

Week 4.1

What are we covering this week?

4.1 - DOM, Dynamic frontends, Connecting FE to BE

4.2 - Chrome developer tools, Why frontend frameworks

What have we covered until now

0 to 1

Foundation

Foundation Javascript, async nature of JS
Node.js and its runtime

Databases (NoSQL/SQL)
Mongo and Postgres deep dive

Typescript beginner to advance

Backend

Backend communication protocols
Express basic to advance

ORMs

Middlewares, routes, status codes, global catches
Zod

MonoRepos, turborepo

Serverless Backends

OpenAPI Spec

Autogenerated clients

Authentication using external libraries

Scaling Node.js, performance benchmarks

Deploying npm packages

Frontend

Reconcilers and Frontend frameworks

React beginner to advance

Internals of state, Context API

State management using recoil

CSS you need to know of, Flexbox, basic styling

Frontend UI frameworks, Deep dive into Tailwind

Containerization, Docker

Next.js

Custom hooks

In house auth using next auth

Basic Devops

Docker end to end

Deploying to AWS servers

Newer clouds like fly/Remix

Nginx and reverse proxies

Projects

GSoC Project setting up and issue solving

Building Paytm/Wallet End to End

Now we'll move on to the frontend, catching up with backend from time to time

0 to 1

Foundation

Foundation Javascript, async nature of JS
Node.js and its runtime
Databases (NoSQL/SQL)
Mongo and Postgres deep dive
Typescript beginner to advance

Backend

Backend communication protocols
Express basic to advance
ORMs
Middlewares, routes, status codes, global catches
Zod
MonoRepos, turborepo
Serverless Backends
OpenAPI Spec
Autogenerated clients
Authentication using external libraries
Scaling Node.js, performance benchmarks
Deploying npm packages

Frontend

Reconcilers and Frontend frameworks

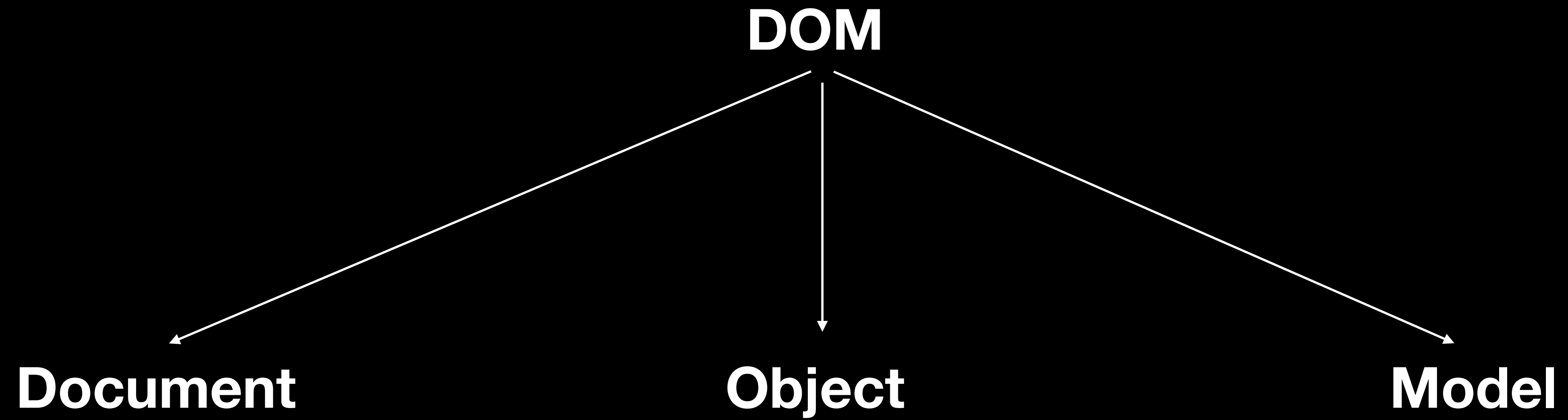
React beginner to advance
Internals of state, Context API
State management using recoil
CSS you need to know of, Flexbox, basic styling
Frontend UI frameworks, Deep dive into Tailwind
Containerization, Docker
Next.js
Custom hooks
In house auth using next auth

Basic Devops

Docker end to end
Deploying to AWS servers
Newer clouds like fly/Remix
Nginx and reverse proxies

Projects

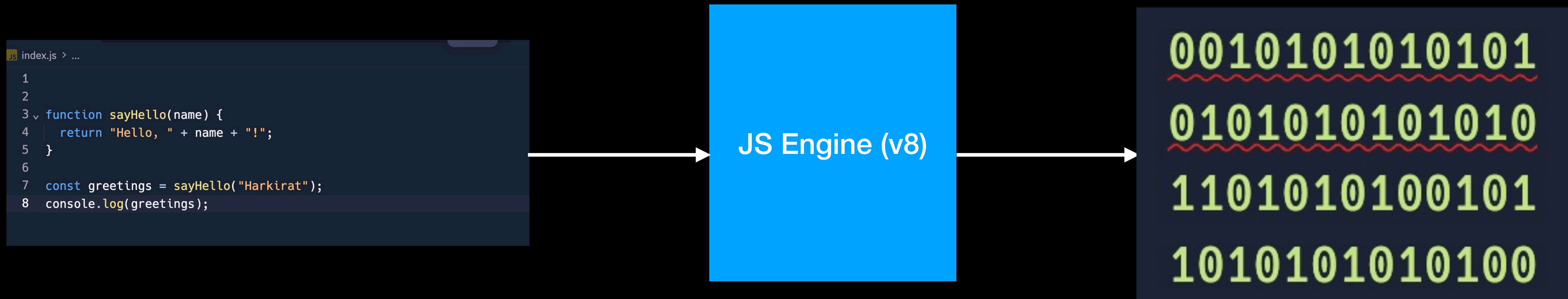
GSoC Project setting up and issue solving
Building Paytm/Wallet End to End



The DOM (Document Object Model) API is a programming interface for web documents. It represents the page so that programs can change the document structure, style, and content. The DOM represents the document as a tree of objects; each object represents a part of the page.

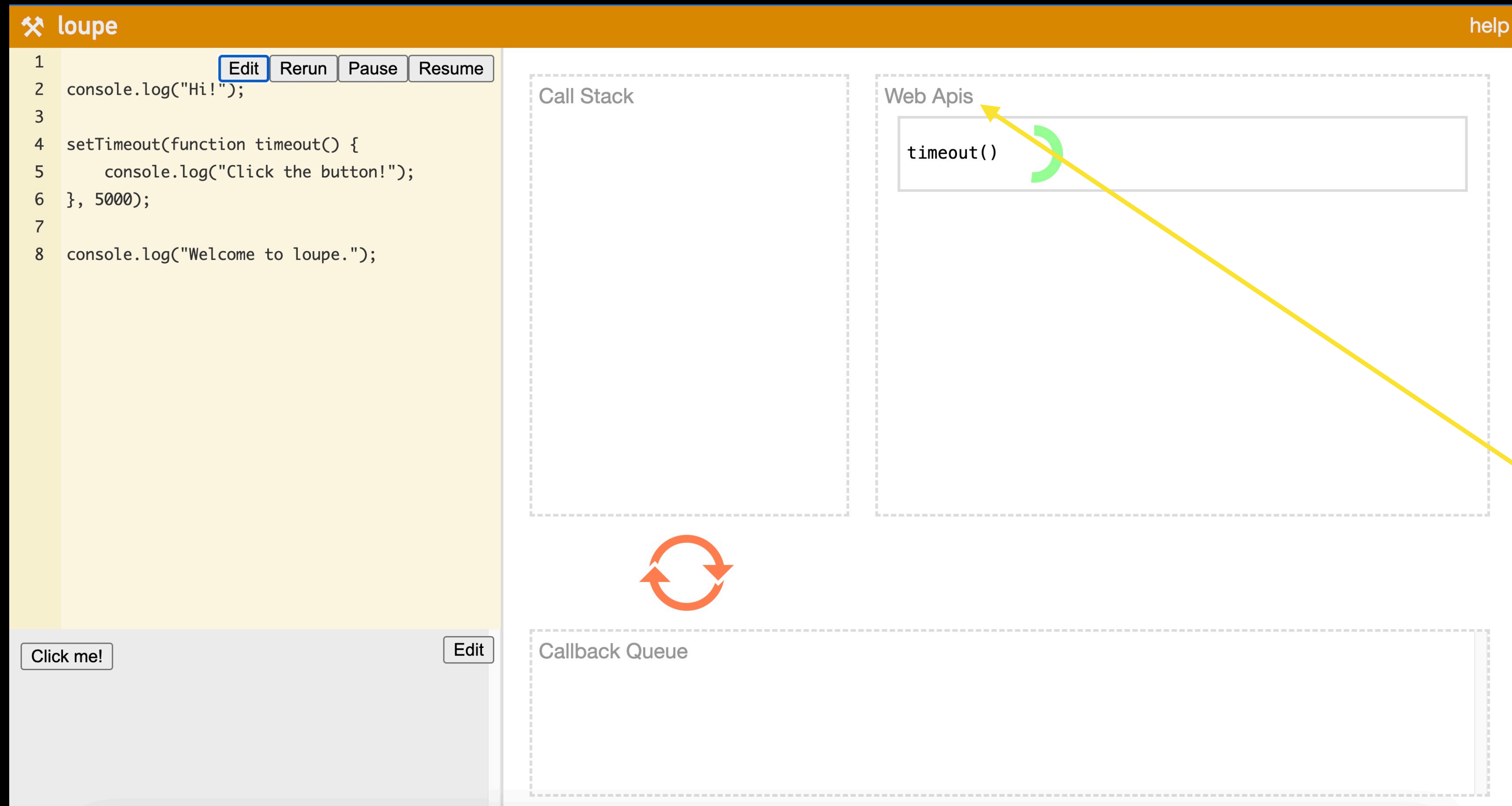
What was Javascript?

It was an implementation of the ECMAScript spec



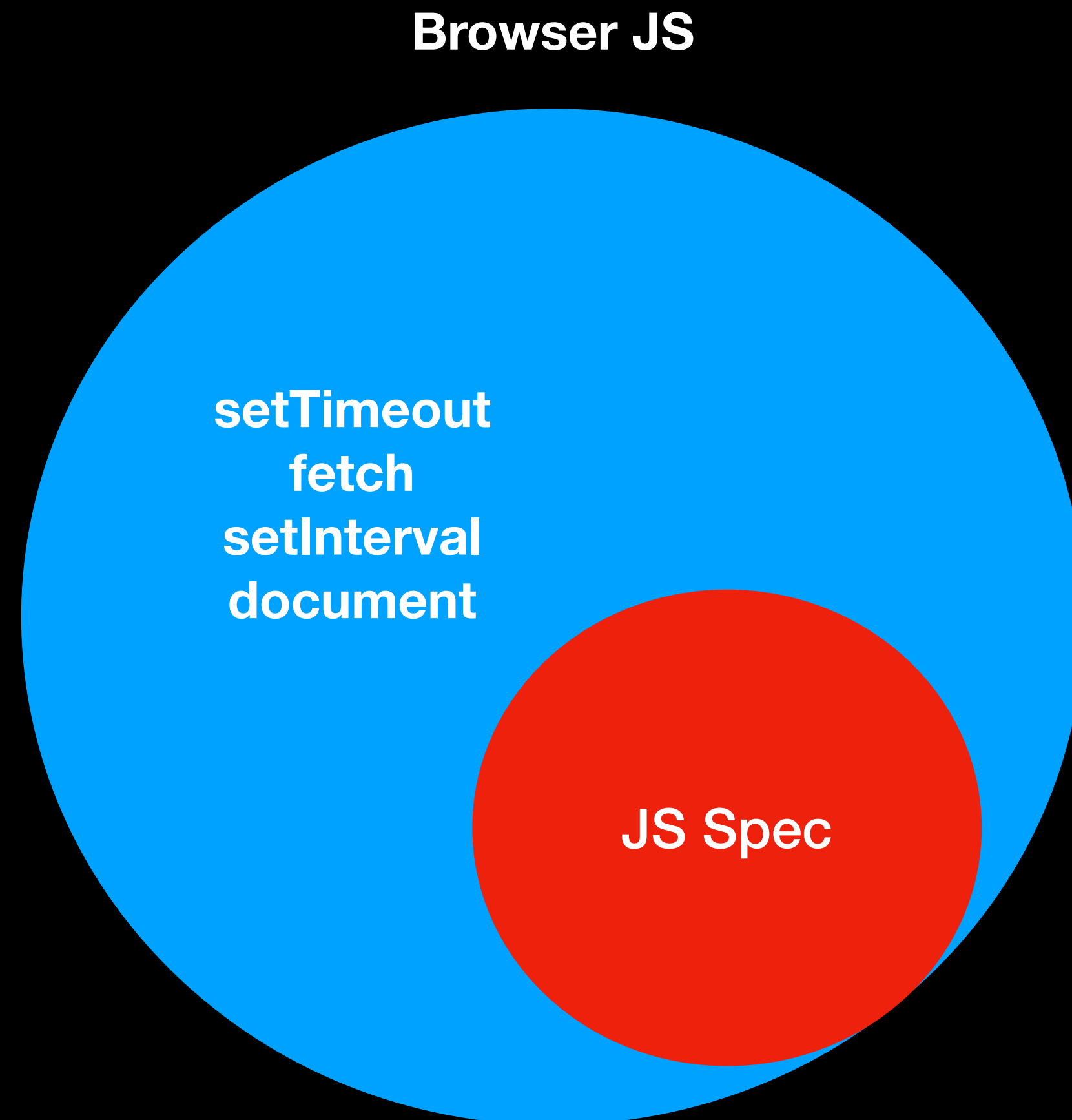
What was Javascript?

But the Javascript that runs in your browser has some extra functionality



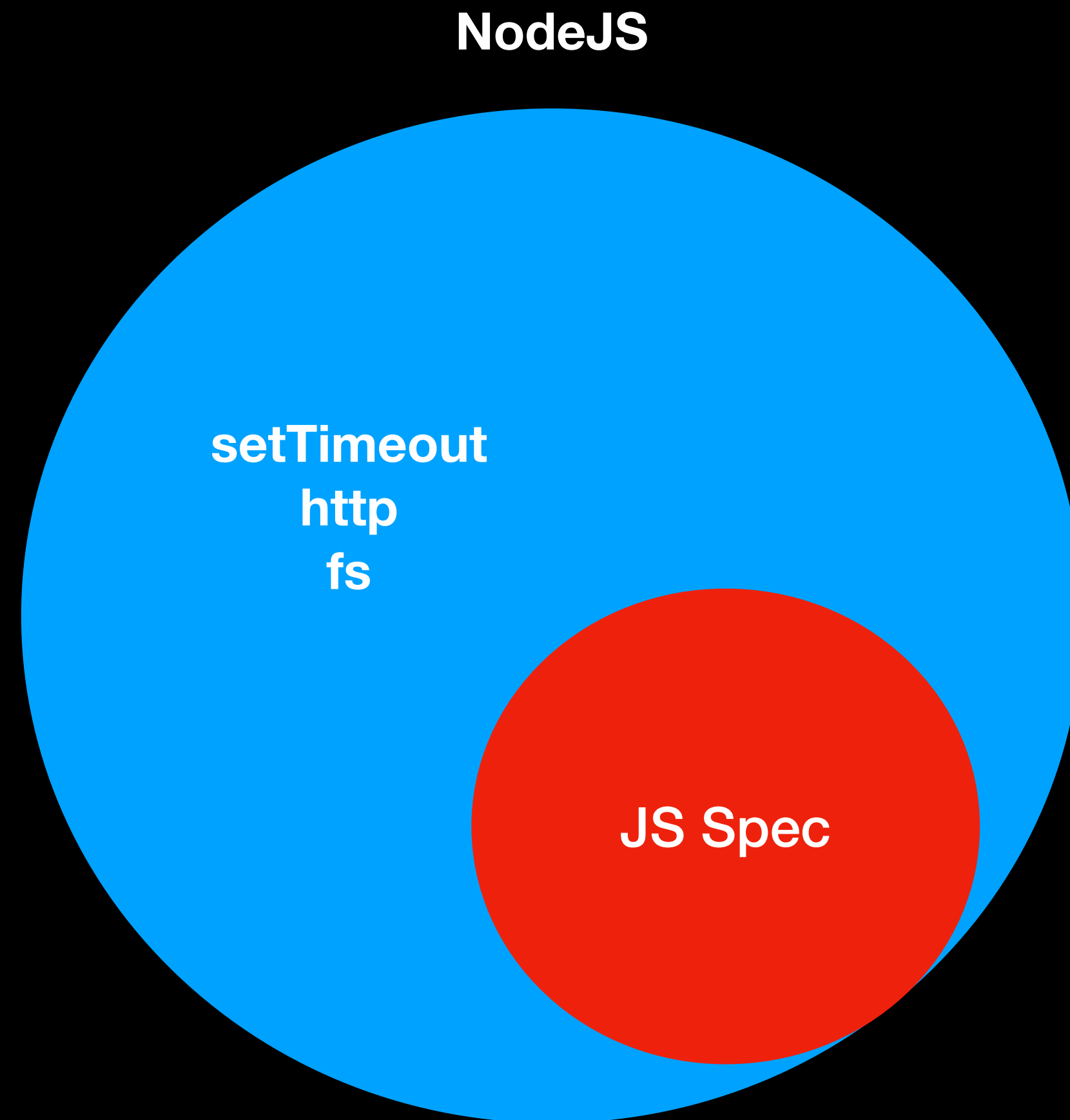
What was Javascript?

But the Javascript that runs in your browser has some extra functionality



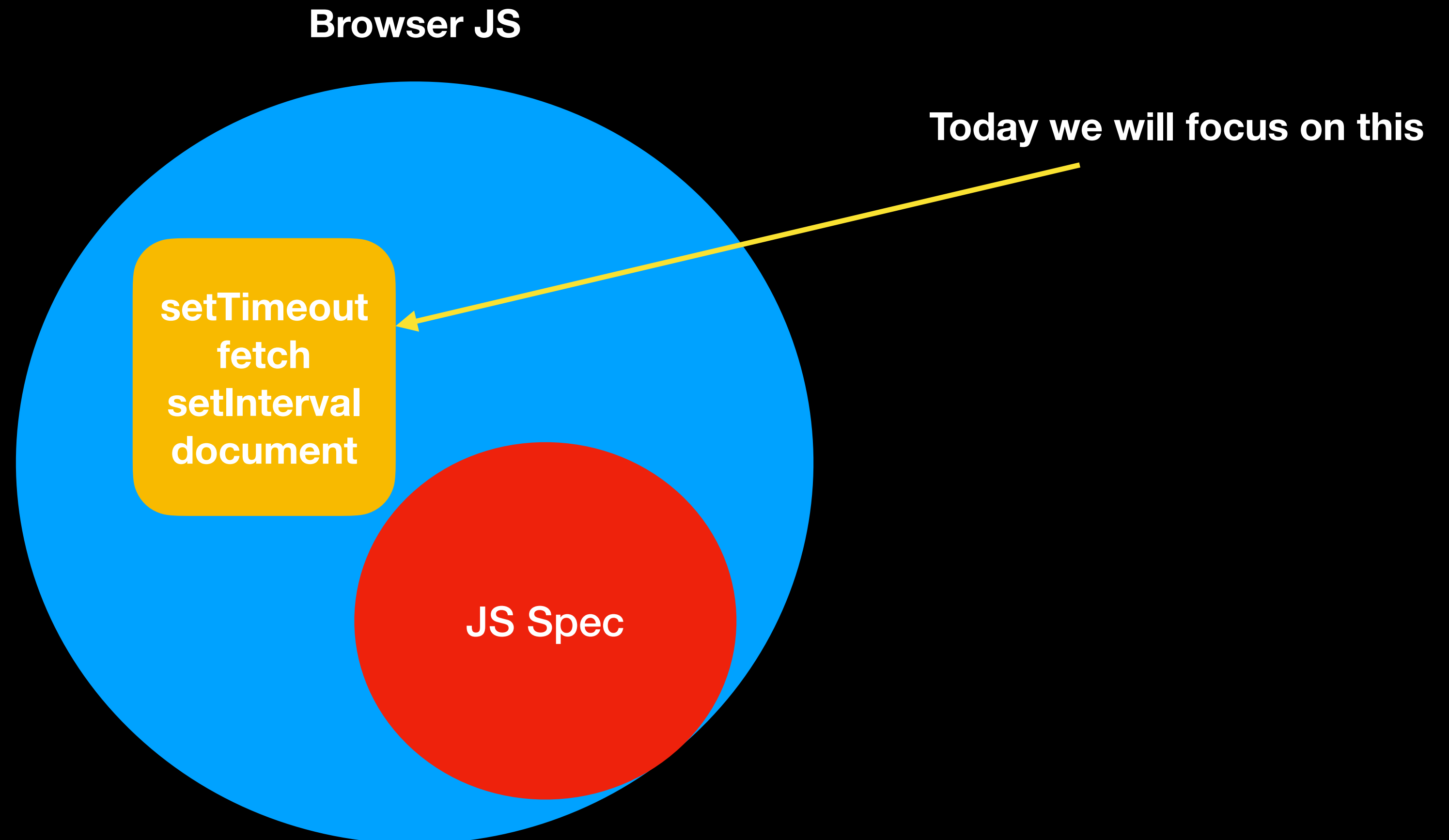
What was Javascript?

But the Javascript that runs in your browser has some extra functionality



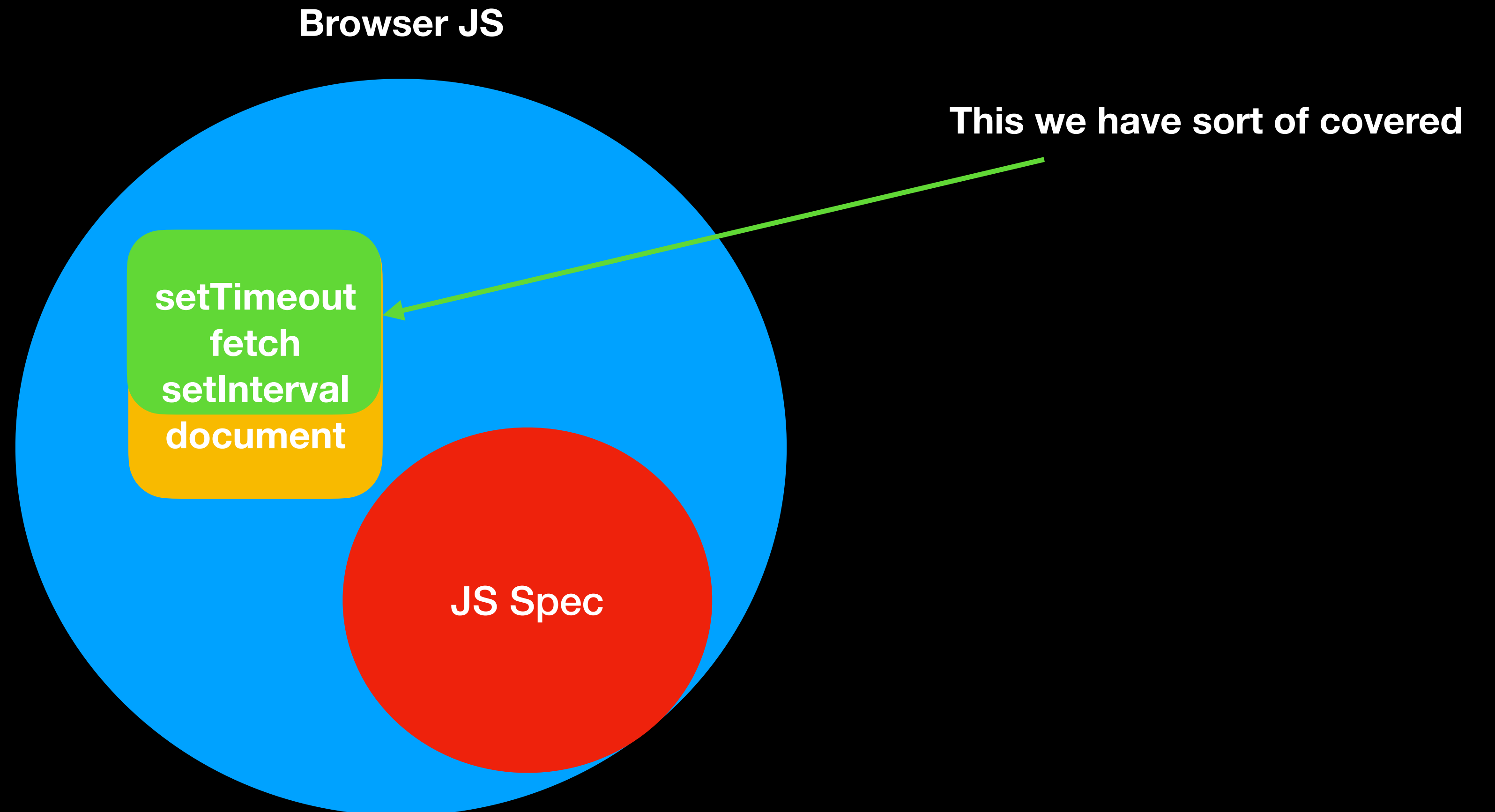
What was Javascript?

But the Javascript that runs in your browser has some extra functionality



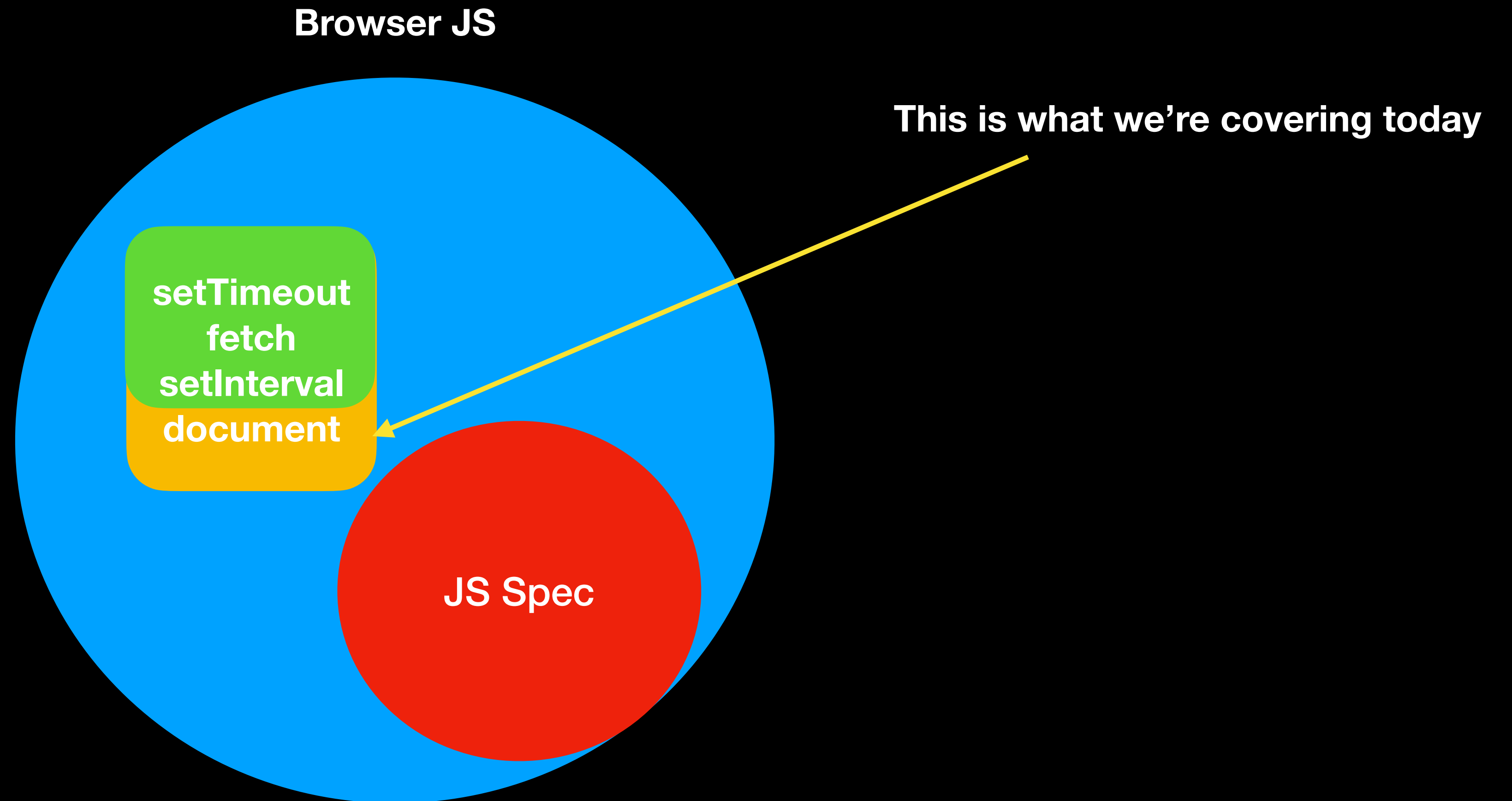
What was Javascript?

But the Javascript that runs in your browser has some extra functionality



What was Javascript?

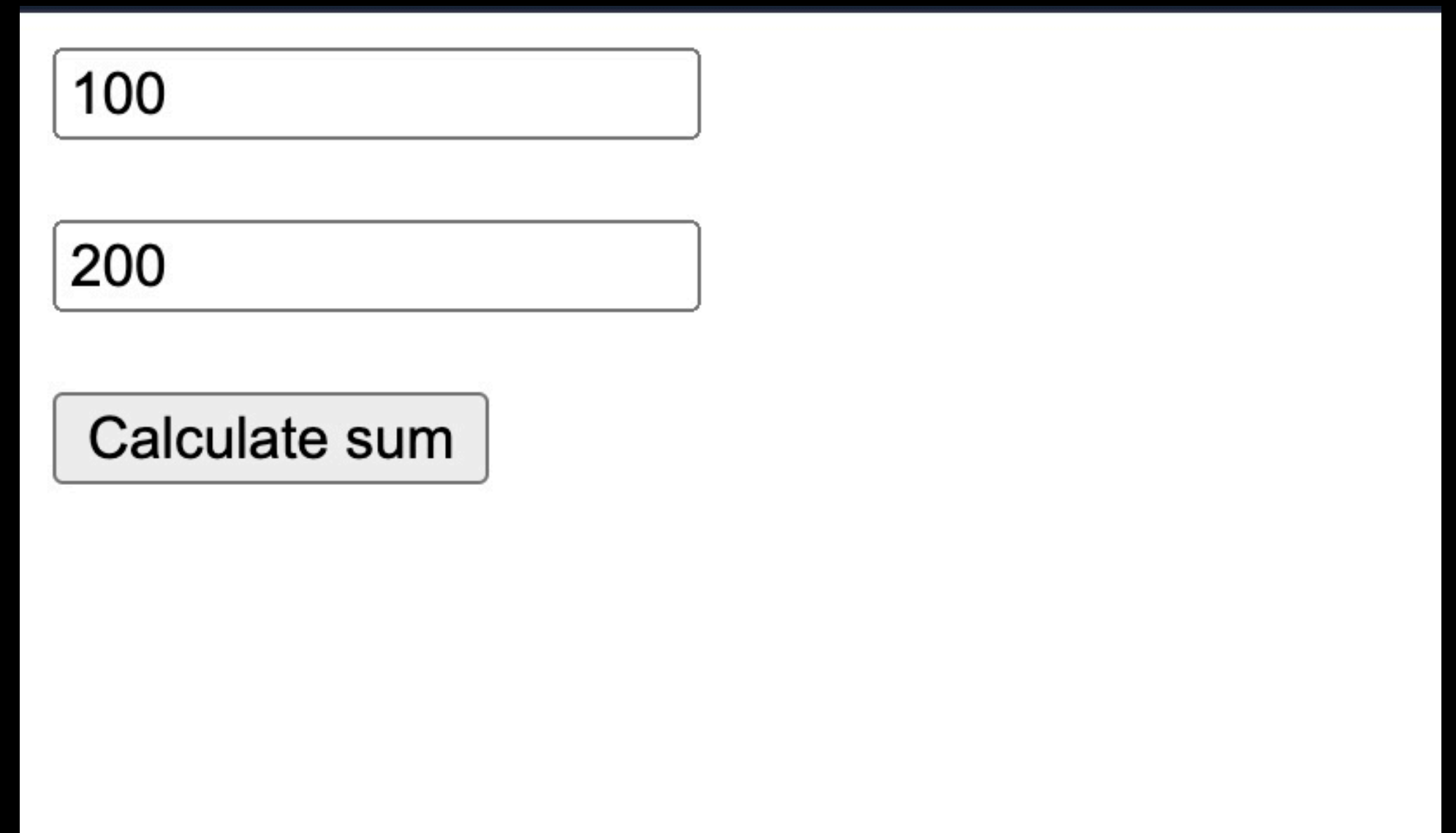
But the Javascript that runs in your browser has some extra functionality



DOM / Document object

Lets create a simple website to calculate the sum of two numbers

This is a static website, clicking the button does nothing



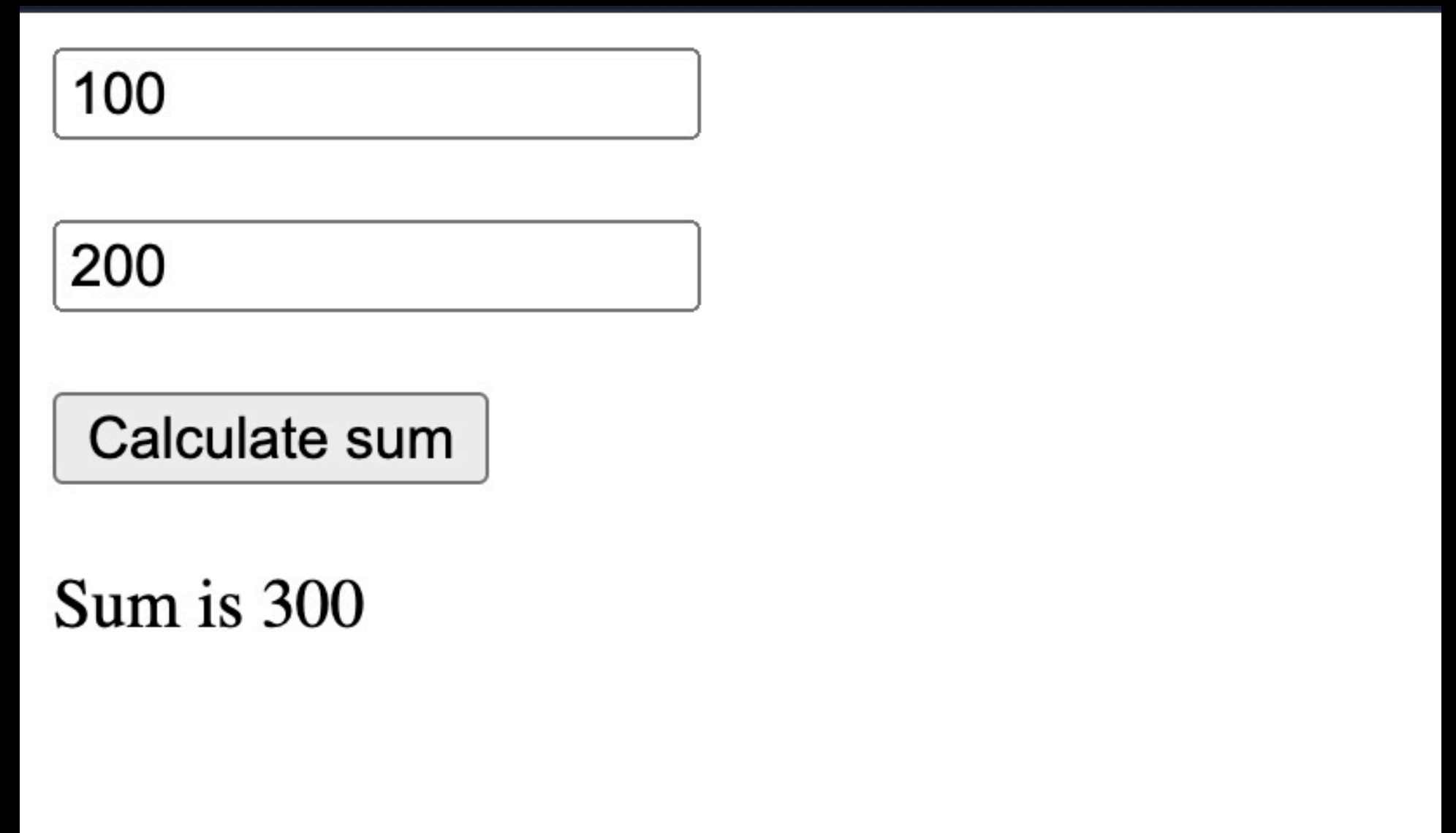
A screenshot of a web form. It features two text input fields stacked vertically. The first input field contains the number '100' and the second contains '200'. Below these fields is a button with the text 'Calculate sum'. The entire form is set against a light gray background.

DOM / Document object

But making them dynamic is hard

What do I mean, when I say dynamic -

1. Changing the elements on the website once the website is loaded
2. Actually calculating the sum based on the inputs and **rendering** it on the screen



100

200

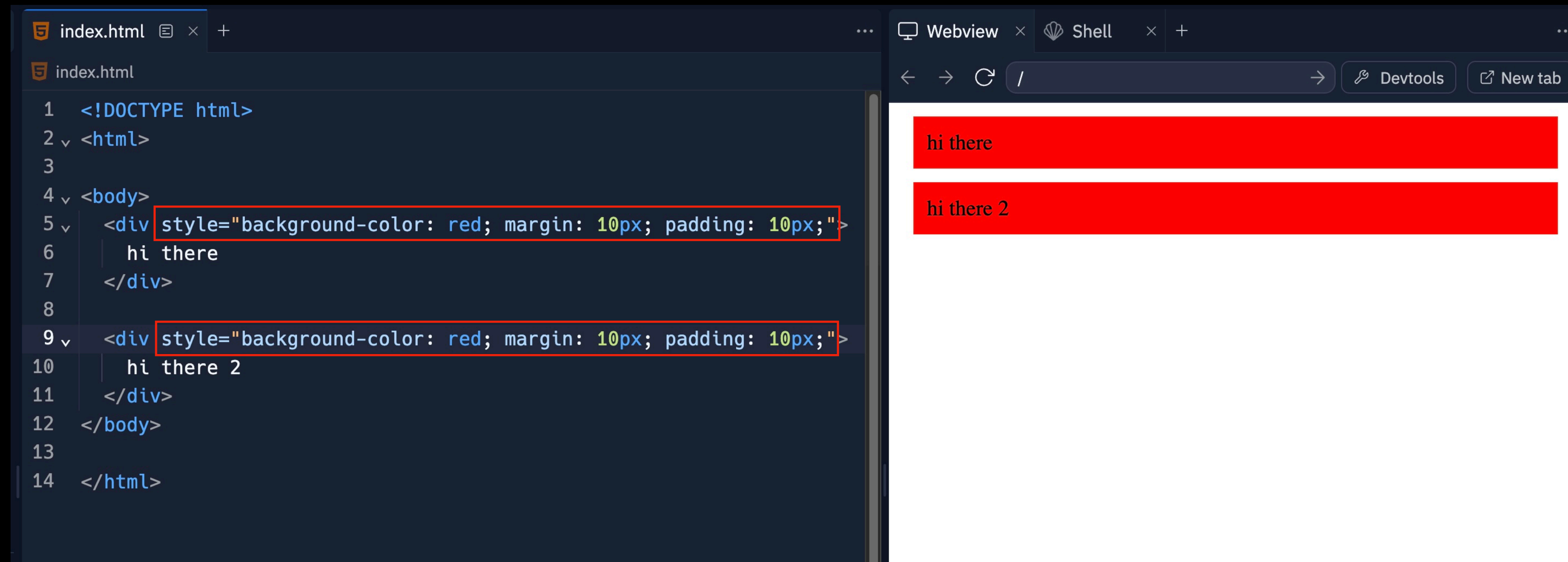
Calculate sum

Sum is 300

DOM / Document object

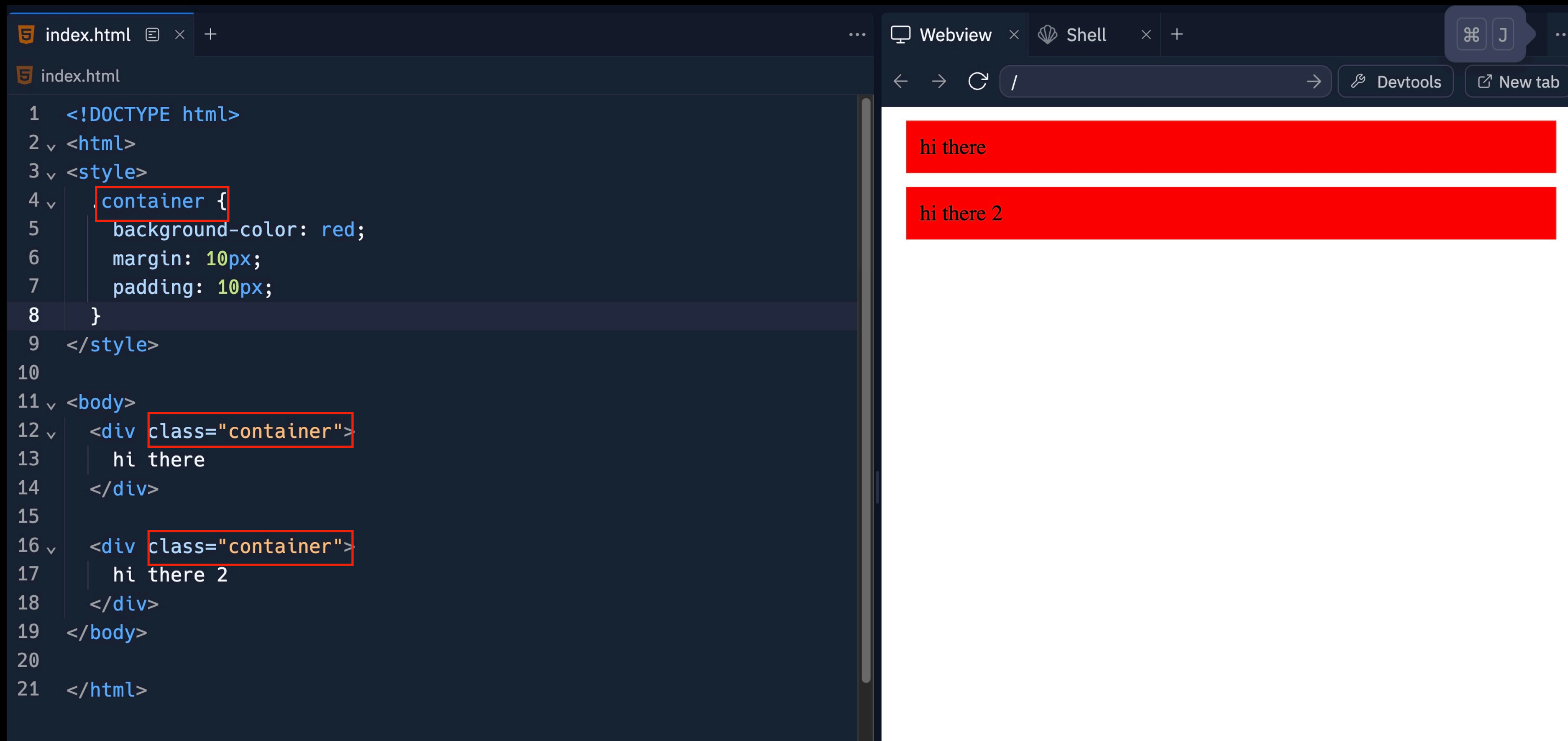
Classes and ids

Lets say we want to have the similar divs in two places



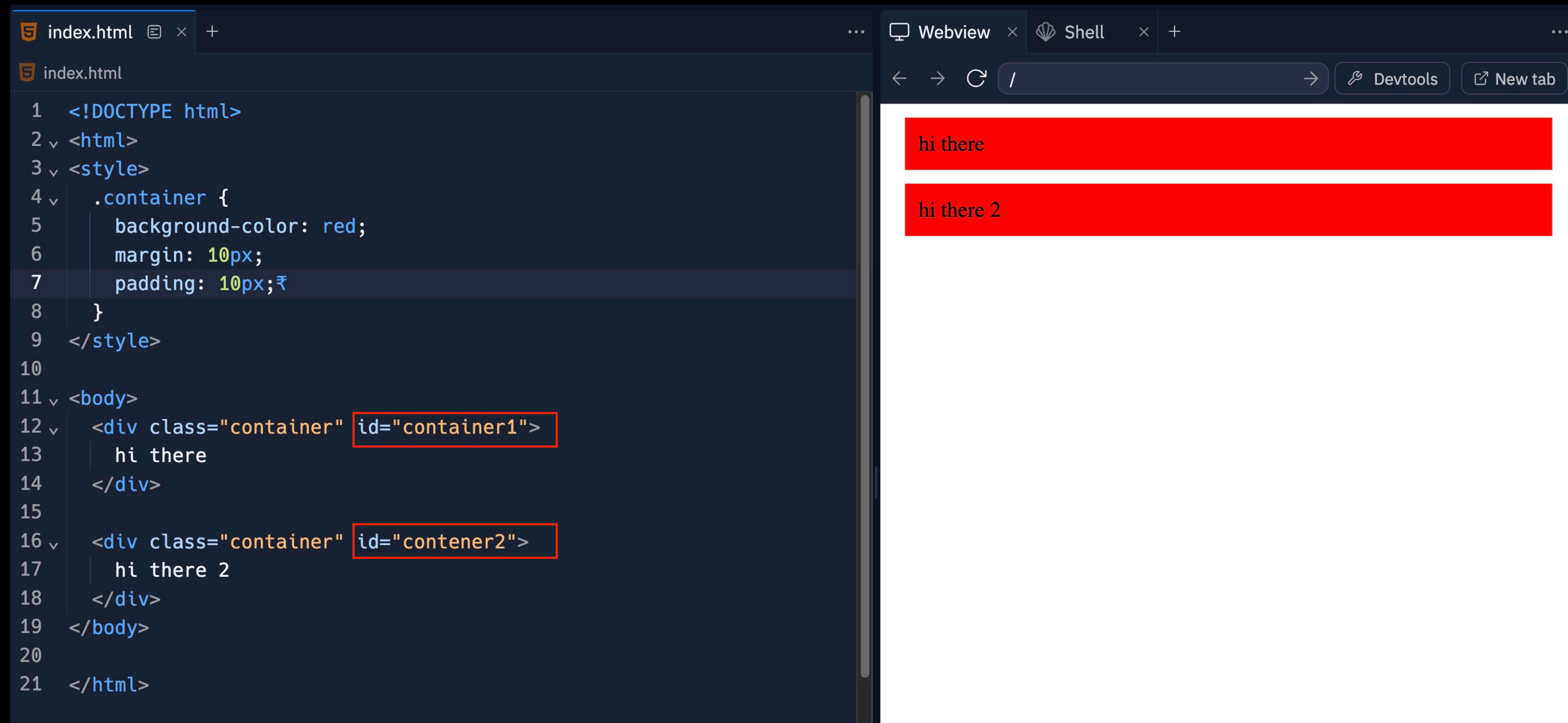
DOM / Document object

Classes



DOM / Document object

Ids are supposed to be unique. They are used to uniquely identify a div/span/input/button



The image shows a VS Code editor window with a file named 'index.html'. The code is as follows:

```
1 <!DOCTYPE html>
2 <html>
3 <style>
4   .container {
5     background-color: red;
6     margin: 10px;
7     padding: 10px;
8   }
9 </style>
10
11 <body>
12   <div class="container" id="container1">
13     hi there
14   </div>
15
16   <div class="container" id="contener2">
17     hi there 2
18   </div>
19 </body>
20
21 </html>
```

The webview on the right displays the rendered HTML. It shows two red rectangular boxes. The top box contains the text 'hi there' and the bottom box contains the text 'hi there 2'. The second box's ID in the code is misspelled as 'contener2'.

DOM / Document object

Why are Ids useful?

Classes let you get rid of code repetition

But what do ids do?

DOM / Document object


They let you access elements via the DOM API

Can you use ids to do CSS? - Yes

But what do you use it for usually - Javascript

DOM / Document object

Lets say I ask you to log the values in these inputs



100

200

Calculate sum

Sum is 300

DOM / Document object

Lets say I ask you to log the values in these inputs

Here is where the global **document** object
Comes into the picture

The diagram illustrates a web page layout with a white background. It contains two text input fields, a button, and a text output. The first input field contains the value '100' and the second contains '200'. Below these is a button labeled 'Calculate sum'. At the bottom, the text 'Sum is 300' is displayed. Two white arrows originate from the left side of the image, pointing to the first and second input fields respectively. The text 'Here is where the global document object Comes into the picture' is positioned to the left of these arrows, indicating that the 'document' object is used to access these DOM elements.

100

200

Calculate sum

Sum is 300

DOM / Document object

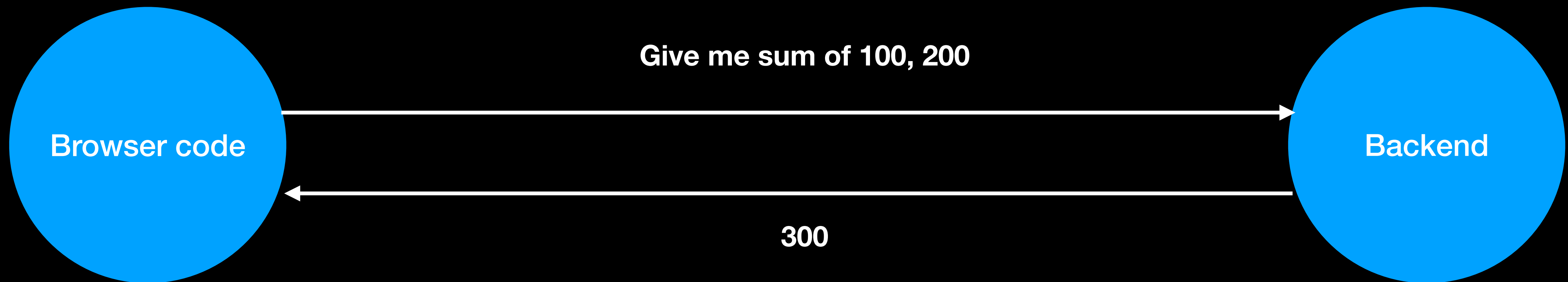
document.getElementById

document.getElementsByClassName

Lets complete this assignment

Sum is 300

**Now, lets say you don't have access to the calculation logic on the frontend
Lets assume its a hard problem that someone has exposed on a backend server
And you need to hit the backend server and get back the value**



Backend server

<https://sum-server.100xdevs.com/sum?a=1&b=2>

Harder assignment

<https://sum-server.100xdevs.com/interest?principal=100&rate=5&time=2>