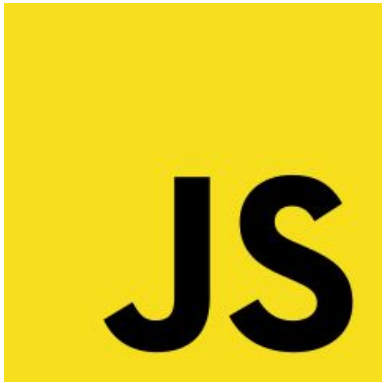
Lecture 07: JS Events and Traversal

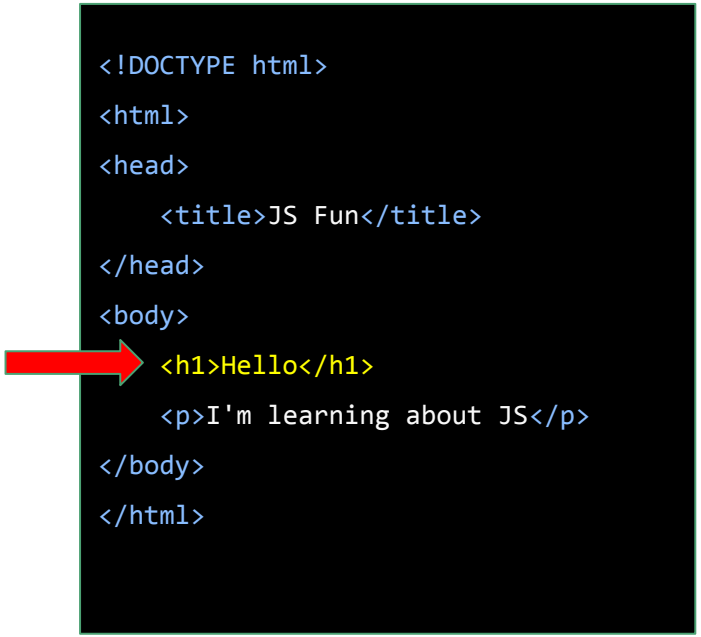
ITP 303 Full-Stack Web Development

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**

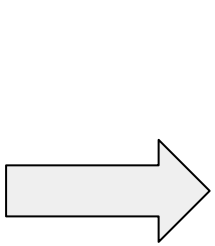
Document Object Model (DOM)

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

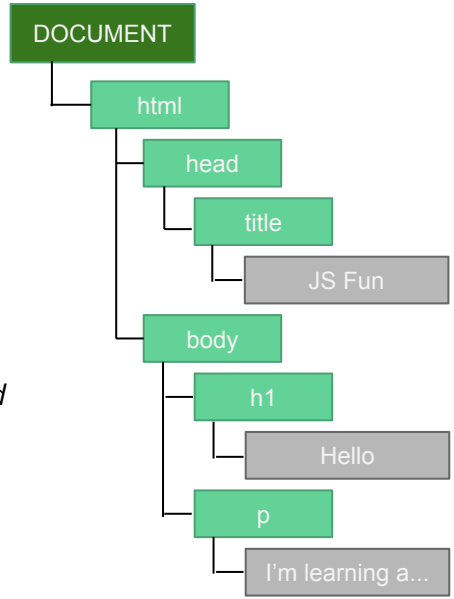
Document Object Model (DOM)

- 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>
```



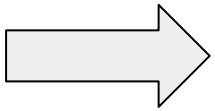
Your HTML file gets read into the browser and loads into DOM



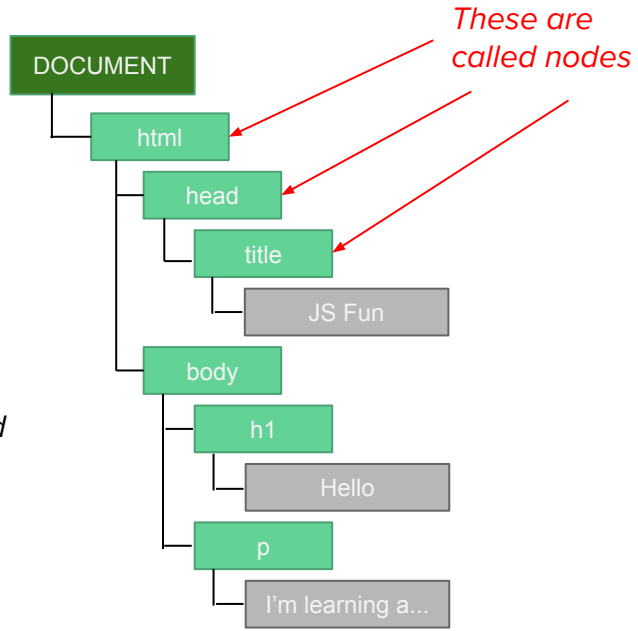
Document Object Model (DOM)

- 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>
```



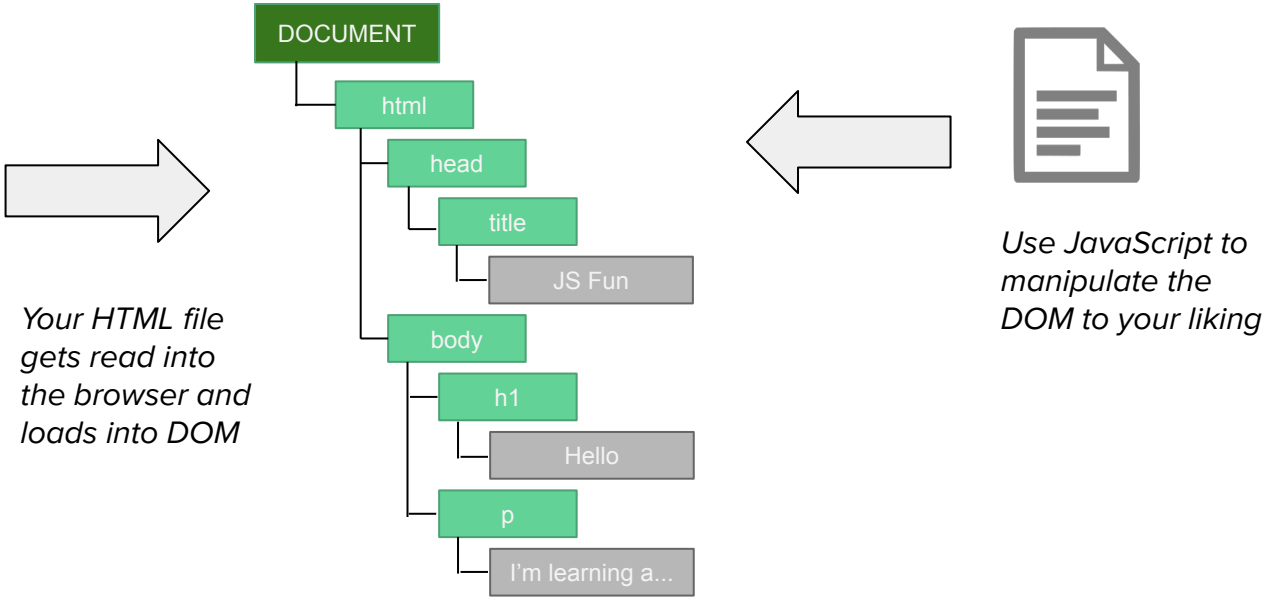
Your HTML file gets read into the browser and loads into DOM



Ok... so what does JS do?

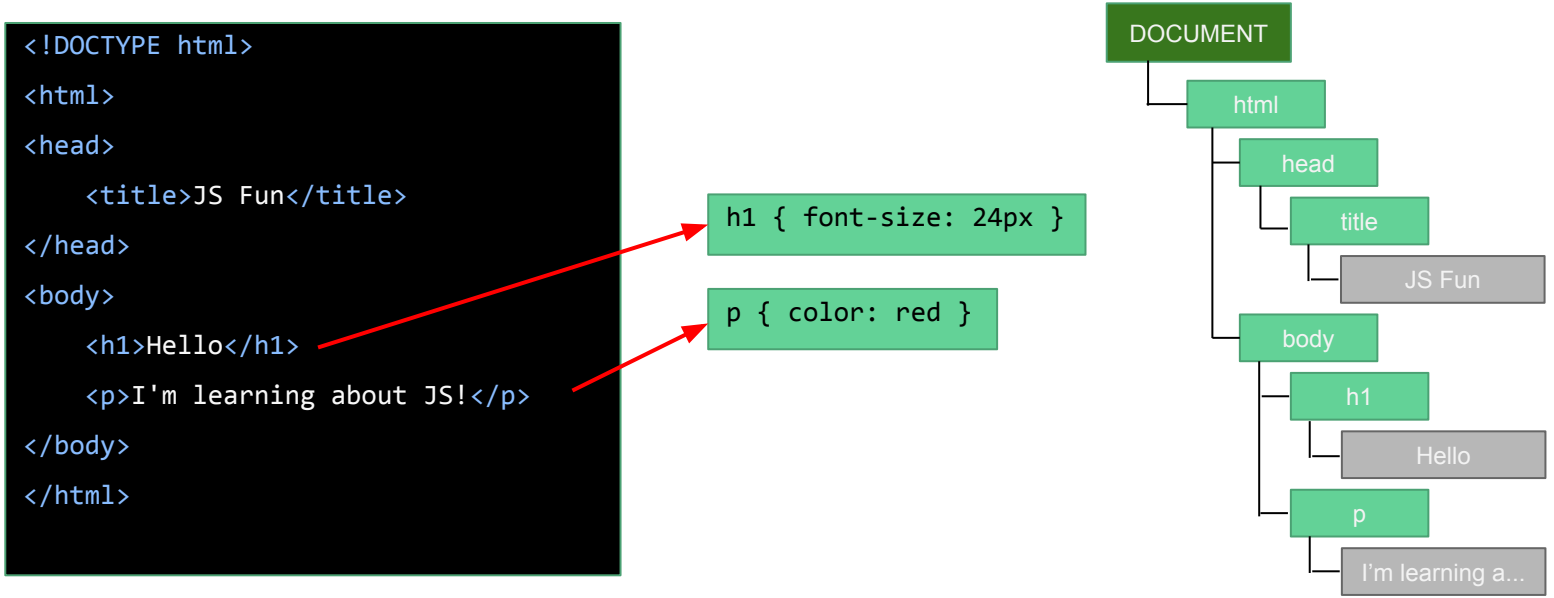
- JavaScript gives us a language to interact with the DOM

```
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



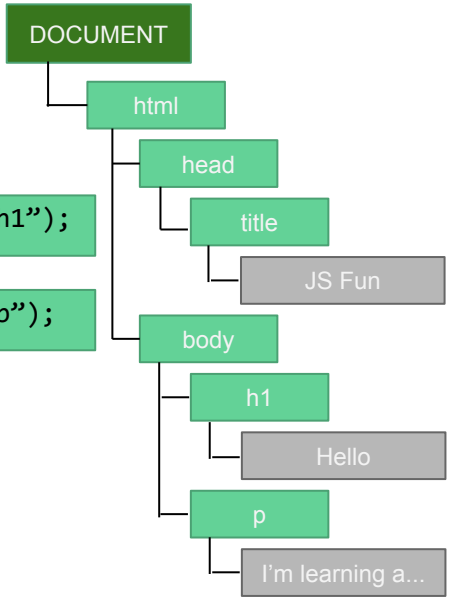
How JS “Finds elements” aka accesses the DOM

- 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 several 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.

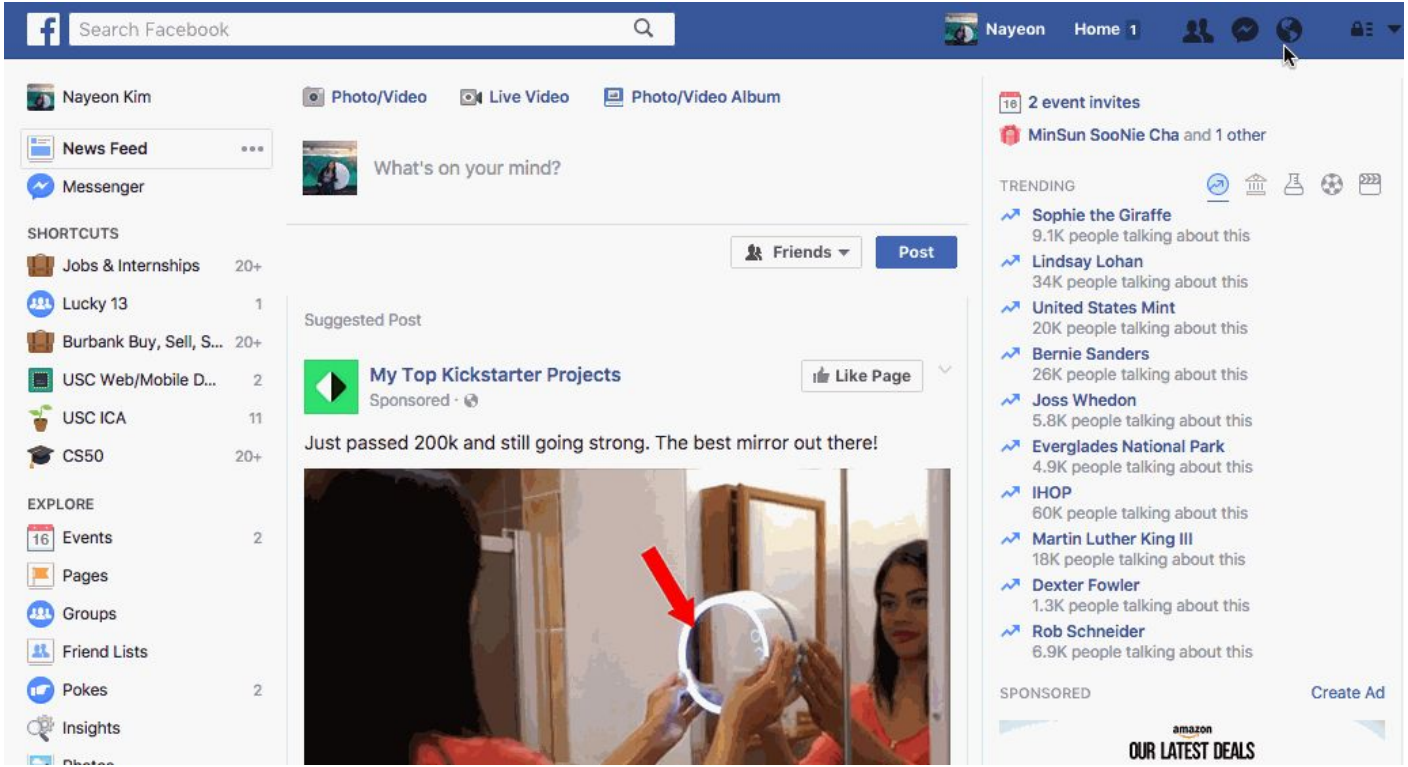
Resource: Full list of DOM Events.

```
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

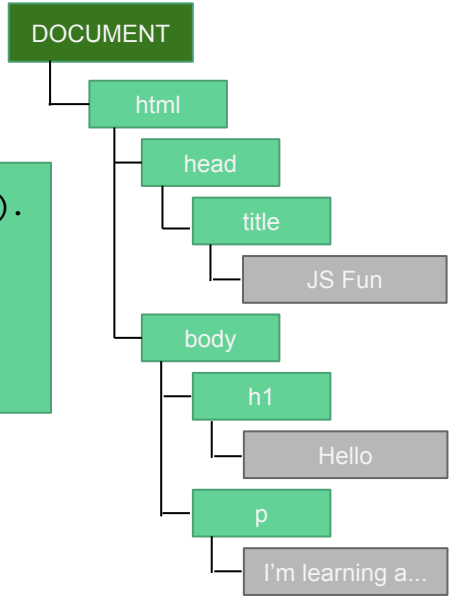


DOM Traversal is possible too

- 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.