GENERAL ASSEMBLY

JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

HELLO!

1. Pull changes from the `svodnik/JS-SF-8-resources` repo to your computer
2. Open the `starter-code` folder in your code editor

Intro to the DOM

LEARNING OBJECTIVES

At the end of this class, you will be able to

- Identify differences between the DOM and HTML.
- Explain and use JavaScript methods for DOM manipulation.

AGENDA

- Intro to the DOM
- DOM manipulation lab

WEEKLY OVERVIEW

WEEK 5	Intro to the DOM / Intro to jQuery
WEEK 6	Ajax & APIs / Asynchronous JavaScript & Callbacks
WEEK 7	Advanced APIs / Project 2 Lab (Feedr)

HOMEWORK — GROUP DISCUSSION

EXERCISE

TYPE OF EXERCISE

- Groups of 3

TIMING

6 min

1. Show off your bot! What can it do?
2. Share a challenge you encountered, and how you overcame it.
3. If you tried something that didn't work, or wanted to add functionality but weren't quite sure how, brainstorm with your group how you might approach it.

HOMEWORK — GROUP DISCUSSION

EXERCISE

TYPE OF EXERCISE

- Groups of 3

TIMING

4 min

1. Share your solutions for the objects homework and for the JSON homework.
2. Share a challenge you encountered, and how you overcame it.
3. Share 1 thing you found challenging. If you worked it out, share how; if not, brainstorm with your group how you might approach it.

EXIT TICKET QUESTIONS

1. The proper way to store an API key locally when working with a public repo - also is Heroku public or private?
2. Didn’t get monkey functions working, and we didn’t go over it.
   I would have liked to get some feedback or see the correct answer for some of the codelabs, though. I got working code, but I don't now if it was good code.
3. I would like to work more on these in examples to really get it and same with hubot.
   I'm wondering about how many codelabs we aren't getting to in class. Is that normal?

EXIT TICKET QUESTIONS (CONTINUED)

4. What are some real world examples of using JSON? I imagine you can be working with massive data structures and I was interested in what that looks like and why it is useful.
5. still a little confused about nested JSON data
6. would prefer individual exercises not group, better to think through on my own

How could you describe the location of the highlighted string “orange” within this HTML code?

```
<html>
  <head>
    <title>Foods</title>
  </head>
  <body>
    <h1><img src=“images/apples.png” alt=“a wood bowl of red apples”></h1>
    <ul class=“foodsList”>
      <li class=“red” id=“mainitem”>apple</li>
      <li class=“orange”>orange</li>
      <li class=“yellow”>banana</li>
    </ul>
  </body>
</html>
```

THE DOCUMENT OBJECT MODEL (DOM)

DOM TREE — HTML FILE

index.html *

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <title>The Evolution of Denim</title>
6  </head>
7  <body>
8
9    <h1>The Evolution of Denim</h1>
10   <p>
11     Chambray retro plaid gentrify letterpress.
       Taxidermy ennui cliche Intelligentsia. Echo
       Park umami authentic before they sold out. <a
       href="https://placekitten.com/">Forage
       wayfarers</a> listicle Kickstarter, Pitchfork
       cray messenger bag fap High Life tilde pug
       Blue Bottle mumblecore.
12   </p>
13   <ul>
14     <li>Dark Wash</li>
15     <li>Stone Wash</li>
16     <li>Chambray</li>
17   </ul>
18
19 </body>
20 </html>
```

DOM TREE

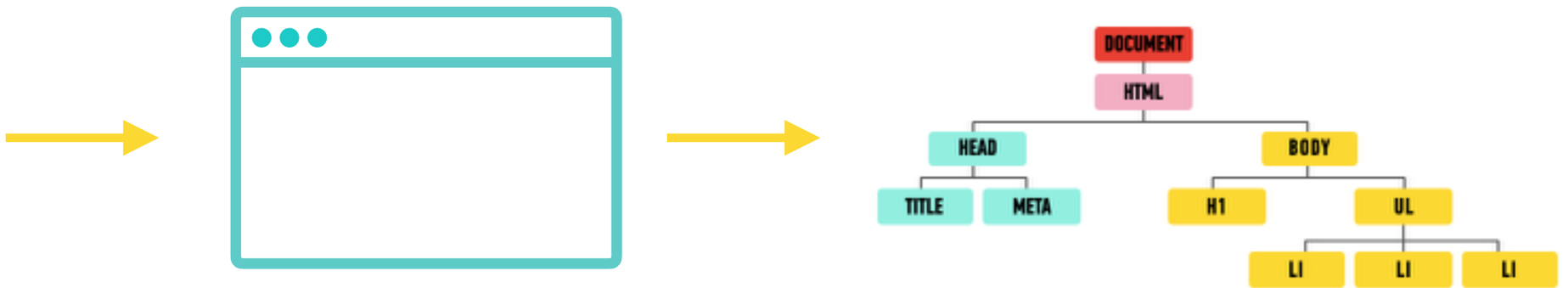
- The browser pulls in this HTML document, analyzes it, and creates an *object model* of the page in memory.
- This model is called the *Document Object Model (DOM)*.
- The DOM is structured like a tree, a DOM Tree, like in the model below:

index.html

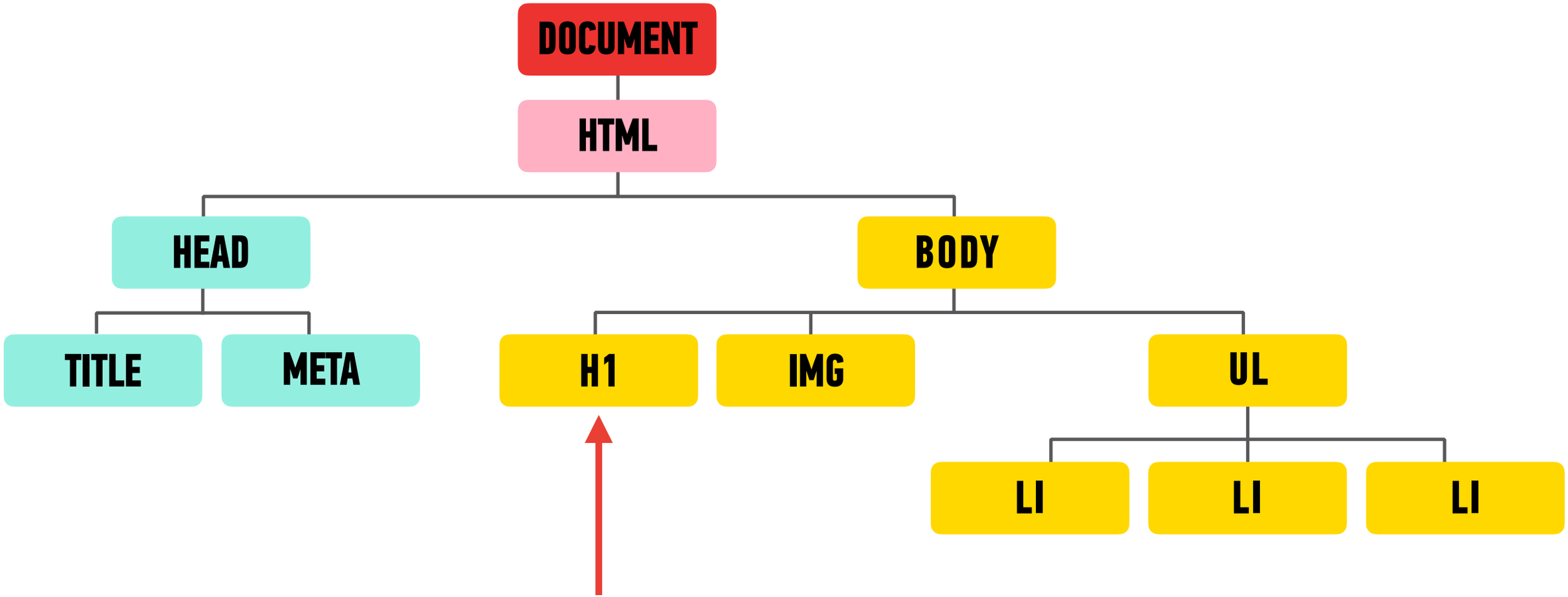
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>The Evolution of Denim</title>
</head>
<body>

  <h1>The Evolution of Denim</h1>
  <p>
    Chambray retro plaid gentrify letterpress.
    Taxidermy ennui cliche Intelligentsia. Echo
    Park umami authentic before they sold out. <a
    href="https://placekitten.com/">Forage
    wayfarers</a> listicle Kickstarter, Pitchfork
    cray messenger bag fap High Life tilde pug
    Blue Bottle mumblecore.
  </p>
  <ul>
    <li>Dark Wash</li>
    <li>Stone Wash</li>
    <li>Chambray</li>
  </ul>

</body>
</html>
```

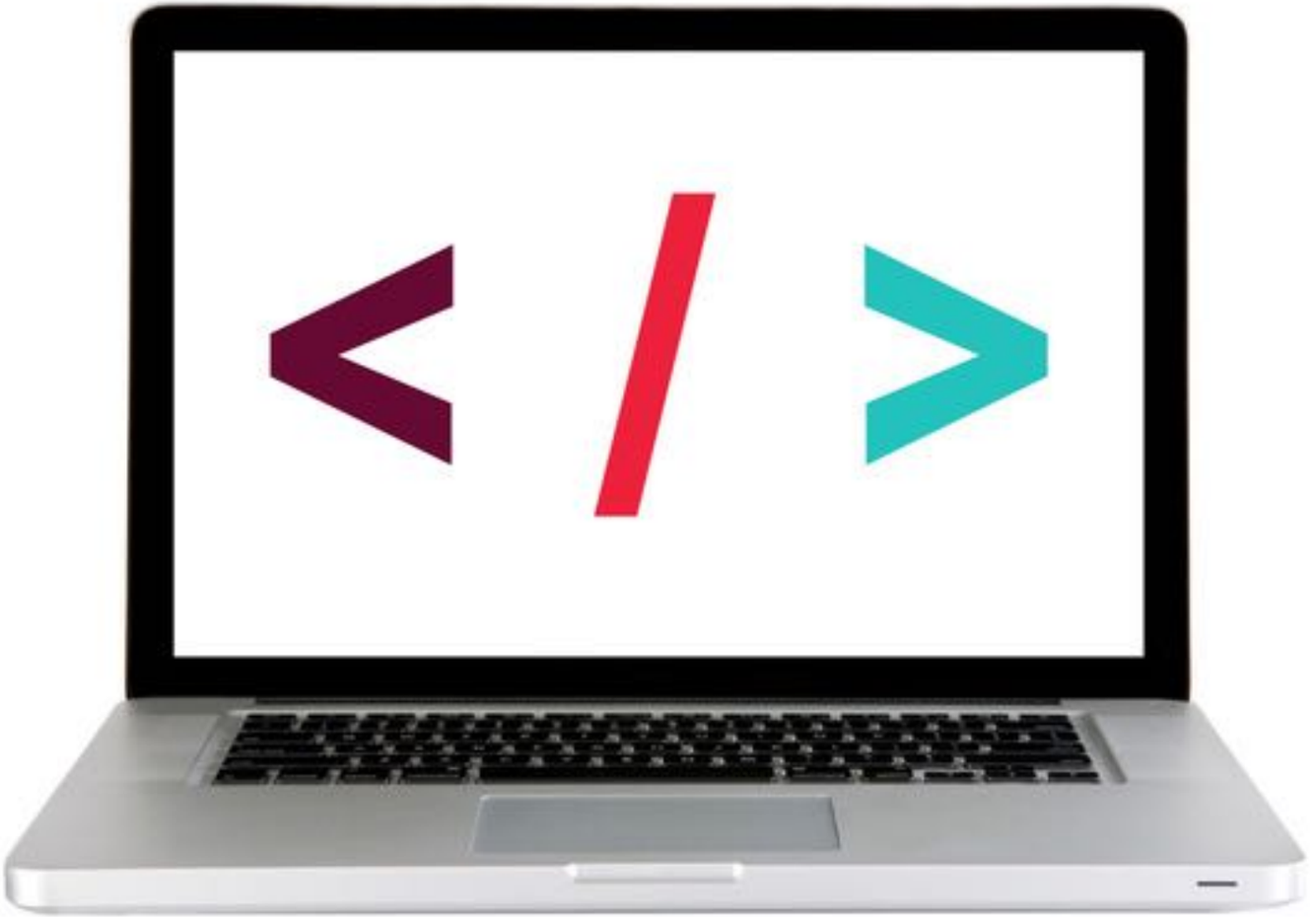


DOM TREE



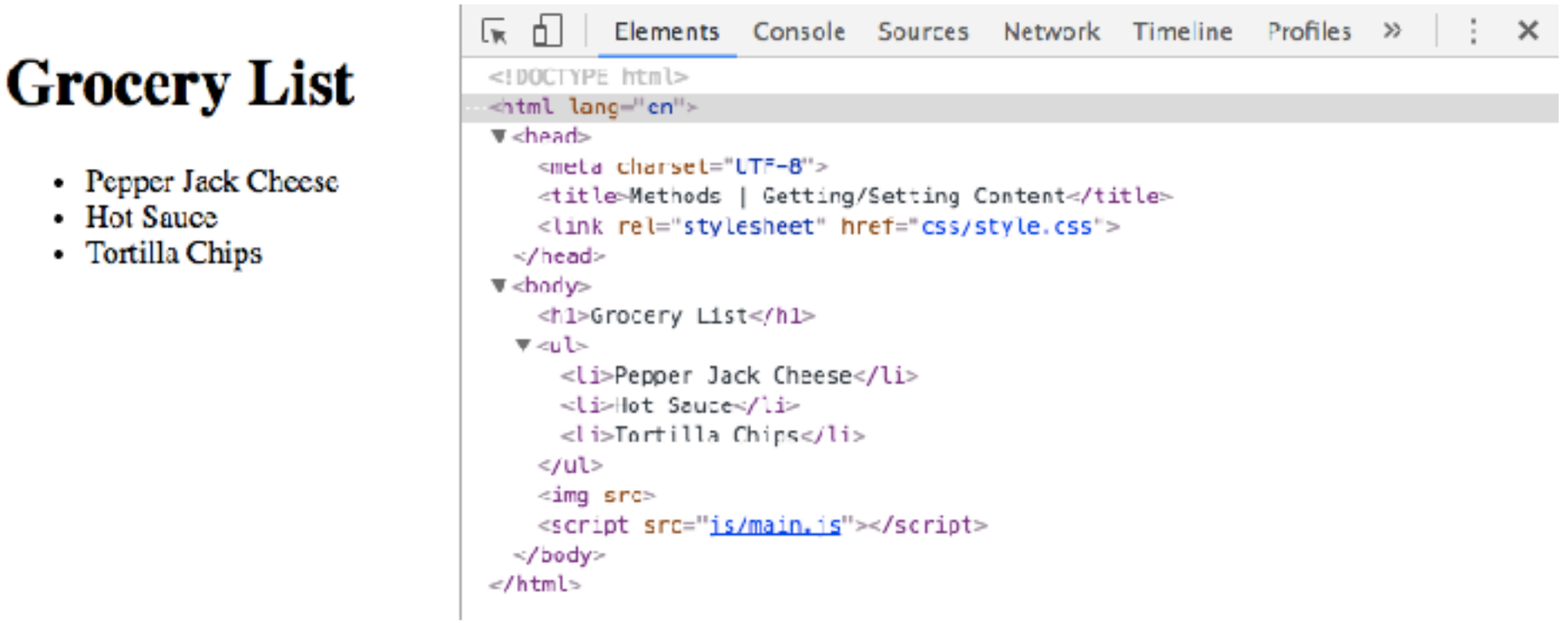
- Each element in the HTML document is represented by a *DOM node*.
- You can think of a node as a live object that you can access and change using JavaScript.
- When the model is updated, those changes are reflected on screen.

LET'S TAKE A LOOK



DOM TREE

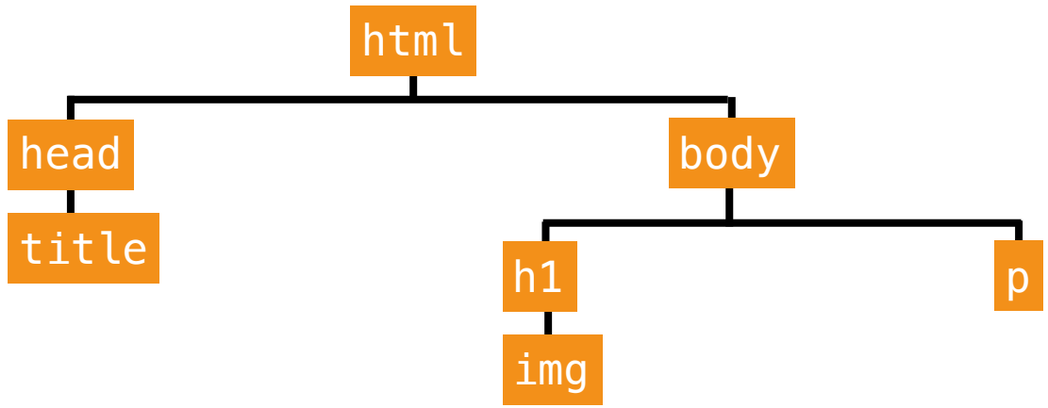
- In Chrome, you can go to View > Developer > Developer Tools and click on the Elements panel to take a look at the DOM tree.



Web page elements

```
<html>
  <head>
    <title>JavaScript Basics</title>
  </head>
  <body>
    <h1>
      <img src="logo.png" alt="JS Basics">
    </h1>
    <p>First, master HTML and CSS.</p>
  </body>
</html>
```

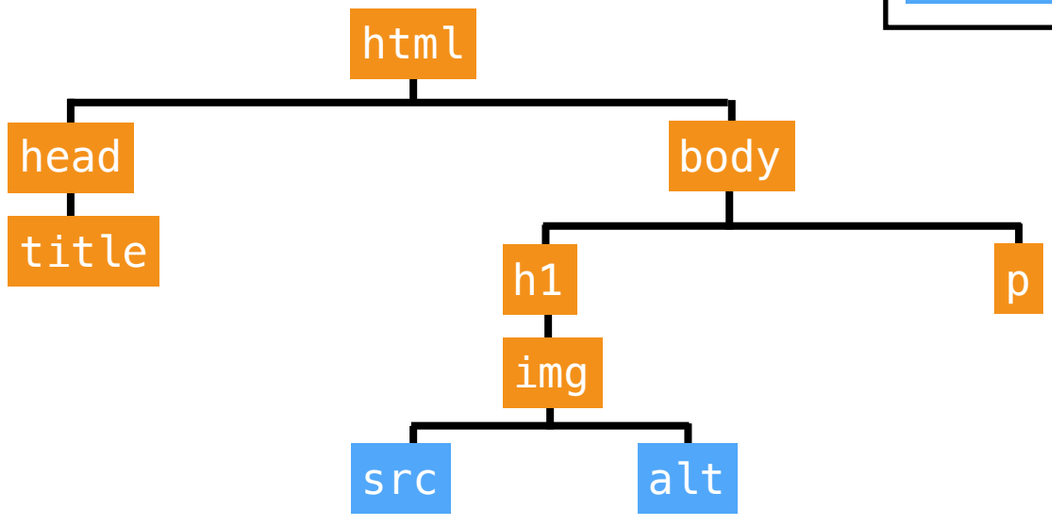
DOM Tree



Web page elements

```
<html>
  <head>
    <title>JavaScript Basics</title>
  </head>
  <body>
    <h1>
      <img src="logo.png" alt="JS Basics">
    </h1>
    <p>First, master HTML and CSS.</p>
  </body>
</html>
```

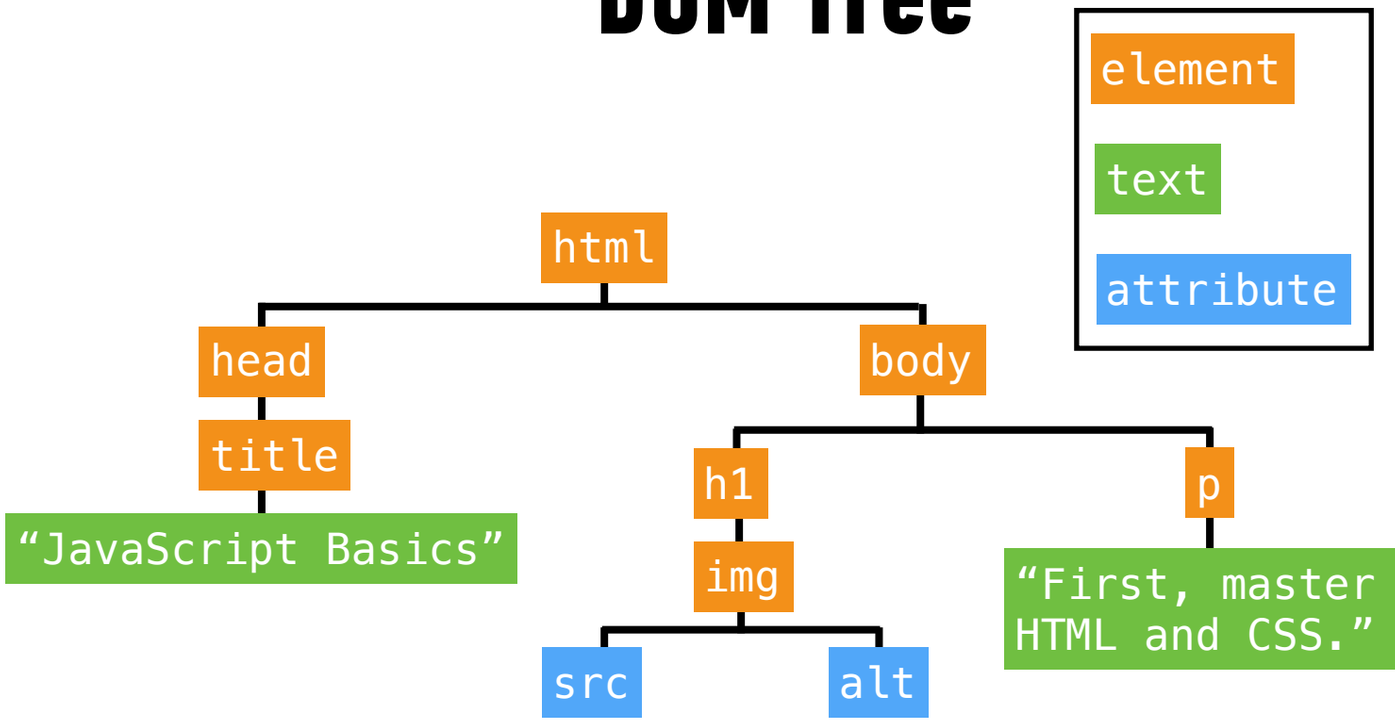
DOM Tree



Web page elements

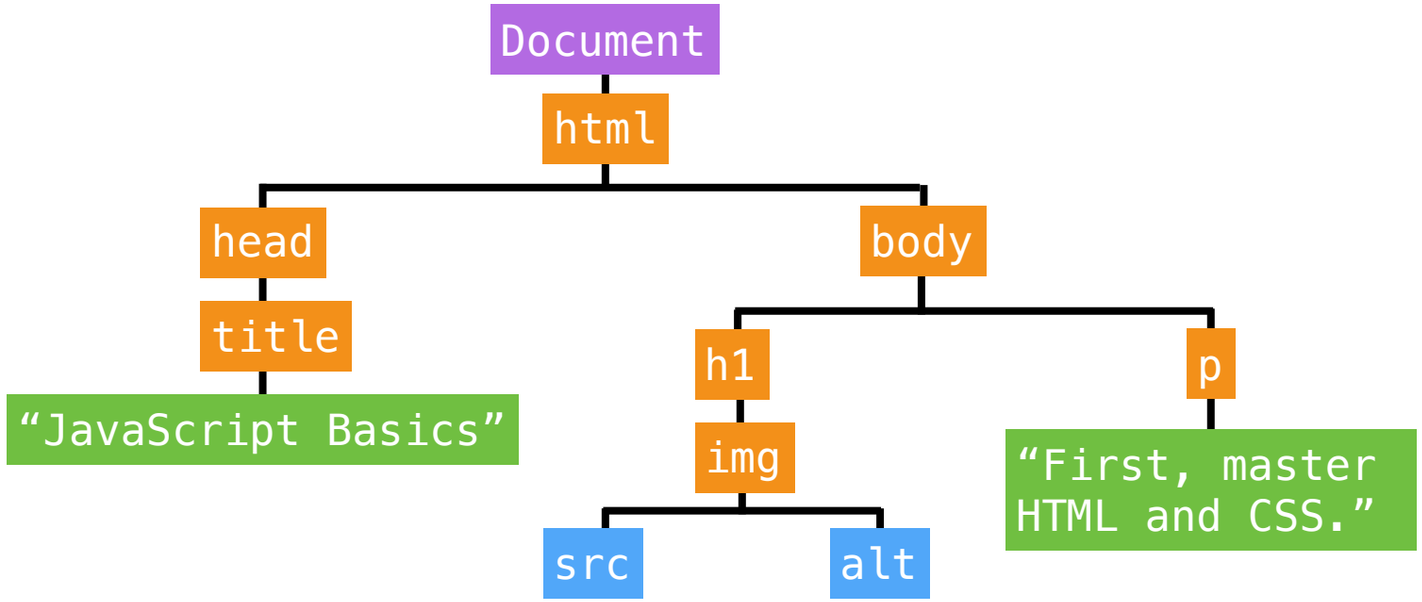
```
<html>
  <head>
    <title>JavaScript Basics</title>
  </head>
  <body>
    <h1>
      <img src="logo.png" alt="JS Basics">
    </h1>
    <p>First, master HTML and CSS.</p>
  </body>
</html>
```

DOM Tree



The `Document` object

- Created by the browser
- Contains all web page elements as descendant objects
- Also includes its own properties and methods



EXERCISE

KEY OBJECTIVE

- Identify differences between the DOM and HTML

TYPE OF EXERCISE

- Pairs

TIMING

2 min

1. How is the DOM different from a page's HTML?

REFERENCING A SCRIPT IN HTML

`script` element at the bottom of the `body` element

just before the closing `</body>` tag

```
<html>
  <head>
  </head>
  <body>
    <h1>JavaScript resources</h1>
    <script src="script.js"></script>
  </body>
</html>
```

Selecting an element in the DOM

- `querySelector()`
- `querySelectorAll()`

querySelector()

- Takes a single argument, a string containing CSS selector

HTML

```
<body>
  …
  <p id="main">Lorem ipsum</p>
  …
</body>
```

JavaScript

```
document.querySelector('#main');
```

querySelector()

- Selects the **first** DOM element that matches the specified CSS selector

```
<body>
  …
  <ul>
    <li>Lorem ipsum</li>
    <li>Lorem ipsum</li>
    <li>Lorem ipsum</li>
  </ul>
…
</body>
```

JavaScript

```
document.querySelector('li');
```

querySelectorAll()

- Takes a single argument, a string containing CSS selector
- Selects all DOM elements that match this CSS selector
- Returns a NodeList, which is similar to an array

```
<body>
  …
  <ul>
    <li>Lorem ipsum</li>
    <li>Lorem ipsum</li>
    <li>Lorem ipsum</li>
  </ul>
…
</body>
```

JavaScript

```
document.querySelectorAll('li');
```

What can we do with a selected element?

- Get and set its text content with the `innerHTML` property
- Get and set its attribute values by referencing them directly (`id`, `src`, etc.)

innerHTML

- Gets the existing content of an element, including any nested HTML tags
- Sets new content in an element

```
var item = document.querySelector('li');

console.log(item.innerHTML) // Gets value: "Lorem ipsum"

item.innerHTML = 'Apples' // Sets value: 'Apples'
```

`className` property

- Gets/sets an element's `class` attribute value
- CSS style sheet contains a style rule for each class
  - » Appearance of element changes based on which class is applied
  - » This is the best practice.

```
var item = document.querySelector('li');

console.log(item.className) // Gets value: "park cf"

item.className = 'park cf expanded'
// Sets value: 'park cf expanded'
```

EXERCISE

LOCATION

▸ `starter-code > 1-dom-exercise`

TIMING

5 min

1. Open index.html in your editor, then scroll to the bottom.
2. Add a reference to the app.js file where indicated, then save your changes.
3. Open app.js in your editor, then follow the instructions.

EXERCISE

LOCATION

▸ `starter-code > 2-dom-attributes-exercise`

TIMING

Review this before going on
Ask students to share answers, question by question.

5 min

1. Open app.js in your editor, then follow the instructions.

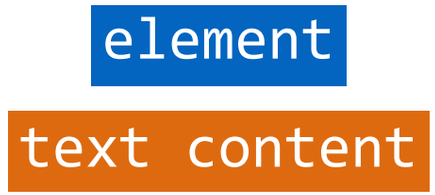
Adding content to the DOM

1. create a new element with `document.createElement()`

`element`

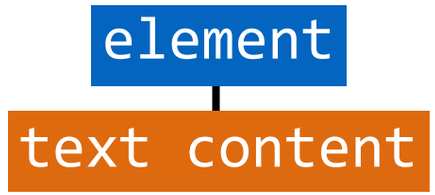
Adding content to the DOM

1. create a new element with `document.createElement()`
2. create new content for that element with `document.createTextNode()`



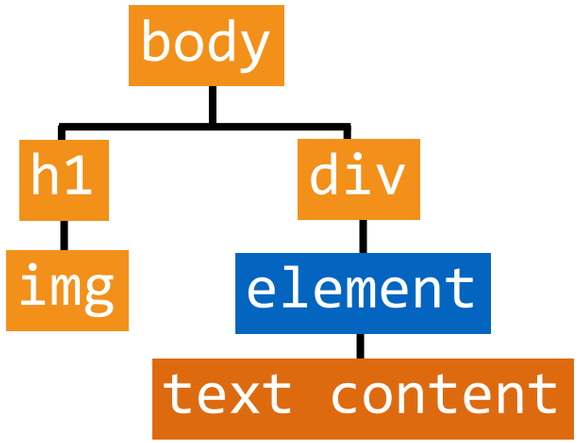
Adding content to the DOM

1. create a new element with `document.createElement()`
2. create new content for that element with `document.createTextNode()`
3. attach the new text content to the new element with `appendChild()`



Adding content to the DOM

1. create a new element with `document.createElement()`
2. create new content for that element with `document.createTextNode()`
3. attach the new text content to the new element with `appendChild()`
4. attach the new element to the DOM with `appendChild()`



createElement()

- Creates a new element

```
document.createElement('li'); // creates an li element
```

- Created element isn't attached to DOM
  - » assign variable when creating so you can reference later

```
let item1 = document.createElement('li');
let item2 = document.createElement('li');
```

createTextNode()

- Creates text content that can be added as the child of another element
- Created text node isn't attached to DOM
  - » assign variable when creating so you can reference later

```
let text1 = document.createTextNode('banana');
let text2 = document.createTextNode('apple');
```

appendChild()

- Attaches element or node as child of specified element
  - » Attaching to an element that's not part of the DOM creates/expands a **document fragment**
- Syntax:
  parent.appendChild(*child*);

```
item1.appendChild(text1);   // adds text1 text to item1 li
item2.appendChild(text2);   // adds text2 text to item2 li
```

appendChild() (continued)

- Attaches element or node as child of specified element
  - » Attaching to a DOM element makes it part of the DOM
- Syntax:
  parent.appendChild(*child*);

```
let list = document.querySelector('ul'); // selects ul element
list.appendChild(item1);     // adds item1 li to list ul
list.appendChild(item2);     // adds item2 li to list ul
```

EXERCISE

KEY OBJECTIVE

- Explain and use JavaScript methods for DOM manipulation.

TYPE OF EXERCISE

- Groups of 3-4

TIMING

2 min

1. Work together to create and complete a list of the four steps in DOM manipulation.
2. For each step in your list, add the method used.

EXERCISE – ADD CONTENT TO A WEB PAGE USING JAVASCRIPT

LOCATION

▸ `starter-code > 3-create-append—exercise`

TIMING

15 min

1. Open preview.png. Your task is to use DOM manipulation to build the sidebar shown in the image and add it to the blog.html web page.
2. Open app.js in your editor, then follow the instructions to create and the “About us” heading and the 2 paragraphs of text to the sidebar.
3. BONUS 1: Open preview-bonus.png, then write JavaScript code to add the image shown to the sidebar. (Filename and location in app.js.)
4. BONUS 2: Create and append the “Recent issues” heading and list.

EVENTS

After we've selected elements, we can use DOM methods to create event listeners

EVENT LISTENERS

selecting element

```
let button = document.querySelector(‘.submitBtn’);
```

element reference

```
button.addEventListener('click', function() {
  // your code here
}, false);
```

EVENT LISTENERS

```
let button = document.querySelector(‘.submitBtn’);
```

method to add event listener

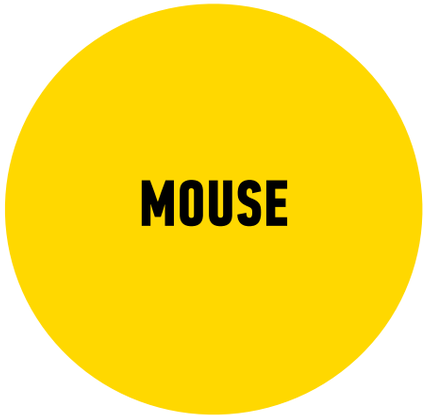
```
button.addEventListener('click', function() {
  // your code here
}, false);
```

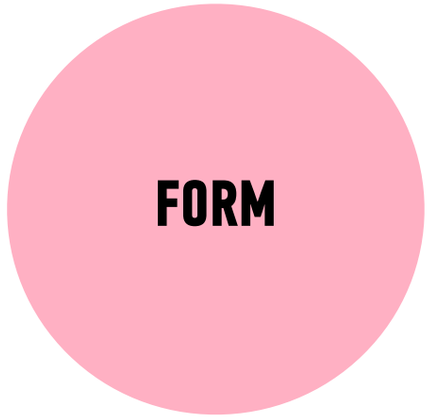
EVENT LISTENERS

```
let button = document.querySelector(‘.submitBtn’);
```

type of event

```
button.addEventListener('click', function() {
  // your code here
}, false);
```





click

dblclick

mouseenter

mouseleave

keypress

keydown

keyup

submit

change

focus

blur

resize

scroll

```
button.addEventListener('eventgoeshere', function() {
  // your code here
}, false);
```

EVENT LISTENERS

```
let button = document.querySelector(‘.submitBtn’);
```

```
button.addEventListener('click', function() {
  // your code here
}, false);
```

function to run when event is triggered

EVENT LISTENERS

```
let button = document.querySelector(‘.submitBtn’);
```

```
button.addEventListener('click', function() {
  // your code here
}, false);
```

final boolean parameter
for backward compatibility

EVENT LISTENERS

element reference

method to add event listener

type of event

```
button.addEventListener('click', function() {
  // your code here
}, false);
```

function to run when event is triggered

final boolean parameter for backward compatibility

ACTIVITY

KEY OBJECTIVE

- Explain and use JavaScript methods for DOM manipulation

TYPE OF EXERCISE

- Individual/Partner

AS A CLASS

10 min

Exercise is in `6-events-exercise` folder

1. Add event listeners to the 3 buttons at the top of the page. Clicking each button should hide the block below it with the corresponding color.
2. Use cheat sheet/slides as a guide for syntax
3. BONUS: Add an event listener for the "Show all blocks" button that removes the hidden class from all the colored block elements.

`preventDefault()`

- Prevents element from executing default behavior in response to an event

Referencing an event

- An object containing information about the triggering event is passed to a function called in response to an event
- Specify a parameter to be able to reference this event in your code
  - » By convention, we use `event`, `evt`, or `e`

```
submitButton.onclick = function(event) {
  event.preventDefault();
  ...
}
```

EXERCISE

LOCATION

▸ `starter-code > 7-js-dom—exercise`

TIMING

until 9:20

1. Open index.html in your browser.
2. Open main.js in your editor, then follow the instructions to make the submit button functional and use DOM manipulation to add items to the list.
3. BONUS: Add functionality that adds a message to the page that alerts the user when they click Submit without typing anything. (Use DOM manipulation, not the `alert` method.)

LEARNING OBJECTIVES - REVIEW

- Identify differences between the DOM and HTML.
- Explain and use JavaScript methods for DOM manipulation.

NEXT CLASS PREVIEW

Intro to jQuery

- Manipulate the DOM by using jQuery selectors and functions.
- Register and trigger event handlers for jQuery events.
- Use event delegation to manage dynamic content.
- Use implicit iteration to update elements of a jQuery selection, and use chaining to place methods on selectors.
- Use ES6 template literals for more abstracted content manipulation.

Exit Tickets!

Q&A