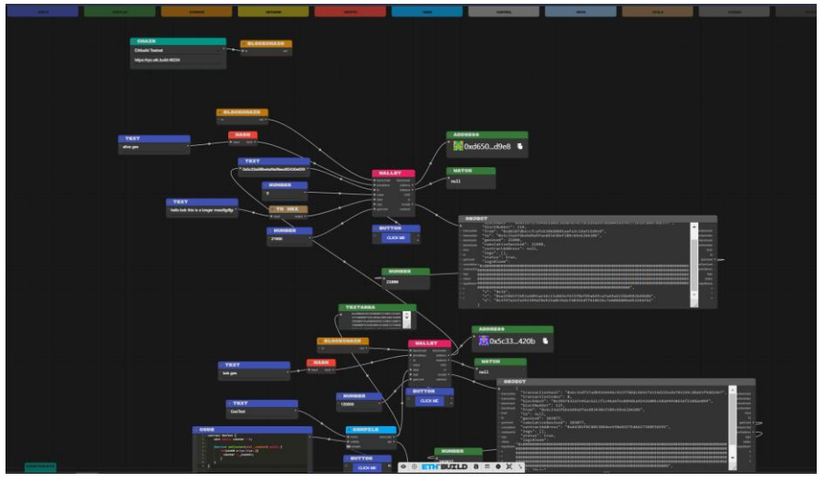
http://2015.dconstruct.org/

Plain nonsense

http://eelslap.com/

https://makeawebsitehub.com/weird-websites/
https://ecommercebooth.com/weird-websites/

Complex interactions



https://eth.build/

Live Internet Stats

5,298,421,482

Internet Users in the world

1,950,093,319

Total number of Websites

116,521,843,528

Emails sent today

3,637,874,313

Google searches today

3,537,008

Blog posts written today

356,297,550

Tweets sent today

3,403,084,520

Videos viewed today
on YouTube

41,283,157

Photos uploaded today
on Instagram

74,505,324

Tumblr posts today

3,149,548,777

Facebook active users

1,117,232,655

Google+ active users

387,786,107

Twitter active users

https://www.internetlivestats.com/

576,000 New Websites Per Day

4,800 zettabytes (ZB) predicted in year 2022

zettabyte as “10^{21} or 1,000,000,000,000,000,000,000 bytes.”

576,000 New Websites Per Day

4,800 zettabytes (ZB) predicted in year 2022

zettabyte as "10^{21} or 1,000,000,000,000,000,000,000 bytes."

From Shruti Jain: If each Terabyte in a Zettabyte were a kilometer, it would be equivalent to 1,300 round trips to the moon and back (768,800 kilometers).

From Arielle Sumits: If each Petabyte in a Zettabyte were a centimeter, then we could reach a height 12 times higher than the Burj Khalifa (the world's tallest building at 828 meters high).

From Usha Andra: If every Gigabyte in a Zettabyte were a meter, it could span the distance of the Amazon River (the world's longest river at 6,992 kilometers) more than150,000 times.

From Taru Khurana: If each Gigabyte in a Zettabyte were a brick, 258 Great Walls of China (made of 3,873,000,000 bricks) could be built.

https://websitesetup.org/news/how-many-websites-are-there/

https://blogs.cisco.com/sp/the-zettabyte-era-officially-begins-how-much-is-that

What is a Web Developer?

1. **Front-end**: focus on presentation, usually have good design skills, interact with web services as black boxes

What is a Web Developer?

1. **Front-end**: focus on presentation, usually have good design skills, interact with web services as black boxes
2. **Back-end:** application logic, database, admin, suck at designing user interfaces

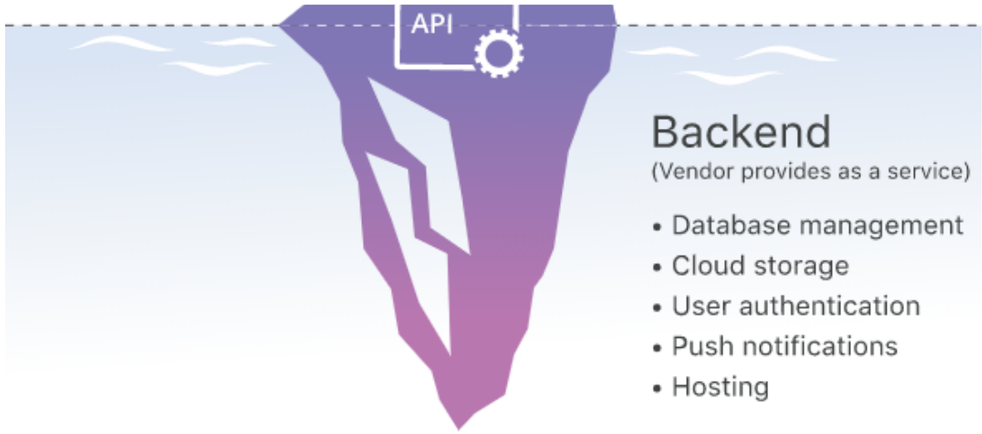


Image source

What is a Web Developer?

1. **Front-end**: focus on presentation, usually have good design skills, interact with web services as black boxes
2. **Back-end:** application logic, database, admin, suck at designing user interfaces
3. **Full-stack**: usually suck at both ☺, but they have a solid understanding of how things glue together

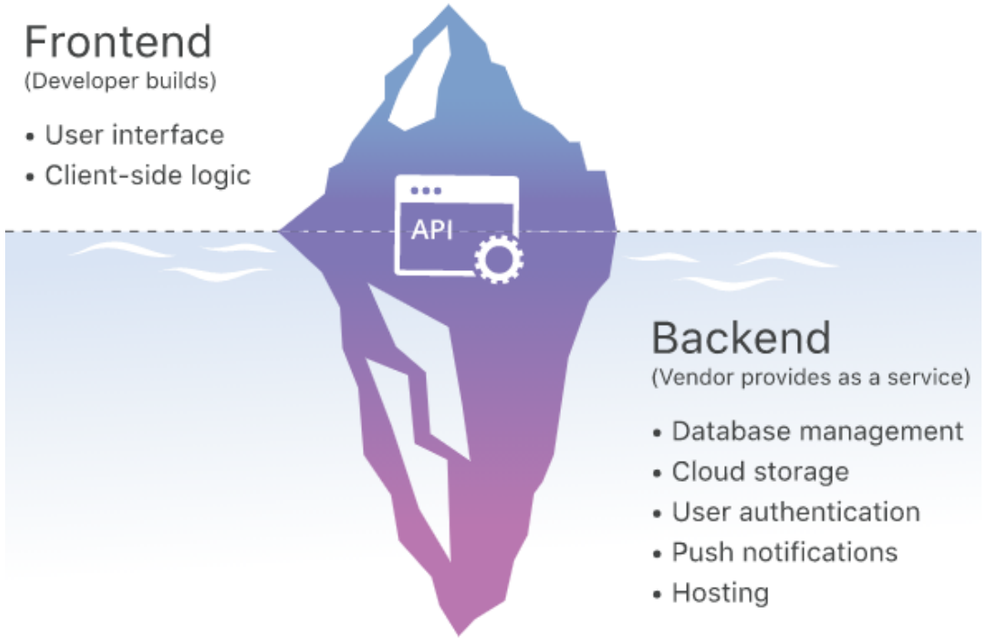


Image source

What is a Web Developer?

1. **Front-end**: focus on presentation, usually have good design skills, interact with web services as black boxes
2. **Back-end:** application logic, database, admin, suck at designing user interfaces
3. **Full-stack**: usually suck at both ☺, but they have a solid understanding of how things glue together

4. Face an **increasing number of frameworks and tools**
5. Pressured between the need to **deliver current projects** and **learning new technology**

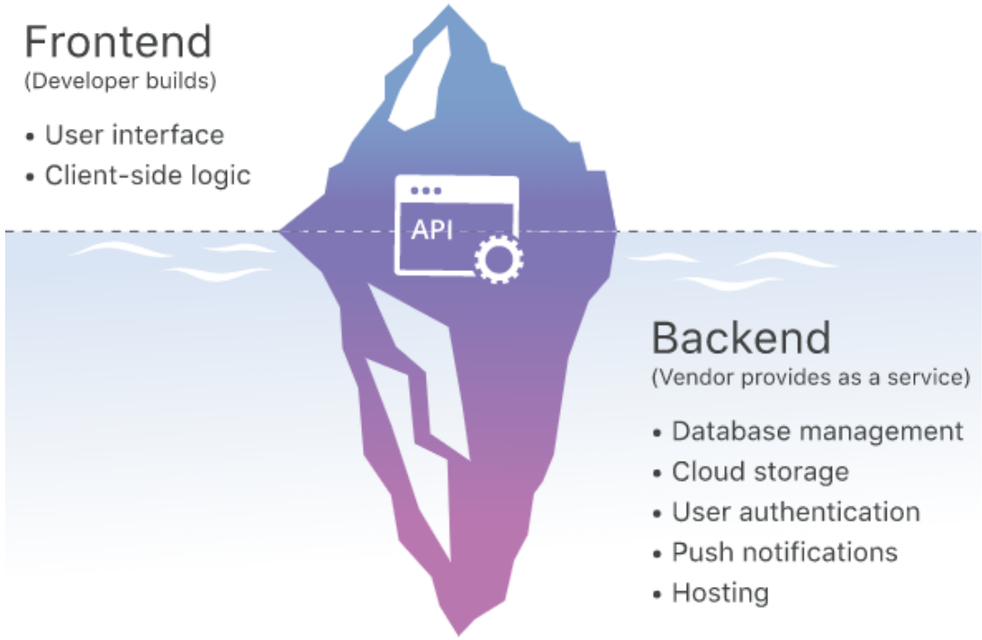


Image source

The Web Development Jungle (2015…)

https://pressupinc.com/blog/**2015**/11/web-development-industry-jungle/

The Web Development Jungle (2015…)

https://pressupinc.com/blog/**2015**/11/web-development-industry-jungle/

- Understand the **technology stack**
- **Review** main frameworks (not exhaustive)
- Hands-on **exercises**
- **Choosing the right approach for your project**

Full-Stack Technologies

https://www.masterborn.com/blog/Frontend_vs_backend_guide

https://www.cybercoders.com/insights/what-hiring-managers-look-for-in-a-full-stack-developer/

Full-Stack Technologies

Covered in intro course (we will have a recap)

Will get a general overview (no demos)
We could have a demo for nodeGame

We will cover Bootstrap and jQuery

https://www.masterborn.com/blog/Frontend_vs_backend_guide

https://www.cybercoders.com/insights/what-hiring-managers-look-for-in-a-full-stack-developer/

Full-Stack Technologies

We will work with Node.JS

https://www.masterborn.com/blog/Frontend_vs_backend_guide

https://www.cybercoders.com/insights/what-hiring-managers-look-for-in-a-full-stack-developer/

Full-Stack Technologies

We will have a quick intro to MongoDB

https://www.masterborn.com/blog/Frontend_vs_backend_guide

https://www.cybercoders.com/insights/what-hiring-managers-look-for-in-a-full-stack-developer/

Full-Stack Technologies

What is **DevOps**?

It's a term indicating the work of someone at the interface of developing and business operations

We will review a few hosting solutions, e.g. Heroku and Digital Ocean

https://www.masterborn.com/blog/Frontend_vs_backend_guide

https://www.cybercoders.com/insights/what-hiring-managers-look-for-in-a-full-stack-developer/

Full-Stack Technologies

We will review Ionic and PWA plus Chrome extensions

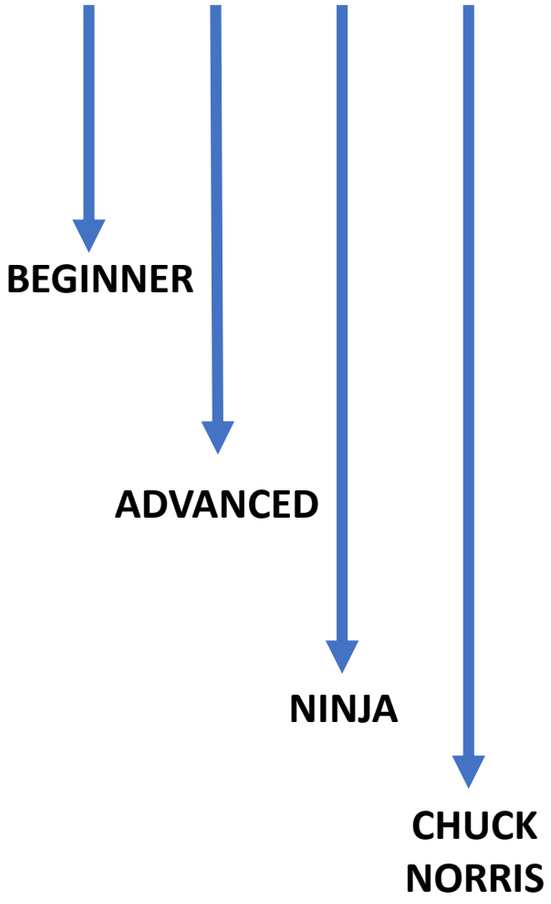
https://www.masterborn.com/blog/Frontend_vs_backend_guide

https://www.cybercoders.com/insights/what-hiring-managers-look-for-in-a-full-stack-developer/

Tentative Schedule

Date	Module	Goals
Nov 11	**Recap, Async code, Frontend Frameworks**	Environment check, recap of intro course (JS, Browser, CSS), NPM and Node, Asynchronous Code (callbacks, promises, async/await, listeners, fetch, axios, REST APIs), jQuery
Nov 12	**More Frontend frameworks, Intro to Express, and Nginx**	Bootstrap v5, Flex, and Grid, Express, Nginx Running a local HTTP server
Nov 18	**Securing Express, Taming Bots, Hosting, Let's Encrypt, MongoDB, nodeGame**	Debugging Backend code, Running MongoDB in the cloud and locally, Choosing an host, Encryption, nodeGame, Honeypots, Captchas
Nov 19	**Chrome Extensions, PWA, Mobile Development**	Mobile First, Overview of Single-page frameworks, Chrome Extensions, PWA, Ionic, Debugging Mobile

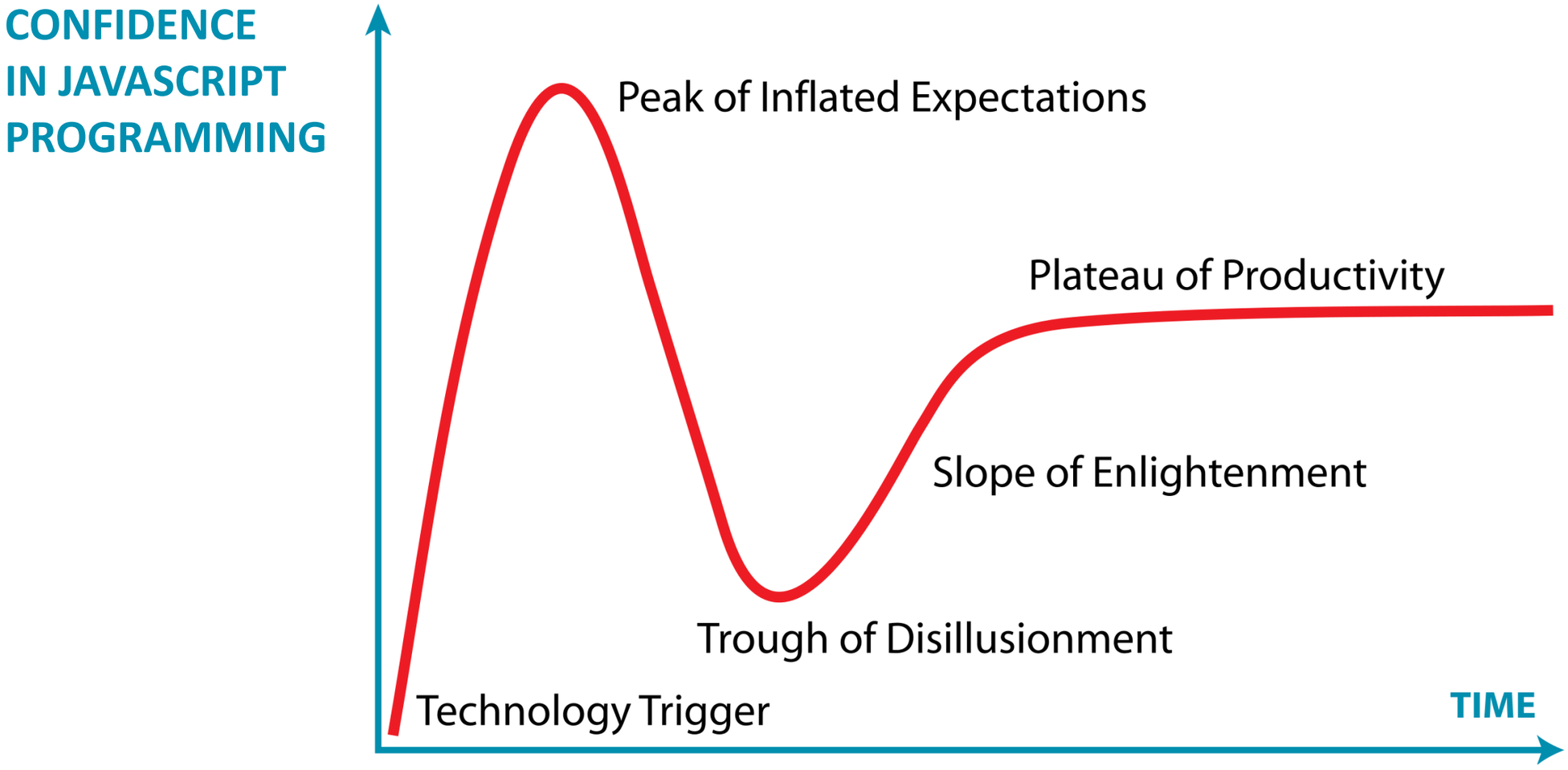
What is a realistic learning goal?



How to get the most out of this course?

- This is a **workshop** rather than a course
- Frontal teaching is the smaller part of learning, you will get the most by doing the **exercises**
- You should do the exercises at **your own pace**. Don't feel compelled to them all
- Set a **plausible learning target** for yourself and *try hard* to achieve it!

Learning Curve

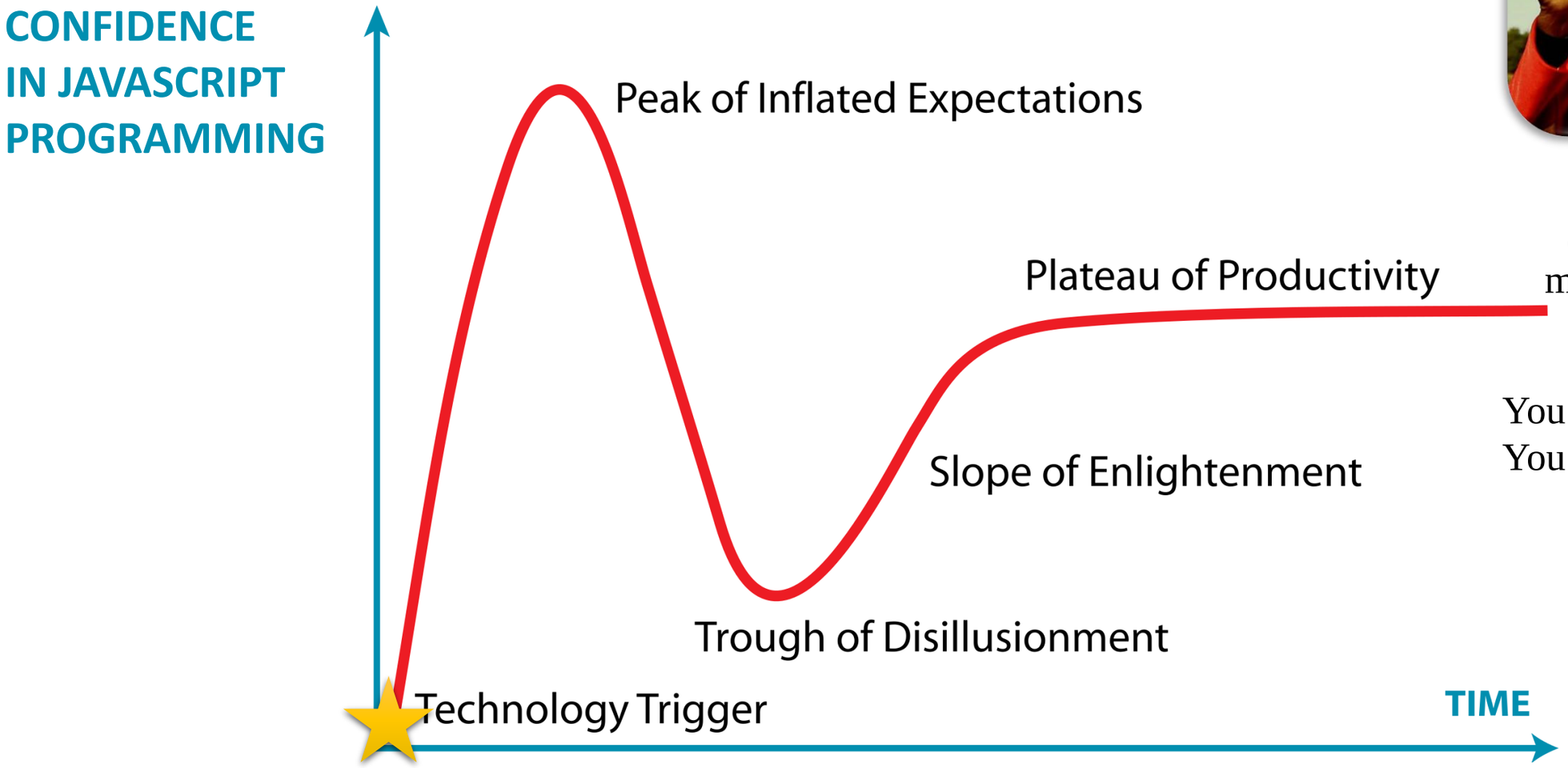


Adapted from: https://en.wikipedia.org/wiki/Hype_cycle

Learning Curve

Dr. Jimmy Cliff is a Jamaican ska & reggae musician, singer and actor.

You can get it if you really want
You can get it if you really want
But you must try,
try and try, try and try
You'll succeed at last

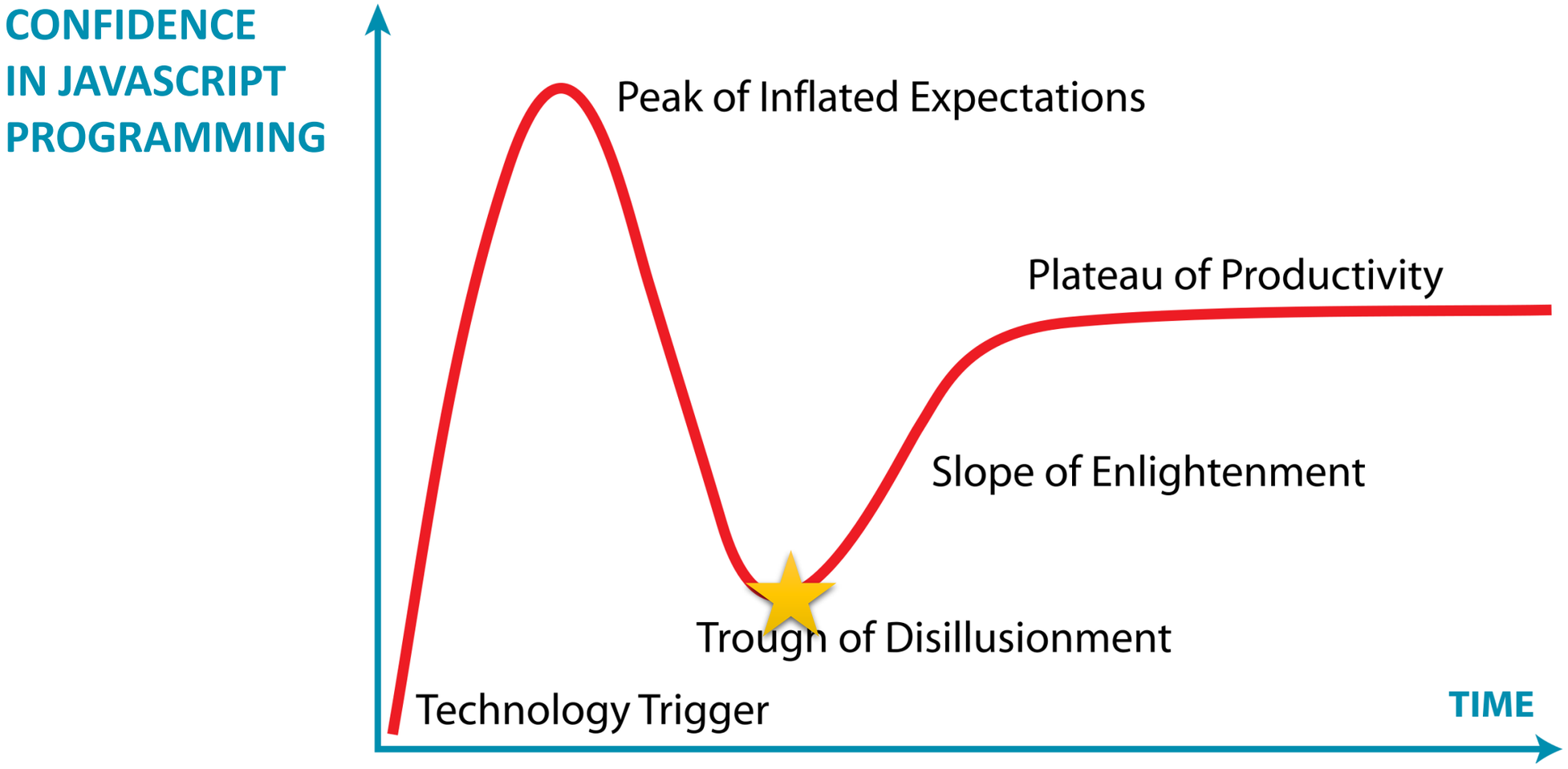


Adapted from: https://en.wikipedia.org/wiki/Hype_cycle

If You Get Stuck in The Exercises

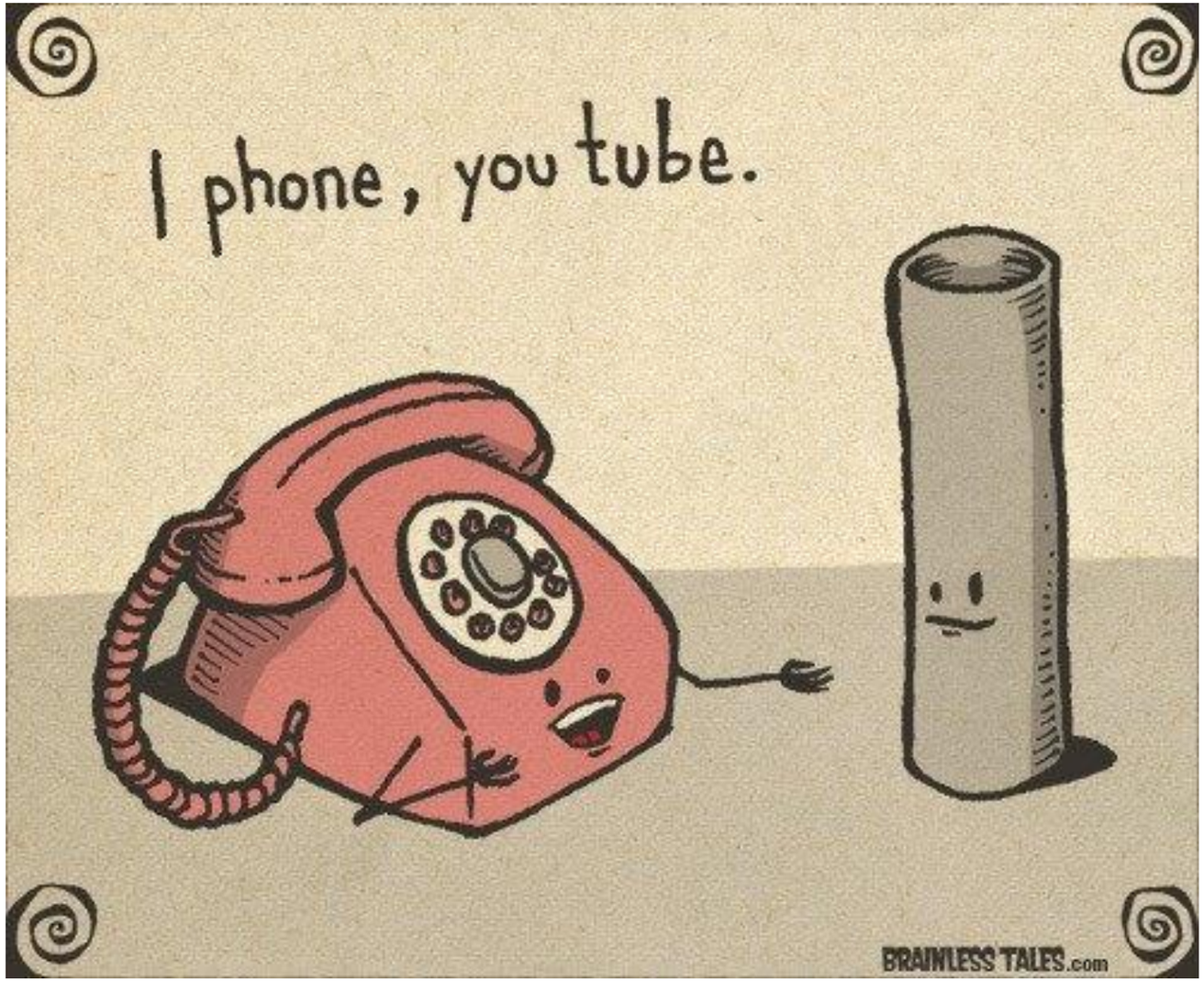
- Within each sheet, exercises are generally in *order of complexity*; some exercises are marked as "Bonus" or "Advanced." Feel free to skip all exercises that don't fit your *learning goal*.
- Ask for **help** in the Discord channel (during or after the lecture time)
- **Anybody** can answer (or attempt to answer) questions
- Share **screen** if necessary (Discord allows to share apps and screens)
- Try to keep your **GitHub** repository up to date when asking for more specific help

Learning Curve



Adapted from: https://en.wikipedia.org/wiki/Hype_cycle

Now Introductions



Your Instructor: Stefano Balietti

http://stefanobalietti.com

Currently

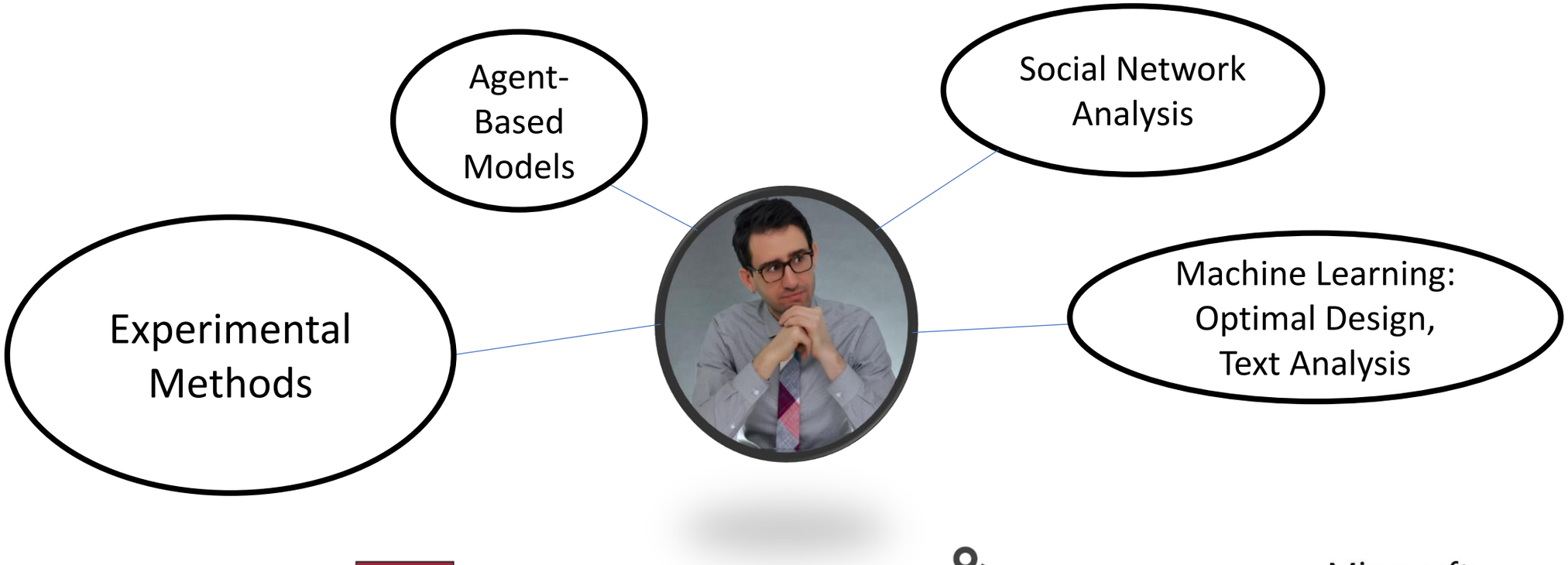
- Fellow in Sociology Mannheim Center for European Social Research (MZES)
- Postdoc at the Alfred Weber Institute of Economics at Heidelberg University

Previously

- Microsoft Research - Computational Social Science New York City
- Postdoc Network Science Institute, Northeastern University
- Fellow IQSS, Harvard University
- PhD, Postdoc, Computational Social Science, ETH Zurich

My Methodology

Interface of computer science, sociology, and economics



ETH zürich

HARVARD UNIVERSITY

network science institute

Microsoft Research

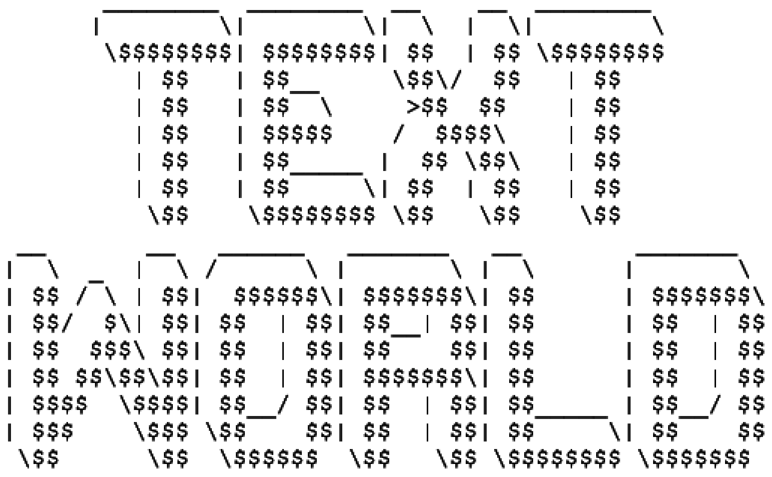
Building Platforms

Garch-in-Gretl (GiG) for econometrics Gretl software

~**5000** weekly downloads

Patterns Configuration Module for Drupal Web Content Management System

2,622 active users, **30,448** downloads



Microsoft

Fast, scalable JavaScript for large-scale real-time online experiments

nodeGAME

v7

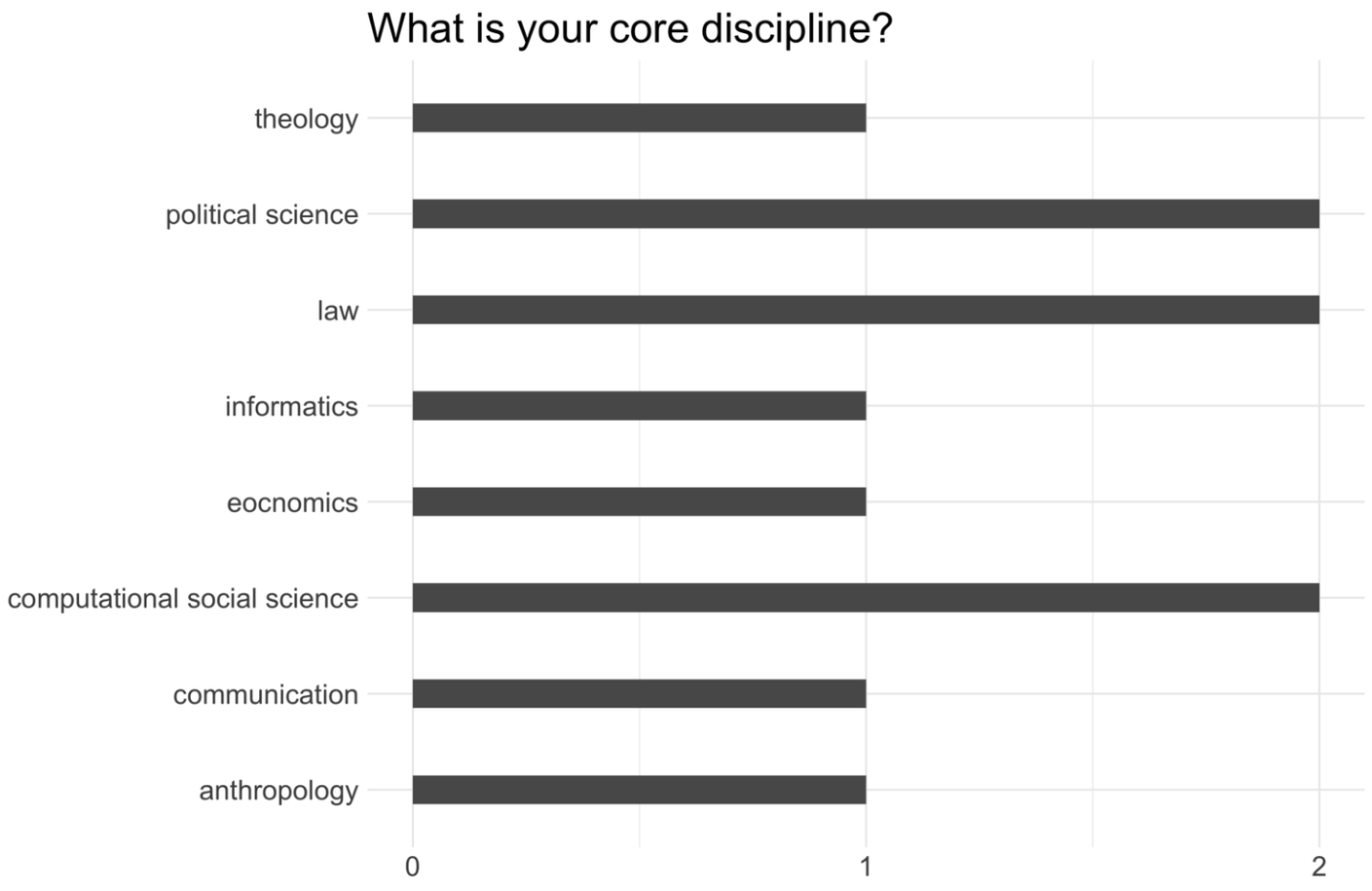
www.nodegame.org

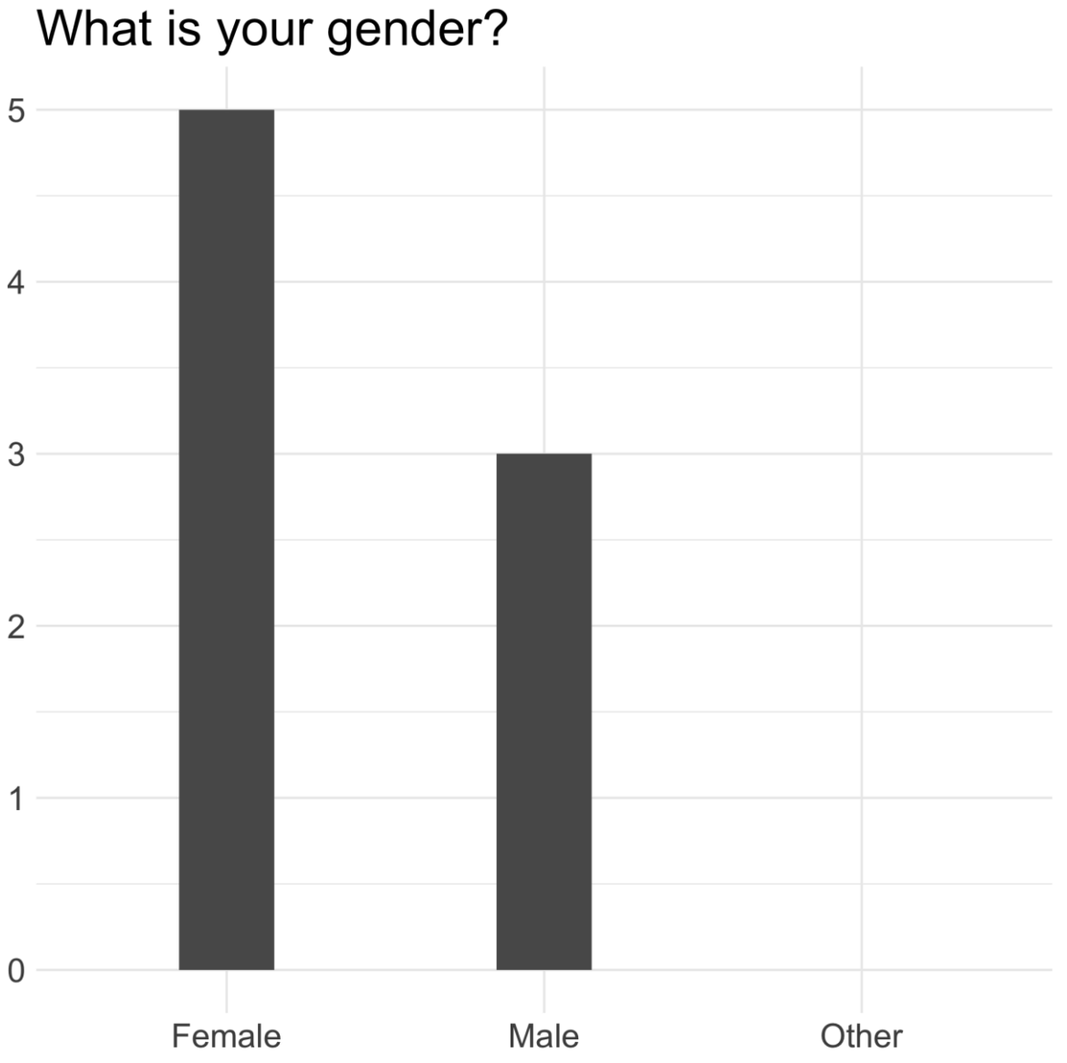
My Vision

Simulating Societal Processes in Virtual Labs

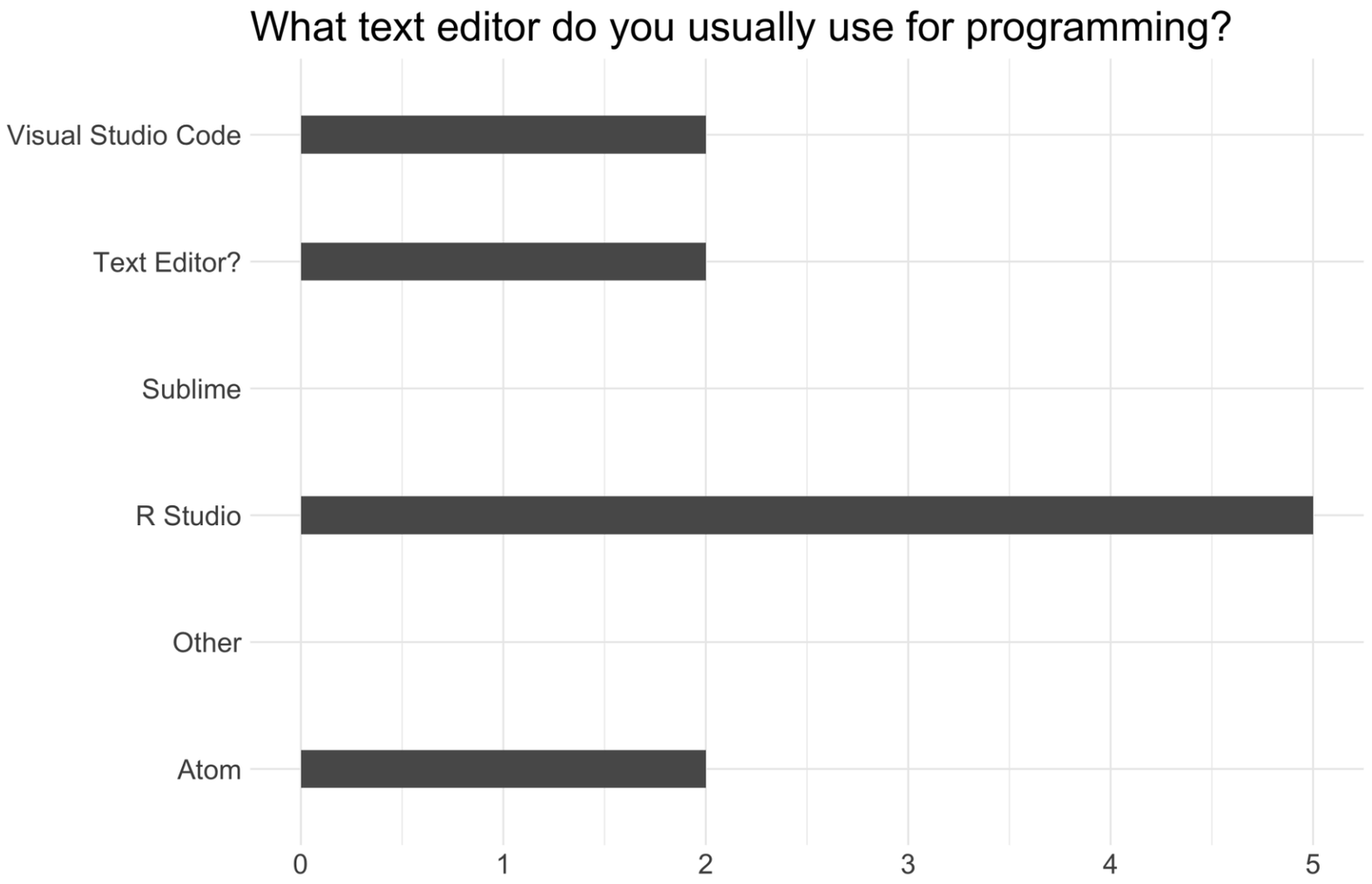
- Consensus, social influence, and polarization
- Group fairness, inequality, redistribution
- Incentives schemes for collective intelligence
- Optimal experimental design

About You (from web survey)

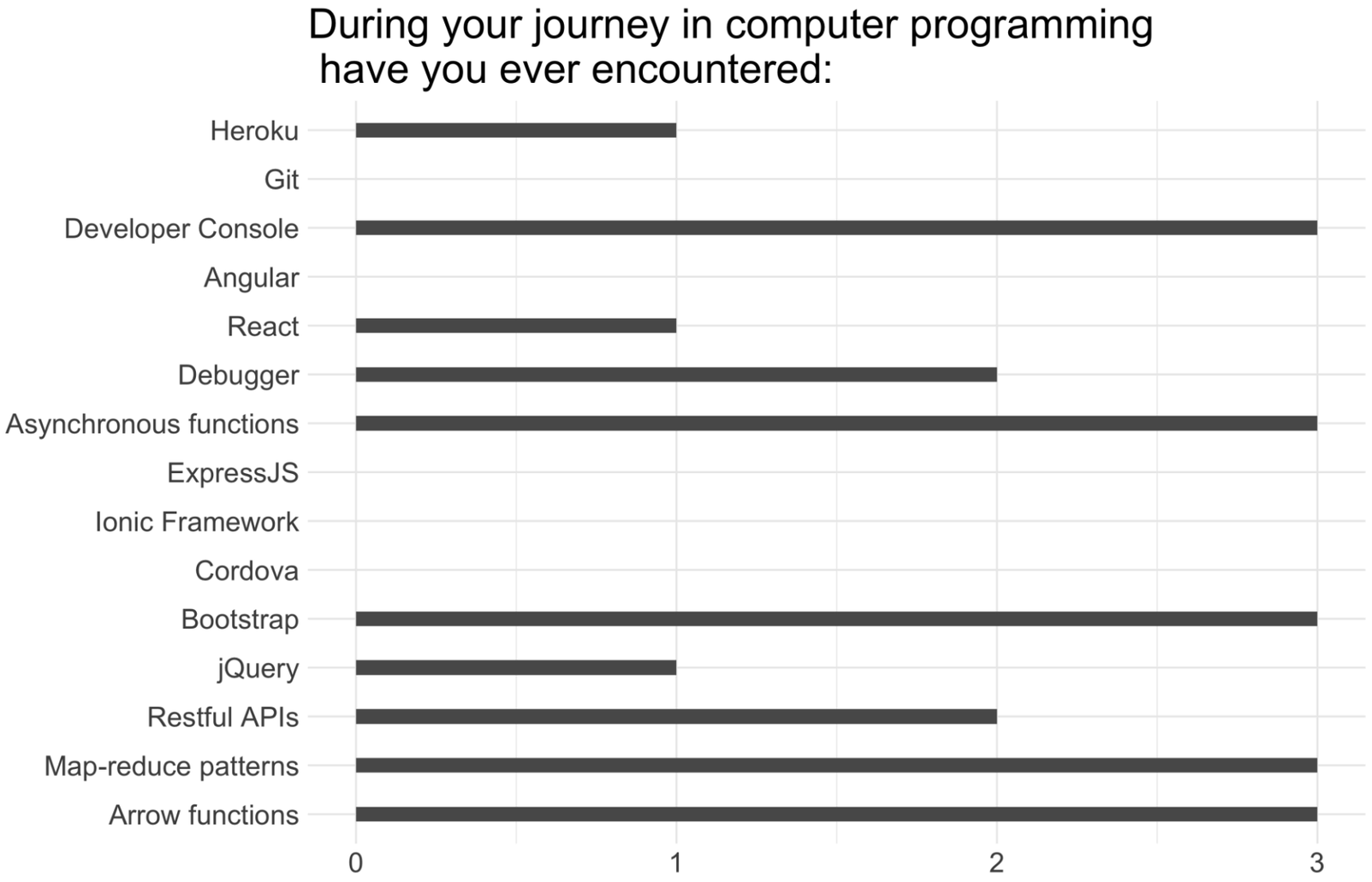




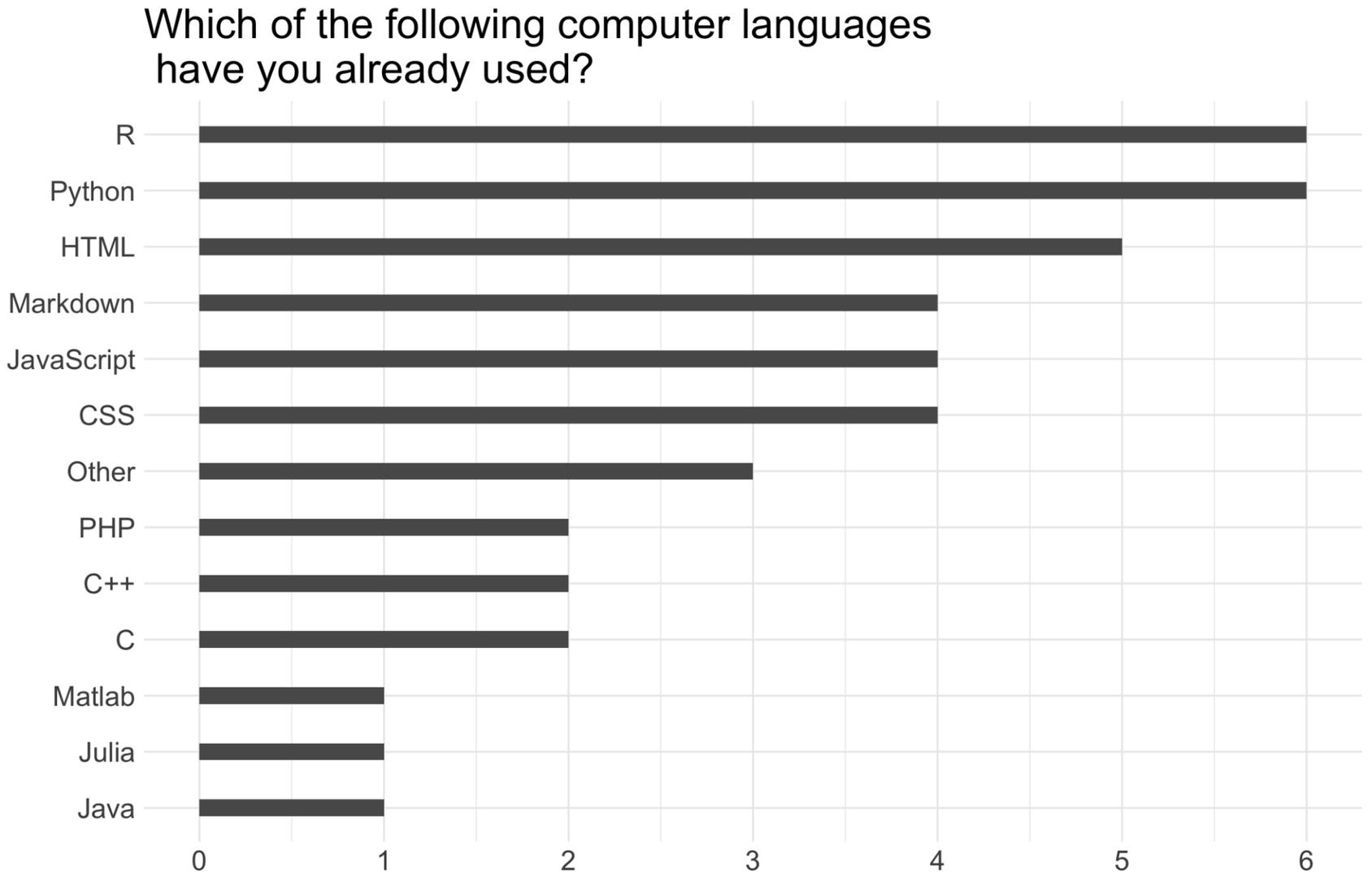
About You (from web survey)



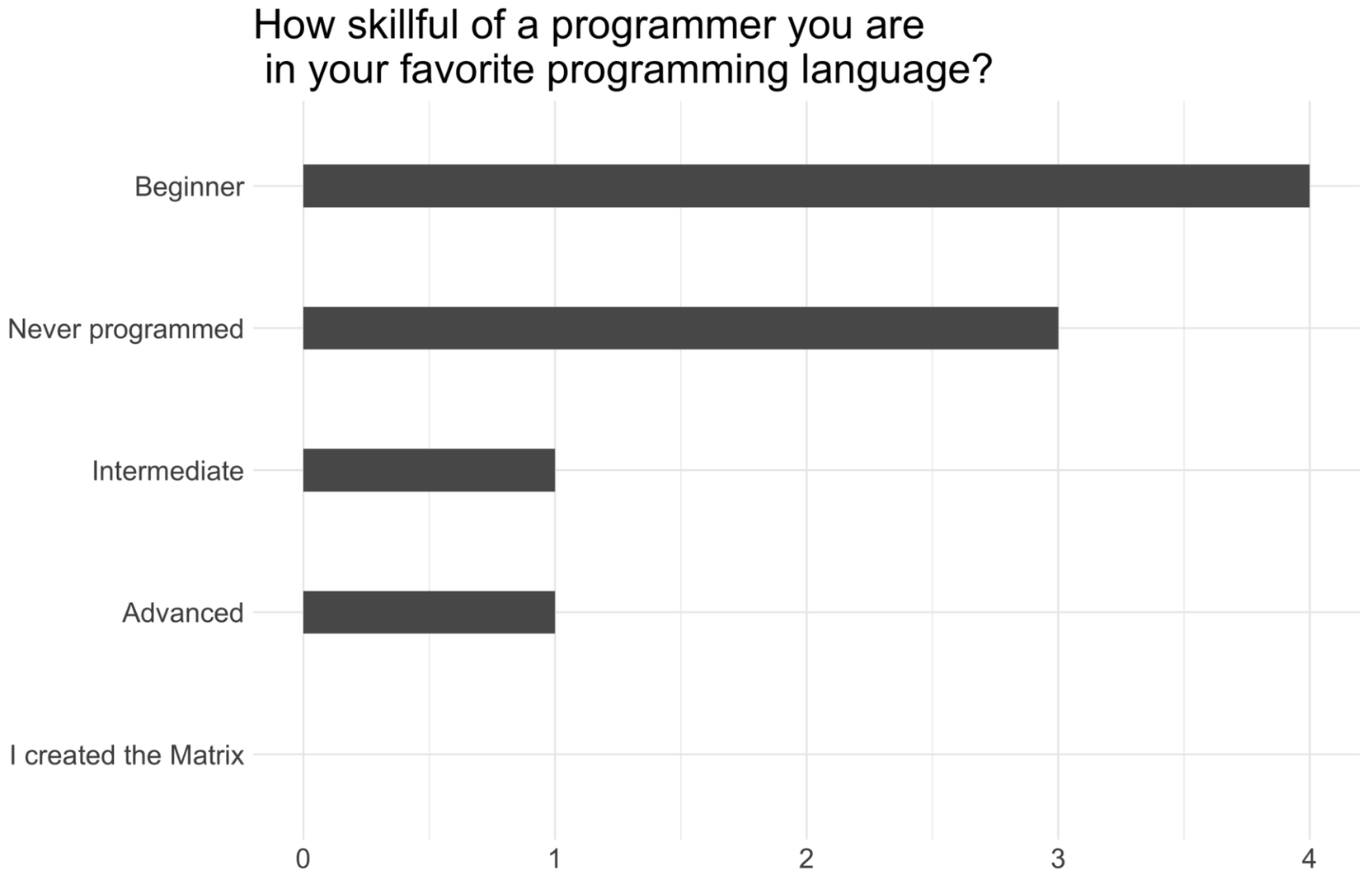
About You (from web survey)



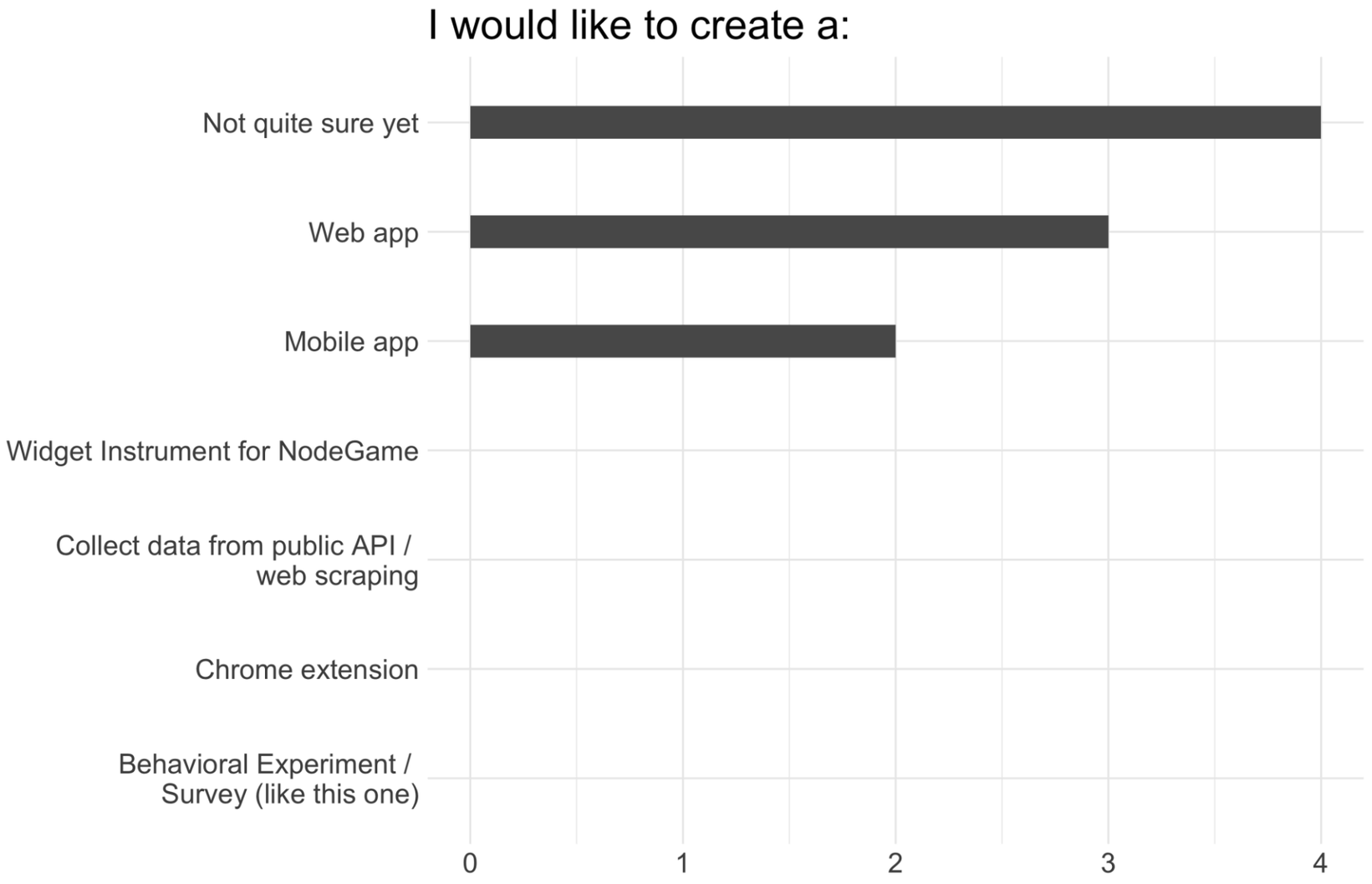
About You (from web survey)



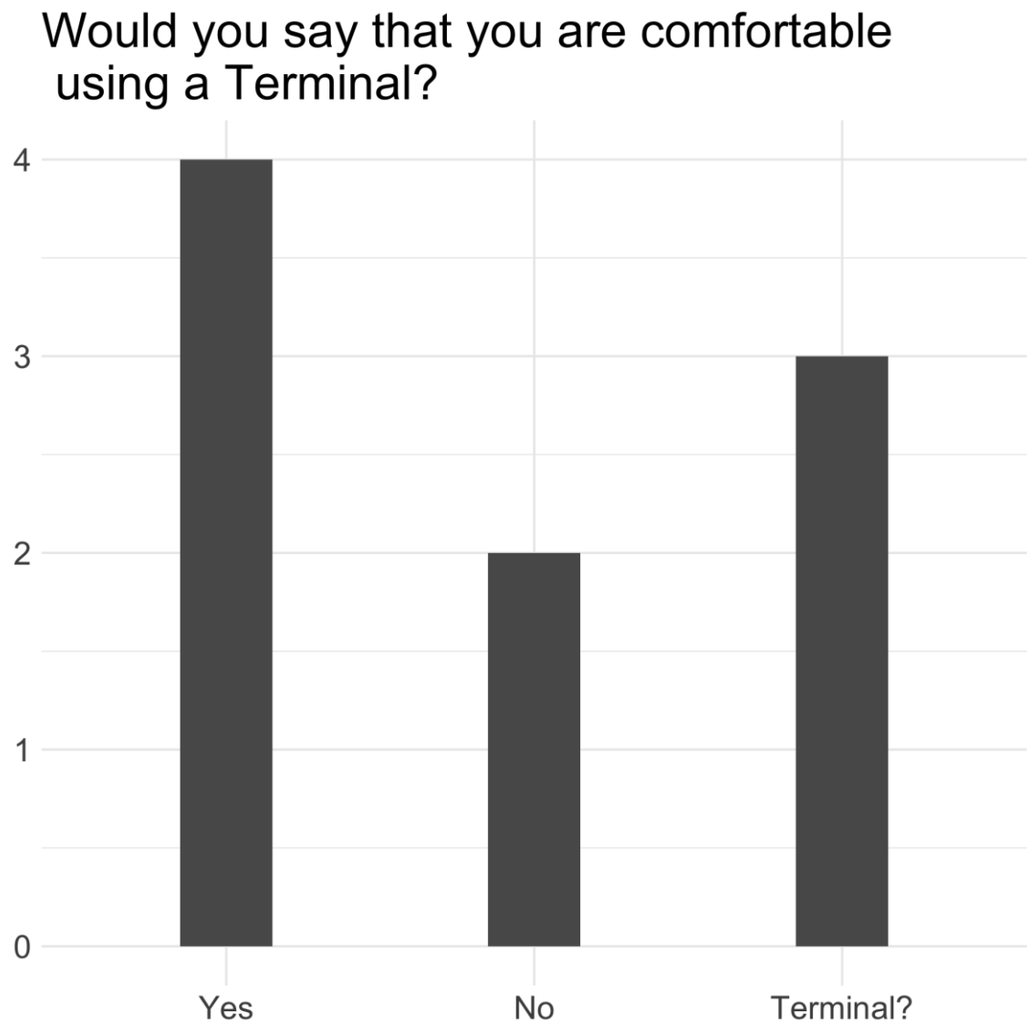
About You (from web survey)

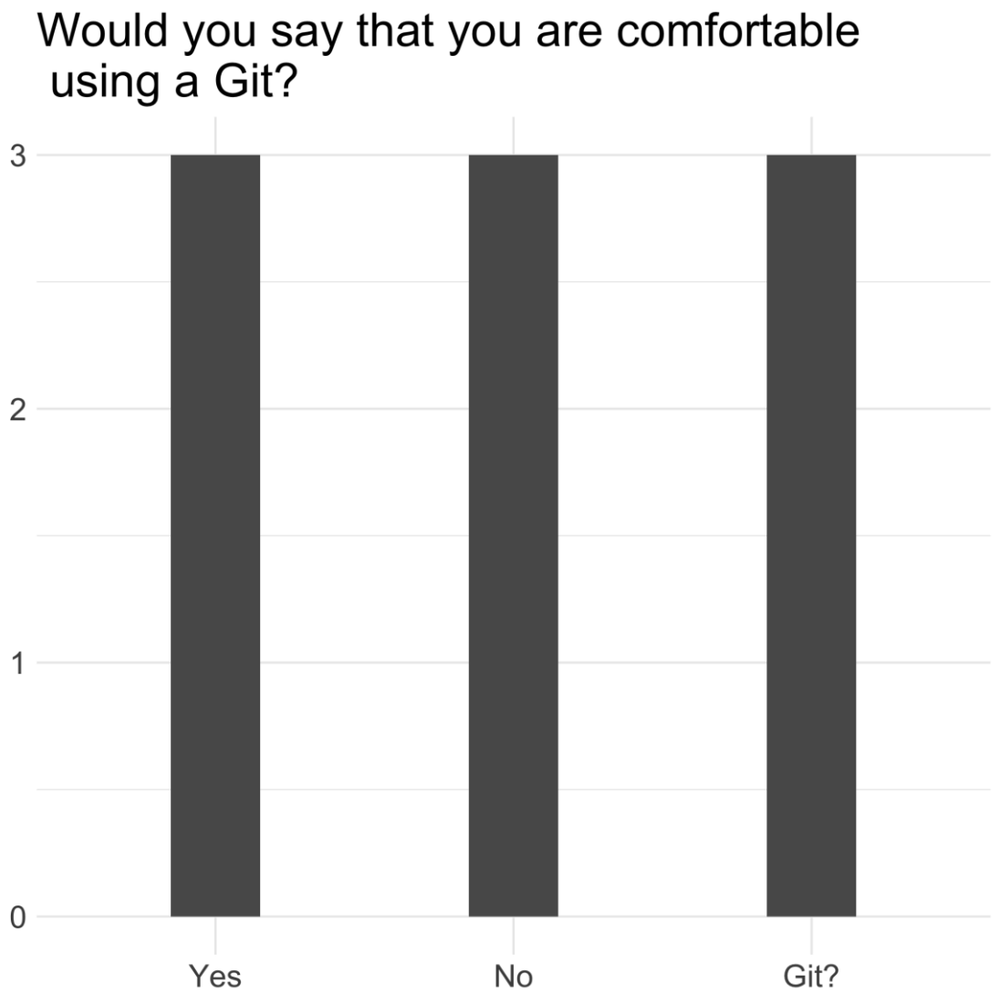


About You (from web survey)



About You (from web survey)





You Get the Certificate If

Attend **all days**.

No problems if you *miss a few hours*.

Recap!

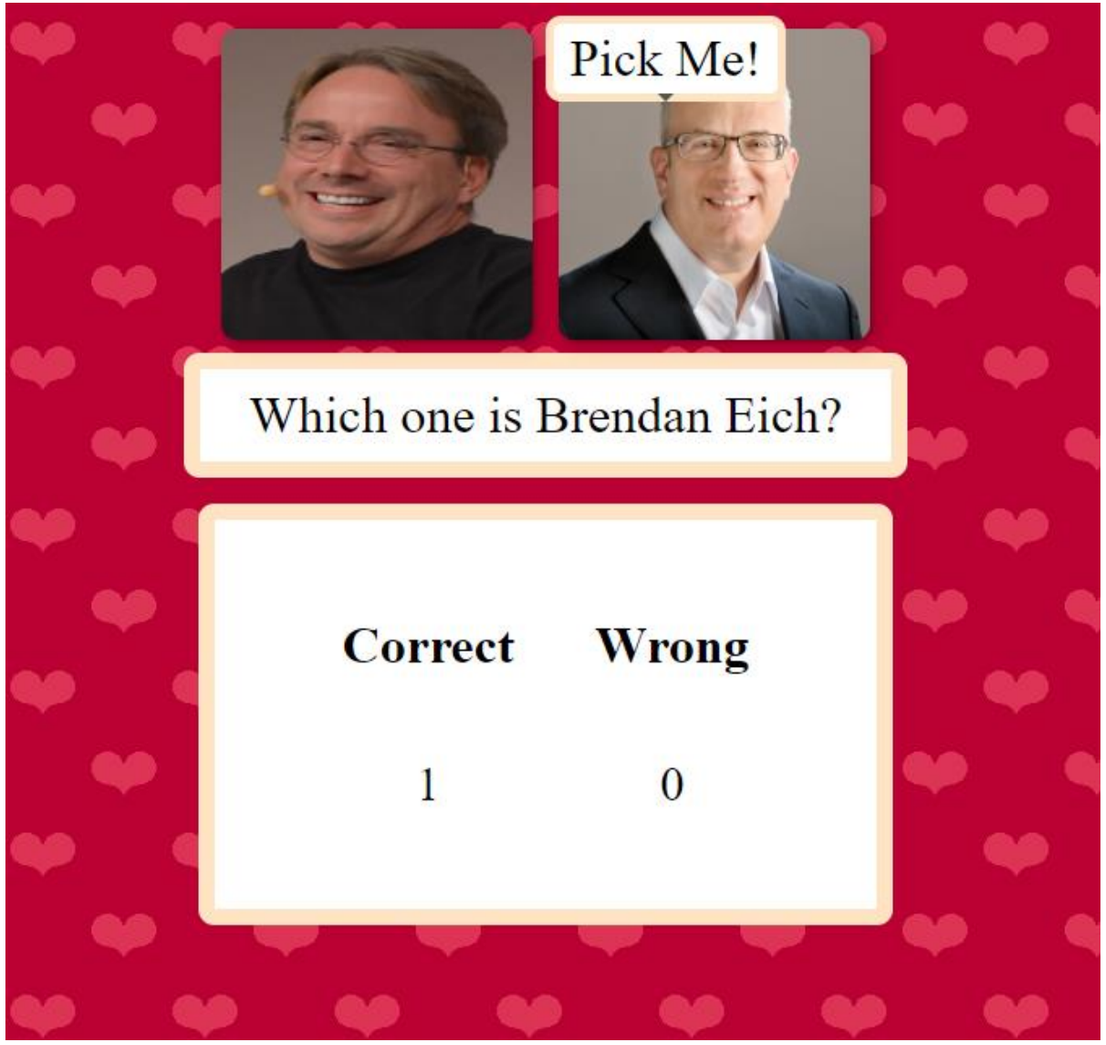
Prerequisites:

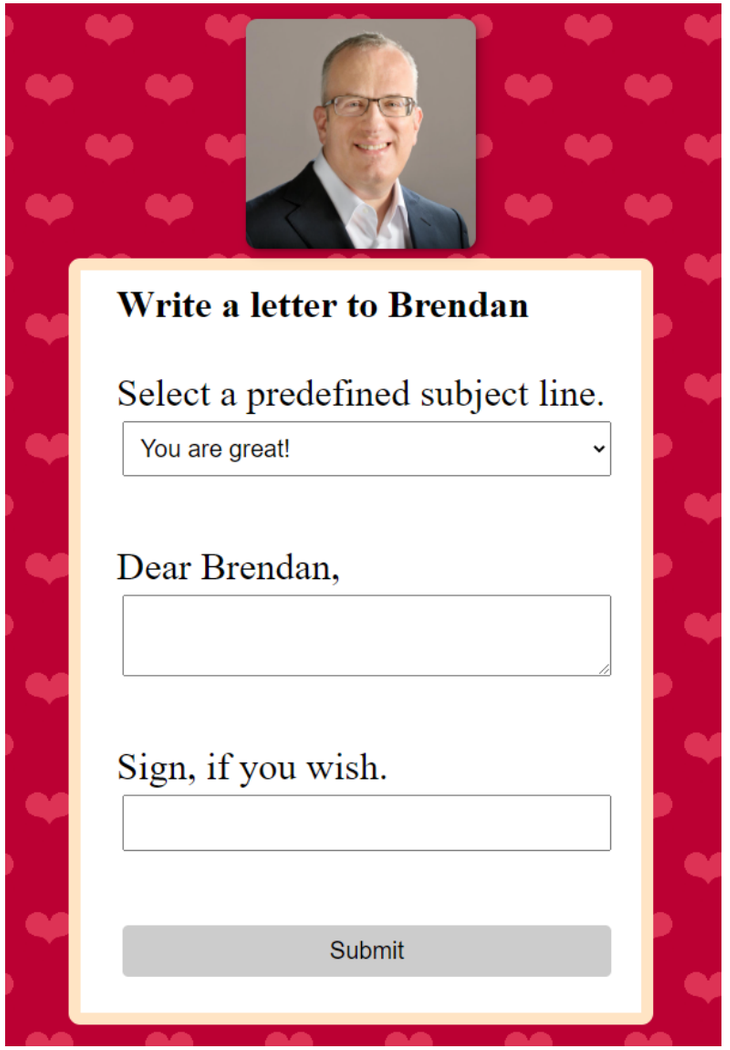
1. **Basic JavaScript / Node.JS programming**

2. **Basic front-end development:** HTML, JavaScript, CSS, debugging front-end code.

3. **Basic understanding of Git/GitHub**

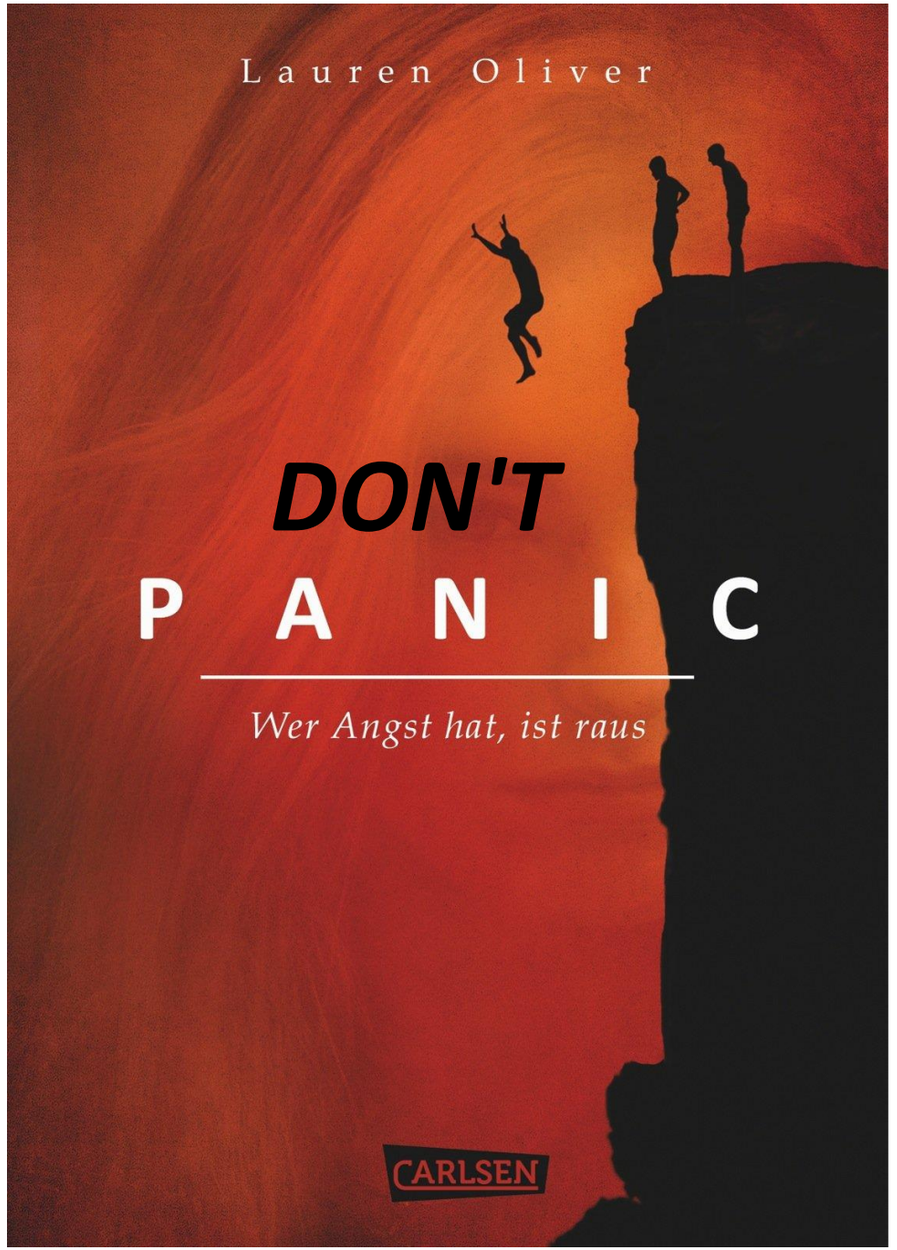
Where we left





OMG I missed the first part or I don't remember a thing…How Do I catch up?

- Intro course is still available on Discord
- https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/
- https://javascript.info/
- https://www.w3schools.com/



Recap: What Is JavaScript?

JavaScript is NOT Java

"Java is to JavaScript as ham is to hamster." (Jeremy Keith)

Image source

JavaScript is NOT Java

"Java is to JavaScript as car is to carpet." (Chris Heilmann)

53.12 incl. VAT

Image source

€61.39 incl. VAT

Image source

JavaScript

- JavaScript was developed in May 1995 by *Brendan Eich* for Netscape Communications Corp
- Was created in **10 days** in order to accommodate the Navigator 2.0 Beta release
- Initially called **Mocha**, later renamed **LiveScript** in September, and later **JavaScript** in the same month

JavaScript

- Microsoft introduced **JScript** as reverse-engineered implementation of Netscape's JavaScript in 1996 in Internet Explorer 3

- In 1996 Netscape submitted JavaScript to European Computer Manufacturers Association (ECMA) to create and industry standard

- In 1997 **ECMAScript** was released

- Between 1997 and 2009 5 standard have been released.

- *July 2015 ECMASCRIPT V6 released.*

JavaScript Releases

- ES2016 a.k.a. ES7
- ES2017 a.k.a. ES8
- ES2018 a.k.a. ES9
- ES2019 a.k.a. ES10
- ES2020 a.k.a. ES11
- ES2021 a.k.a. ES12
- ES2022 a.k.a. ES13

Language improvement proposals discussed here:

https://github.com/tc39/proposals

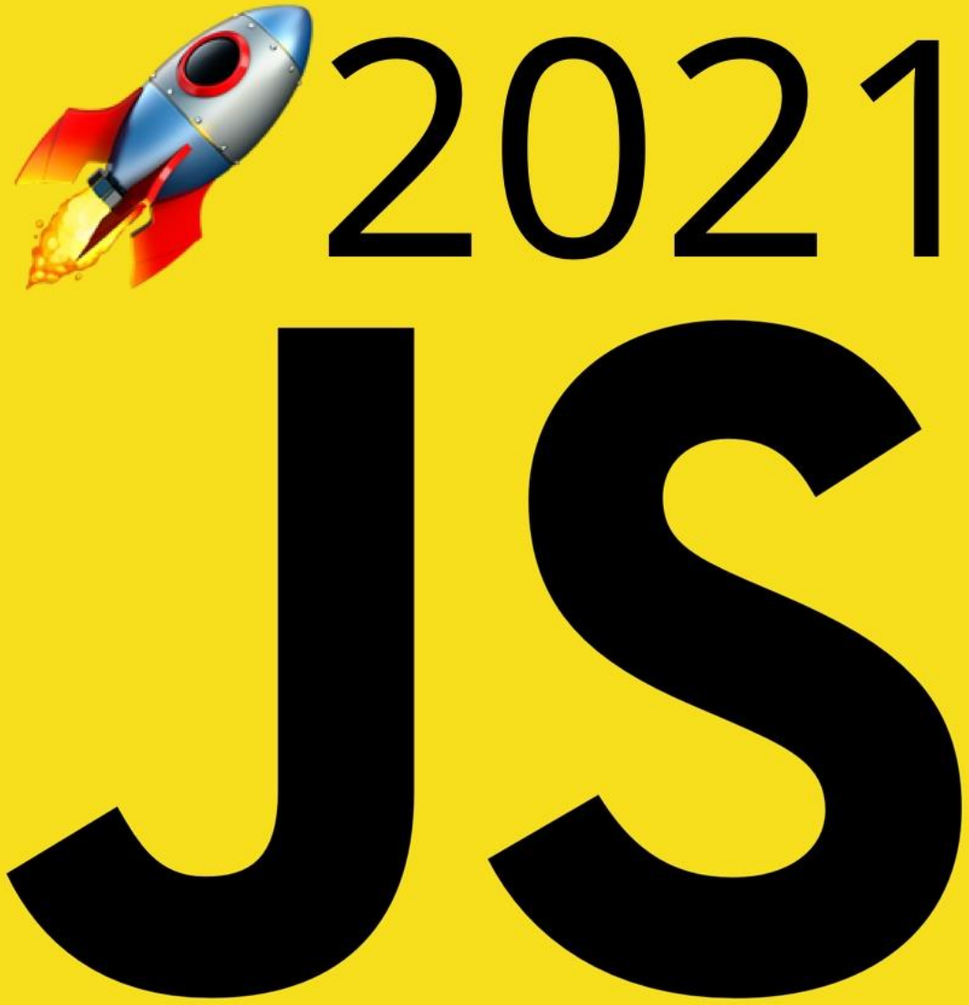
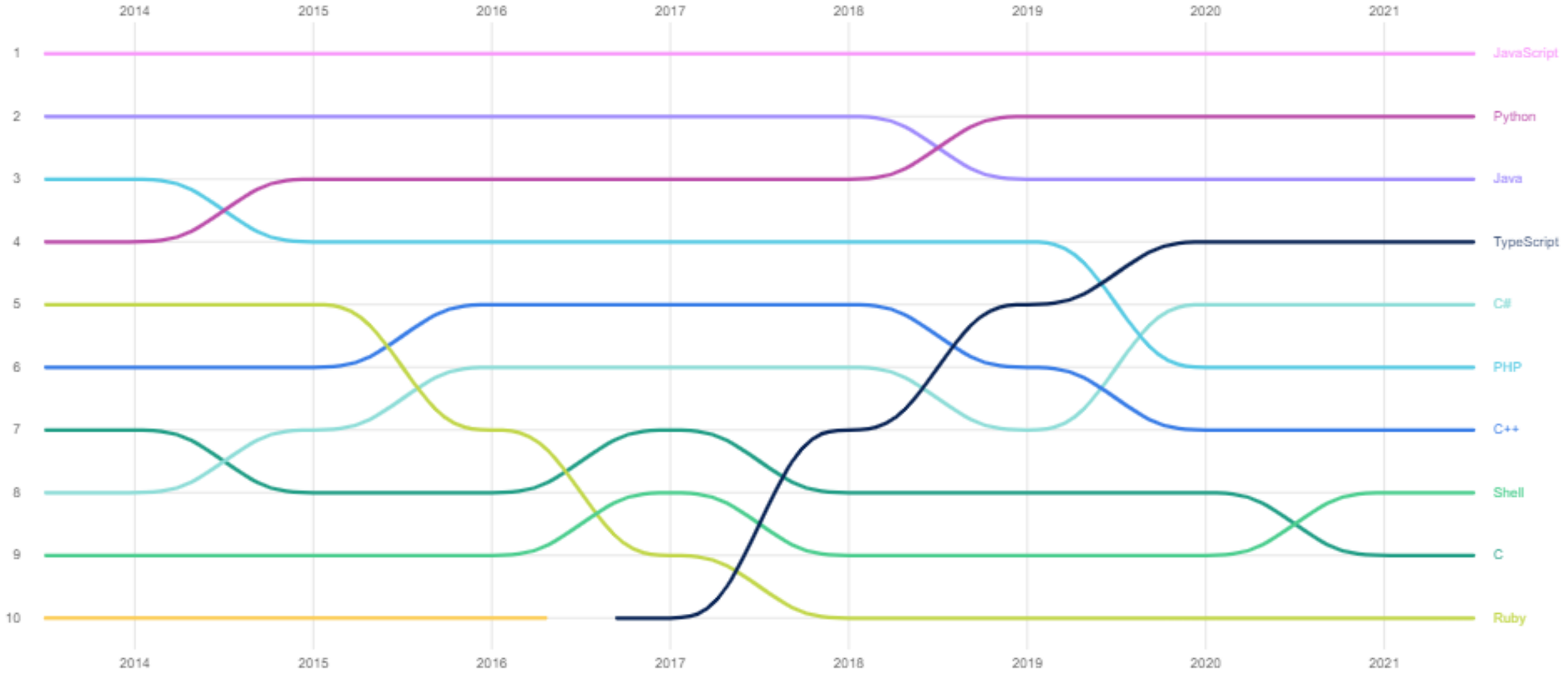


Image source

JavaScript is #1 Language on Github



Github.com 2021

Let's look back at the code and communities built on GitHub this year...

73 M+
Total developers on GitHub

16 M+
New users in 2021

84 %
of Fortune 100 companies use GitHub Enterprise

61 M+
New repositories created in the last year

170 M
Pull requests merged

https://octoverse.github.com/

Github.com 2020

Let's look back at the code and communities built on GitHub this year...

Based on the data collection range of October 2019 - September 2020.

56 M+
total developers on GitHub

72 %
of Fortune 50 companies use GitHub Enterprise

60 M+
new repositories created in the last year

1.9 B+
contributions added in the last year

https://octoverse.github.com/

GitHub.com 2019

40 m+

developers on GitHub, including 10M new users in 2019.*

44 m+

repositories created in the last year—and 44% more developers created their first repository in 2019 than in 2018.*

87 m+

pull requests merged in the last year—and 28% more developers opened their first pull request in 2019 than in 2018.*

20 m+

issues closed in the last year. That's a lot of decisions made, bugs fixed, and boxes checked.*

https://octoverse.github.com/

Quick Setup Checkpoint

You have installed

- NodeJS
- Git
- Visual Studio Code (Code-Runner and Bracket matcher extensions)

Quick Setup Checkpoint

You have installed

- NodeJS
- Git
- Visual Studio Code (Code-Runner and Bracket matcher extensions)

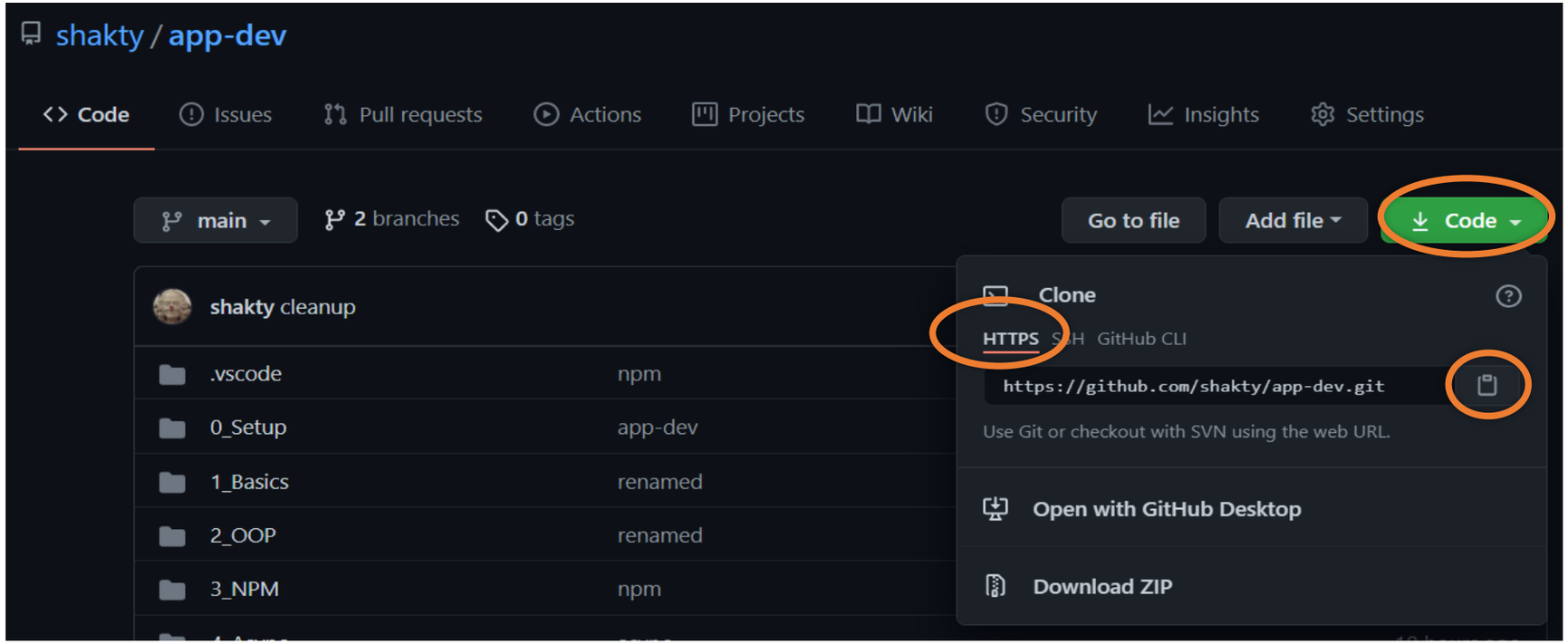
Fork the repository of exercises onto your GitHub account

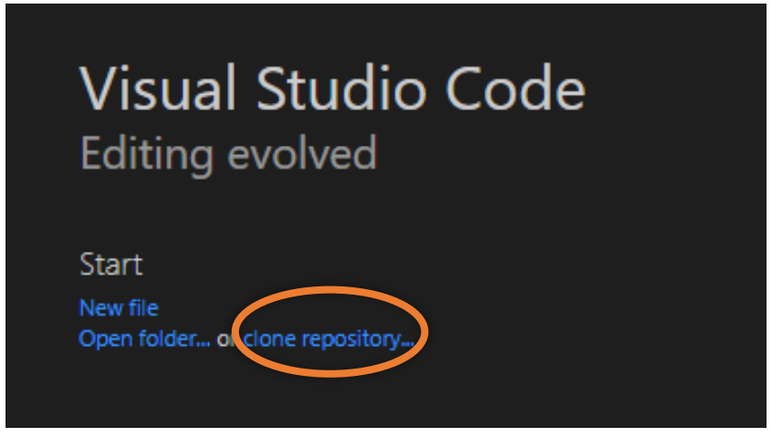
https://github.com/shakty/app-dev-day-1

Clone the forked repository onto your machine

Forking Instructions

https://github.com/shakty/app-dev



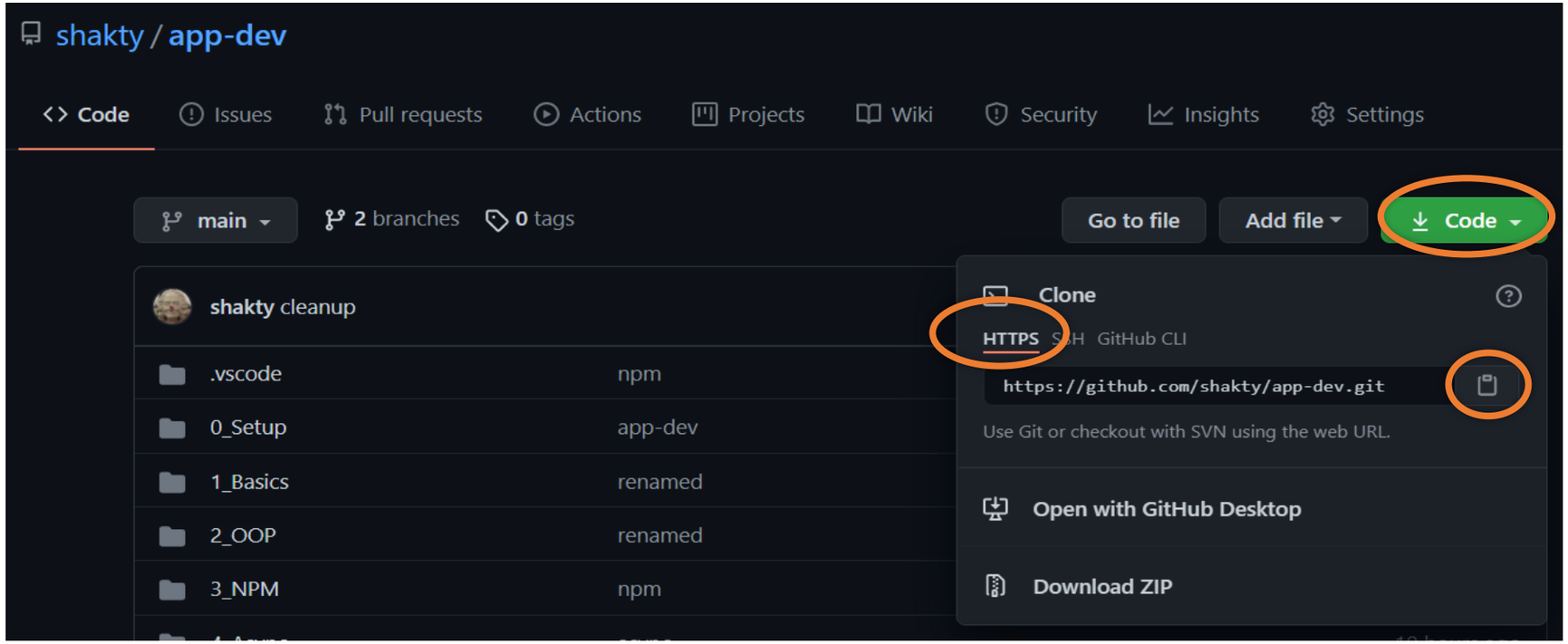


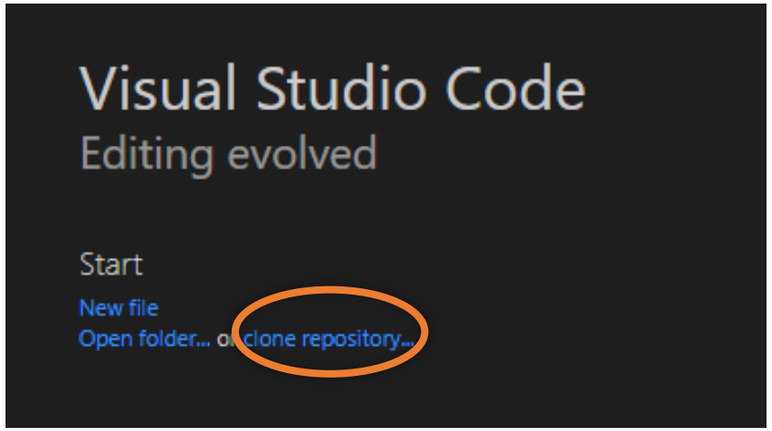
For Atom, I made this video:

https://www.youtube.com/watch?v=MDU2p9YtvlA

Forking Instructions

https://github.com/shakty/app-dev



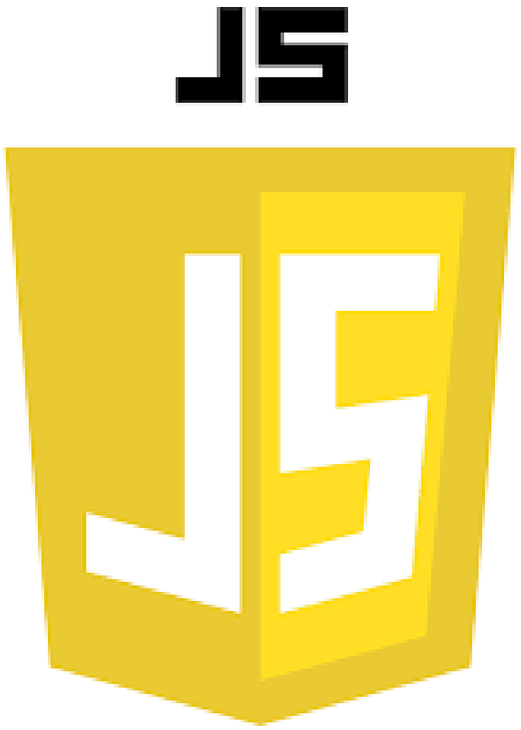


For Atom, I made this video:

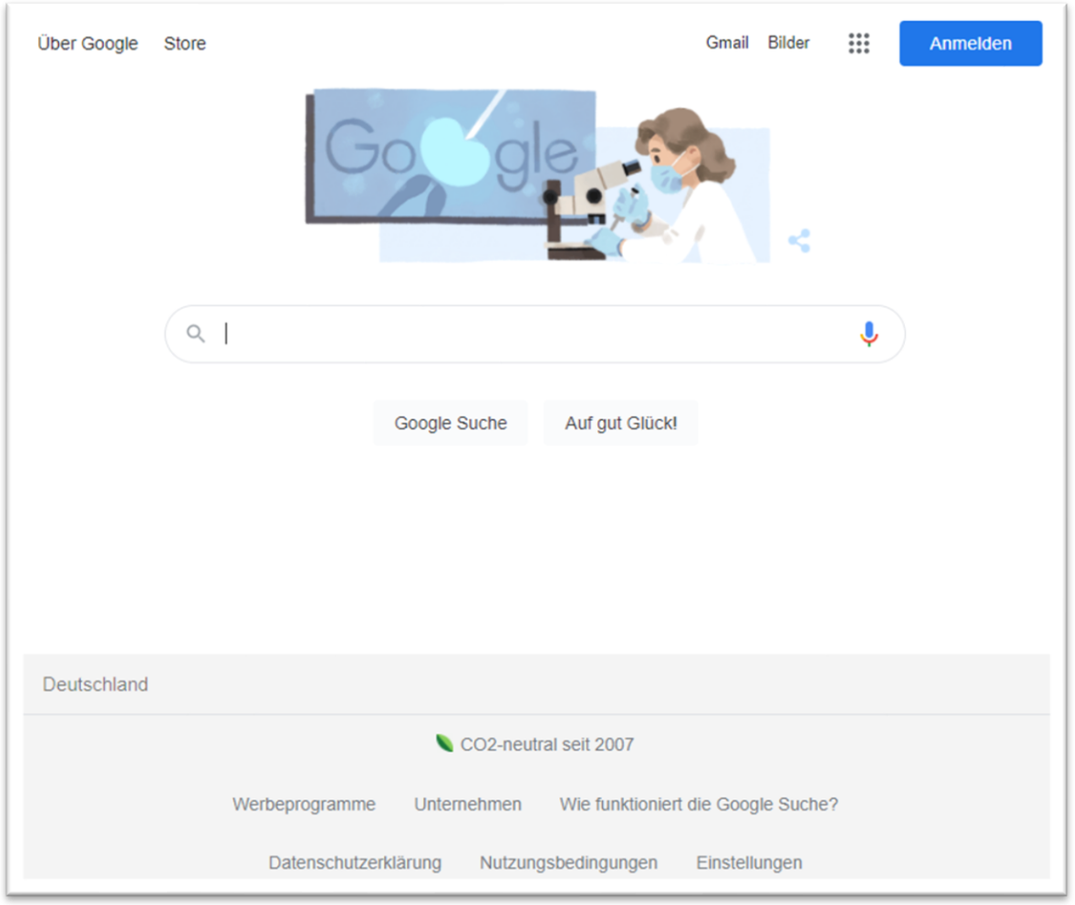
https://www.youtube.com/watch?v=MDU2p9YtvlA

Do not mix up with spooning :)

Recap: HTML, JS, CSS

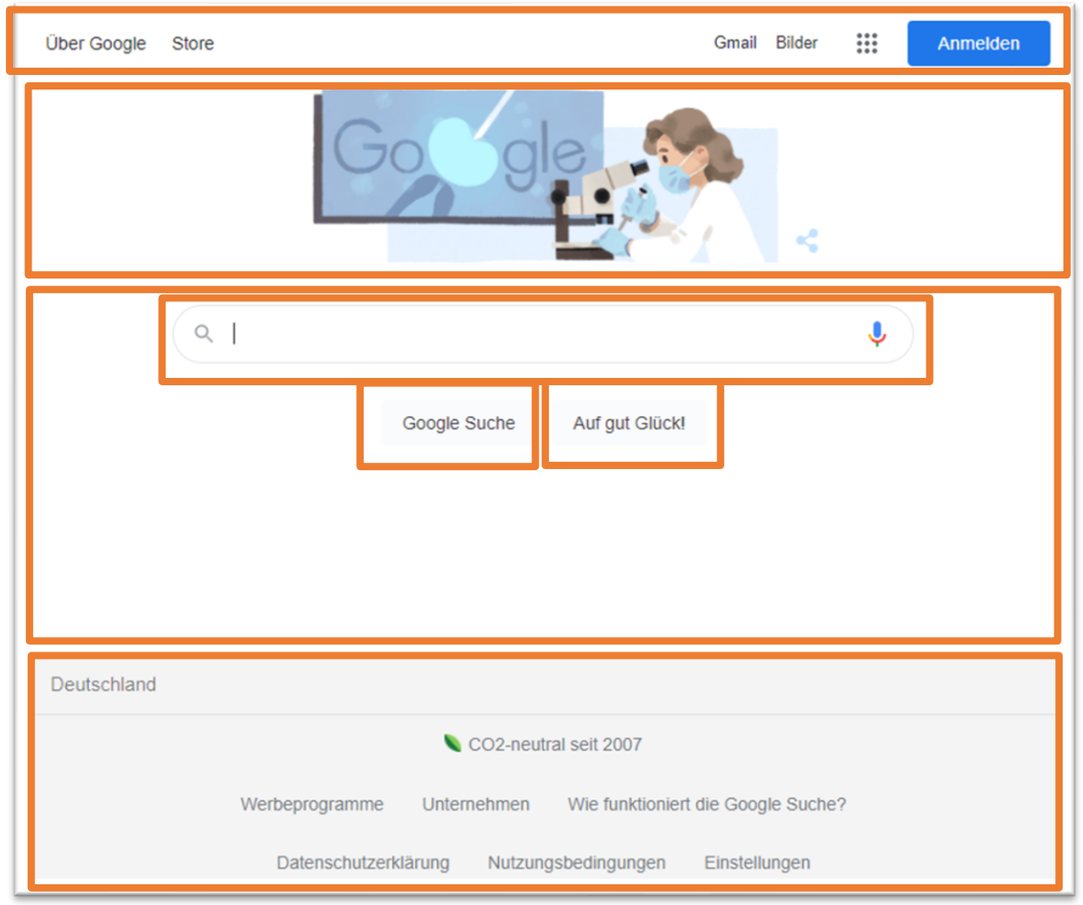


The Browser: Behind the Scenes



Every web page that we visit is rendered by the browser using a combination of the following three technologies:

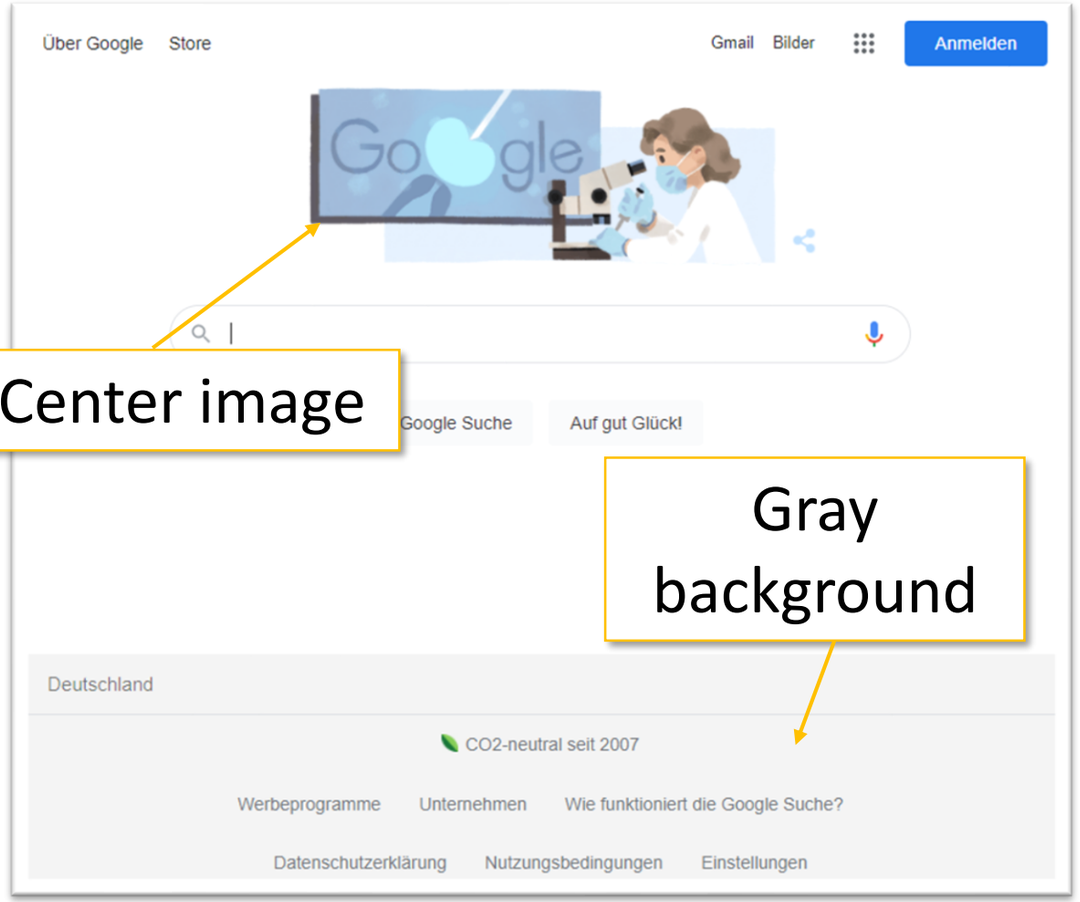
The Browser: Behind the Scenes



Every web page that we visit is rendered by the browser using a combination of the following three technologies:

Structure and Content

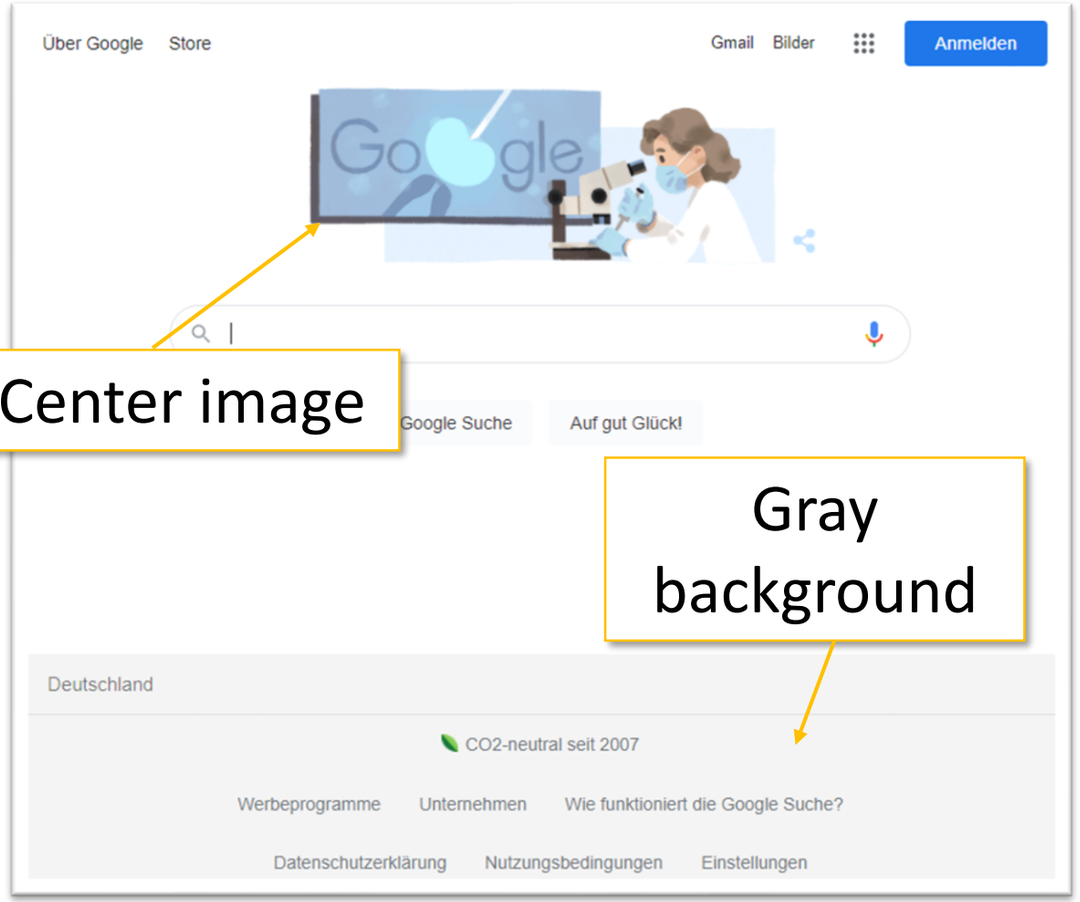
The Browser: Behind the Scenes



Every web page that we visit is rendered by the browser using a combination of the following three technologies:

Styling and simple interactions

The Browser: Behind the Scenes



Every web page that we visit is rendered by the browser using a combination of the following three technologies:

Complex interactions, communication with remote servers, logging, tracking, etc.

Some of the HTML Page's Inhabitants

DOM Tree

```
<HTML>
   <HEAD>
      <LINK>
      <SCRIPT>
   </HEAD>
   <BODY>
   …
   </BODY>
</HTML>
```

Presentation Tags

<P>

<DIV>

<SPAN>

Images and Links

<IMG>

<A>

Forms

<INPUT>

<TEXTAREA>

Attributes

<DIV id=“header”>

<SPAN class=“bold”>

<IMG SRC=“image.jpg” />

<A HREF=“newpage.htm”>

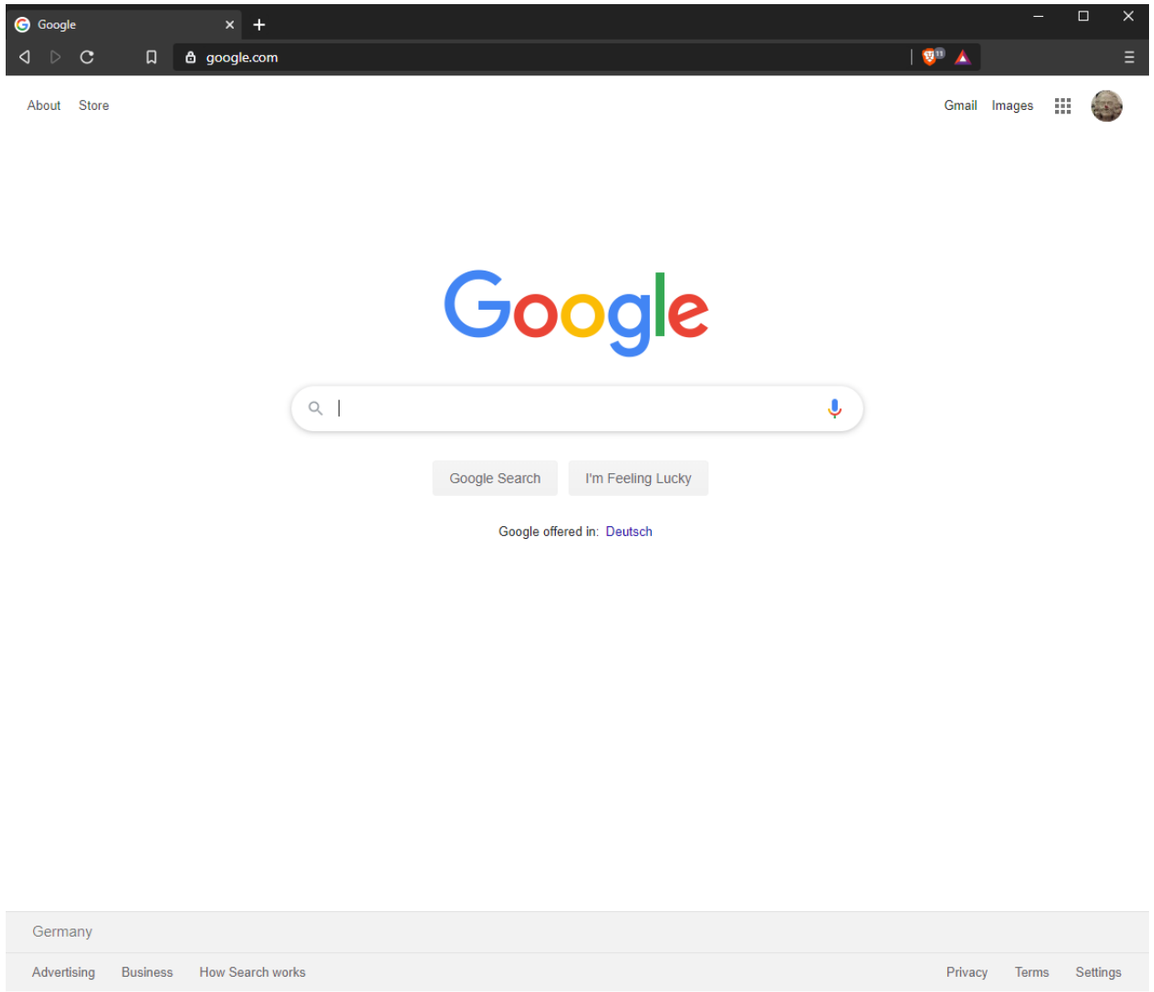
<INPUT disabled>

CSS Declarations

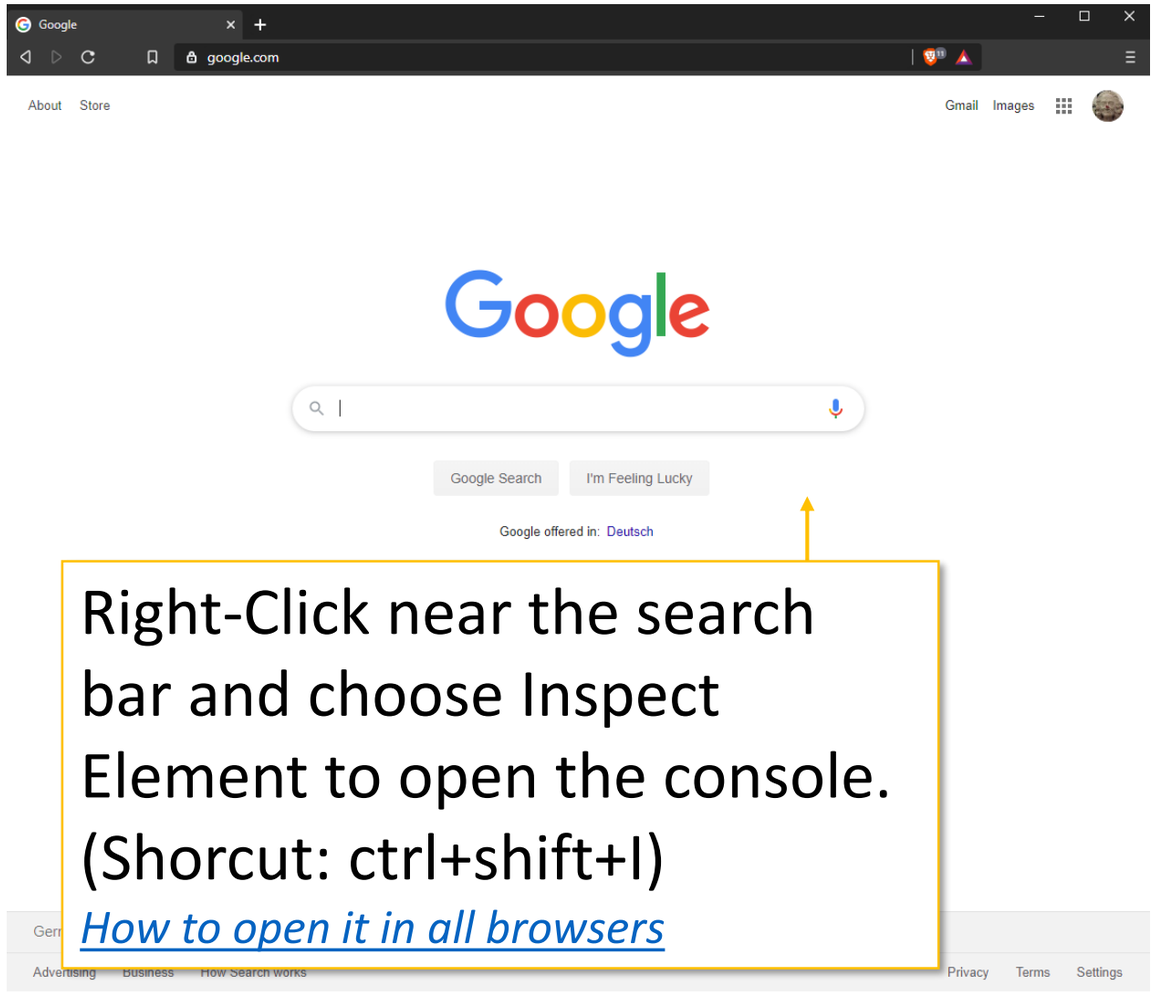
.bold { font-weight: bold };

#header { width: 600px };

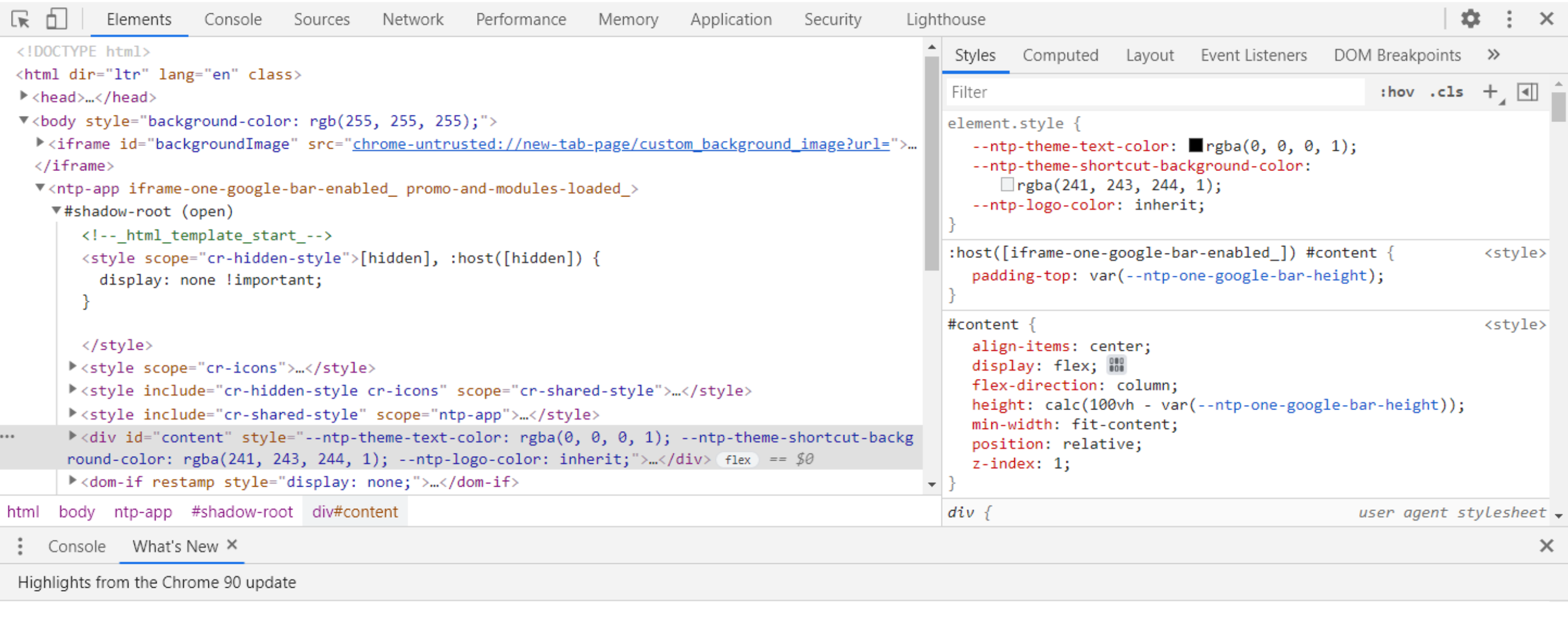
HTML, CSS, JavaScript



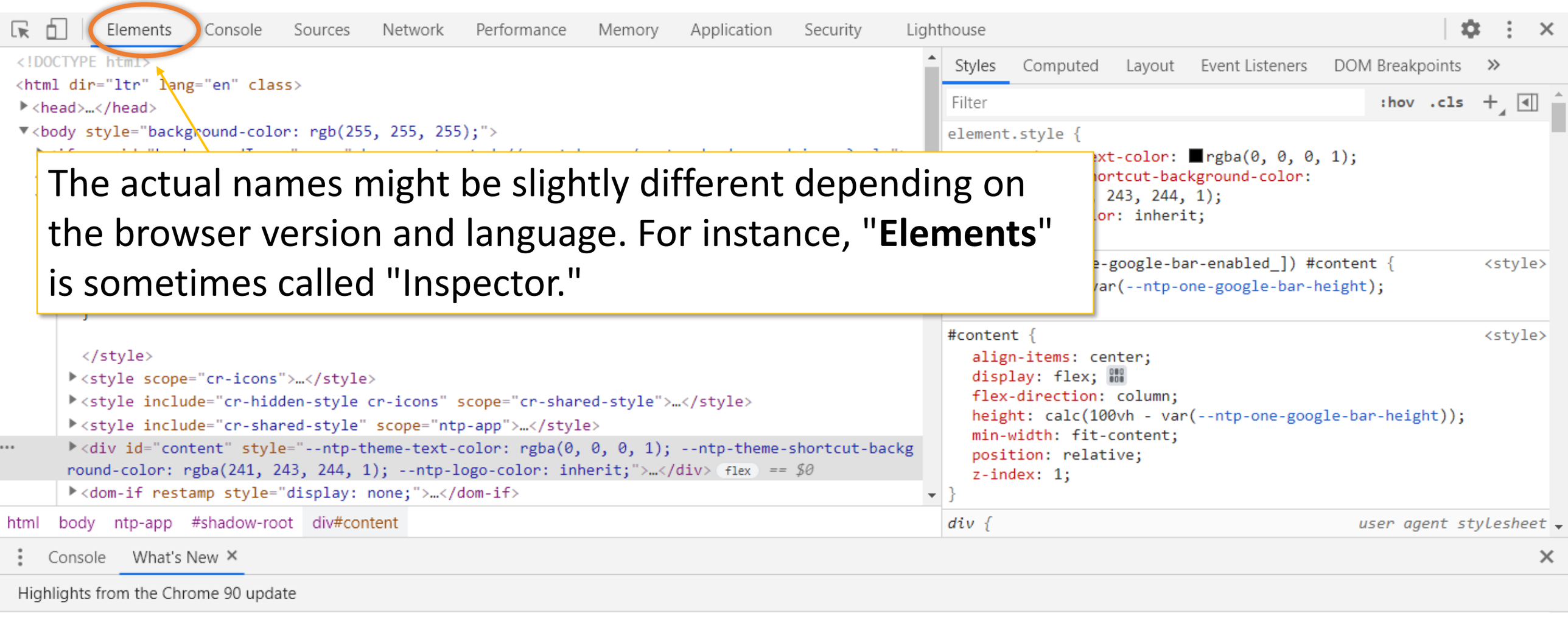
HTML, CSS, JavaScript



Developer Tools: Elements

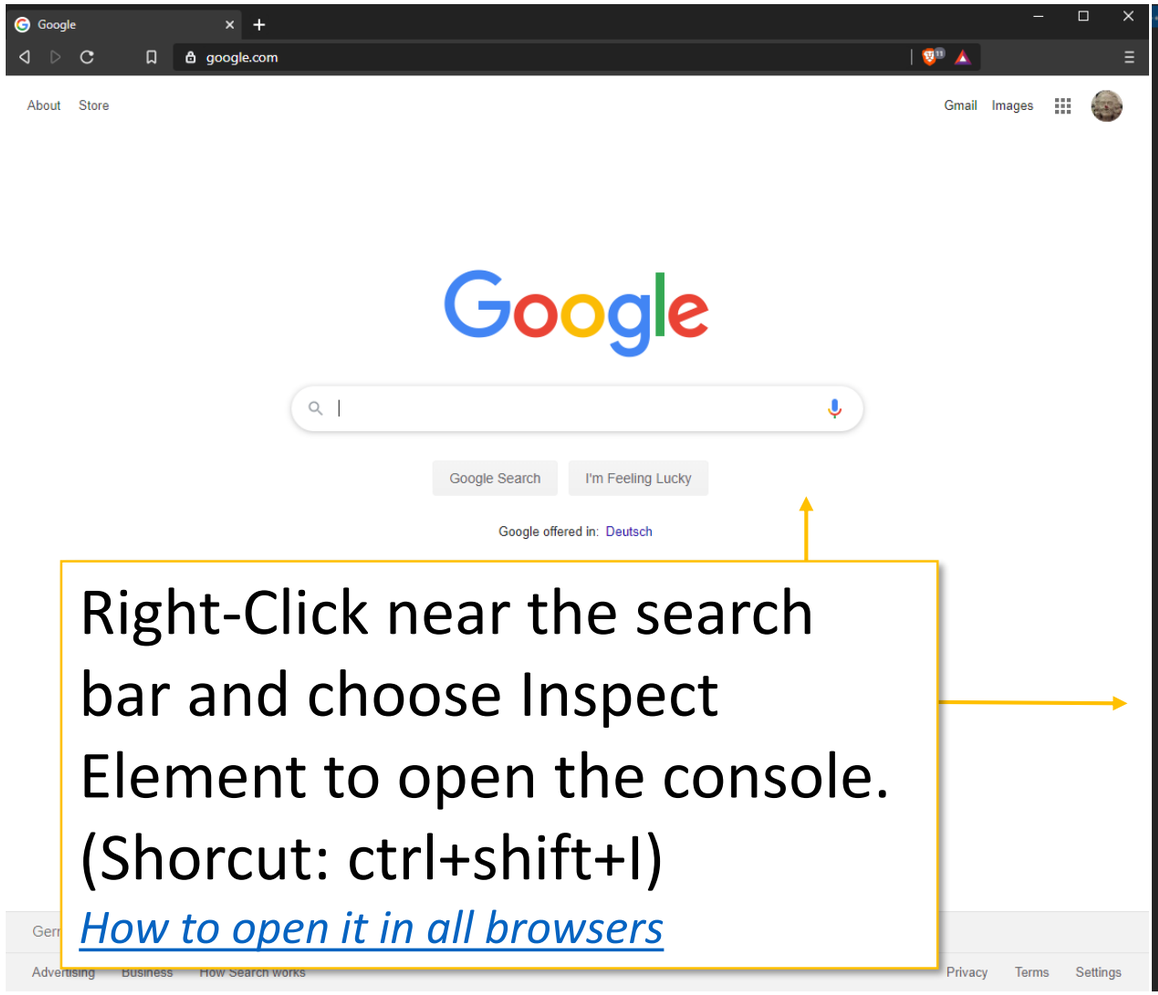


Developer Tools: Elements



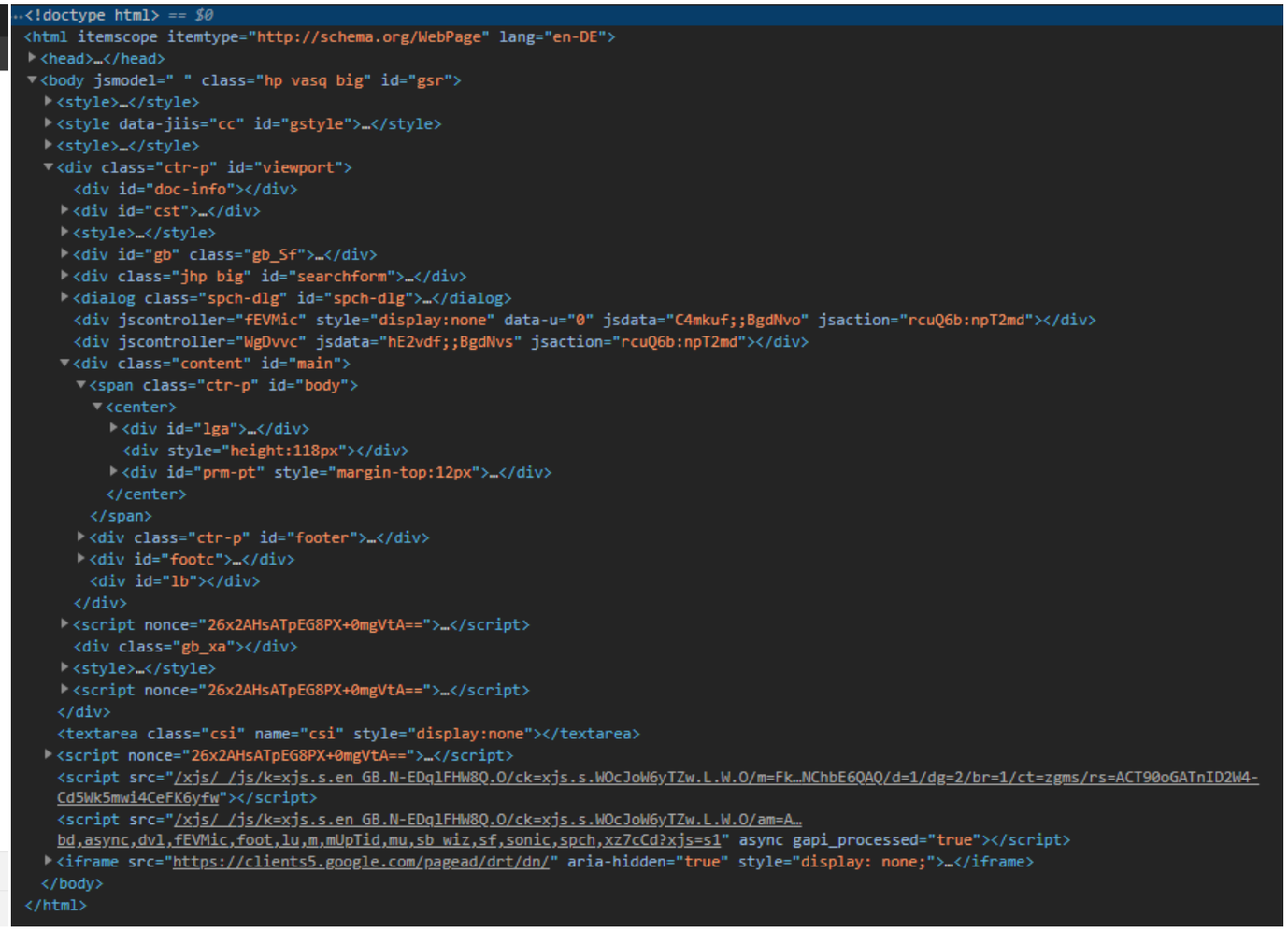
The actual names might be slightly different depending on the browser version and language. For instance, "**Elements**" is sometimes called "Inspector."

HTML, CSS, JavaScript

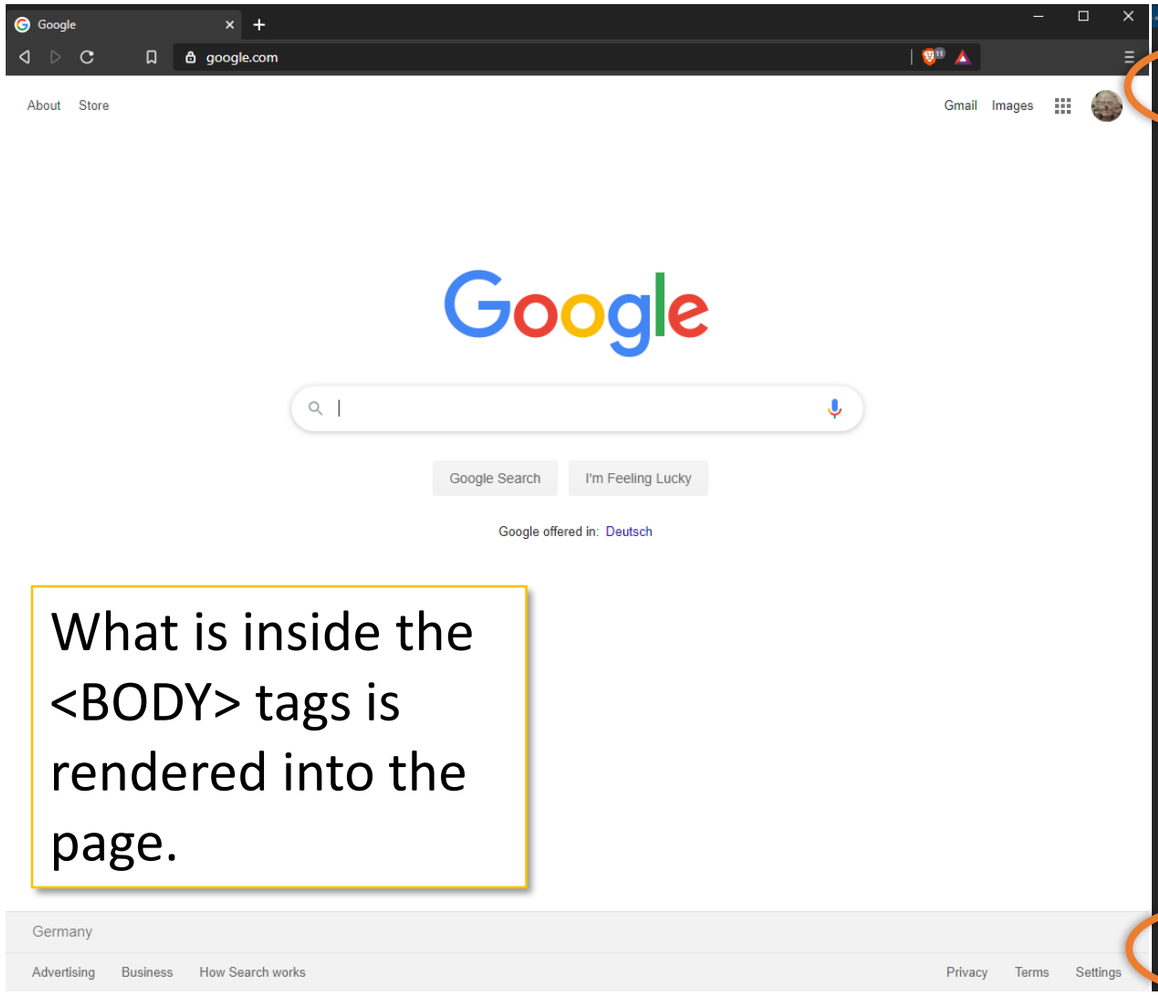


Right-Click near the search bar and choose Inspect Element to open the console. (Shorcut: ctrl+shift+I)

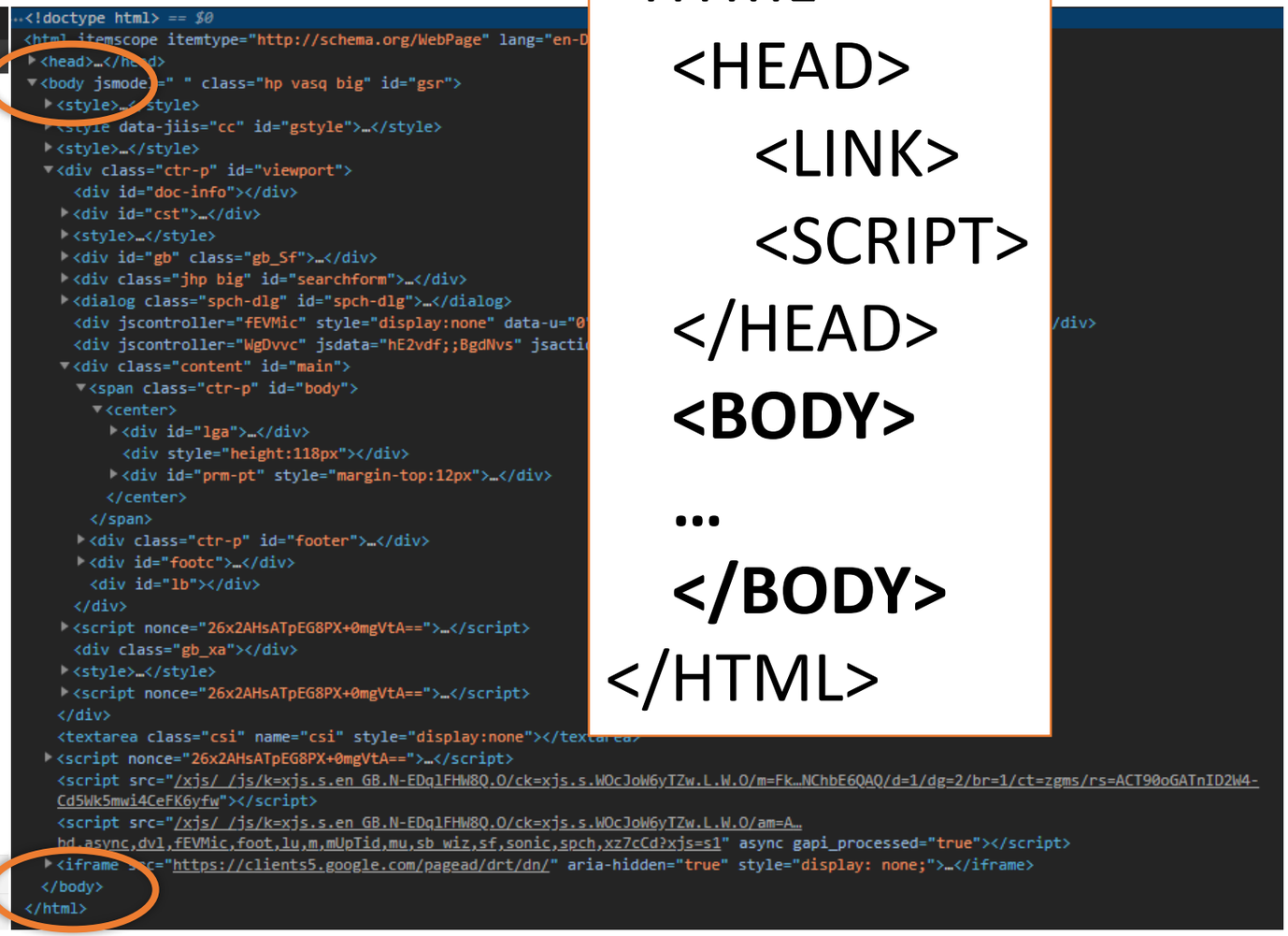
How to open it in all browsers



HTML, CSS, JavaScript



What is inside the <BODY> tags is rendered into the page.



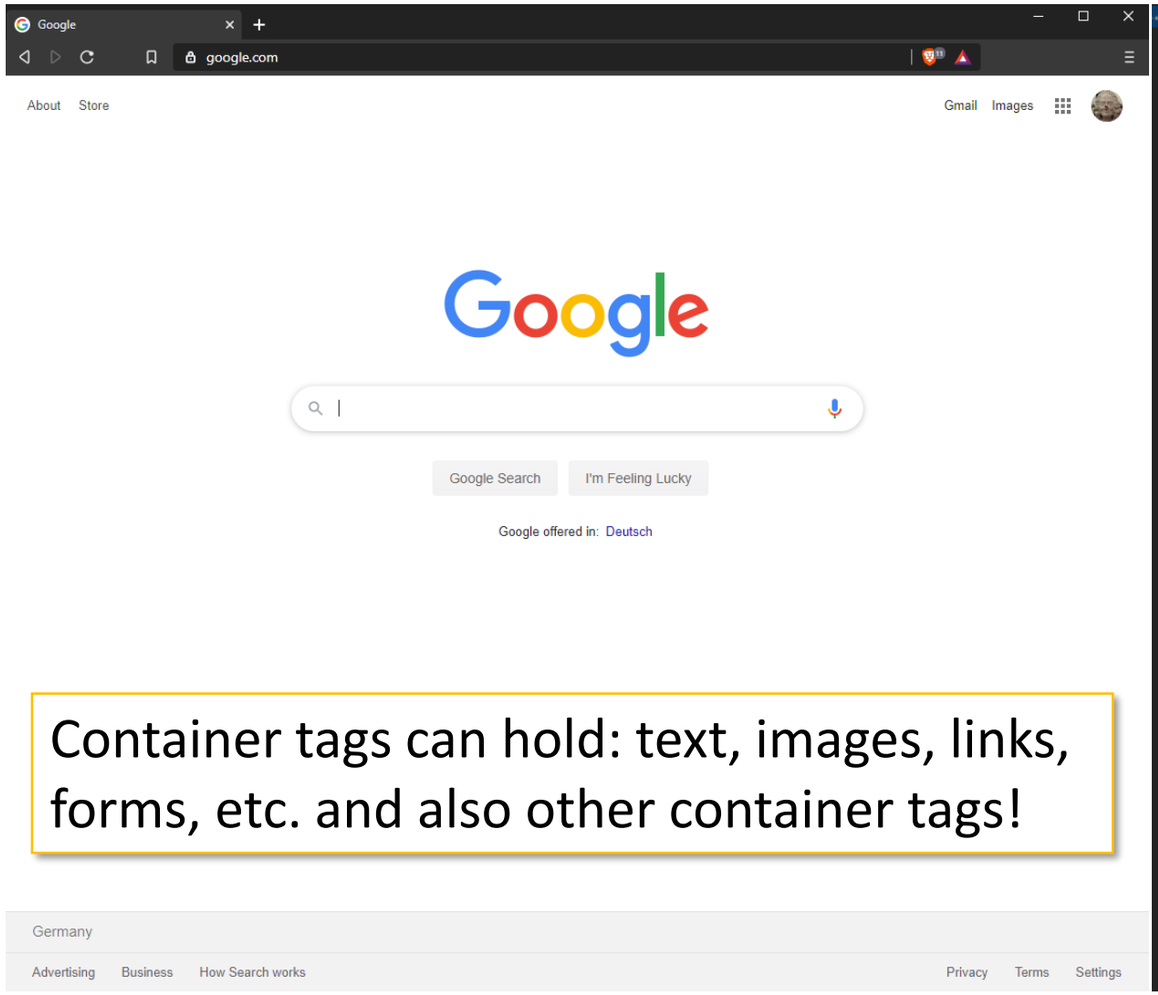
DOM Tree

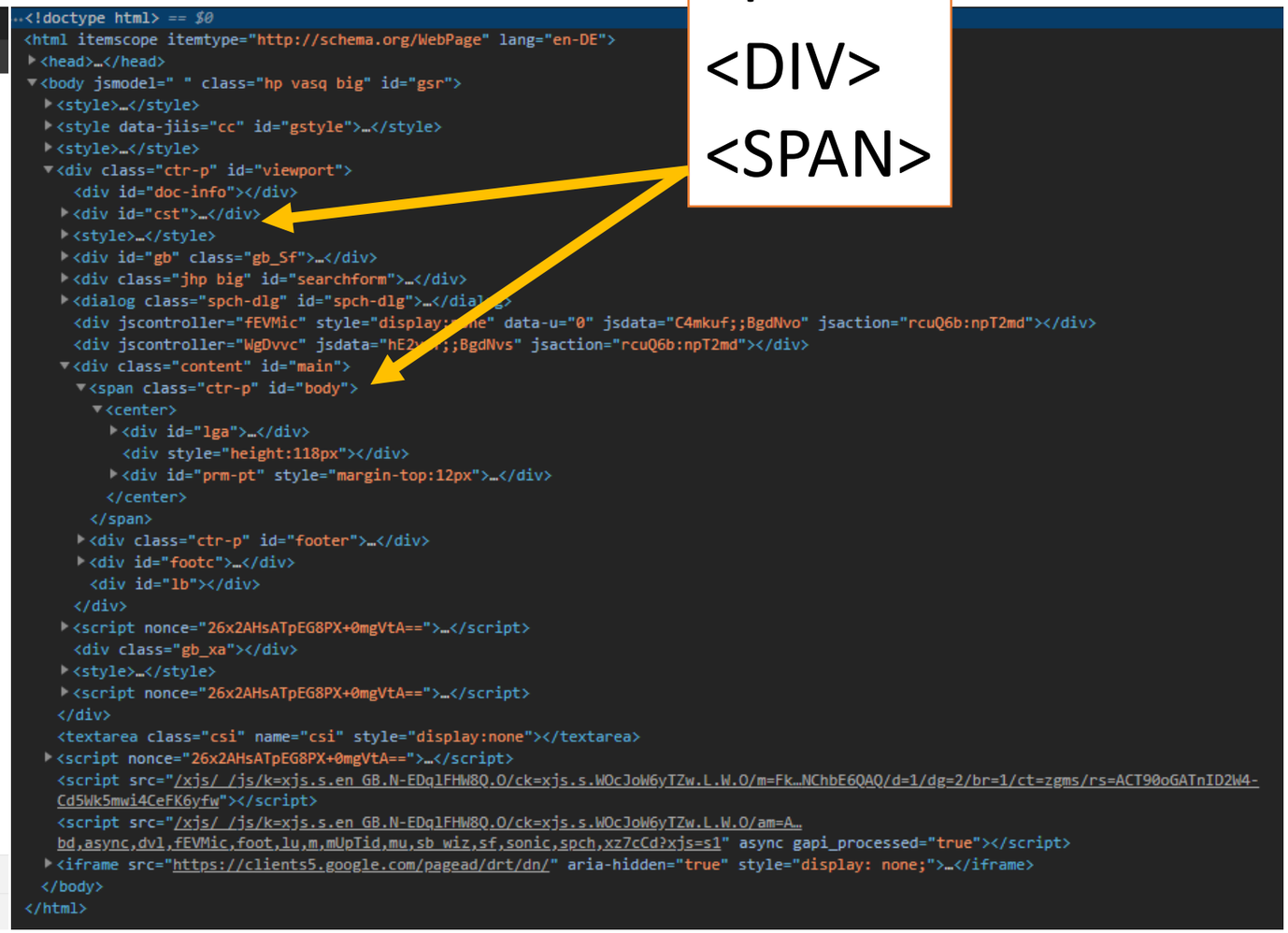
<HTML>
  <HEAD>
    <LINK>
    <SCRIPT>
  </HEAD>
  <BODY>
  …
  </BODY>
</HTML>

HTML, CSS, JavaScript

Container Tags

<P>
<DIV>
<SPAN>





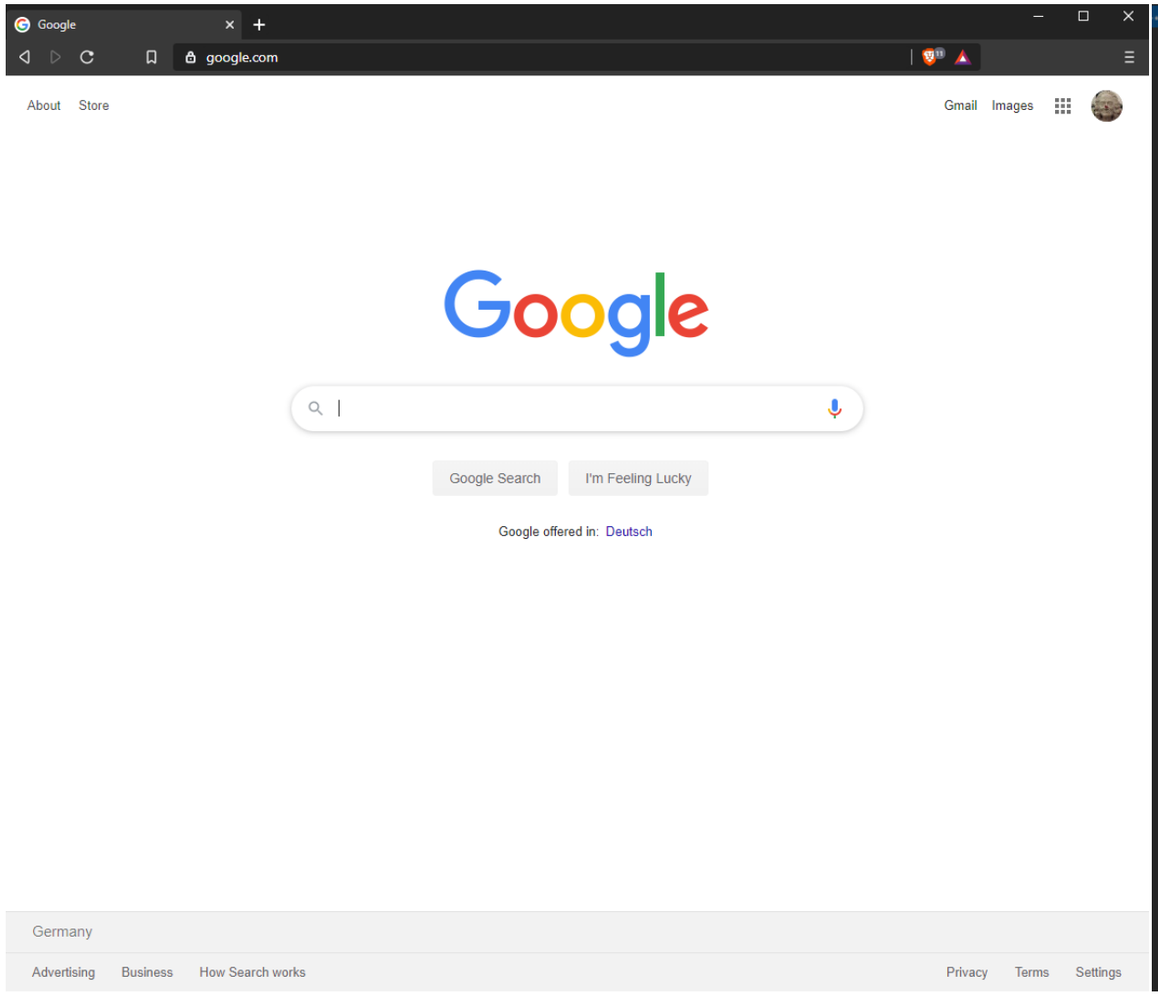
Container tags can hold: text, images, links, forms, etc. and also other container tags!

https://stackoverflow.com/questions/30879707/why-is-a-div-called-a-div-why-is-a-span-called-a-span

HTML, CSS, JavaScript

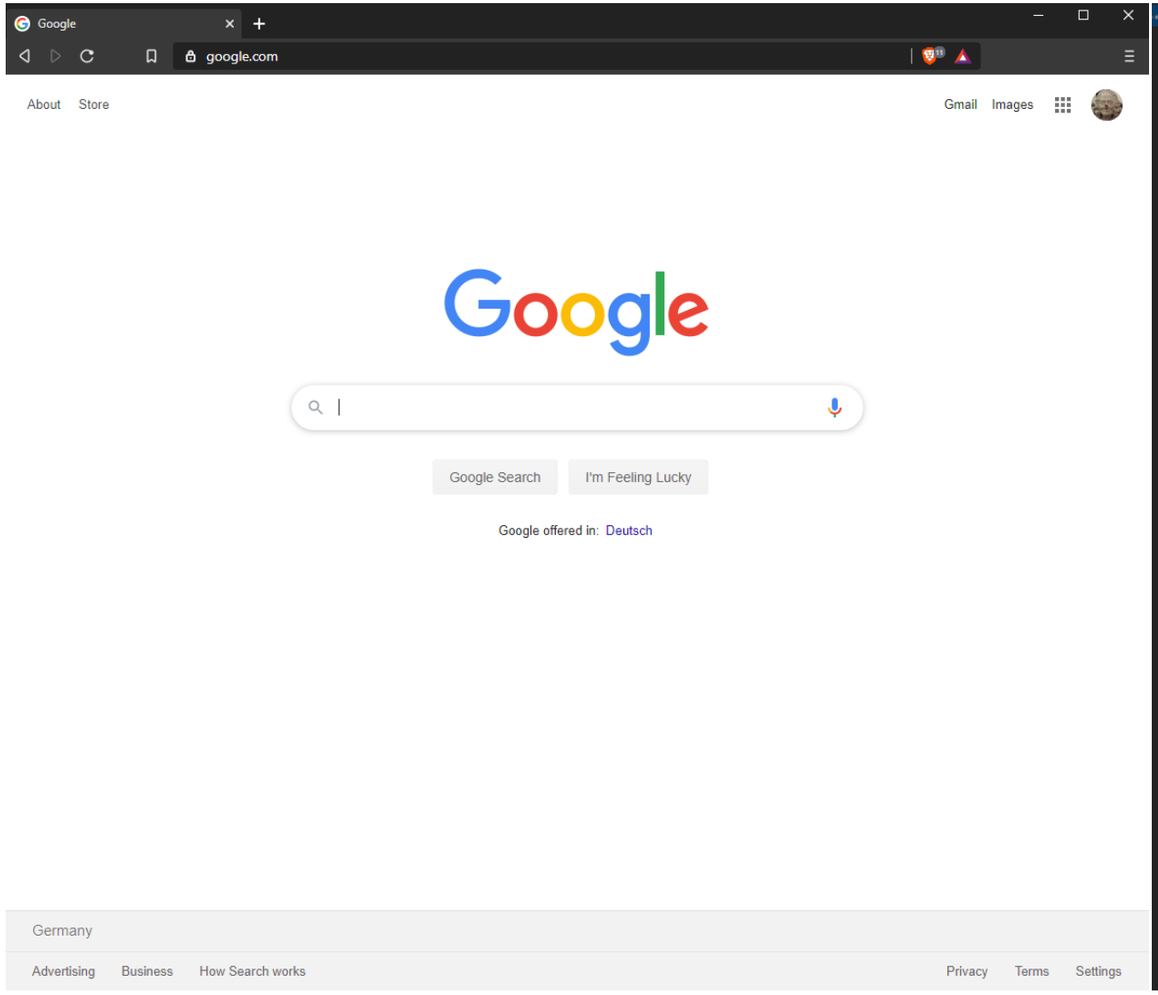
Style Tags

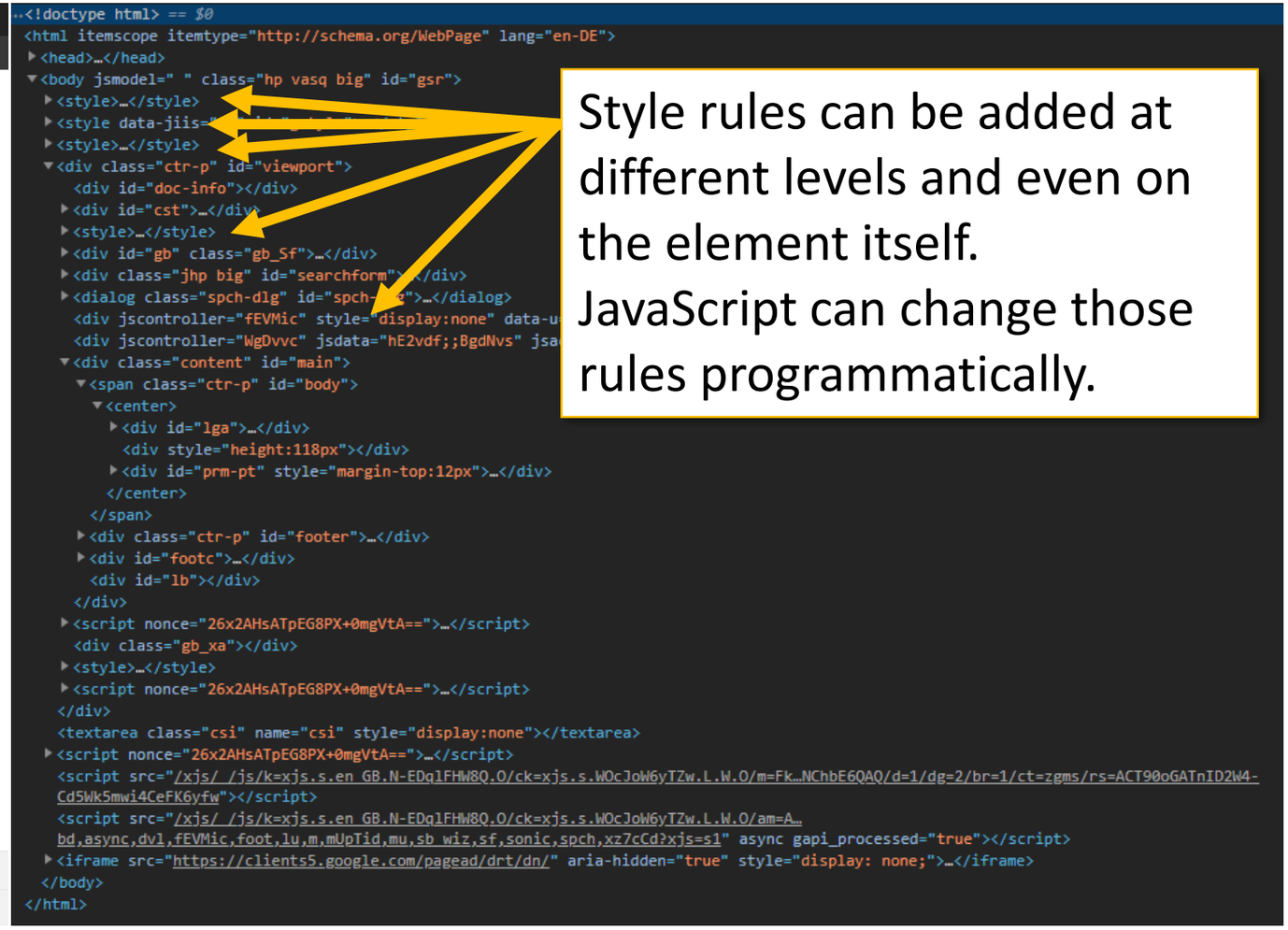
<STYLE>
<LINK>



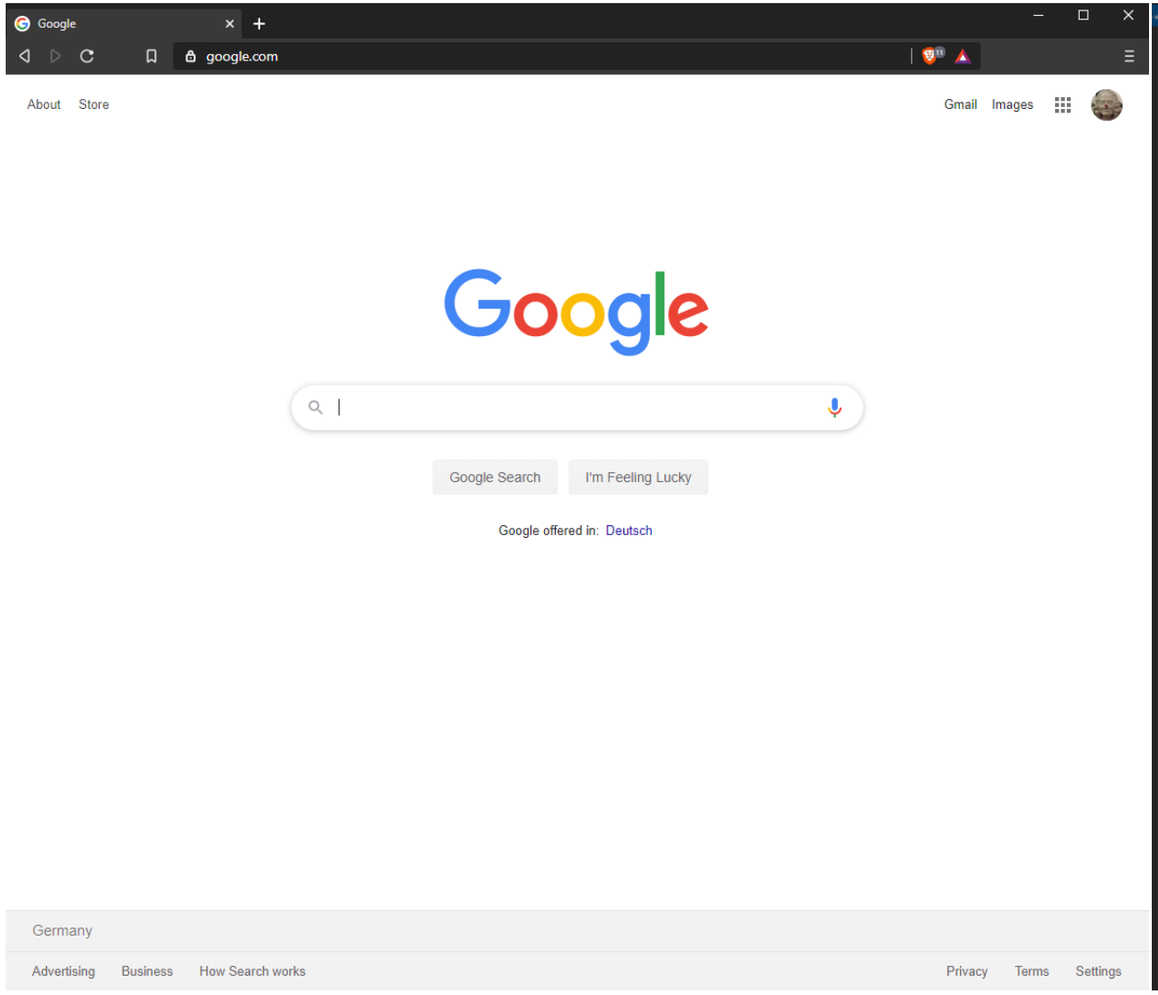

```
<!doctype html> == $0
<html itemscope itemtype="http://schema.org/WebPage" lang="en-DE">
 <head>…</head>
 <body jsmodel=" " class="hp vasq …">
  <style>…</style>
  <style data-jiis="cc" id="gstyle">…</style>
  <style>…</style>
  <div class="ctr-p" id="viewport">
   <div id="doc-info"></div>
   <div id="cst">…</div>
   <style>…</style>
   <div id="gb" class="gb_Sf">…</div>
   <div class="jhp big" id="searchform">…</div>
   <dialog class="spch-dlg" id="spch-dlg">…</dialog>
   <div jscontroller="fEVMic" style="display:none" data-u="0" jsdata="C4mkuf;;BgdNvo" jsaction="rcuQ6b:npT2md"></div>
   <div jscontroller="WgDvvc" jsdata="hE2vdf;;BgdNvs" jsaction="rcuQ6b:npT2md"></div>
   <div class="content" id="main">
    <span class="ctr-p" id="body">
     <center>
      <div id="lga">…</div>
      <div style="height:118px"></div>
      <div id="prm-pt" style="margin-top:12px">…</div>
     </center>
    </span>
    <div class="ctr-p" id="footer">…</div>
    <div id="footc">…</div>
    <div id="lb"></div>
   </div>
   <script nonce="26x2AHsATpEG8PX+0mgVtA==">…</script>
   <div class="gb_xa"></div>
   <style>…</style>
   <script nonce="26x2AHsATpEG8PX+0mgVtA==">…</script>
  </div>
  <textarea class="csi" name="csi" style="display:none"></textarea>
  <script nonce="26x2AHsATpEG8PX+0mgVtA==">…</script>
  <script src="/xjs/_/js/k=xjs.s.en_GB.N-EDqlFHW8Q.O/ck=xjs.s.WOcJoW6yTZw.L.W.O/m=Fk…NChbE6QAQ/d=1/dg=2/br=1/ct=zgms/rs=ACT90oGATnID2W4-Cd5Wk5mwi4CeFK6yfw"></script>
  <script src="/xjs/_/js/k=xjs.s.en_GB.N-EDqlFHW8Q.O/ck=xjs.s.WOcJoW6yTZw.L.W.O/am=A…bd,async,dvl,fEVMic,foot,lu,m,mUpTid,mu,sb_wiz,sf,sonic,spch,xz7cCd?xjs=s1" async gapi_processed="true"></script>
  <iframe src="https://clients5.google.com/pagead/drt/dn/" aria-hidden="true" style="display: none;">…</iframe>
 </body>
</html>
```

HTML, CSS, JavaScript

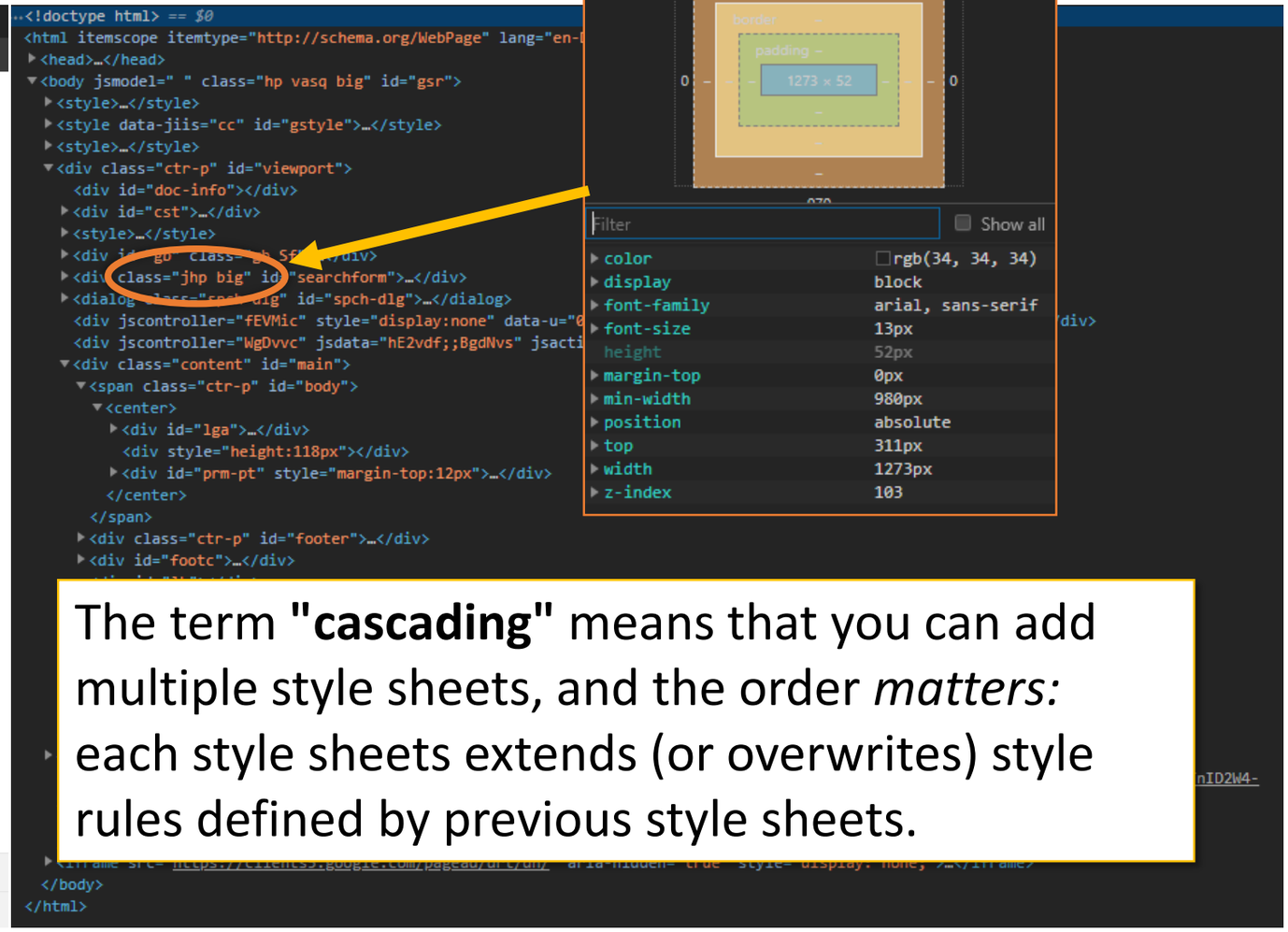




HTML, CSS, JavaScript



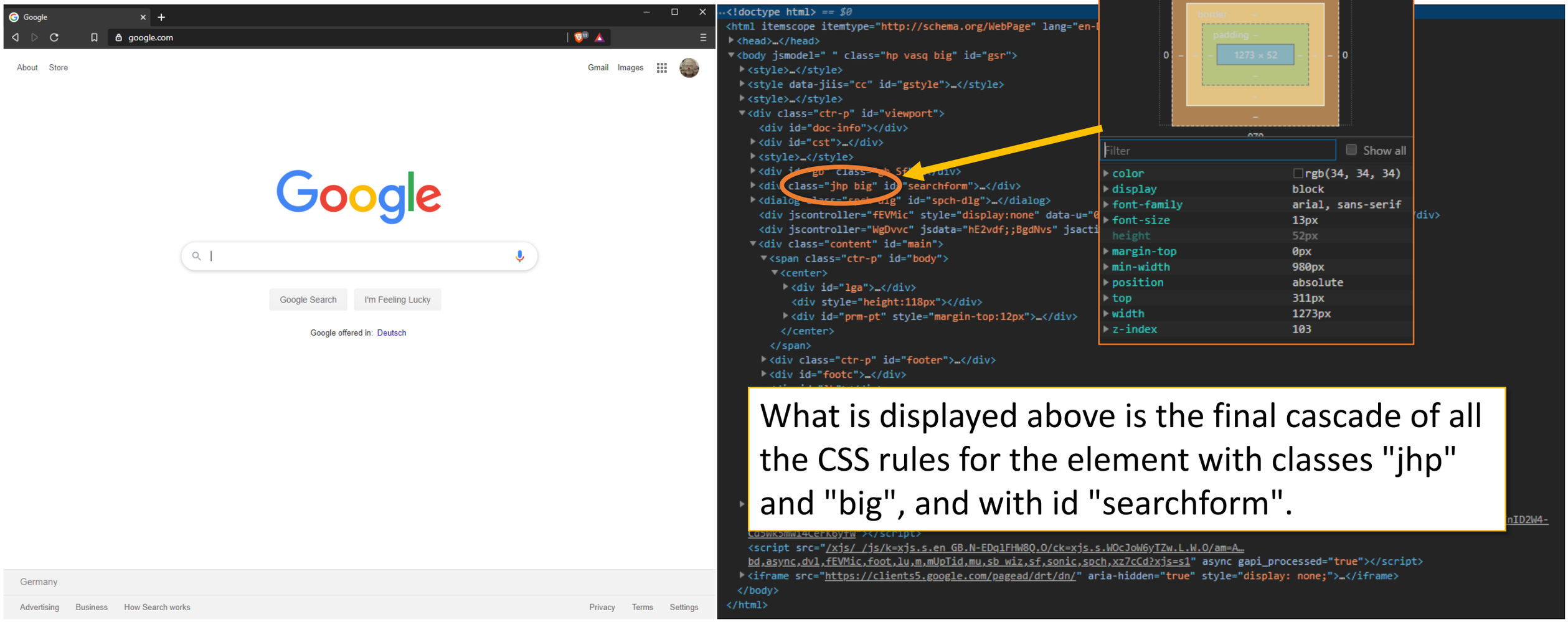
CSS (Cascading Style Sheets)



The term **"cascading"** means that you can add multiple style sheets, and the order *matters:* each style sheets extends (or overwrites) style rules defined by previous style sheets.

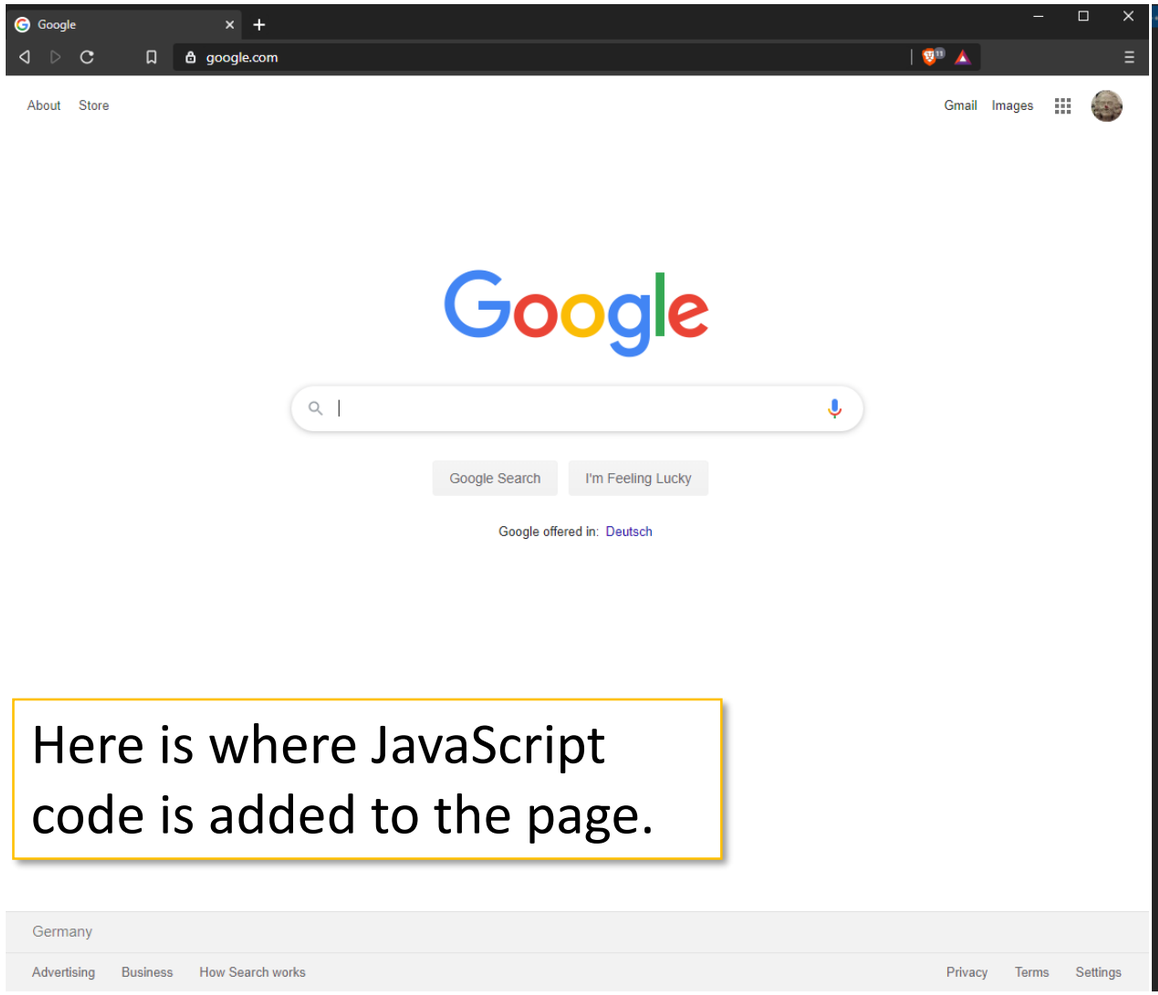
HTML, CSS, JavaScript

CSS (Cascading Style Sheets)

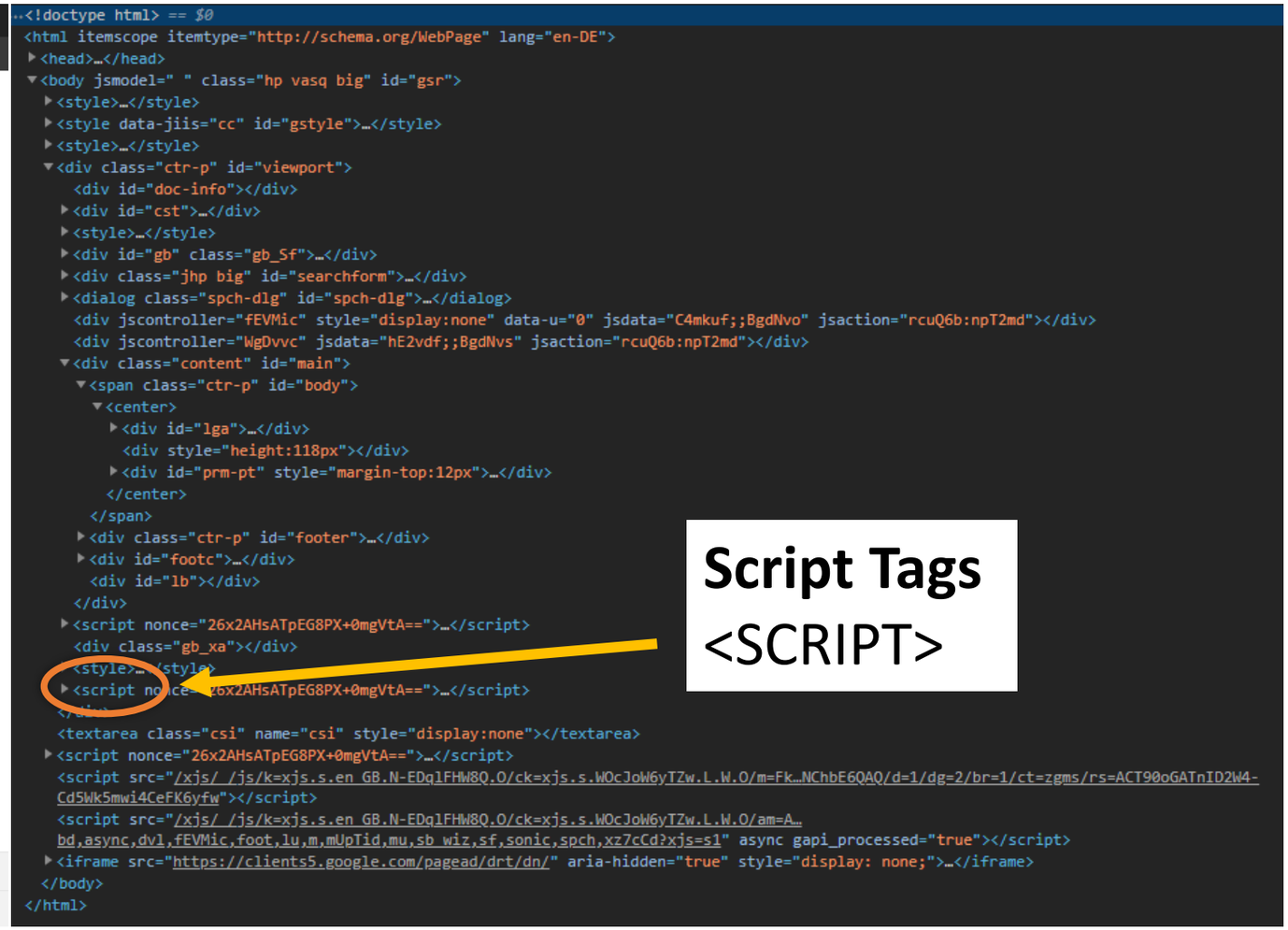


What is displayed above is the final cascade of all the CSS rules for the element with classes "jhp" and "big", and with id "searchform".

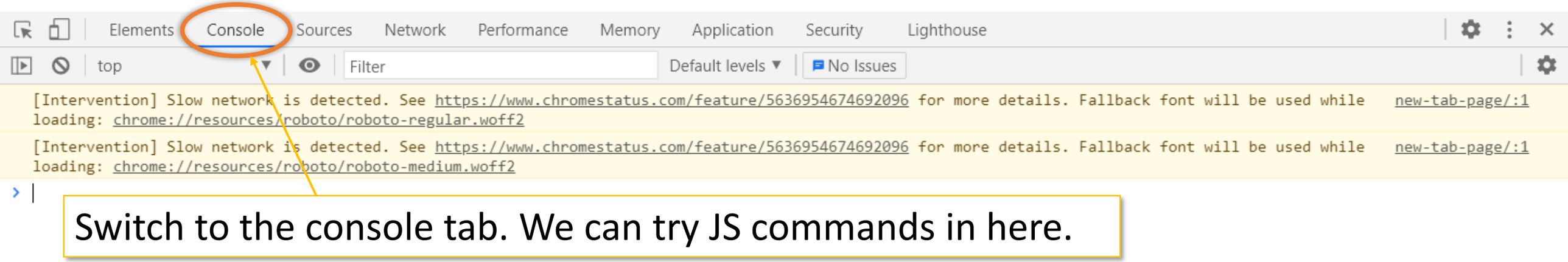
HTML, CSS, JavaScript



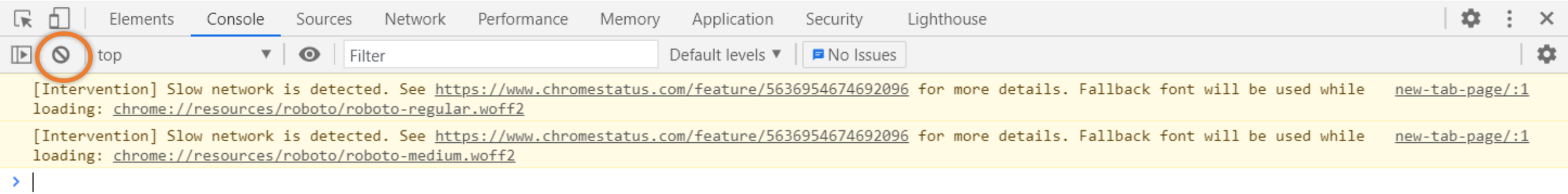
Here is where JavaScript code is added to the page.



Developer Tools: Console



Developer Tools: Console

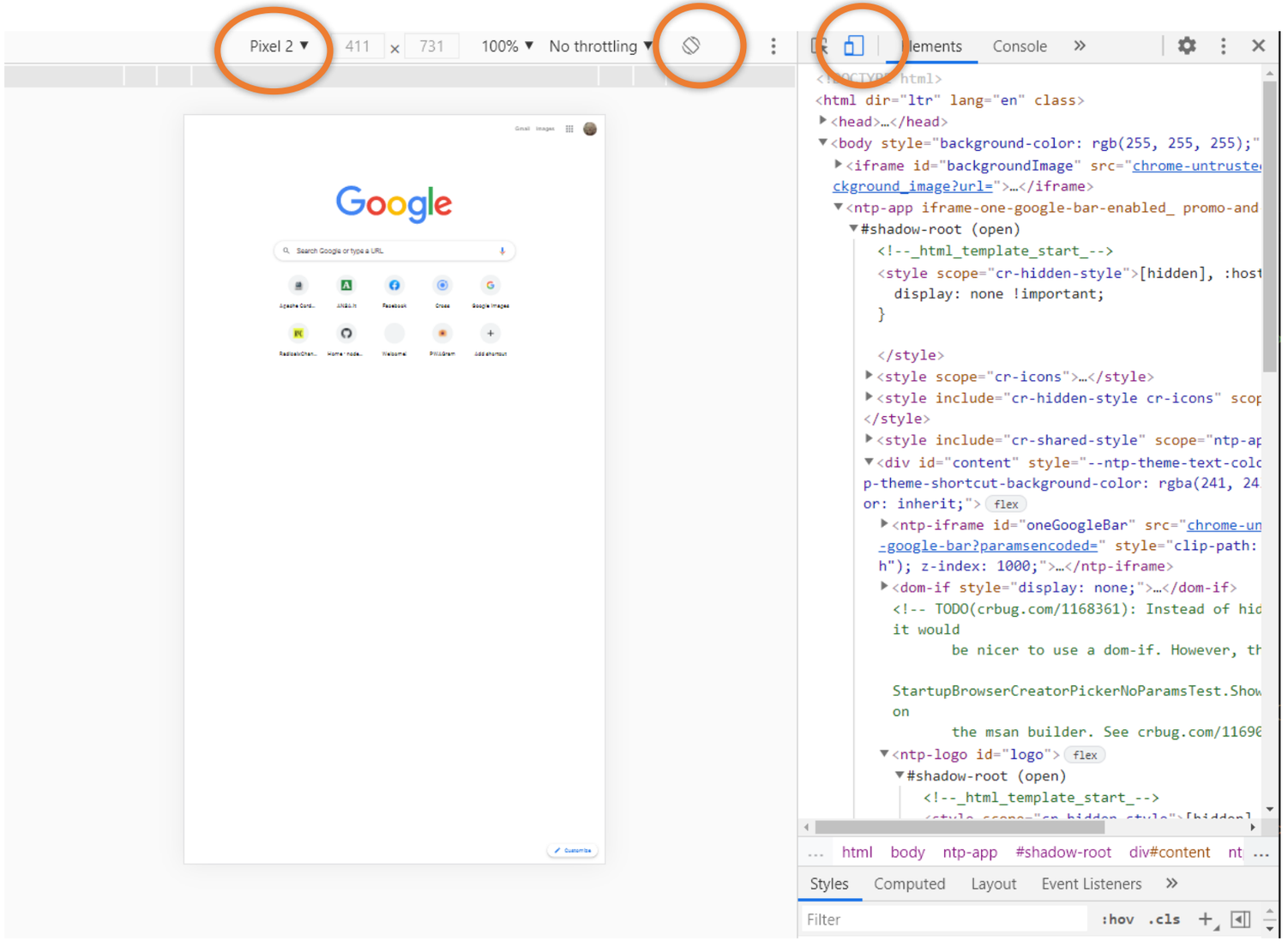


Clear any pre-existing output: click on the button or type: `clear()`

Then, write something of your own with `console.log()`

```
console.log('Hello World');
```

Test in Mobile View



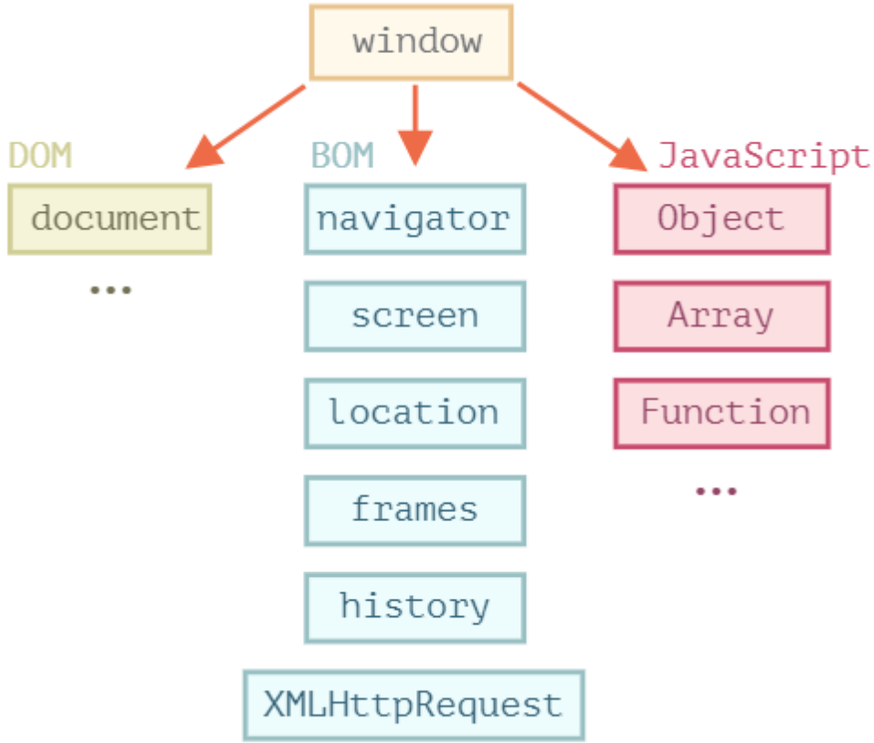
What Can JS in the Browser do?

Every JS object in the browser is child of the `window` object, including:

1. DOM (Document Object Model) objects (i.e., all things displayed on the page) exposed through the `document` object

2. Extra info and methods about the browser itself

3. JavaScript language itself



Img source: JavaScript.info

JS,HTML,CSS: References

- https://www.stefanobalietti.com/teaching/programming-fundamentals/
- https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/
- https://javascript.info/
- https://developer.mozilla.org/en-US/docs/Web
- https://css-tricks.com/
- https://www.w3schools.com/html/
- https://www.w3schools.com/css/

Module 1: NodeJS and NPM

Module 1: NodeJS and NPM

Learning Goals

- You should already know some JavaScript, soft reboot
- Search and install NodeJS packages from NPM
- What is the package.json file
- The node_modules directory
- Load packages into NodeJS programs
- Requiring and exporting local files

Node.JS

- **Node.JS** was invented in 2009 by *Ryan Dahl* and other developers working at Joyent

- Combination of Google's V8 JavaScript engine, an event loop, and a low-level I/O API

- **npm**, the node package manager, in 2011

- Versions: 0.10, 0.12, 4.0 … 16.0!

Node.JS

- **Node.JS** was invented in 2009 by *Ryan Dahl* and other developers working at Joyent

- Combination of Google's V8 JavaScript engine, an event loop, and a low-level I/O API

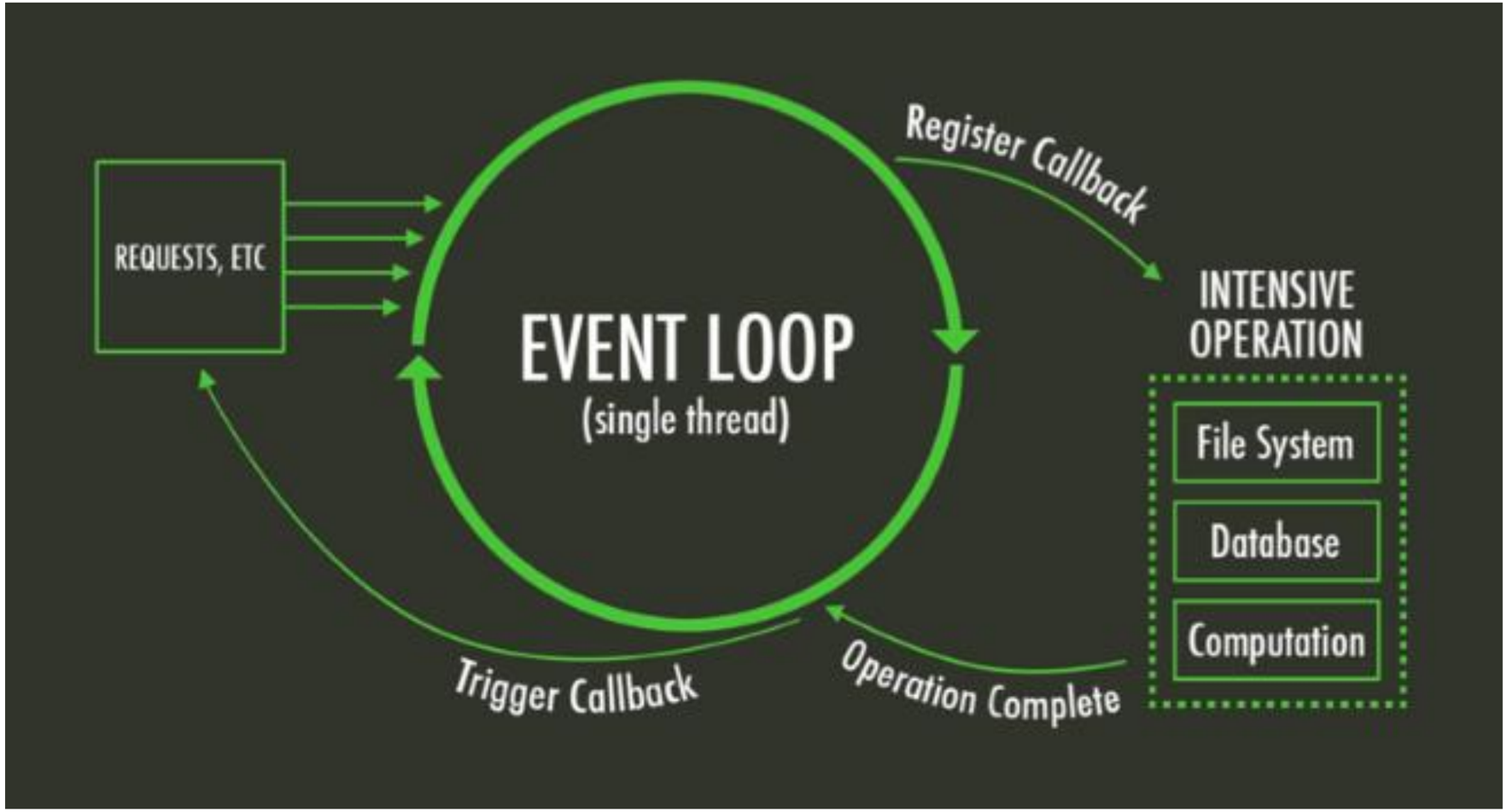
- **npm**, the node package manager, in 2011

- Versions: 0.10, 0.12, 4.0 … 16.0!

Module 1: References

- https://nodejs.org/en/
- https://www.npmjs.com/
- https://docs.npmjs.com/cli/v6/configuring-npm/package-json
- https://www.geeksforgeeks.org/node-js-modules/

Module 2: Asynchronous Code



Picture source

Module 2: Asynchronous Code

Learning Goals

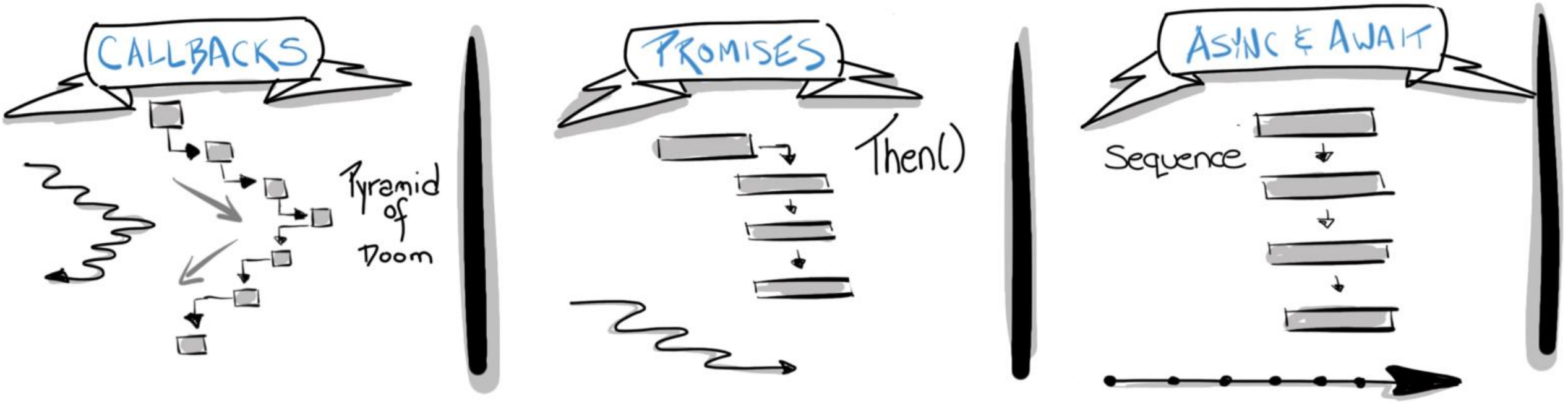
- Writing **async code** with:
  - callbacks
  - promises,
  - async/await pattern

- Understanding **event-listeners** in the browser

- Understanding **functions**: timeouts, arrow functions, anonymous, self-executing

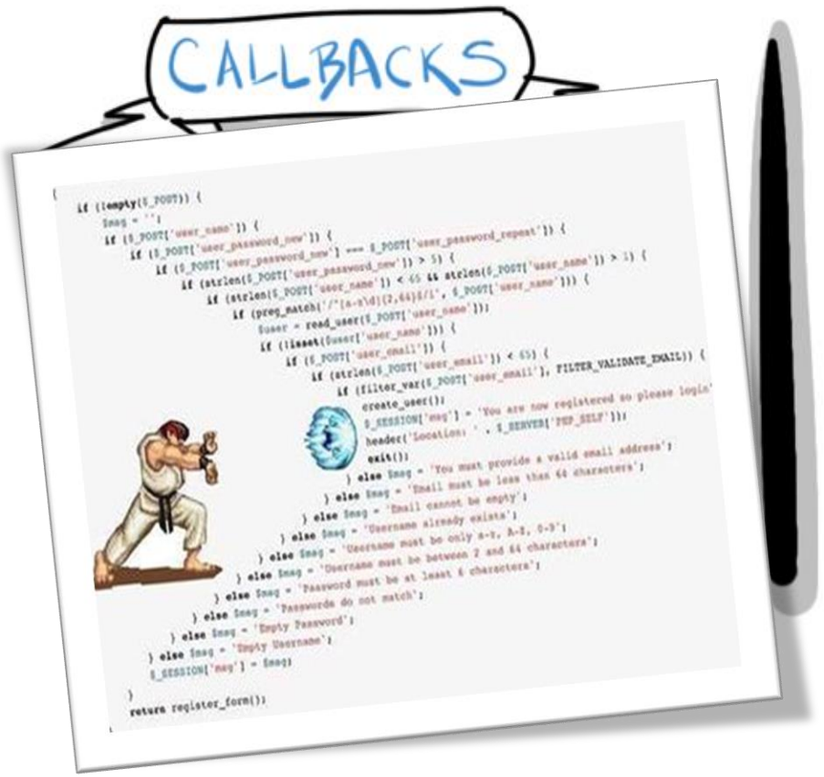
- Use a real-world **REST API** to fetch data programmatically with **fetch** and **axios**

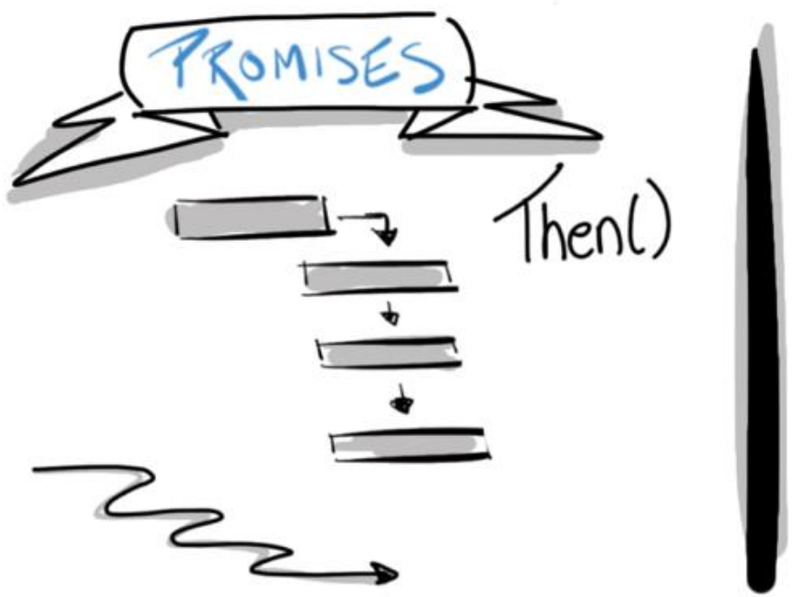
Asynchronous Code

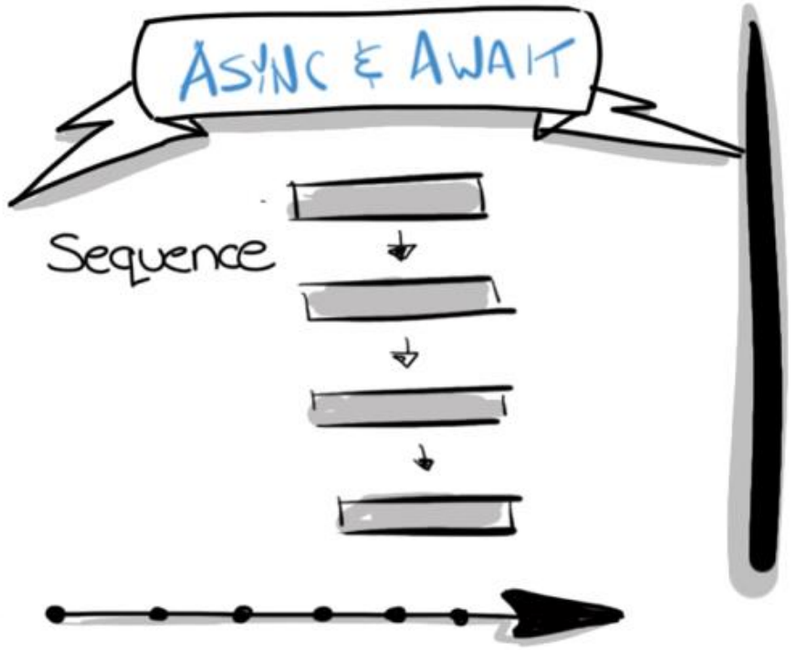


Picture source

Asynchronous Code







[Picture source]

Callbacks

Callbacks are functions that are passed as parameters to another function.

```
function contactServer(payload, callback) {

    // Do something.
    // Then call the callback.
}
```

Is callback execution necessarily asynchronous?

Callbacks

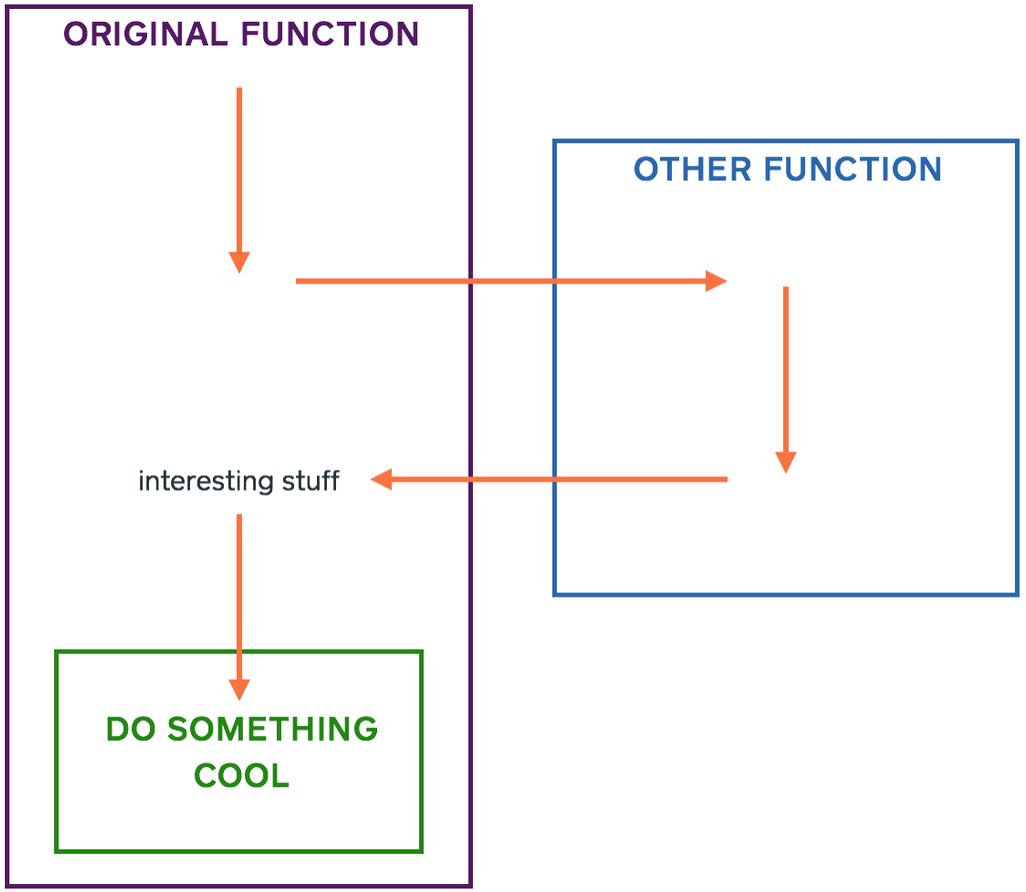
Callbacks are functions that are passed as parameters to another function.

```
function contactServer(payload, callback) {

    // Do something.
    // Then call the callback.
}
```

Is callback execution necessarily asynchronous?

NO, it can also be sequential

What is Asynchronous Code?

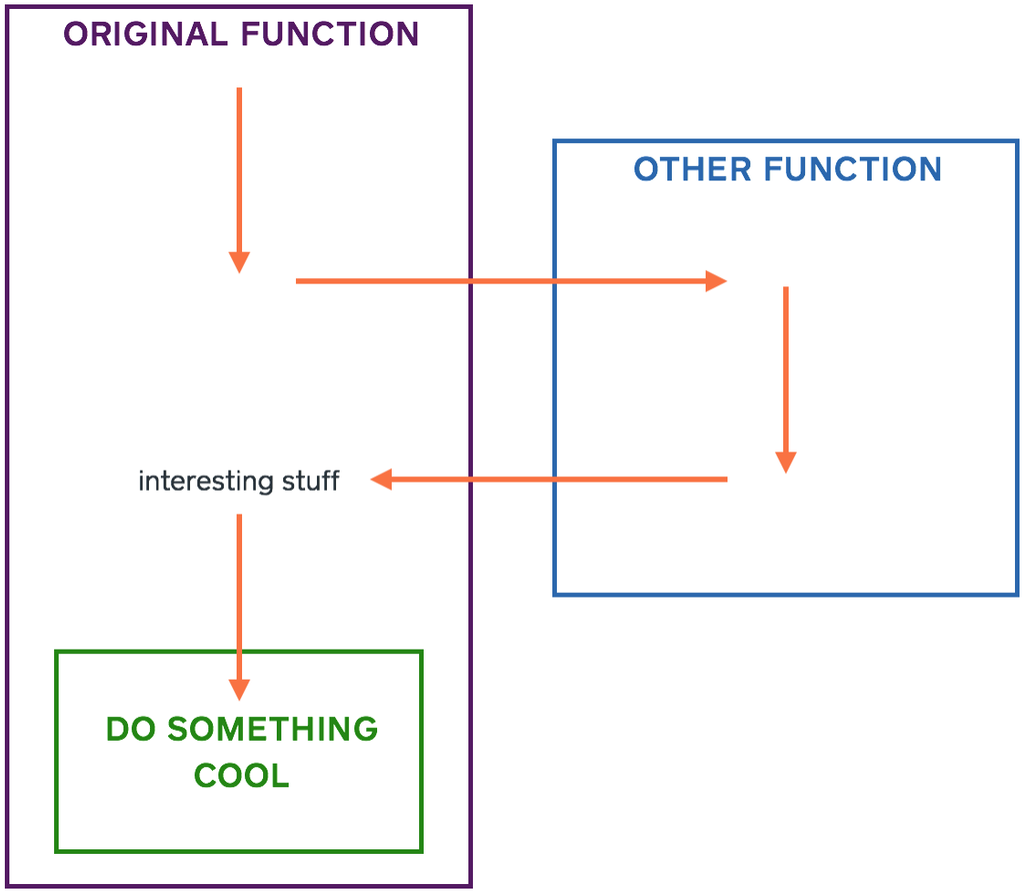


Img Source

Consider the execution flow on the left

Is this asynchronous?

What is Asynchronous Code?



Img Source

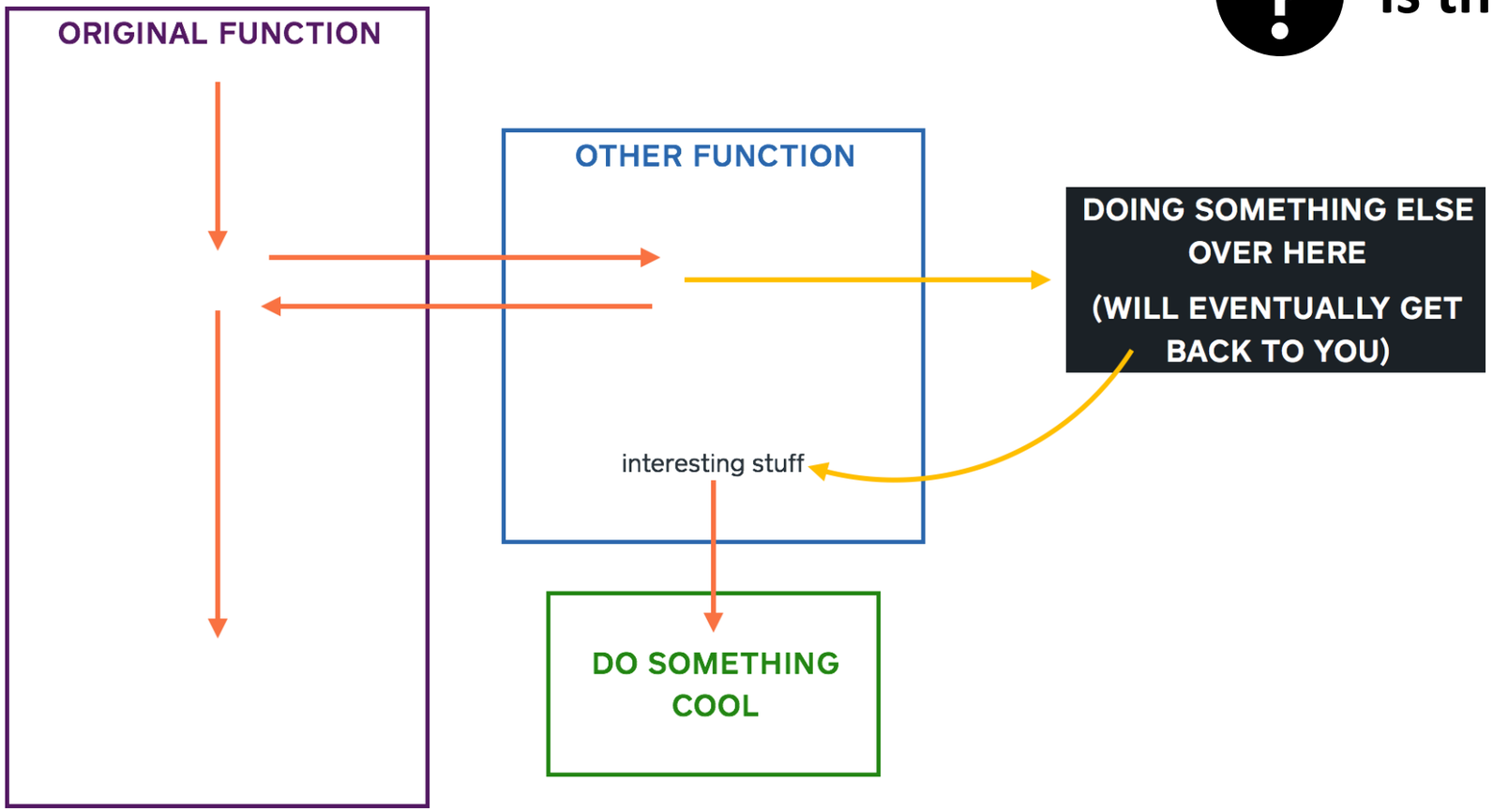
Consider the execution flow on the left

Is this asynchronous?

NO, it is sequential. The context of execution switches from one function to another, *which runs until the end*, then it returns to the original calling function.

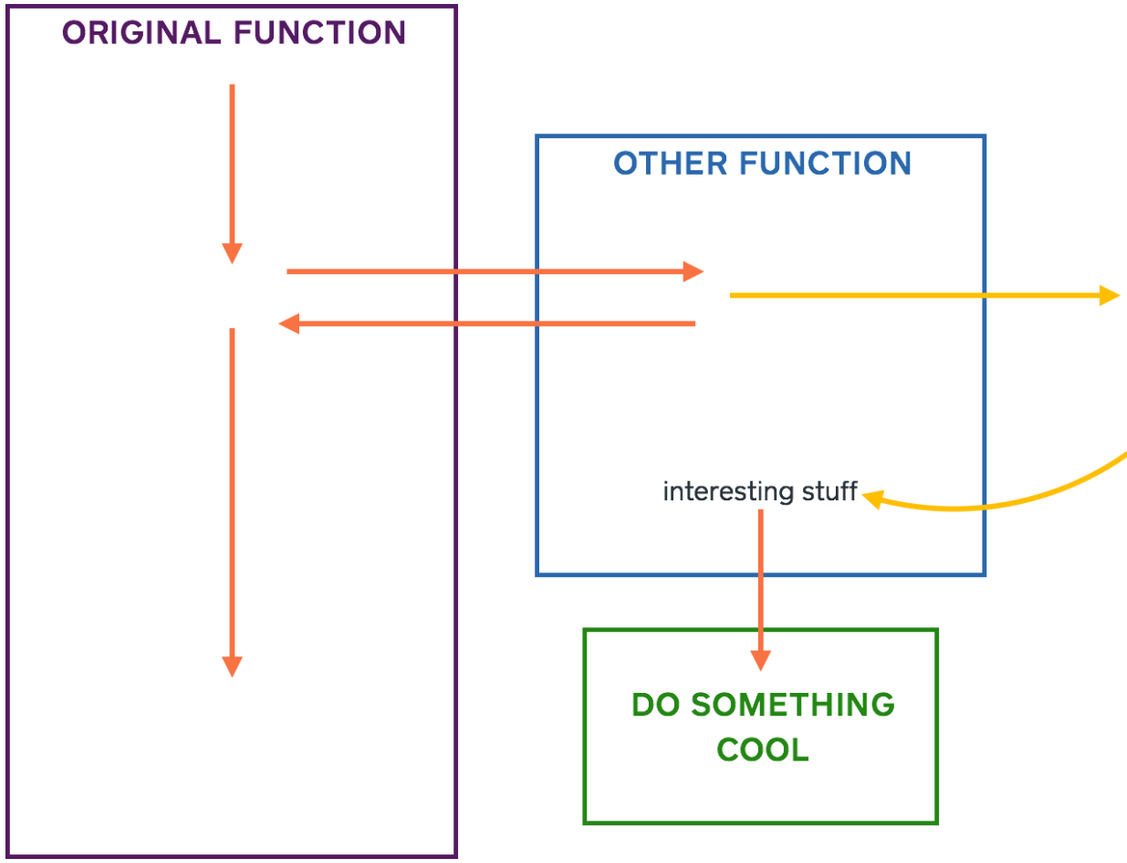
What is Asynchronous Code?

Is this asynchronous?



Img Source

What is Asynchronous Code?

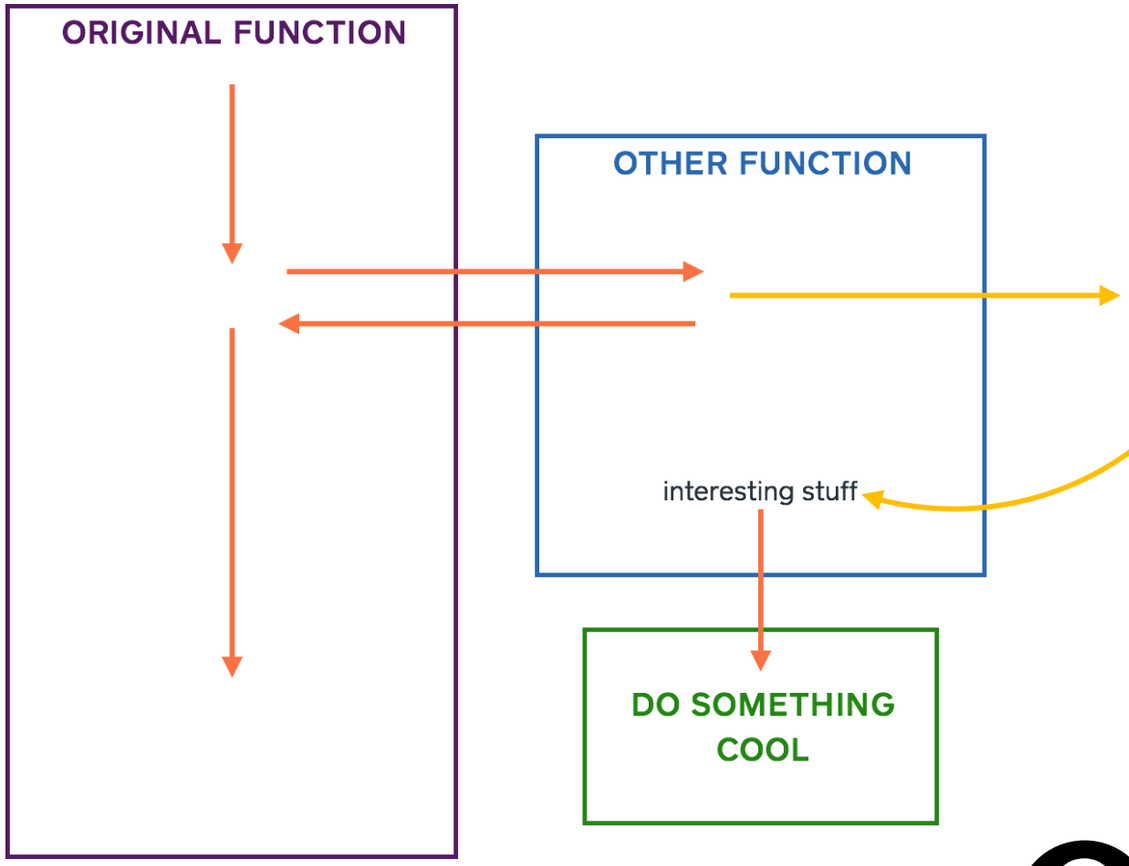


Is this asynchronous?

YES, it is! The context of execution switches from one function to another, it returns *immediately* to the original calling function, while the called function runs *in parallel*.

Img Source

What is Asynchronous Code?



Img Source

Is this asynchronous?

YES, it is! The context of execution switches from one function to another, it returns *immediately* to the original calling function, while the called function runs *in parallel*.

But how does the original function access the return value from the other function?

Callbacks

Callbacks are functions that are passed as parameters to another function.

```
function contactServer(payload, callback) {

    // Do something.
    // Then call the callback.
}
```

Callbacks

Callbacks are functions that are passed as parameters to another function.

```
function contactServer(payload, callback) {

    // Sequential (synchronous) execution.
    // if (DATA_IN_CACHE) {
        callback(DATA_IN_CACHE);
    }
}
```

Callbacks

```
function contactServer(payload, callback) {

    // Sequential (synchronous) execution.
    // if (DATA_IN_CACHE) {
        callback(DATA_IN_CACHE);
    }
    // Asynchronous execution.
    else {
        fetch(SERVER_ADDRESS, callback);
    }
}
```

Callbacks

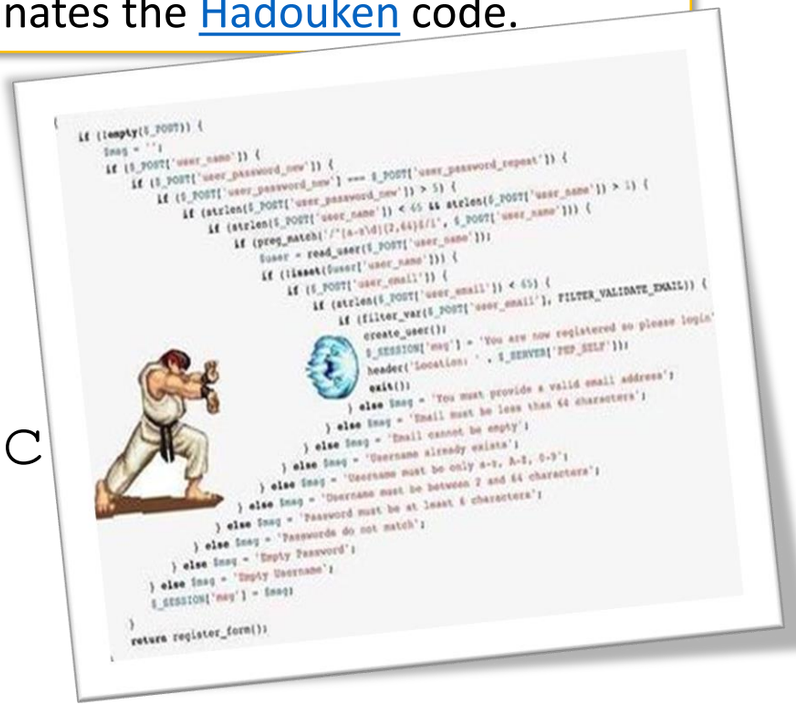
```
function contactServer(payload, callback) {

    // Sequential (synchronous) e
    // if (DATA_IN_CACHE) {
        callback(DATA_IN_CACHE);
    }
    // Asynchronous execution.
    else {
        fetch(SERVER_ADDRESS, callback);
    }
}
```

The callback function may executed synchronously or asynchronously within the same function.

Callbacks

```
function contactServer(payload, callback) {

    // Sequential (synchronous) e
    // if (DATA_IN_CACHE) {
        callback(DATA_IN_CACHE);
    }
    // Asynchronous execution.
    else {
        fetch(SERVER_ADDRESS, callbac
    }
}
```

The callback function may executed synchronously or asynchronously within the same function.
When multiple callback calls are nested, it originates the Hadouken code.



Simplest, Really Common Callback

```
setTimeout(function() {
    console.log('I am alive!);
}, 2000);
```

Simplest, Really Common Callback

```
setTimeout(function() {
    console.log('I am alive!);
}, 2000);
```

What is the name of this callback?

It has no name, it is an **anonymous** function

Simplest, Really Common Callback

```
setTimeout(function() {
    console.log('I am alive!);
}, 2000);
```

What is the name of this callback?

It has no name, it is an **anonymous** function

Could it have a name?

Yes, but what for? This function is used and thrown away.

Simplest, Really Common Callback

```
setTimeout(function() {
    console.log('I am alive!);
}, 2000);
```

What is the name of this callback?

It has no name, it is an **anonymous** function

Could it have a name?

Yes, but what for? This function is used and thrown away.

What is the advantage of being anonymous?

Does not pollute the *namespace.* Slightly better performance because of lower memory load. You type less.

Simplest, Really Common Callback

```
setTimeout(function() {
    console.log('I am alive!);
}, 2000);

// Arrow function equivalent.
```

Simplest, Really Common Callback

```
setTimeout(function() {
    console.log('I am alive!);
}, 2000);

// Arrow function equivalent.
setTimeout(() => console.log('I am alive'), 2000);
```

Most Common Callbacks: Event Listeners

Events are actions or occurrences happening in the system in which you are programming (e.g., Node.JS or Browser)

The system produces or "fires" a signal of some kind when an event occurs,

A mechanism is in place by which an event can be "caught" when the event occurs (that is, some code running).

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks/Events

Common Events in The Browser

The user **clicks** a certain element.
The user **hovers** the cursor over a certain element.
The user chooses a key on the keyboard.
The user resizes or closes the browser window.
A web page finishes **loading**.
A form is **submitted**.
A video is played, paused, or finishes.
An error occurs.

https://mdn.github.io/learning-area/javascript/building-blocks/events/random-color-eventhandlerattributes.html

How to Define Event Listeners

Get a reference to an **element** on the page (or the page itself)

Select the **event** you want to **listen** to

Add a **function** to be executed when the event is **fired**

Shake well.

How to Define Event Listeners

Get a reference to an **element** on the page (or the page itself)

Select the **event** you want to **listen** to

Add a **function** to be executed when the event is **fired**

Shake well.

As you might expect, JavaScript has multiple ways of registering event listeners (also called event **handlers**)

The Old Way (still valid)

```
const btn = document.querySelector('button');

btn.onclick = function() {

};
```

The Old Way (still valid)

```
const btn = document.querySelector('button');

btn.onclick = function() {
    const rndCol =
     `rgb(random(255), random(255), random(255))`;

    document.body.style.backgroundColor = rndCol;
};
```

https://www.w3schools.com/cssref/func_rgb.asp

The Old Way (still valid)

```
const btn = document.querySelector('button');

// Removing and event listener on onclick.
btn.onclick = null;
```

The Modern Way

```
const btn = document.querySelector('button');

const changeColor = function() {
  // Code as before.
};
// Add listener.
btn.addEventListener('click', changeColor);
```

Notice an important difference!
Before it was **"onclick"** here is simply **"click"**

The Modern Way

```
const btn = document.querySelector('button');

const changeColor = function() {
  // Code as before.
};
// Add listener.
btn.addEventListener('click', changeColor);
// Remove it.
btn.removeEventListener('click', changeColor);
```

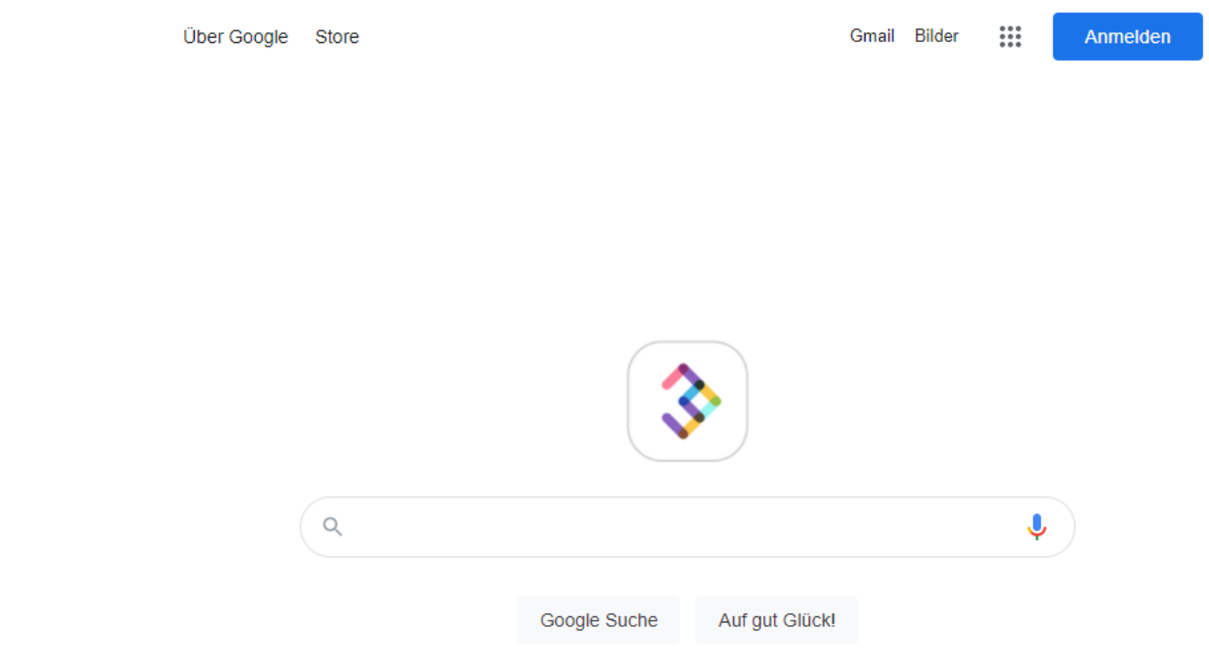
Comparison

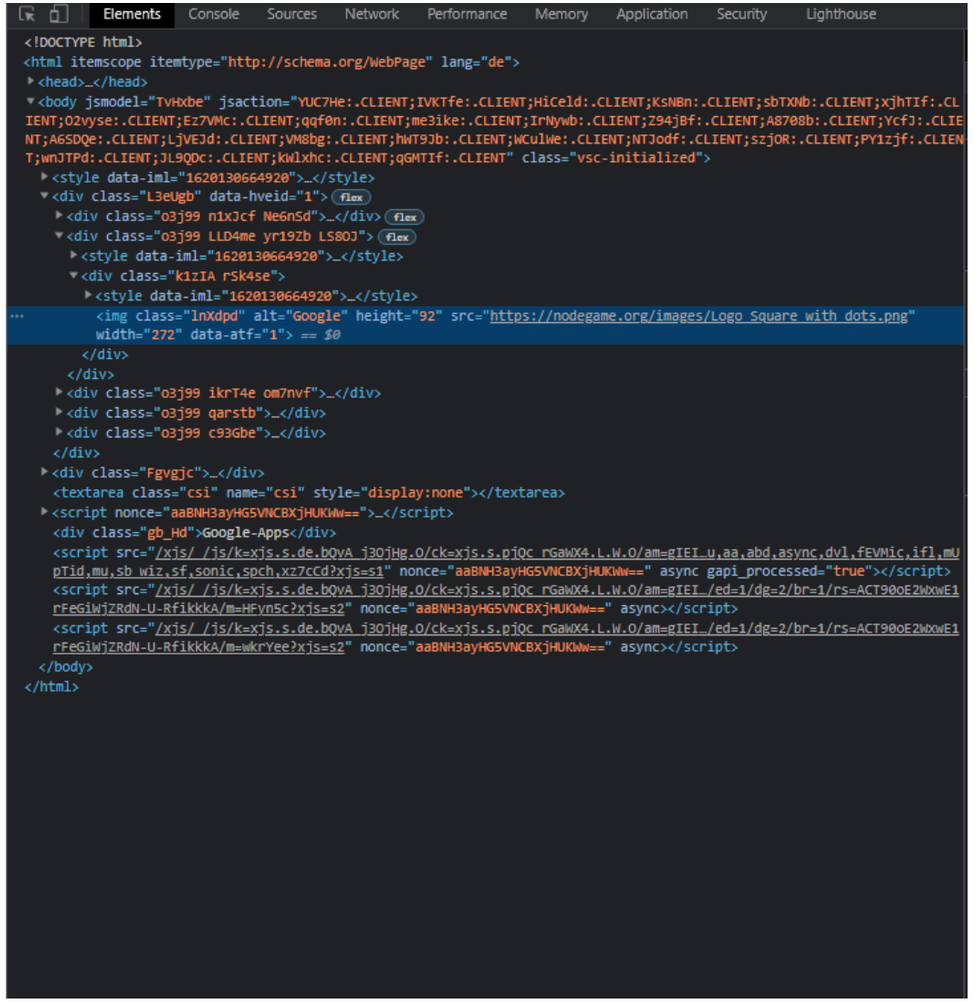
Event Handler Properties (e.g., onclick)	add|RemoveEventListener
Easier to use	More complicated to use
Less powerful	More powerful: give further options on the how/when catching the event
Only one event listener allowed	Multiple event listeners allowed

If you are using an external framework (e.g., Bootstrap, React, Angular, etc.), there are good chances to create conflicts with its own event listeners.

Hands On: Messing Around with Google.Com

Go to Google.com, open DevTools and change the logo to something else.





Hands On: Messing Around with Google.Com

Let's manipulate the page elements programmatically:

```
// Locate the HTML element with given id.
let logo = document.getElementById("logo");
```

If Google shows a special logo, you should check the DOM for the right ID/class. For instance, this command might be an alternative:

let logo = document.querySelector('.lnXdpd');

Check the id/class assigned to the logo image in the browser's Inspect tab!

Hands On: Messing Around with Google.Com

Changes in the Inspector are immediately reflected on the page.

For example, if add a rule:

"display: none"

the selected element will be hidden in the page.

Hands On: Messing Around with Google.Com

Go to Google.com and manipulate the page elements programmatically:

```
// Locate the HTML element holding with given id.
let logo = document.getElementById("logo");
```

How to change the image displayed?

Hands On: Messing Around with Google.Com

Go to Google.com and manipulate the page elements programmatically:

```
// Locate the HTML element holding with given id.
let logo = document.getElementById("logo");
```

How to change the image displayed?

DOM objects are glorified JavaScript objects with properties and methods.
The browser reads those properties and displays them accordingly.

Hands On: Messing Around with Google.Com

Go to Google.com and manipulate the page elements programmatically:

```
// Locate the HTML element holding with given id.
let logo = document.getElementById("logo");
// Change one of its attributes (pick any image you like).
logo.srcset = "https://nodegame.org/images/Logo_Square_with_dots.png";
```

Google does thousands of A/B testing, so the exact name of the property might be slightly different from mine. If not working, try setting `srcset` to null, and set the property `src`.

Hands On: Messing Around with Google.Com

Go to Google.com and manipulate the page elements programmatically:

```
// Locate the HTML element holding with given id.
let logo = document.getElementById("logo");
// Change one of its attributes (pick any image you like).
logo.srcset = "https://nodegame.org/images/Logo_Square_with_dots.png";
// Defines an onclick event-handler (anonymous function).
logo.onclick = function() {

                    Let's do Something Here!

};
```

Hands On: Messing Around with Google.Com

Go to Google.com and manipulate the page elements programmatically:

```
// Locate the HTML element holding with given id.
let logo = document.getElementById("logo");
// Change one of its attributes (pick any image you like).
logo.srcset = "https://nodegame.org/images/Logo_Square_with_dots.png";
// Defines an onclick event-handler (anonymous function).
logo.onclick = function() {
    // Redirect to a new page using the location object.
    window.location.href = "https://nodegame.org";
};
```

List of Events

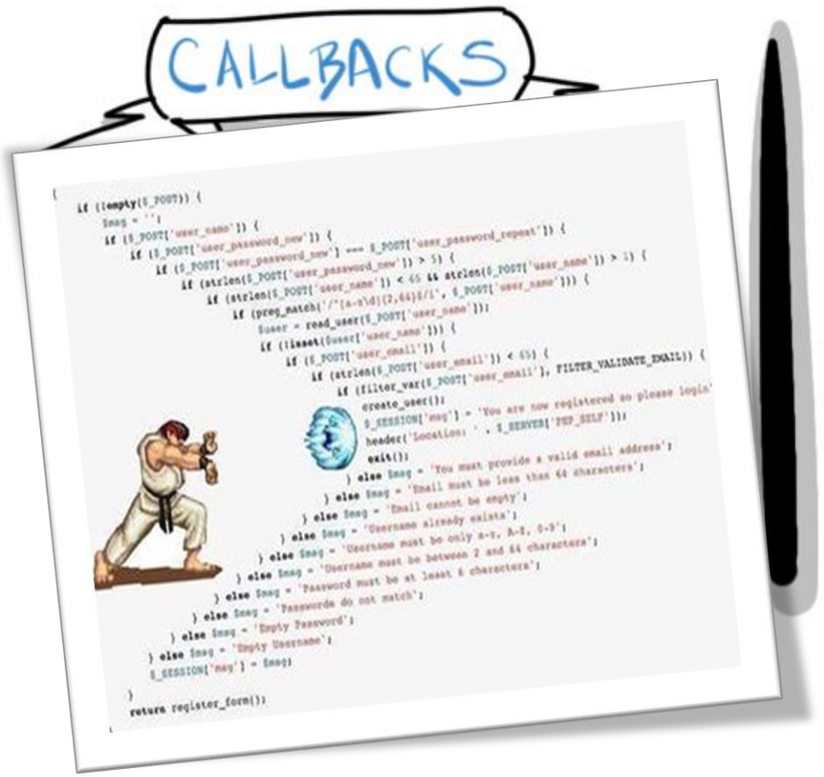
Most common events, and examples.

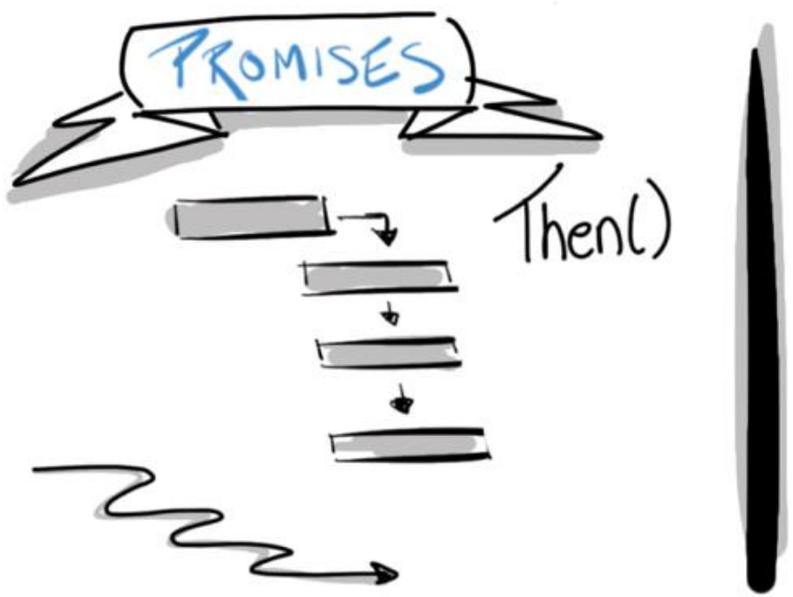
https://javascript.info/introduction-browser-events

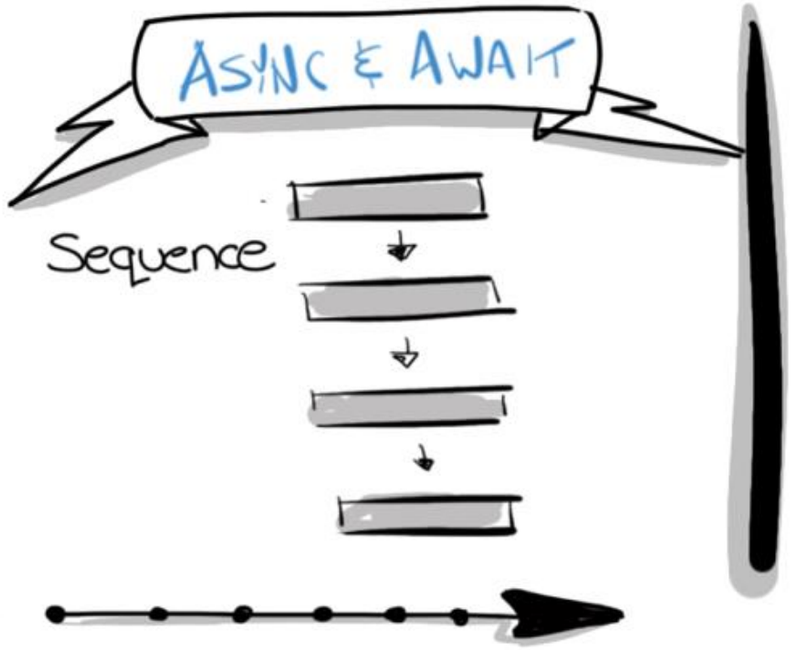
All events, by browser. No longer updated ☹

https://perimeterx.github.io/map-events-website/

Asynchronous Code







[Picture source]

Promises

Promises are a new **paradigm** to execute callbacks (**ES6**)

A response to the need of a more principle way to run multiple callbacks

Promises

Promises are a new **paradigm** to execute callbacks (**ES6**)

A response to the need of a more principle way to run multiple callbacks

Promises shift some of the complexity at Promise creation to simplify its execution.

Promises

You need to create a Promise with the **new** operator before using it.

The **new** operator is used in Object Oriented Programming (OOP) to create a new instance of a class, that is an object.

```
let promise = new Promise(…);
```

Some OOP Terminology

```
let promise = new Promise(…);

// promise is an instance of the Promise (capital P) class.

// As an instance of big Promise, little promise
// inherits some properties and method from its parent.

// Promise is the constructor method instantiating the
// objects of class Promise.
```

Back to Promises

Promises are really simple.

- The constructor takes one **callback** function

```
let promise = new Promise(function(…) {…});
```

Back to Promises

Promises are really simple.

- The constructor takes one **callback** function
- This callback function takes **two input parameters**

```
let promise = new Promise(function (a, b) {…});
```

Back to Promises

Promises are really simple.

- The constructor takes one **callback** function
- This callback function takes **two input parameters**
- These input parameters are also callback functions, usually called **resolve and reject**

```
let promise = new Promise(function (resolve, reject) {…});
```

Back to Promises

Promises are really simple.

- The constructor takes one **callback** function
- This callback function takes **two input parameters**
- These input parameters are also callback functions, usually called **resolve and reject**

```
let promise = new Promise(function (resolve, reject) {…});
```

Your task as a developer, is to write the logic executing the resolve callback on *success*, and the reject callback on *failure*.

A Promise is Created

After a promised is created you can call it elegantly

```
let promise = new Promise(function (resolve, reject) {…});

promise
    .then(() => console.log('I am a success'));
    .catch(() => console.log('I am a failure'));
```

A Promise is Created

After a promised is created you can call it elegantly

```
let promise = new Promise(function (resolve, reject) {…});

promise
    .then(() => console.log('I am a success'));
    .catch(() => console.log('I am a failure'));
```

Is promise execution necessarily asynchronous?

A Promise is Created

After a promised is created you can call it elegantly

```
let promise = new Promise(function (resolve, reject) {…});

promise
    .then(() => console.log('I am a success'));
    .catch(() => console.log('I am a failure'));
```

Is promise execution necessarily asynchronous?

NO, promises are masked callbacks, hence they can also be sequential.

Let's Promisify Our Previous Callback

```
function contactServer(payload, callback) {

    // Sequential (synchronous) execution.
    // if (DATA_IN_CACHE) {
        callback(DATA_IN_CACHE);
    }
    // Asynchronous execution.
    else {
        fetch(SERVER_ADDRESS, callback);
    }
}
```

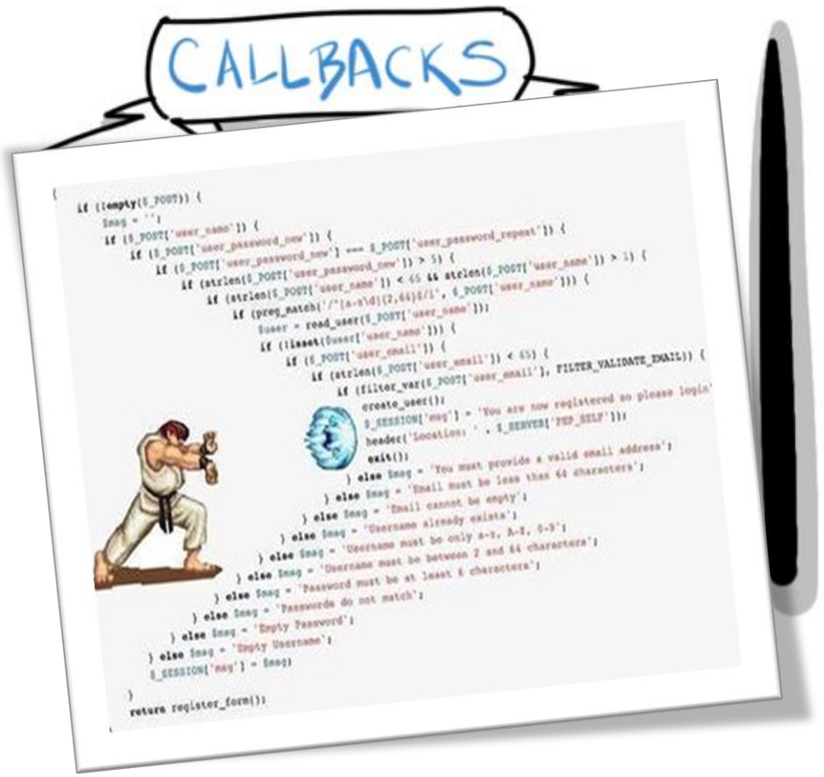
Let's Promisify Our Previous Callback

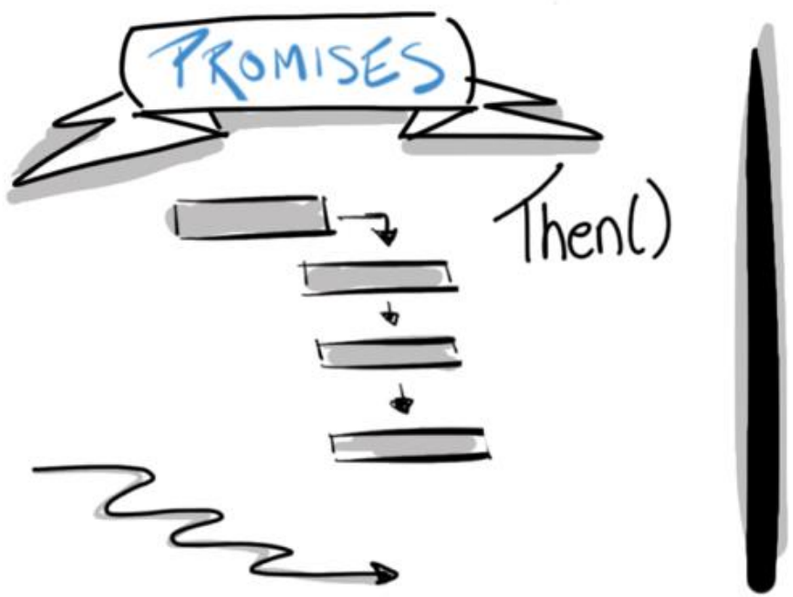
```
let promise = new Promise(function (resolve, reject) {



});
```

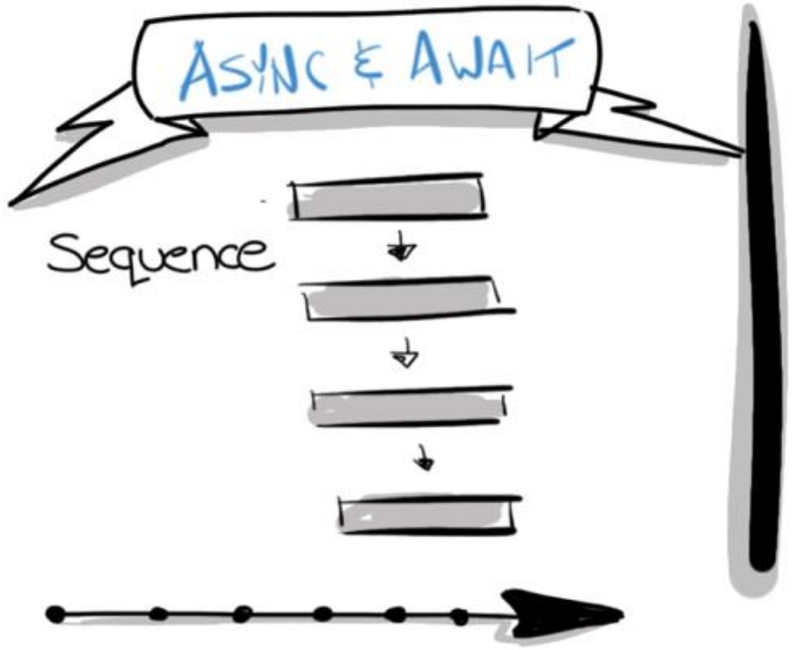
Let's Promisify Our Previous Callback

```
let promise = new Promise(function (resolve, reject) {
    // if (DATA_IN_CACHE) {
        resolve(DATA_IN_CACHE);
    }
    //
    else {
        fetch(SERVER_ADDRESS, function(res) {
            if (res.error) reject(res);
            else resolve(res);
        });
    }
});
```

Asynchronous Code







[Picture source]

Async/Await

The await/async pattern is so called "**sugar coating**" over the Promise syntax

It means that it makes writing code involving promises easier and faster.

You don't even realize you are writing a Promise!

Async/Await

The await/async pattern is so called "**sugar coating**" over the Promise syntax

It means that it makes writing code involving promises easier and faster.

You don't even realize you are writing a Promise!

ASYNC

```
async function hello() {
   return "Hello"
};
```

Async/Await

The await/async pattern is so called "**sugar coating**" over the Promise syntax

It means that it makes writing code involving promises easier and faster.

You don't even realize you are writing a Promise!

ASYNC

```
async function hello() {
   return "Hello"
};

hello().then(res => {
   console.log(res)
});
```

Async/Await

The await/async pattern is so called "**sugar coating**" over the Promise syntax

It means that it makes writing code involving promises easier and faster.

You don't even realize you are writing a Promise!

ASYNC

```
async function hello() {
   return "Hello"
};

hello().then(res => {
   console.log(res)
});
```

AWAIT

```
let result = await promise;
```

Async/Await

The await/async pattern is so called "**sugar coating**" over the Promise syntax

It means that it makes writing code involving promises easier and faster.

You don't even realize you are writing a Promise!

ASYNC

```
async function hello() {
    return "Hello"
};

hello().then(res => {
    console.log(res)
});
```

AWAIT

```
let result = await promise;
```

CAVEATS:

- You need to have a promise to begin with
- Works only inside an **async** function.

```
// We must be inside an async function
let hello = async () => {

  // We must have a promise.
  let promise = new Promise((resolve, reject) => {
    setTimeout(() => resolve("Hello"), 1000);
  });

  // We can finally use await.
  let word = await promise;
  console.log(word);
};

hello();
```

Having to create:
- a wrapper function, and
- a promise

in order to use await is a bit cumbersome

However, if you have a method that already returns a promise, it's much easier.

Star Wars API

Let's use **fetch** or **axios** method in the browser to connect to this API and fetch Star Wars characters!

Fetch	Axios (https://axios-http.com)
POST and GET requests	**POST and GET requests**
Native in modern browsers	External library
Not supported in older browsers https://caniuse.com/?search=fetch	**Supported in older browsers**
Available in Node.js via npm module node-fetch (slight differences exist)	**Available in Node.js via npm module axios**
Must explicitly convert response to JSON	**Automatic JSON conversion**
Rejects only if request does not complete	**Rejects also with error responses (e.g., 404)**
No interceptors	**Interceptors to modify HTTP headers**

https://www.pluralsight.com/guides/axios-vs-fetch

https://www.blog.duomly.com/fetch-vs-axios-what-is-better-in-2020/

Fetch	Axios (https://axios-http.com)
POST and GET requests	**POST and GET requests**
Native in modern browsers	External library
Not supported in older browsers https://caniuse.com/?search=fetch	**Supported in older browsers**
Available in Node.js via npm module node-fetch (slight differences exist)	**Available in Node.js via npm module axios**
Must explicitly convert response to JSON	**Automatic JSON conversion**
Rejects only if request does not complete	**Rejects also with error responses (e.g., 404)**
No interceptors	**Interceptors to modify HTTP headers**

Quick prototype, smaller projects

Small and large projects

https://www.pluralsight.com/guides/axios-vs-fetch

https://www.blog.duomly.com/fetch-vs-axios-what-is-better-in-2020/

```
(async() => {

// Our code in here.

})();
```

What is this weird construct?

```
(async() => {

// Our code in here.

})();
```

What is this weird construct?

It is a **self-executing**, **anonymous** function.

```
(async() => {

// Our code in here.

})();
```

What is this weird construct?

It is a **self-executing**, **anonymous** function.

It is called a **closure** because a new **variables scope** separated from the main one.

The difference here is that this is an **async** space.

```
(async() => {
  // SWAPI details.
  const ENDPOINT = "https://swapi.dev/api/";
  let query = 'people/1';

  // Asynchronous fetch call.
})();
```

```
(async() => {
  // SWAPI details.
  const ENDPOINT = "https://swapi.dev/api/";
  let query = 'people/1';

  // Asynchronous fetch call.
  const res = await fetch(ENDPOINT + query);
  console.log(res);

})();
```

```
(async() => {
  // SWAPI details.
  const ENDPOINT = "https://swapi.dev/api/";
  let query = 'people/1';

  // Asynchronous fetch call.
  const res = await fetch(ENDPOINT + query);
  console.log(res);

})();
```

Doesn't look good?

The response object from fetch is **NOT** the JSON response body.

It is the entire **HTTP** response.

We need to extract the JSON body with the **asynchronous .json() method**, which returns a **promise**.

```
(async() => {
  // SWAPI details.
  const ENDPOINT = "https://swapi.dev/api/";
  let query = 'people/1';

  // Asynchronous fetch call.
  const res = await fetch(ENDPOINT + query);
  console.log(res);

  // Asynchronous parsing into JSON.
  const user = await res.json();
  console.log('We got ASYNC/AWAIT: ', user.name);

})();
```

```
(async() => {
  // SWAPI details.
  const ENDPOINT = "https://swapi.dev/api/";
  let query = 'people/1';

  // Asynchronous fetch call.
  const res = await fetch(ENDPOINT + query);
  console.log(res);

  // Asynchronous parsing into JSON.
  const user = await res.json();
  console.log('We got ASYNC/AWAIT: ', user.name);

})();
```

Nice, but what if an error occurs while fetching?
How to handle errors with async/await?

```
(async() => {
  // SWAPI details.
  const ENDPOINT = "https://swapi.dev/api/";
  let query = 'people/1';

  // Asynchronous fetch call.
  const res = await fetch(ENDPOINT + query);
  console.log(res);

  // Asynchronous parsing into JSON.
  const user = await res.json();
  console.log('We got ASYNC/AWAIT: ', user.name);

})();
```

Nice, but what if an error occurs while fetching?
How to handle errors with async/await? **Use Try-and-Catch blocks.**

With **axios**

```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>


(async() => {
  // SWAPI details.
  const ENDPOINT = "https://swapi.dev/api/";
  let query = 'people/1';

  // Asynchronous fetch call.
  const json = await axios(ENDPOINT + query);

  console.log('We got ASYNC/AWAIT: ', json.data.name);

})();
```

Library needs to be imported.

JSON automatically parsed

data under json.data

Test APIs

- https://github.com/public-apis/public-apis

Cryptocurrency

API	Description	Auth	HTTPS	CORS
Binance	Exchange for Trading Cryptocurrencies based in China	apiKey	Yes	Unknown
BitcoinAverage	Digital Asset Price Data for the blockchain industry	apiKey	Yes	Unknown
BitcoinCharts	Financial and Technical Data related to the Bitcoin Network	No	Yes	Unknown
Bitfinex	Cryptocurrency Trading Platform	apiKey	Yes	Unknown
Bitmex	Real-Time Cryptocurrency derivatives trading platform based in Hong Kong	apiKey	Yes	Unknown
Bittrex	Next Generation Crypto Trading Platform	apiKey	Yes	Unknown
Block	Bitcoin Payment, Wallet & Transaction Data	apiKey	Yes	Unknown
Blockchain	Bitcoin Payment, Wallet & Transaction Data	No	Yes	Unknown
BlockFacts	Real-time crypto data from multiple exchanges via a single unified API, and much more	apiKey	Yes	Unknown

References

- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous
- https://javascript.info/async
- https://javascript.info/introduction-browser-events
- https://perimeterx.github.io/map-events-website/
-
- https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch
- https://swapi.dev

Module 4: jQuery

Image source

Image source

Module 4: jQuery

Learning Goals

- Learn how to import and use jQuery
- Select elements
- Perform simple animations

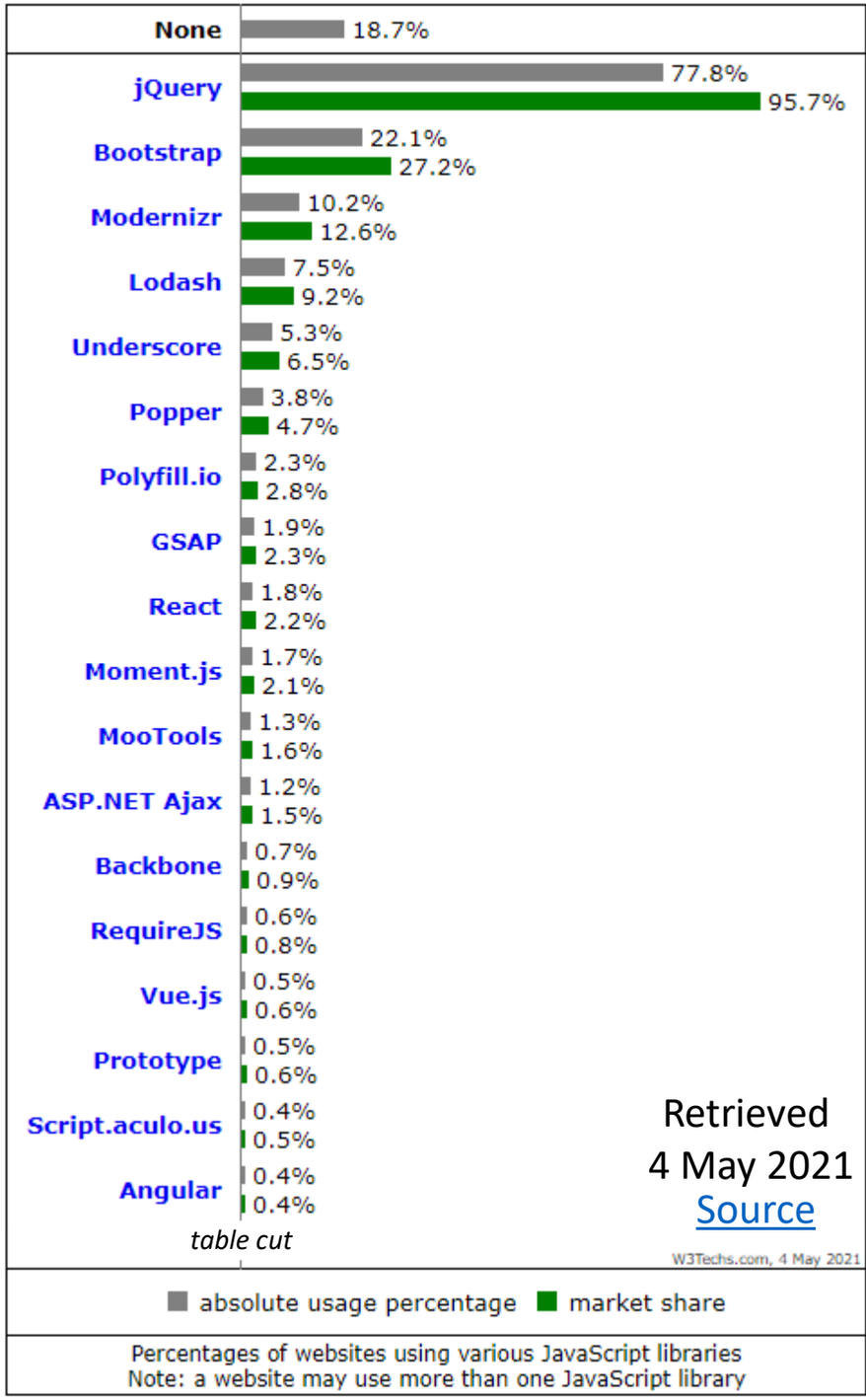
Module 4: jQuery

- **Free** and **open source** JS library to simplify:
  - DOM traversal and manipulation,
  - event handling,
  - CSS animation,
  - Ajax requests.

Module 4: jQuery

- **Free** and **open source** JS library to simplify:
  - DOM traversal and manipulation,
  - event handling,
  - CSS animation,
  - Ajax requests.

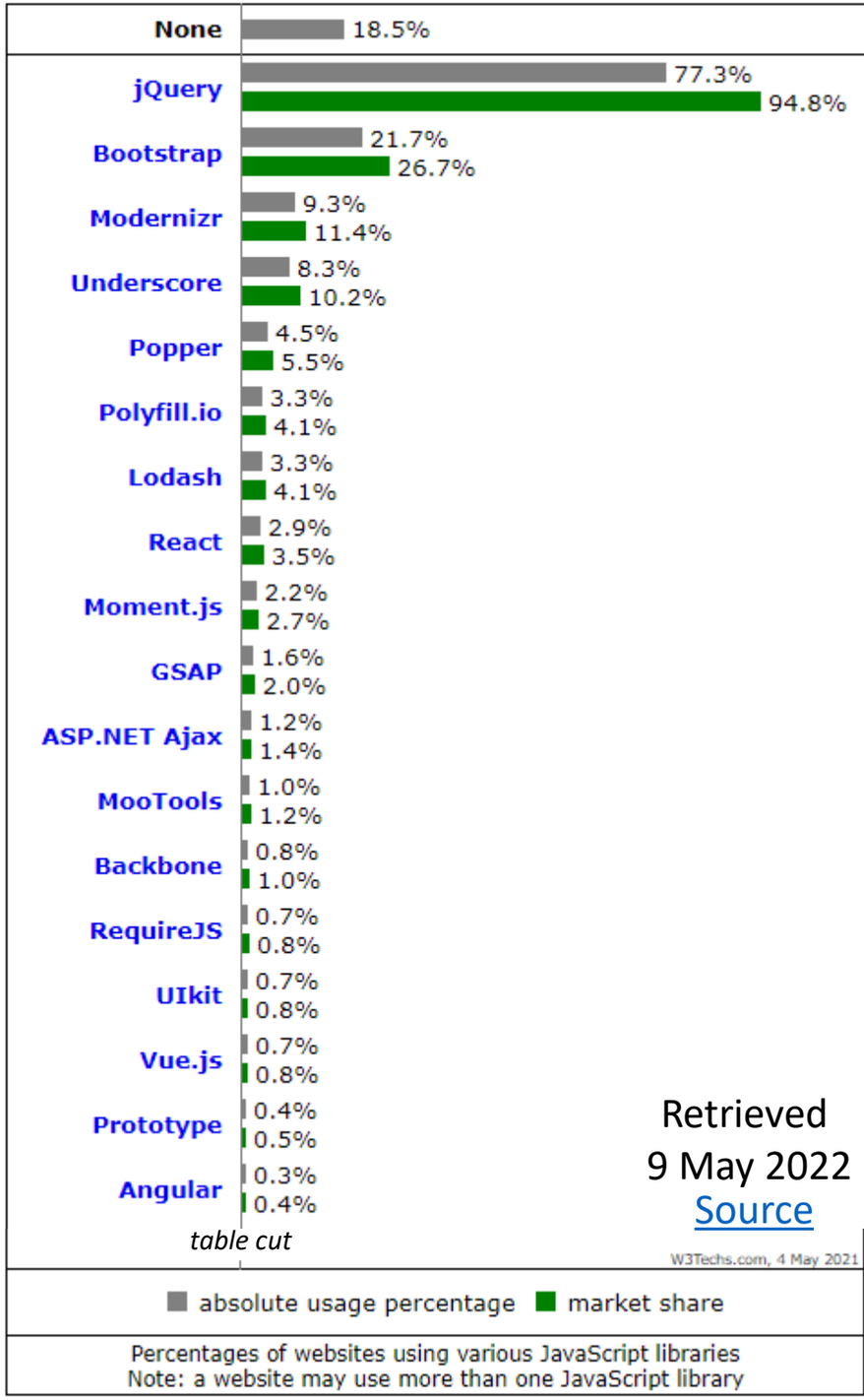
- Most widely deployed JS library, 3 to 4 times more usage than any other JS library



Module 4: jQuery

- **Free** and **open source** JS library to simplify:
  - DOM traversal and manipulation,
  - event handling,
  - CSS animation,
  - Ajax requests.

- Most widely deployed JS library, 3 to 4 times more usage than any other JS library



Module 4: jQuery

- Easy to use
- Easy to embed

Module 4: jQuery

- Easy to use
- Easy to embed

But,

- relatively slow
- Not a framework to build large, complex apps, such as Vue, Angular, or React

jQuery Basics

- `jQuery` or simply `$` object available in the browser after loading library
- The basic idea is to have a very simple syntax:
- `$("SELECTOR").method(…);`
- Methods can be chained
- `$("SELECTOR").method1(…).method2(…);`

Selectors (Refresh)

#ID Selects the element with id "ID"

.class: Select the element/s with class "class"

button Selects the element with tag <button>

What is this code doing?

```
$(document).ready(function(){

   $("p").click(function(){
      $(this).hide();
   });

});
```

Solution: https://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_hide

What is this code doing?

We now would like to make **the disappearing a little less abrupt**…how can we do it?

Let 's check the jQuery API: https://api.jquery.com/

Then let's change the code below:

```
$(document).ready(function(){
  $("p").click(function(){
    $(this).hide();
  });
});
```

Can we do the same with vanilla JS/CSS?

We can just define a CSS class and apply it to the desired element

```
.fadeout {
    opacity: 0 !important;
    transition: opacity 0.5s;
}
```

```
$(document).ready(function(){
   $("p").click(function(){
      $(this).addClass('fadeout');
   });
});
```

Is this enough?
The element is still "displayed", just with zero opacity

Module 4: References

- https://jquery.com/
- https://github.com/jquery/jquery
- https://www.w3schools.com/jquery/
- https://polyfill.io/