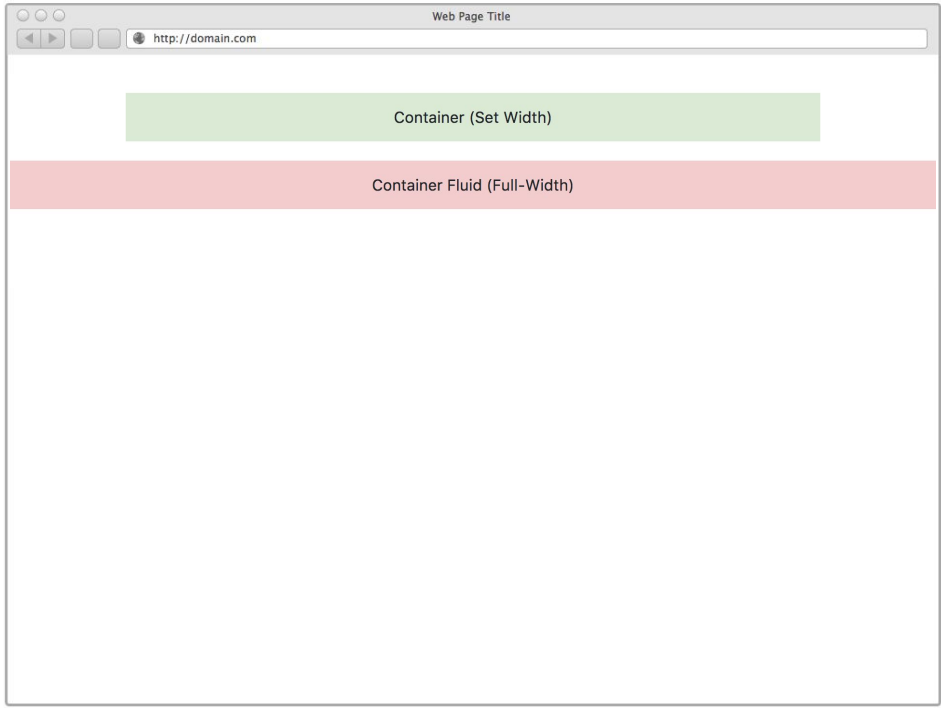
Lecture 06: Bootstrap continued, Intro to JavaScript

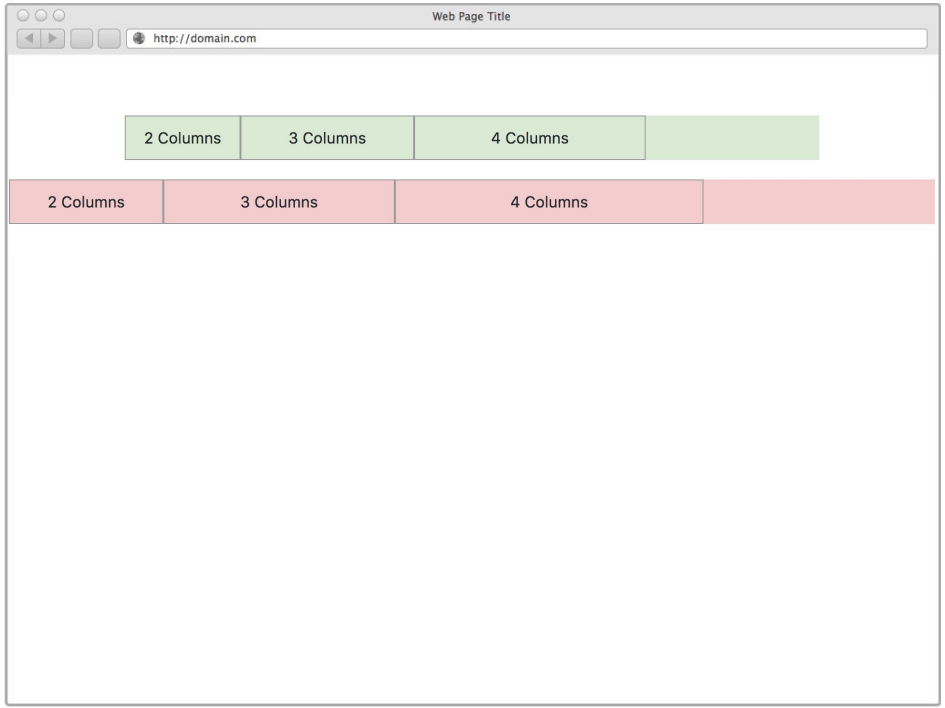
ITP 303 Full-Stack Web Development

Bootstrap: Grid Structure



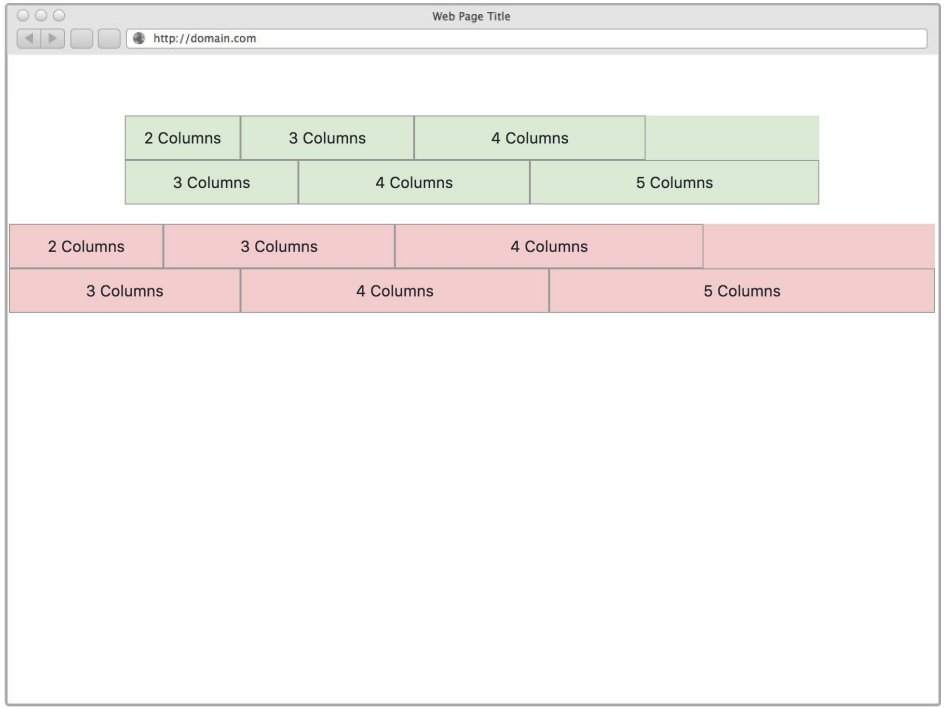

```
<div class="container">
  Container (Set Width)
</div> <!-- .container -->

<div class="container-fluid">
  Container Fluid (Full-Width)
</div> <!-- .container-fluid -->
```

Bootstrap: Grid Structure




```
<div class="container">
  <div class="row">
    <div class="col-2">2 Columns</div>
    <div class="col-3">3 Columns</div>
    <div class="col-4">4 Columns</div>
  </div> <!-- .row -->
</div> <!-- .container -->

<div class="container-fluid">
  <div class="row">
    <div class="col-2">2 Columns</div>
    <div class="col-3">3 Columns</div>
    <div class="col-4">4 Columns</div>
  </div> <!-- .row -->
</div> <!-- .container-fluid -->
```

Bootstrap: Grid Structure




```
<div class="container">
  <div class="row">
    <div class="col-2">2 Columns</div>
    <div class="col-3">3 Columns</div>
    <div class="col-4">4 Columns</div>
  </div> <!-- .row -->
  <div class="row">
    <div class="col-3">3 Columns</div>
    <div class="col-4">4 Columns</div>
    <div class="col-5">5 Columns</div>
  </div> <!-- .row -->
</div> <!-- .container -->

<div class="container-fluid">
  <div class="row">
    <div class="col-2">2 Columns</div>
    <div class="col-3">3 Columns</div>
    <div class="col-4">4 Columns</div>
  </div> <!-- .row -->
  <div class="row">
    <div class="col-3">3 Columns</div>
    <div class="col-4">4 Columns</div>
    <div class="col-5">5 Columns</div>
  </div> <!-- .row -->
</div> <!-- .container-fluid -->
```

Bootstrap: Breakpoints

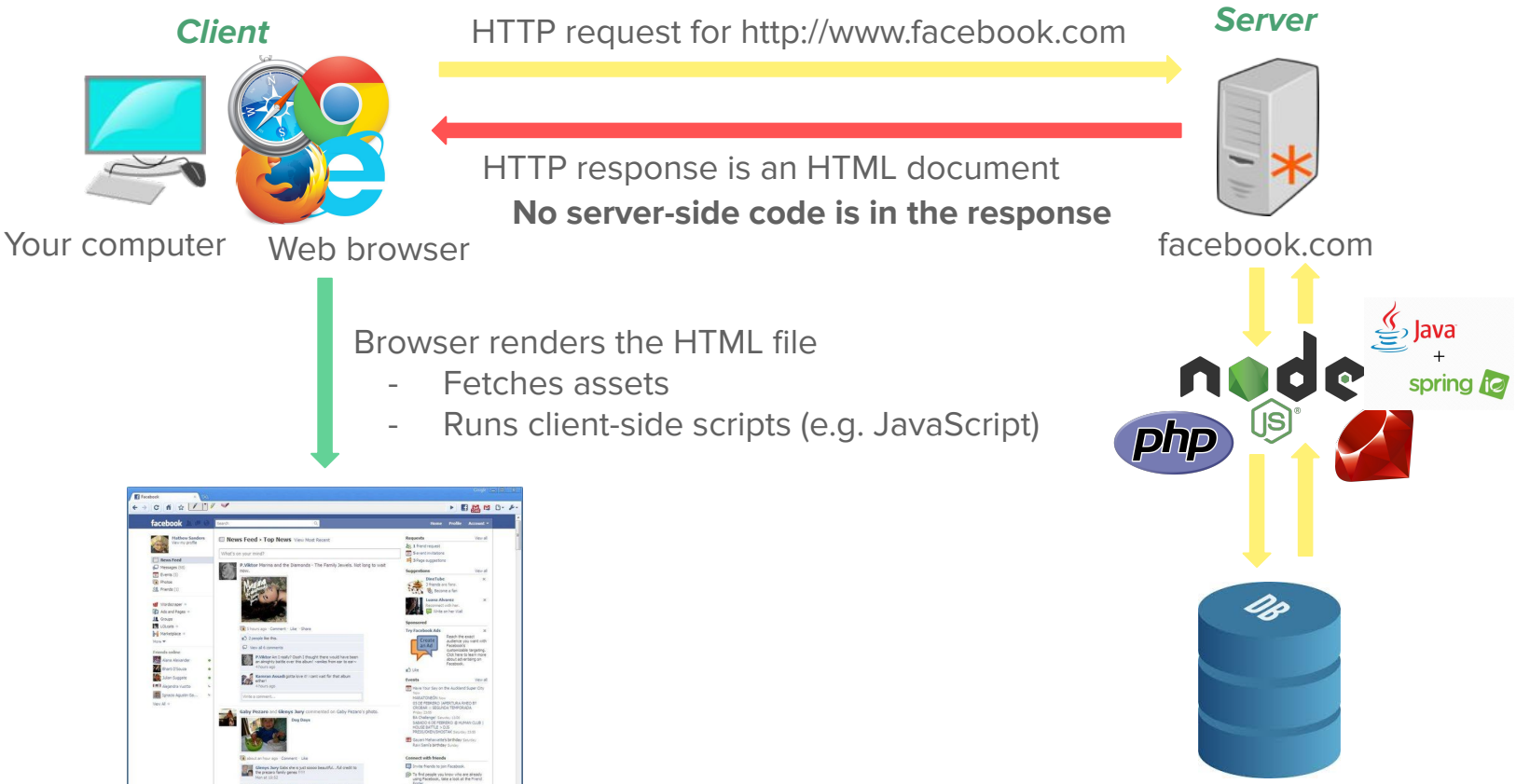
Mobile-First Responsive Design.

Name	Size	Devices
Extra Small	≤ 575px	Portrait Phones
Small	576px - 767px	Landscape Phones
Medium	768px - 991px	Tablets
Large	992px - 1199px	Desktops
Extra Large	≥ 1200px	Large Desktops

```
/* Extra small devices (portrait phones, less than 576px) */
/* No media query since this is the default in Bootstrap */

/* Small devices (landscape phones, 576px and up) */
@media (min-width: 576px) { ... }

/* Medium devices (tablets, 768px and up) */
@media (min-width: 768px) { ... }

/* Large devices (desktops, 992px and up) */
@media (min-width: 992px) { ... }

/* Extra large devices (large desktops, 1200px and up) */
@media (min-width: 1200px) { ... }
```

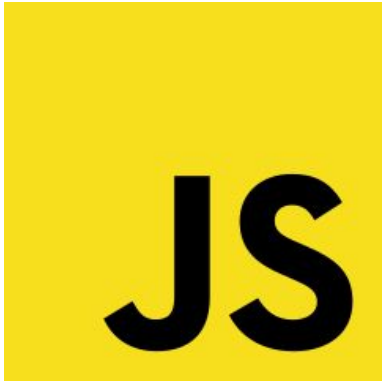
Intro to JavaScript

The journey so far...

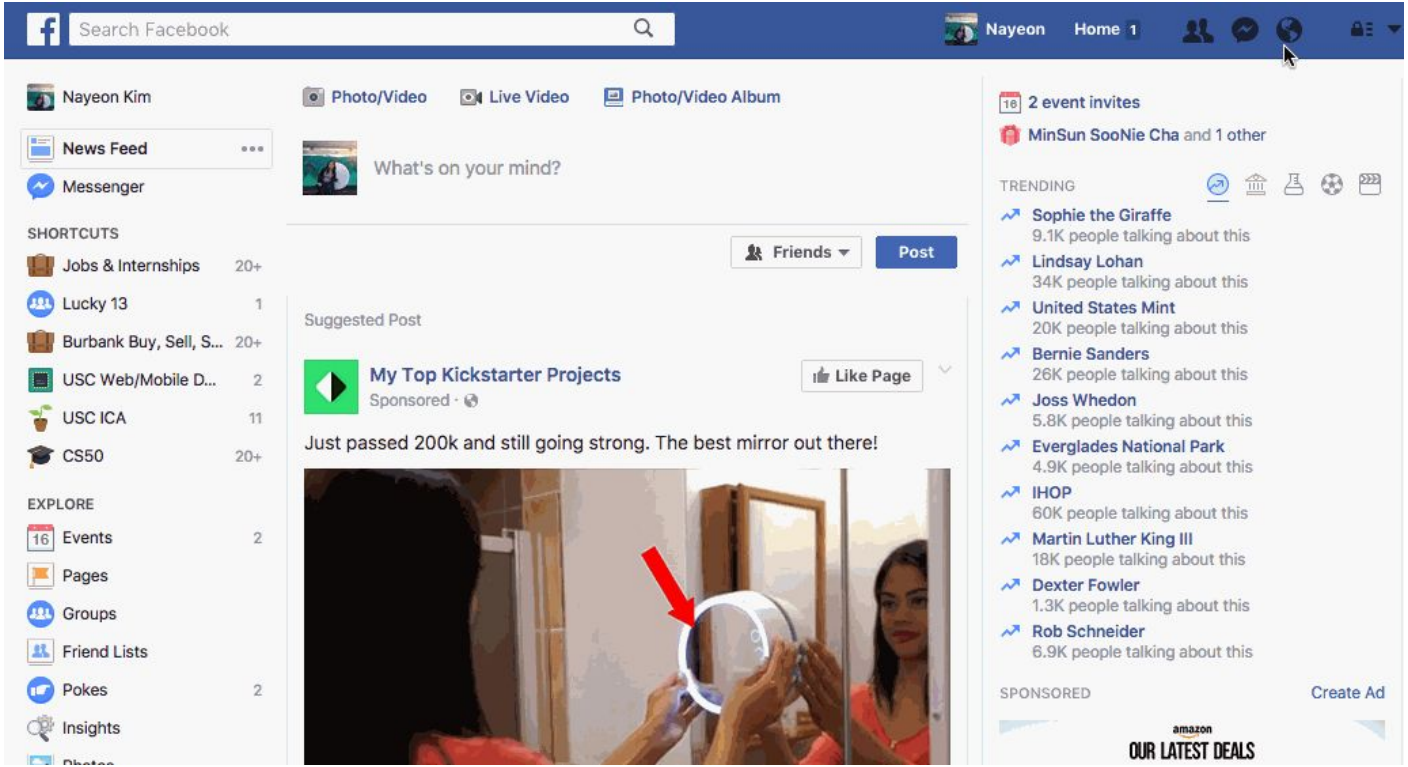


JavaScript (JS)

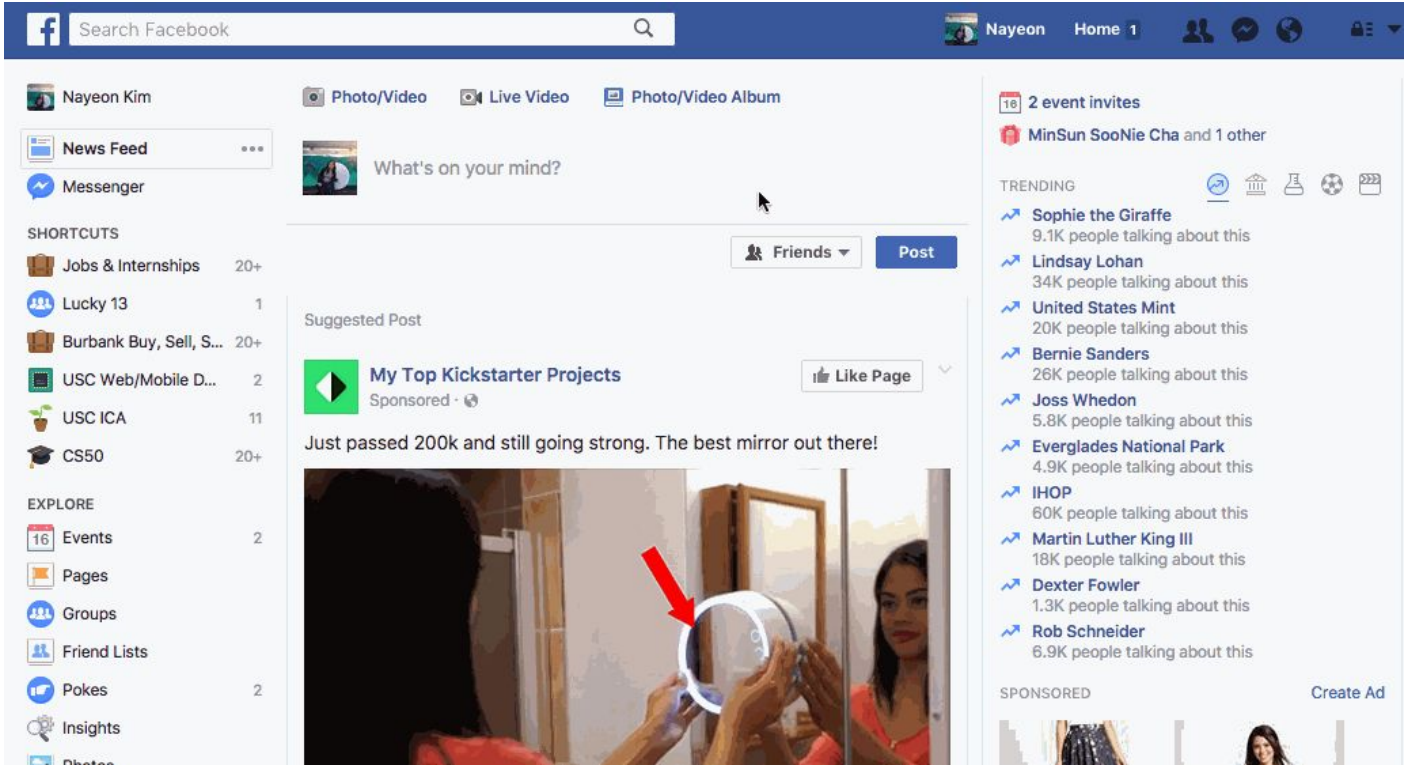
- A programming language that can be read and executed by the browser.
- While many ideas are borrowed from the language Java, **Java and JavaScript are are two entirely different languages**.
- Primarily used on client-side, but with NodeJS on the backend, it's possible to build an entire web application with just JavaScript
- Some key things JS is good at:
  - DOM manipulation
  - Listening to user events (click, hover, keypress, etc)
  - Client-server communication without reloading pages
  - And more...



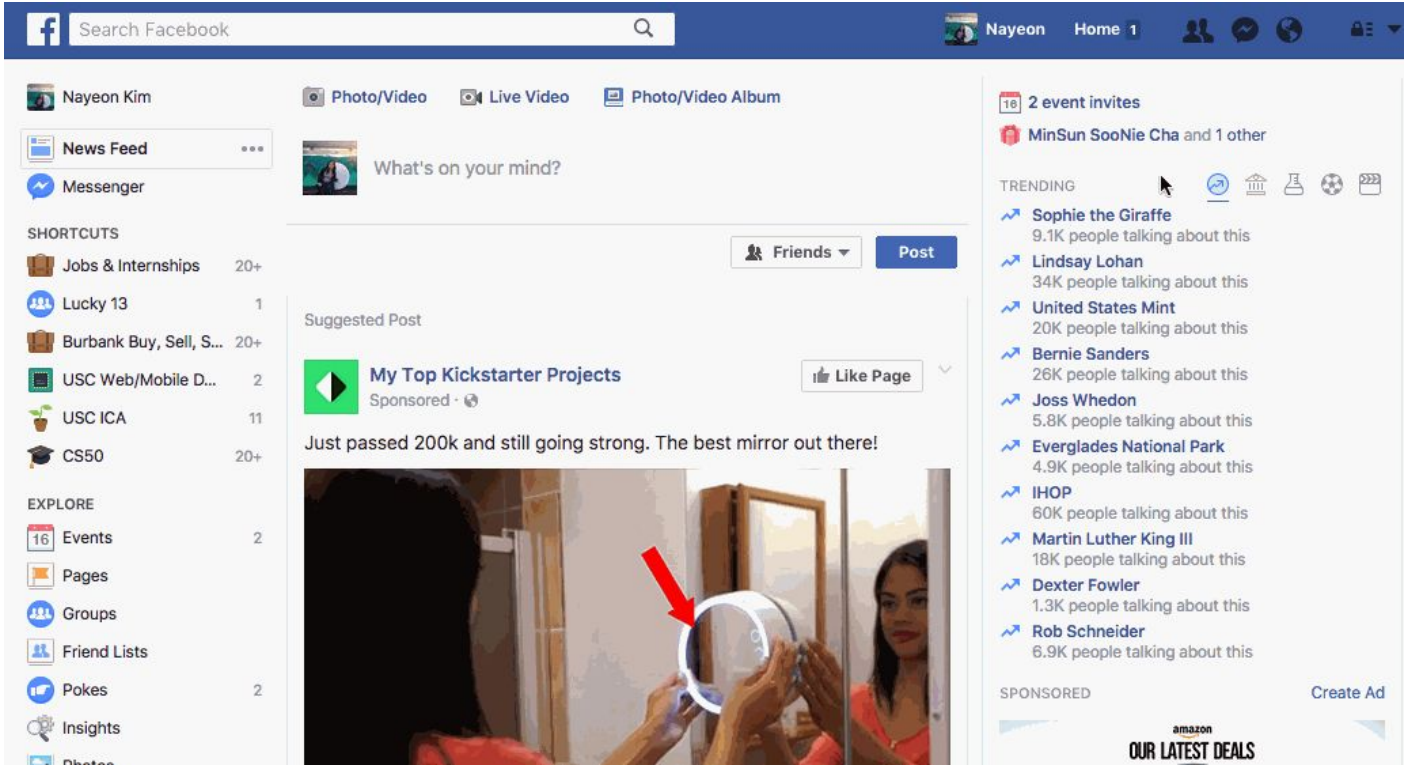
JavaScript in live action



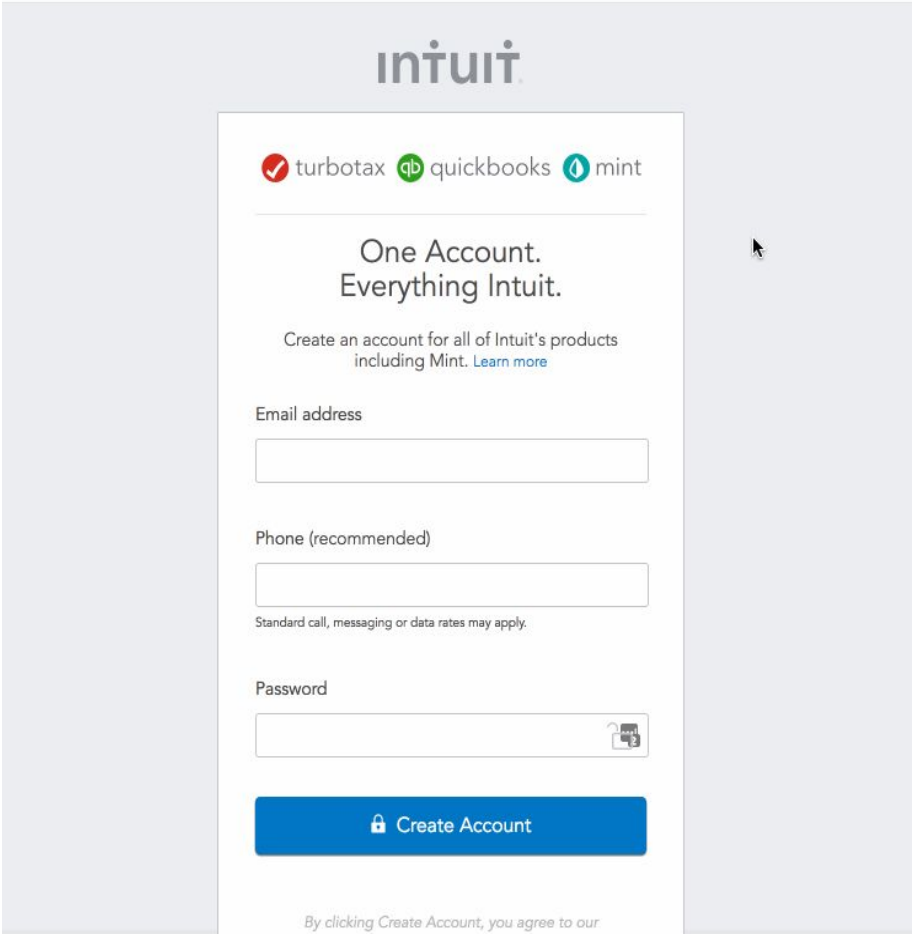
JavaScript in live action



JavaScript in live action



JavaScript in live action



JS syntax

```
var name = “nayeon”;

var favoriteNumber = 2;

var isInstructor = true;
```

JS syntax

```
let name = “nayeon”;

let favoriteNumber = 2;

let isInstructor = true;
```

Resource: More on let

JS syntax

```
let book = {
  title: “Jane Eyre”,
  author: “Charlotte Bronte”,
  published: 1847
}
```

JS syntax

```
if(5 > 7) {
  // some code
}
else {
  // some code
}
```

JS syntax

```
for (let i = 0; i<list.length; i++) {

  // code to iterate here

}
```

JS syntax

```
let numberArray = [1,2,3,4,5];
```

JS syntax

```
let numberArray = [1,2,3,4,5];

numberArray.push(6);

// now numberArray is has 1,2,3,4,5,6
```

JS syntax

```
function sayHello() {
  console.log(“Hello!”);
}
```

JS syntax

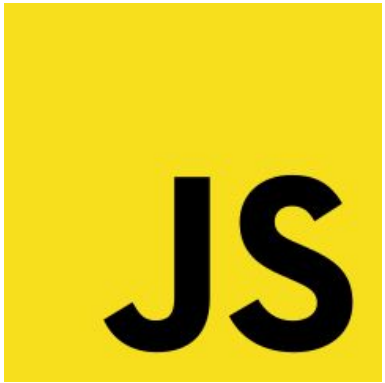
```
let sayHello = function() {

  console.log(“Hello!”);

}
```

JS syntax

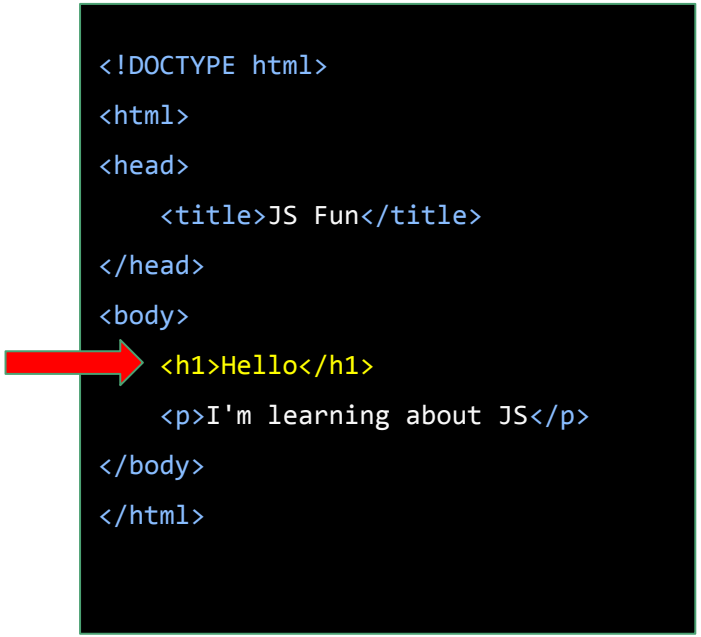
```
let sayHello = function() {

  console.log(“Hello!”);

}

// To call the function

sayHello();
```

Two key features of JS

1. Searching and changing elements in the DOM
2. Listening to user events



Searching elements??? What does that mean?



JavaScript allows us to easily **find** any specific element in our HTML file.

To fully understand this, we need to first learn about the **Document Object Model**

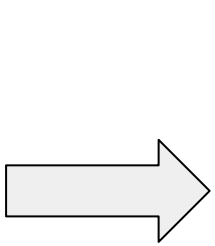
Document Object Model (DOM)

- A tree-like structure that represents a web page which can be utilized quickly access elements using JS

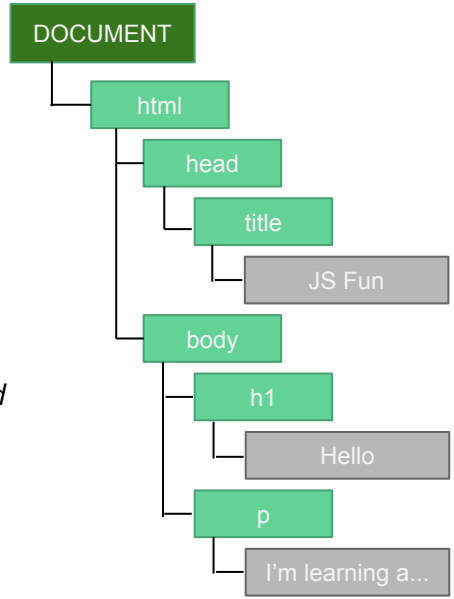
Document Object Model (DOM)

- A tree-like structure that represents a web page which can be utilized quickly access elements using JS

```
<!DOCTYPE html>
<html>
<head>
    <title>JS Fun</title>
</head>
<body>
    <h1>Hello</h1>
    <p>I'm learning about JS!</p>
</body>
</html>
```



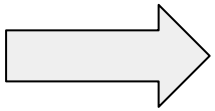
Your HTML file gets read into the browser and loads into DOM



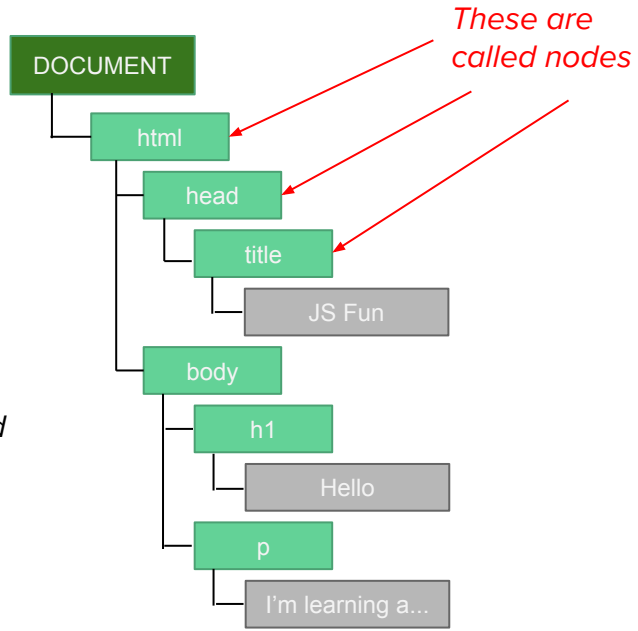
Document Object Model (DOM)

- A tree-like structure that represents a web page which can be utilized quickly access elements using JS

```
<!DOCTYPE html>
<html>
<head>
    <title>JS Fun</title>
</head>
<body>
    <h1>Hello</h1>
    <p>I'm learning about JS!</p>
</body>
</html>
```



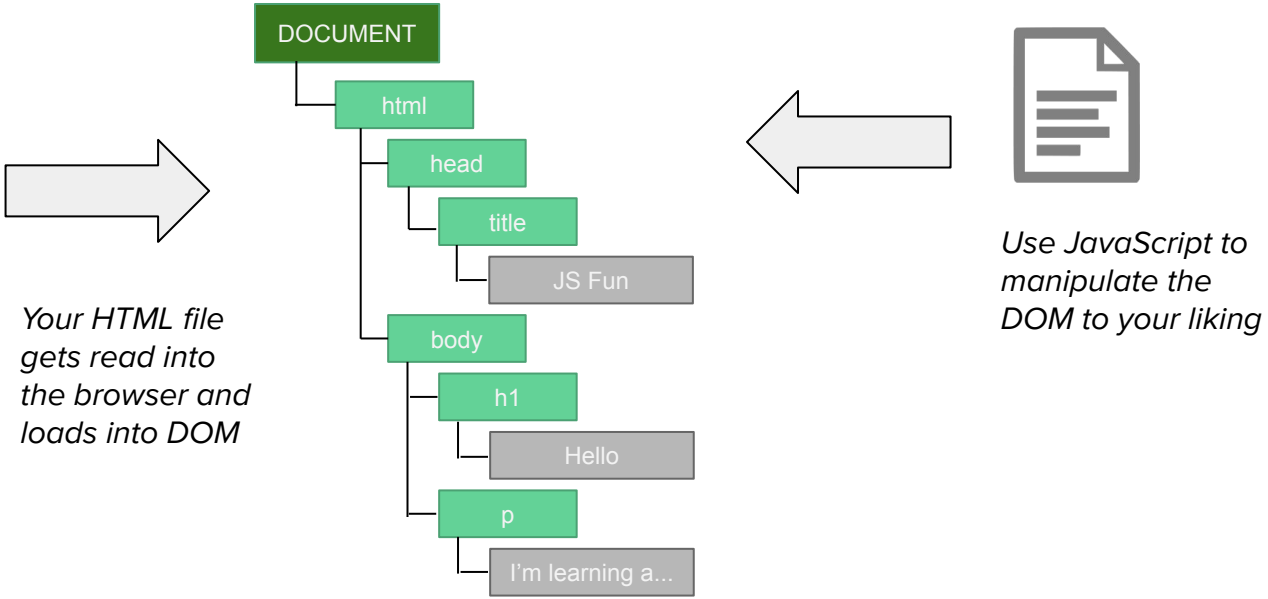
Your HTML file gets read into the browser and loads into DOM



Ok... so what does JS do?

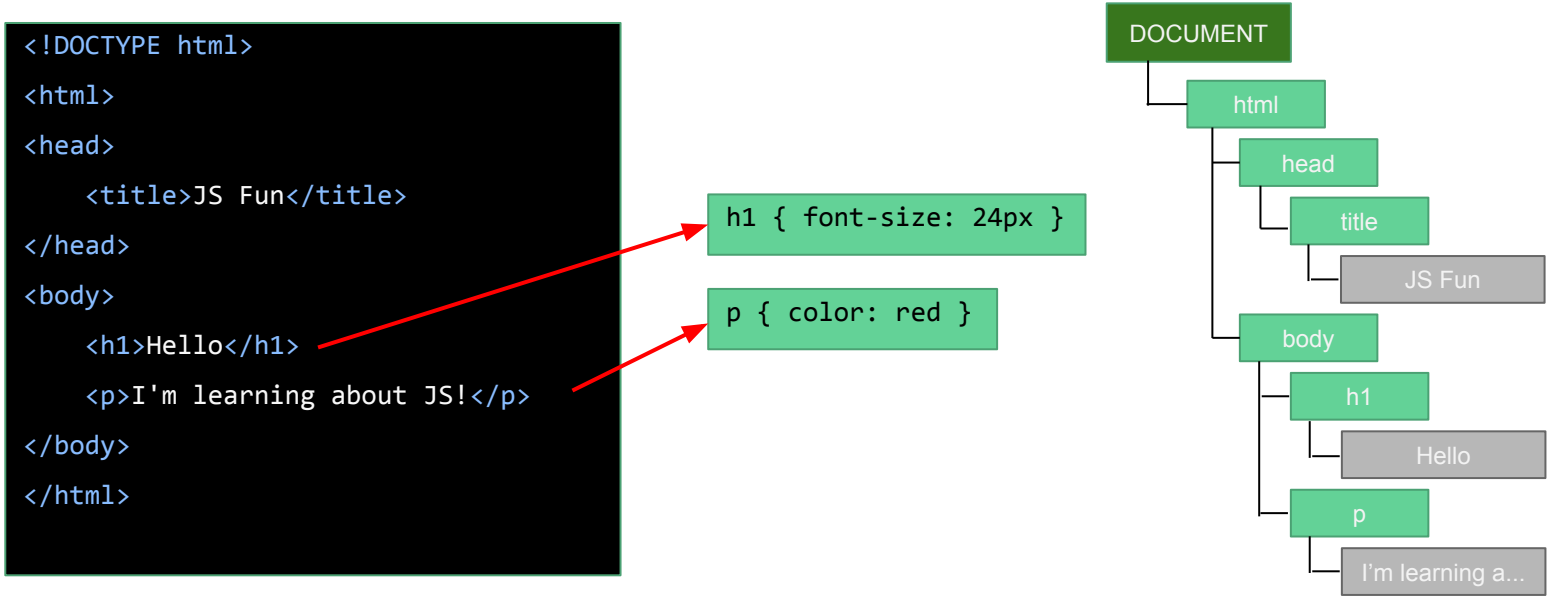
- JavaScript gives us a language to interact with the DOM

```
TYPE html>
>
>
title>JS Fun</title>
d>
>
h1>Hello</h1>
p>I'm learning about JS!</p>
y>
l>
```



How JS “Finds elements” aka accesses the DOM

- We can select any DOM elements with JS by using the same syntax as CSS selectors



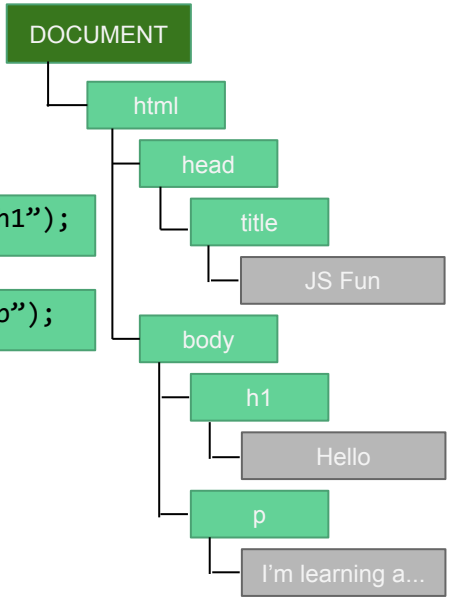
How JS “Finds elements” aka accesses the DOM

- We can select any DOM elements with JS by using the same syntax as CSS selectors

```
<!DOCTYPE html>
<html>
<head>
    <title>JS Fun</title>
</head>
<body>
    <h1>Hello</h1>
    <p>I'm learning about JS!</p>
</body>
</html>
```

```
document.querySelector(“h1”);
```

```
document.querySelector(“p”);
```



Accessing DOM Nodes

There are few ways to select DOM Nodes (HTML Elements) using JS:

- `document.querySelector(` *`CSS Selector`*`)`
- `document.querySelectorAll(` *`CSS Selector`*`)`
- `document.getElementById(` *`ID`*`)`
- `document.getElementsByClassName(` *`Class`*`)`
- `document.getElementsByTagName(` *`Tag`*`)`

DOM Events

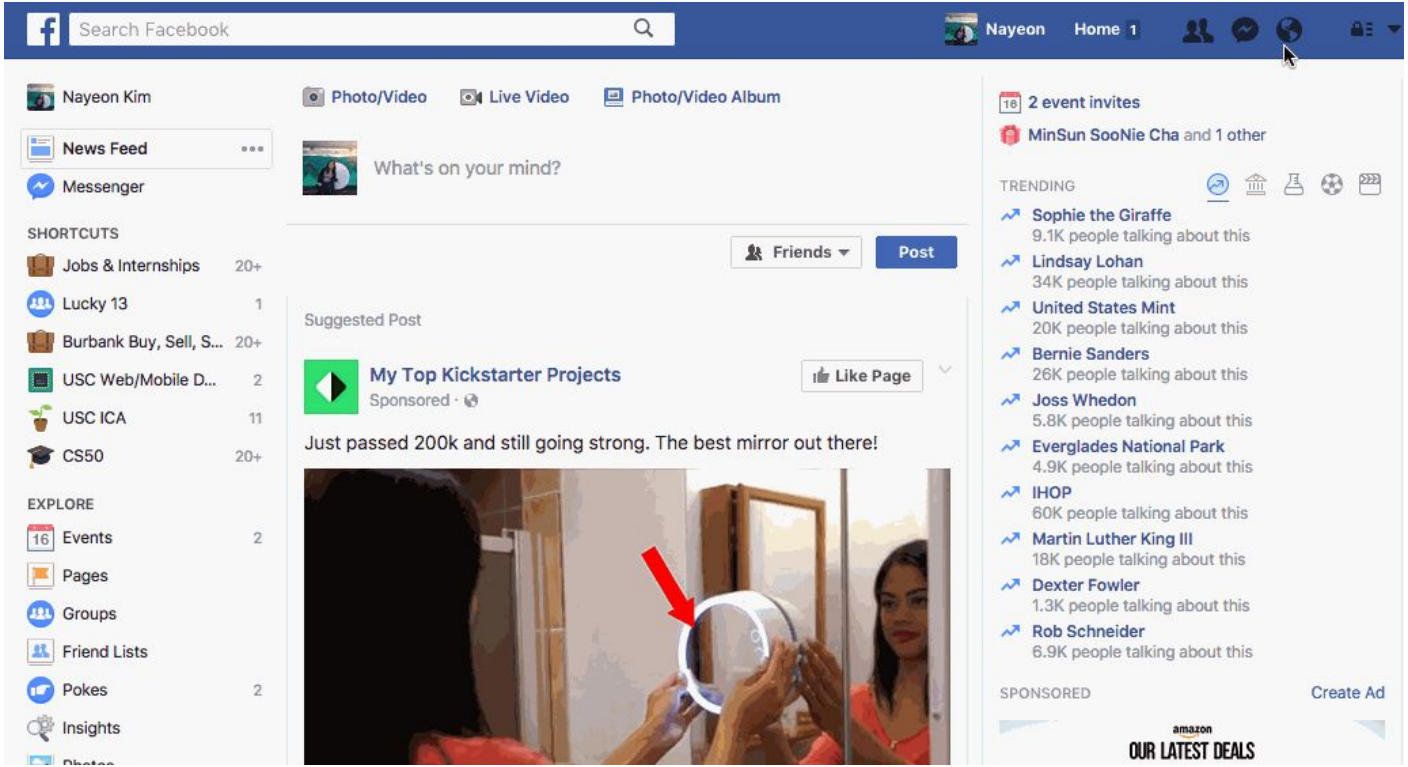
Now that we can “find” elements, we can also wait and listen for an event to trigger the element.

`onclick`	Mouse right-click on an element.
`onmouseenter`	Mouse enters an element.
`onmouseleave`	Mouse leaves an element.
`onmouseover`	Mouse enters an element or its children.
`onmouseout`	Mouse leaves an element or its children.

Resource: Full list of DOM Events.

```
document.querySelector('#content').onclick = function(){
  this.style.backgroundColor = '#FC0';
};

document.querySelector('div').onmouseenter = function(){
  document.querySelector('#name').innerHTML = 'Tommy';
};

var items = document.querySelectorAll('.item');
for (var i=0; i < items.length; i++) {
  items[i].onmouseleave = function(){
    this.href = 'https://www.usc.edu/';
  }
}
```

JavaScript in live action

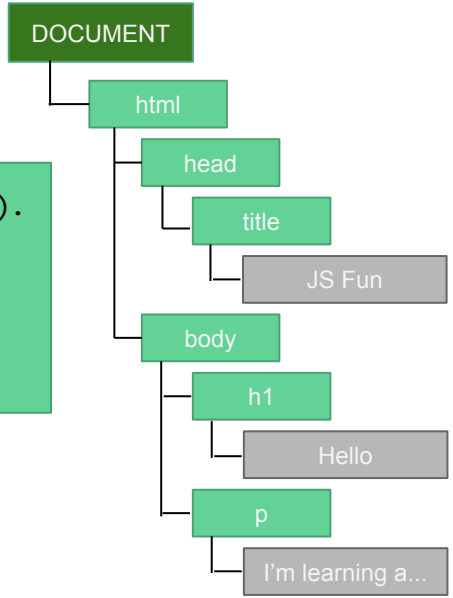


DOM Traversal is possible too

- Without specifying an element, can find neighboring elements like sibling, parent, child, etc

```
<!DOCTYPE html>
<html>
<head>
    <title>JS Fun</title>
</head>
<body>
    <h1>Hello</h1>
    <p>I'm learning about JS!</p>
</body>
</html>
```

```
document.querySelector("h1").
nextSibling;
// returns <p>I'm learning
about JS!</p>
```



DOM Traversal properties

`parentNode`	Parent element.
`children`	Children elements.
`nextSibling`	Next sibling, including whitespace (text) nodes.
`nextElementSibling`	Next sibling, excluding whitespace (text) nodes.
`previousSibling`	Previous sibling, including whitespace (text) nodes.
`previousElementSibling`	Previous sibling, excluding whitespace (text) nodes.