

JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

HELLO!

1. Pull changes from the `svodnik/JS-SF-7` repo to your computer
2. Open the `starter-code` folder in your code editor

JAVASCRIPT DEVELOPMENT

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

Checkin and questions

- The **most significant thing I've learned** about objects and JSON is _____.
- My **biggest outstanding question** about objects and JSON is _____.

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

```
<html>
  <head>
    <title>Foods</title>
  </head>
  <body>
    <h1></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)

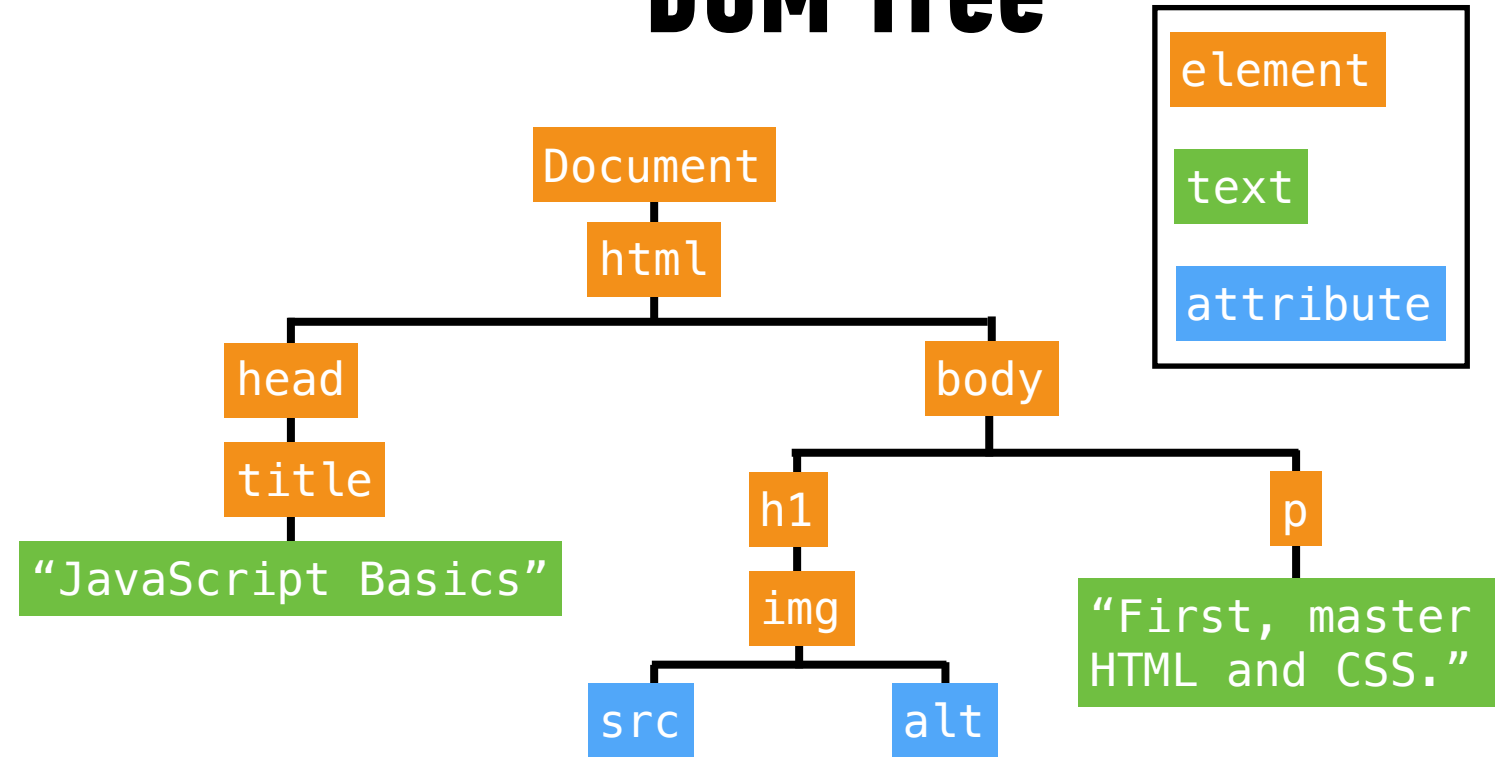
DOCUMENT OBJECT MODEL (DOM)

- The **Document Object Model (DOM)** is a way of representing the contents of a web page.
- The DOM lets you access the structure of a web page, and change the structure, style, and content of the page.
- The DOM breaks up a page into objects and nodes, which have properties and methods.
- At its core, the DOM is a way of accessing web pages with scripts.

HTML code

```
<html>
  <head>
    <title>JavaScript Basics</title>
  </head>
  <body>
    <h1>
      
    </h1>
    <p>First, master HTML and CSS.</p>
  </body>
</html>
```

DOM Tree



The Document object

- Created by the browser
- Contains properties and methods of the web page content
- Also contains all web page elements as descendant objects

EXERCISE



EXERCISE

KEY OBJECTIVE

- Identify differences between the DOM and HTML

TYPE OF EXERCISE

- Pairs

TIMING

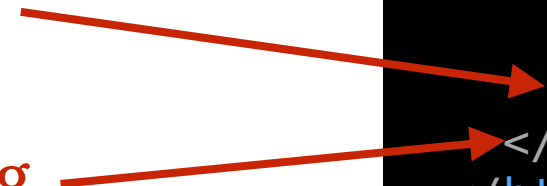
2 min

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>
```

- For DOM manipulation, always place the `script` element at the bottom of the `body` element, just before the closing `</body>` tag
- This ensures that all the content has loaded before we start loading scripts, showing users content as quickly as possible

Selecting an element in the DOM

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

getElementById()

- Takes a single argument, a string containing an ID value
- Selects the DOM element whose id attribute has this value

HTML

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

JavaScript

```
document.getElementById('main');
```

querySelector()

- Takes a single argument, a string containing CSS selector
- Selects the **first** DOM element that matches this 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`, `class`, `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



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.

Adding content to the DOM

1. create a new element
2. create new content for that element
3. add the new content to the element
4. add the new element containing the content to the DOM itself

createElement()

- Creates a new element

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

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

```
var list = document.createElement('ul');  
var item1 = document.createElement('li');  
var 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

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


appendChild()

- Attaches element or node as child of specified element
 - » Attaching to a DOM element makes it part of the DOM
 - » 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
list.appendChild(item1);  // adds item1 li to list ul
list.appendChild(item2);  // adds item2 li to list ul
document.appendChild(list); // adds list ul to document
```

EXERCISE



EXERCISE

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 append the sidebar, the “About us” heading, 2 paragraphs, and image.
3. BONUS: Create and append the “Recent issues” heading and list.

EVENTS

- Commonly used events
 - » load
 - » click
- full list at <https://developer.mozilla.org/en-US/docs/Web/Events>

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



EXERCISE

LOCATION

► starter-code > 5-js-dom-exercise

TIMING

30 min

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.)

INTRO TO THE DOM

Intro to jQuery

WHAT IS JQUERY?

- jQuery is a JavaScript library.
- It provides an application programming interface (API)
- Makes it easier to write code to select elements, manipulate the DOM, and perform other tasks.
- jQuery is the most widely used JavaScript library on the web

HOW IS JQUERY USEFUL?

- jQuery lets us select DOM elements using CSS syntax
- We can do this with JavaScript using the following methods:
 - `querySelector()`
 - `querySelectorAll()`
- But jQuery syntax is less verbose

SELECTING THE FIRST `` IN A LIST

```
<ul class="foodsList">
  <li>apple</li>
  <li>orange</li>
  <li>banana</li>
</ul>
```

```
var firstItem = document.querySelector(".foodsList li:first-child");
```

vanilla JS

```
var firstItem = jQuery(".foodsList li:first-child");
```

← shorter!

jQuery

JQUERY OBJECTS

- Selecting elements with jQuery returns a jQuery object
- This is essentially the results of our search, wrapped in an object with a set of properties and methods, which let you more easily
 - change styling
 - add event listeners for specific events
 - write brand new content into the web page
- You can also work with the results using array notation

HOW DO WE USE JQUERY?

- jQuery is stored in an external .js file just like any other JavaScript code we include in our web pages
- We reference it in a `script` element in our HTML, such as

```
<script src="http://code.jquery.com/jquery-3.1.0.min.js"></script>
```

REFERENCE JQUERY FILE BEFORE YOUR OWN FILES!

- The jQuery code must be loaded before any other .js files that use jQuery syntax; otherwise, they will not work.

REFERENCING THE JQUERY LIBRARY

the jQuery file is loaded first, telling the browser how to interpret jQuery syntax

our JavaScript file uses jQuery syntax, so we load it after the jQuery JavaScript file

```
<html>
  <head>
  </head>
  <body>
    <h1>JavaScript resources</h1>
    <script src="jquery-3.1.0.js"></script>
    <script src="script.js"></script>
  </body>
</html>
```

SELECTING AN ELEMENT WITH JQUERY

- Simply specify the CSS selector within `$ ()`

```
$('#item') // select item by id
```

```
$('.open') // select item(s) by class
```

```
$('h2') // select item(s) by tag
```

CREATING VARIABLES WITH JQUERY

```
var $openTab = $( '.open' );
```

- › Best practice: include \$ as the first character of any variable defined using jQuery code
- › This is not required by jQuery, but helps us keep track of what parts of our code rely on the jQuery library

CREATE A NEW ELEMENT WITH JQUERY

```
var $summary = $( '<p>' );
```

- Even when an element takes a closing tag, we only have to specify the opening tag to create an instance of that element with jQuery

CREATE AND APPEND A TEXT NODE

▸ `.text()`

```
$summary.text("It all comes down to this.");
```

APPEND AN ELEMENT

▸ `.append()`

```
$section1.append($summary);
```

RUN CODE ONLY AFTER THE DOM HAS LOADED

```
$(document).ready(function() {  
    ...  
})
```

- More commonly used shorthand version:

```
$(function() {  
    ...  
})
```

SPECIFY AN EVENT HANDLER

▸ `.on()`

```
$button.on('click', function(event) {  
  
});
```

▸ more at <https://learn.jquery.com/events/>

WORK WITH CSS STYLES

- jQuery lets us access and change specific CSS styles for an element.
- However, the best practice is to do all work with CSS in the stylesheet (.css file) and use jQuery only to change the class value assigned to an element with `.addClass()` and `.removeClass()`

```
$phoneBox.addClass('placeholderText');  
$phoneBox.removeClass('placeholderText');
```

CHANGE THE HTML CONTENT OF AN ELEMENT

▸ `.html()`

```
$summary.html('Everything you need to know.');
```

```
$sidebar.html('<img src="sparrow.jpg" alt="a  
sparrow"');
```

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.
- Implement advanced 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.

Exit Tickets!

Q&A