JavaScript: The Basics

CS50 Seminar by Sela Kasepa
What is JavaScript and What can you do with it?
JavaScript:

- Programming language
- Single-threaded, asynchronous language
- Originally built only to run in browsers (Client-Side JavaScript):
  - Browsers have embedded JavaScript engines (e.g. Firefox - SpiderMonkey, Chrome - v8)
- Run outside browsers via node.js framework (Server-Side JavaScript)
Working with JavaScript in the Browser
JavaScript and the DOM
What is the DOM?

- Programming interface for HTML and XML documents
- A single object that represents an entire web page so that programs can change the document structure, style, and content.
- Represents the document and objects
Accessing the DOM: commonly used interfaces

- `document.getElementById(id)`
- `document.querySelector(selector)`
- `document.querySelectorAll(selector)`
- `element.innerHTML`
- `window.onload`
- `element.addEventListener()`
What is JavaScript

Single-threaded, asynchronous language
What does it mean for JavaScript to be Single-Threaded

Can only do one thing at a time

Has a single Call Stack

What is a Call Stack?

Data Structure that records where we are in the program
JavaScript Environment

Asynchronous language

- The JavaScript runtime can only do one thing at a time
- Our browsers have extra features that allow us to perform tasks

Asynchronously:
- WebAPIs
- Event Loop

- What happens when function provided by web API is called:
  - function in web API is called
  - function is pushed off stack whilst being executed
  - when done - function is added to task queue
  - event loop monitors task queue; when call stack is empty, it pushes the first function in task queue onto the stack
Promises, Async ..... Await
● Promise - an object that may produce a value some time in the future.

● Promises can be in one of the following states:
  ○ fulfilled
  ○ rejected
  ○ pending

● Promises enable us to dictate when we want functions to execute.
Constructing a promise

code:

```javascript
const promiseExample = new Promise((resolve, reject) => {
    // asynchronous task
    // resolve()
    // or
    // reject
});
```