

JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

HELLO!

1. Pull changes from the `vodnik/JS-SF-15-resources` repo to your computer
2. Open the `06-json-dom` folder in your editor

JAVASCRIPT DEVELOPMENT

JSON & INTRO TO THE DOM

LEARNING OBJECTIVES

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

- Implement and work with JSON data
- Identify differences between the DOM and HTML.
- Use vanilla JavaScript methods and properties to create and modify DOM nodes.

AGENDA

- JSON
- Lab: Work with JSON
- Intro to the DOM
- Getting and setting DOM elements

JSON & INTRO TO THE DOM

WEEKLY OVERVIEW

WEEK 4

JSON & Intro to DOM / Events & jQuery

WEEK 5

Advanced jQuery / Ajax & APIs

WEEK 6

Asynchronous JS & callbacks / Advanced APIs

EXIT TICKET QUESTIONS

SLACK BOT LAB

HOMEWORK REVIEW

HOMEWORK — GROUP DISCUSSION



EXERCISE

TYPE OF EXERCISE

- Groups of 2-3

TIMING

4 min

1. Share your solutions to the homework.
2. Share a challenge you encountered, and how you overcame it.

HOMEWORK — GROUP DISCUSSION



EXERCISE

TYPE OF EXERCISE

- Groups of 2-3

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.

JSON

JSON IS A DATA FORMAT BASED ON JAVASCRIPT

object

```
let instructor = {  
  firstName: 'Sasha',  
  lastName: 'Vodnik',  
  city: 'San Francisco',  
  classes: [  
    'JSD', 'FWD'  
  ],  
  classroom: 3,  
  launched: true,  
  dates: {  
    start: 20190611,  
    end: 20190814,  
  },  
};
```

JSON

```
{  
  "firstName": "Sasha",  
  "lastName": "Vodnik",  
  "city": "San Francisco",  
  "classes": [  
    "JSD", "FWD"  
  ],  
  "classroom": 3,  
  "launched": true,  
  "dates": {  
    "start": 20190611,  
    "end": 20190814  
  }  
}
```

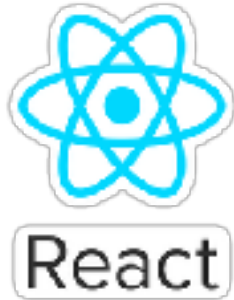
JSON

- Easy for humans to read and write
- Easy for programs to parse and generate

```
{  
  "firstName": "Sasha",  
  "lastName": "Vodnik",  
  "city": "San Francisco",  
  "classes": [  
    "JSD", "FWD"  
  ],  
  "classroom": 3,  
  "launched": true,  
  "dates": {  
    "start": 20190611,  
    "end": 20190814  
  }  
}
```

JSON IS NOT JAVASCRIPT-SPECIFIC

- Used across the web by programs written in many languages



ANGULARJS



JSON IS EVERYWHERE!

10 Going • 79 Interested

Share this event with your friends

Invite

Response contained invalid JSON. Reason: JSON Parse error: Unexpected identifier 'for' for (,);({ '__ar':1,'error':1357004,'errorSummary':'Sorry, something went wrong','errorDescription':'Please try closing and re-opening your browser window','payload':null,'bootloadable':{,'ixData':{,'gkxData':{,'lid':'6537387516854408944'})



Share in Messenger

To: Choose friends

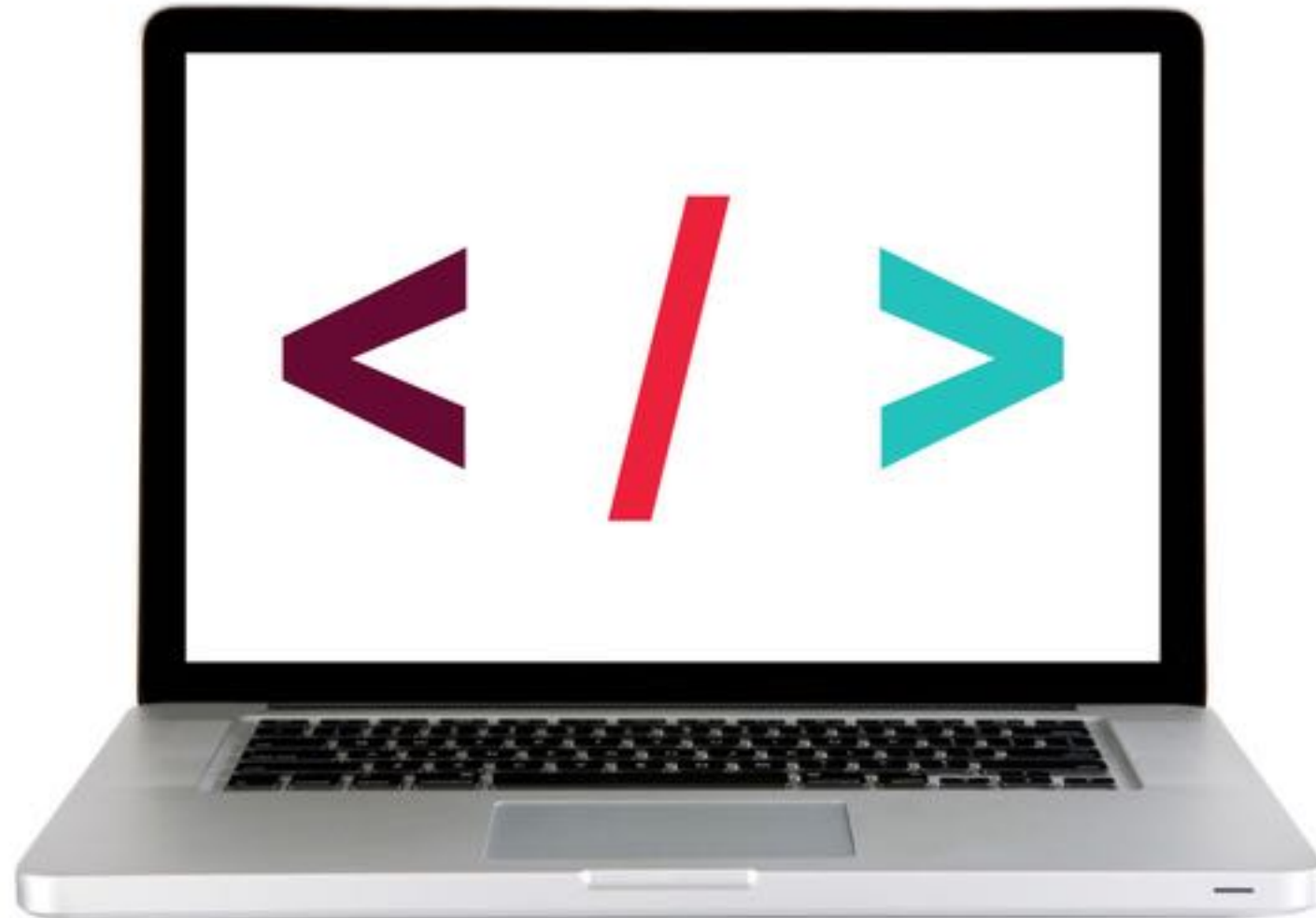
Add a message...

Response contained invalid JSON. Reason: JSON Parse error: Unexpected identifier 'for' for (,);({ '__ar':1,'error':1357004,'errorSummary':'Sorry, something went wrong','errorDescription':'Please try closing and re-opening your browser window','payload':null,'bootloadable':{,'ixData':{,'gkxData':{,'lid':'6537387517222066818'})

Response contained invalid JSON. Reason: JSON Parse error: Unexpected identifier 'for' for (,);({ '__ar':1,'error':1357004,'errorSummary':'Sorry, something went wrong','errorDescription':'Please try closing and re-opening your browser window','payload':null,'bootloadable':{,'ixData':{,'gkxData':{,'lid':'6537387515791219178'})

Response contained invalid JSON. Reason: JSON Parse error: Unexpected identifier 'for' for (,);({ '__ar':1,'error':1357004,'errorSummary':'Sorry, something went wrong','errorDescription':'Please try closing and re-opening your browser window','payload':null,'bootloadable':{,'ixData':{,'gkxData':{,'lid':'6537387516228648313'})

LET'S TAKE A LOOK



JSON RULES

- Property names must be double-quoted strings.
- Trailing commas are forbidden.
- Leading zeroes are prohibited.
- In numbers, a decimal point must be followed by at least one digit.
- Most characters are allowed in strings; however, certain characters (such as ' , " , \ , and newline/tab) must be 'escaped' with a preceding backslash (\) in order to be read as characters (as opposed to JSON control code).
- All strings must be double-quoted.
- No comments!

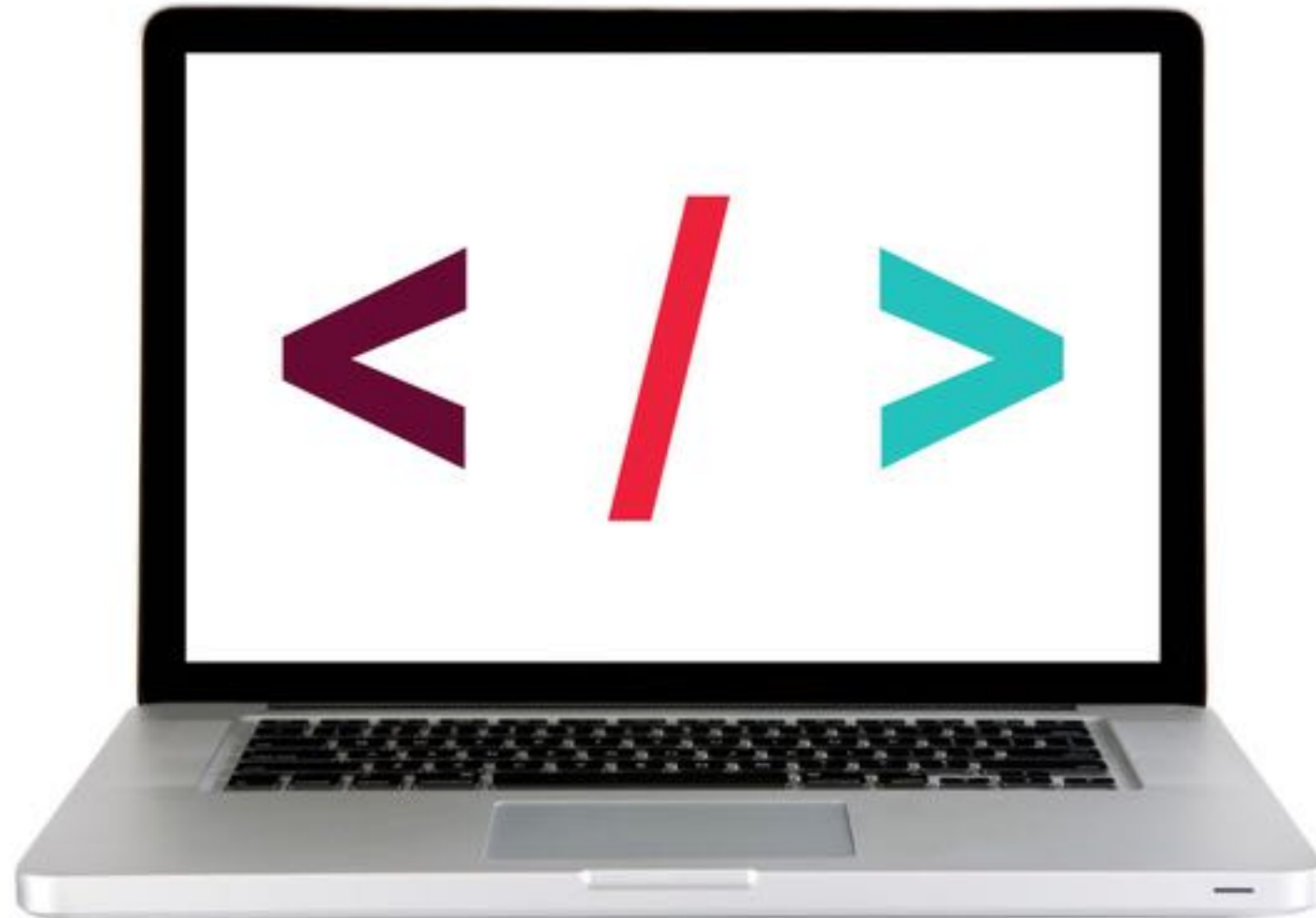
TO CONVERT AN OBJECT TO JSON

```
JSON.stringify(object);
```

TO CONVERT JSON TO AN OBJECT

```
JSON.parse(json);
```

LET'S TAKE A LOOK



EXERCISE — JSON



EXERCISE

KEY OBJECTIVE

- Implement and work with JSON data

TYPE OF EXERCISE

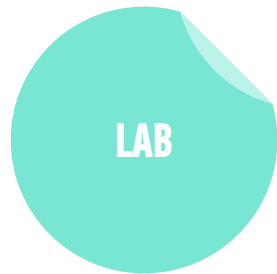
- Pairs

TIMING

3 min

1. Write JSON code that contains an error.
2. Write your code on the wall.
3. When everyone's code is done, we will look at the code together as a class and practice identifying errors.

LAB — JSON



KEY OBJECTIVE

- Implement and work with JSON data

TYPE OF EXERCISE

- Individual or pair

TIMING

10 min

1. Open `starter-code > 1-json-exercise > app.js` in your editor.
2. Follow the instructions to write code that produces the stated output.

WORKING WITH NESTED DATA STRUCTURES

YAY, I GOT SOME DATA!

```
let person = '{"firstName":  
"Sasha","lastName": "Vodnik","city":  
"San Francisco","classes": ["JSD",  
"FWD"],"classroom": 3,"launched":  
true,"dates": {"start": 20190611,"end":  
20190814}}';
```

WAIT, WHAT?!

WORKING WITH NESTED DATA STRUCTURES

1. PARSE THE JSON TO A JAVASCRIPT OBJECT (OR ARRAY!)

2. VIEW THE RESULTING DATA STRUCTURE

3. LOCATE THE DATA YOU WANT TO REFERENCE

4. IDENTIFY THE DATA TYPE OF THE TOP LEVEL, THEN WRITE CODE TO REFERENCE IT

5. IF NECESSARY, MOVE DOWN A LEVEL, THEN REPEAT PREVIOUS STEP

WORKING WITH NESTED DATA STRUCTURES

1. PARSE THE JSON TO A JAVASCRIPT OBJECT (OR ARRAY!)

```
let person = '{"firstName":  
"Sasha","lastName": "Vodnik","city":  
"San Francisco","classes": ["JSD",  
"FWD"],"classroom": 3,"launched":  
true,"dates": {"start": 20190611,"end":  
20190814}}';
```



```
let personObject = JSON.parse(person);
```

WORKING WITH NESTED DATA STRUCTURES

2. VIEW THE RESULTING DATA STRUCTURE

```
let personObject = JSON.parse(person);  
console.log(personObject);  
>
```



```
city: "San Francisco"  
▼ classes: Array(2)  
  0: "JSD"  
  1: "FEWD"  
  length: 2  
  ► __proto__: Array(0)  
classroom: 8  
▼ dates:  
  end: 20171113  
  start: 20170906  
  ► __proto__: Object  
firstName: "Sasha"  
lastName: "Vodnik"  
launched: true
```

WORKING WITH NESTED DATA STRUCTURES

3. LOCATE THE DATA YOU WANT TO REFERENCE


```
city: "San Francisco"
▼ classes: Array(2)
  0: "JSD"
  1: "FEWD"
  length: 2
  ► __proto__: Array(0)
classroom: 8
▼ dates:
  end: 20171113
  start: 20170906
  ► __proto__: Object
firstName: "Sasha"
lastName: "Vodnik"
launched: true
```

WORKING WITH NESTED DATA STRUCTURES

4. IDENTIFY THE DATA TYPE OF THE TOP LEVEL, THEN WRITE CODE TO REFERENCE IT

direct property:

```
console.log(personObject.city);  
> "San Francisco"
```



```
city: "San Francisco"  
▼ classes: Array(2)  
  0: "JSD"  
  1: "FEWD"  
  length: 2  
  ► __proto__: Array(0)  
classroom: 8  
▼ dates:  
  end: 20171113  
  start: 20170906  
  ► __proto__: Object  
firstName: "Sasha"  
lastName: "Vodnik"  
launched: true
```

WORKING WITH NESTED DATA STRUCTURES

4. IDENTIFY THE DATA TYPE OF THE TOP LEVEL, THEN WRITE CODE TO REFERENCE IT

5. IF NECESSARY, MOVE DOWN A LEVEL, THEN REPEAT PREVIOUS STEP

```
city: "San Francisco"
▼ classes: Array(2)
  0: "JSD"
  1: "FEWD"
  length: 2
  ► __proto__: Array(0)
classroom: 8
▼ dates:
  end: 20171113
  start: 20170906
  ► __proto__: Object
firstName: "Sasha"
lastName: "Vodnik"
launched: true
```

direct property > array element

```
console.log(personObject.classes);
> ["JSD", "FEWD"]
```

```
console.log(personObject.classes[0]);
> "JSD"
```

WORKING WITH NESTED DATA STRUCTURES

4. IDENTIFY THE DATA TYPE OF THE TOP LEVEL, THEN WRITE CODE TO REFERENCE IT

5. IF NECESSARY, MOVE DOWN A LEVEL, THEN REPEAT PREVIOUS STEP

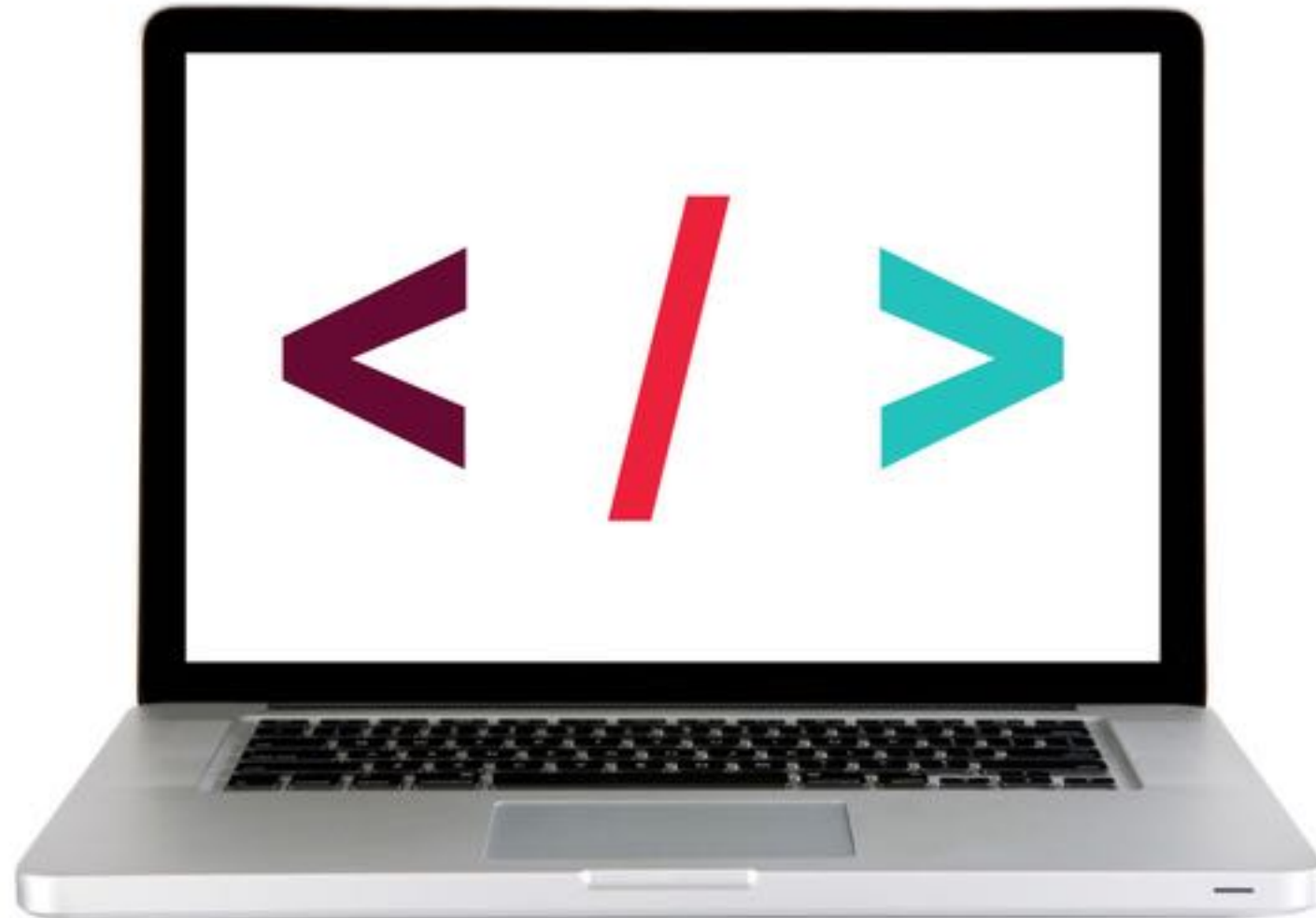
```
city: "San Francisco"
▼ classes: Array(2)
  0: "JSD"
  1: "FEWD"
  length: 2
  ► __proto__: Array(0)
classroom: 8
▼ dates:
  end: 20171113
  start: 20170906
  ► __proto__: Object
firstName: "Sasha"
lastName: "Vodnik"
launched: true
```

direct property > nested object property

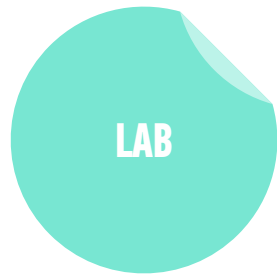
```
console.log(personObject.dates);
> {end:20171113,start:20170906}
```

```
console.log(personObject.dates.start);
> 20170906
```

LET'S TAKE A LOOK



LAB — JSON



KEY OBJECTIVE

- Implement and work with JSON data

TYPE OF EXERCISE

- Individual or pair

TIMING

5 min

1. Open `starter-code > 2-data-structure-exercise > app.js` in your editor.
2. Follow the instructions to write code that produces the stated output.

THE DOCUMENT OBJECT MODEL (DOM)

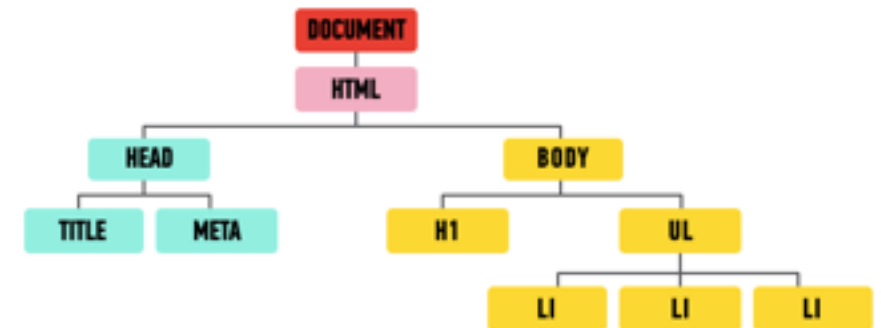
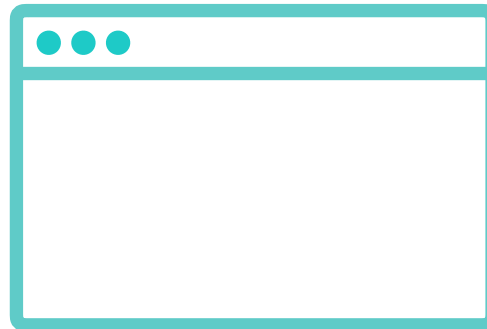
DOM TREE — HTML FILE

```
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>
```

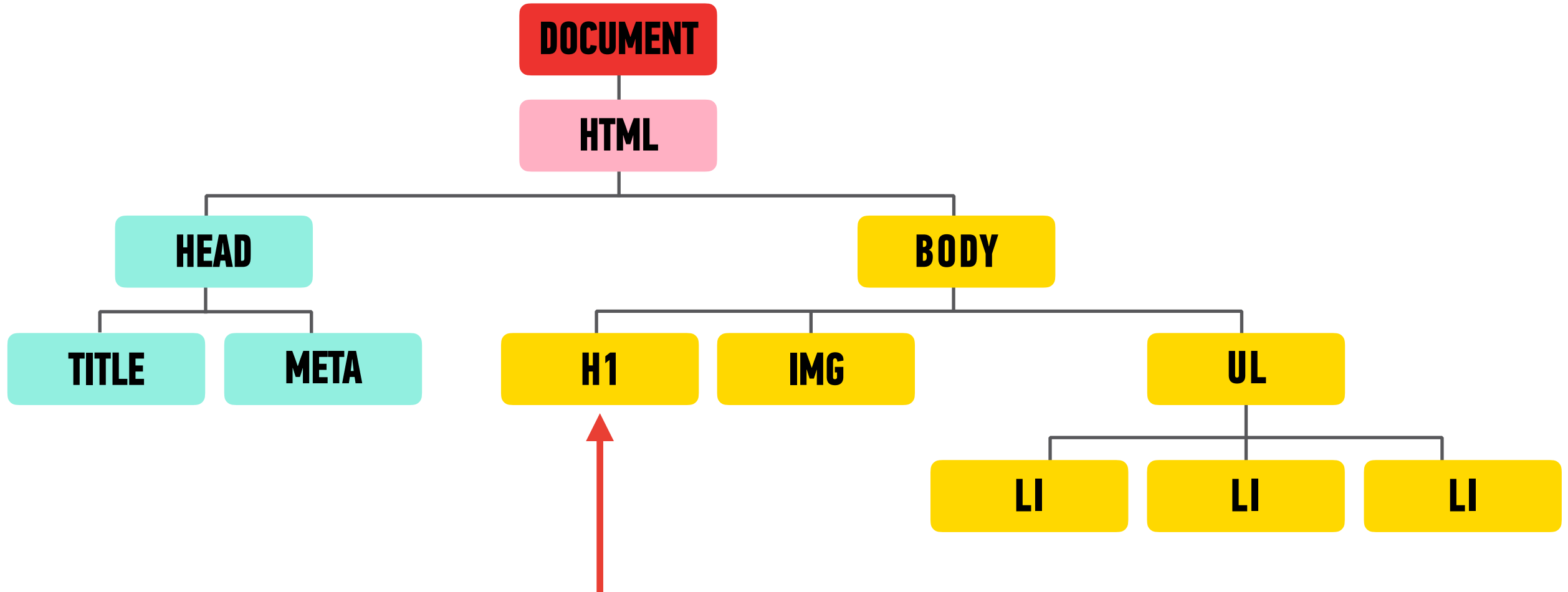
DOM TREE

- ▶ 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
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   <h1>The Evolution of Denim</h1>
9   <p>
10    Chambray retro plaid gentrify letterpress.
11    Taxidermy ennui cliche Intelligentsia. Echo
12    Park umami authentic before they sold out. <a
13    href="https://placekitten.com/">Forage
14    wayfarers</a> listicle Kickstarter, Pitchfork
15    cray messenger bag tap High Life tilde pug
16    Blue Bottle mumblecore.
17  </p>
18  <ul>
19    <li>Dark Wash</li>
20    <li>Stone Wash</li>
21    <li>Chambray</li>
22  </ul>
23 </body>
24 </html>
```



DOM TREE



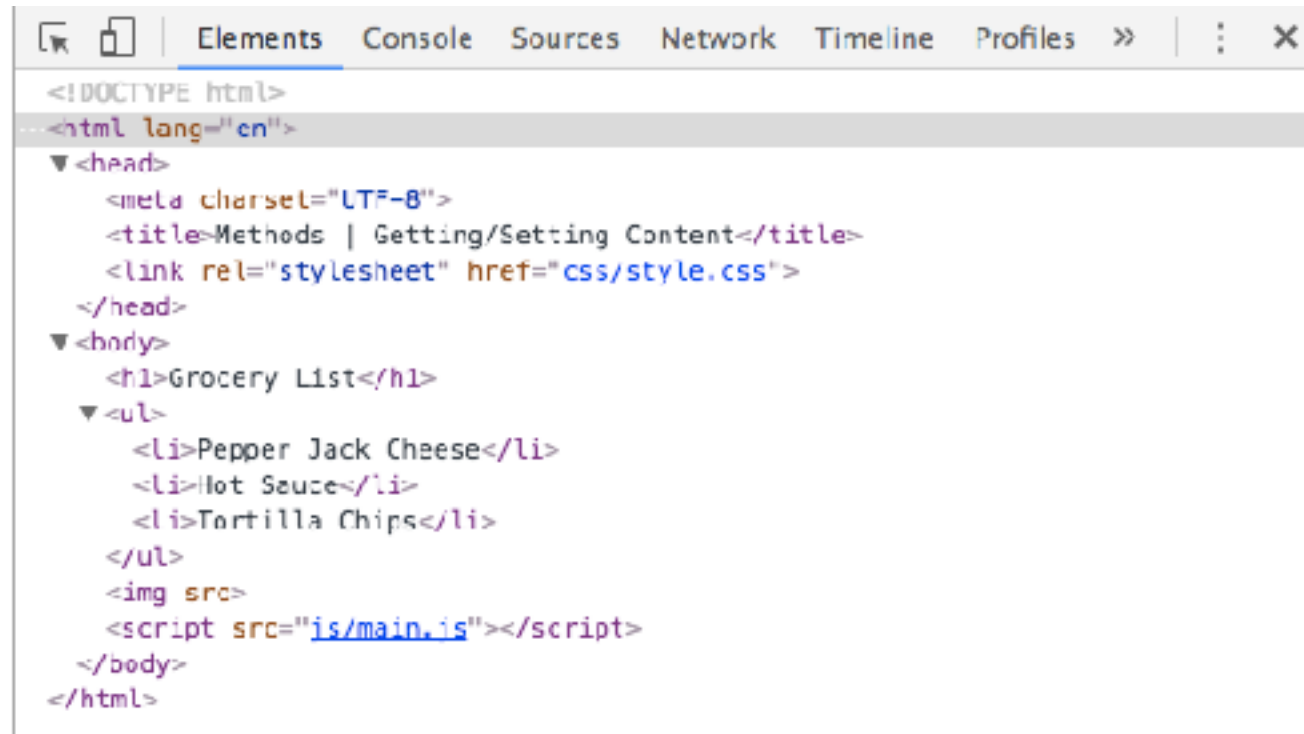
- ▶ Each element in the HTML document is represented by a *DOM node*.
- ▶ You can think of a node as a live object that you can access and change using JavaScript.
- ▶ When the model is updated, those changes are reflected on screen.

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.

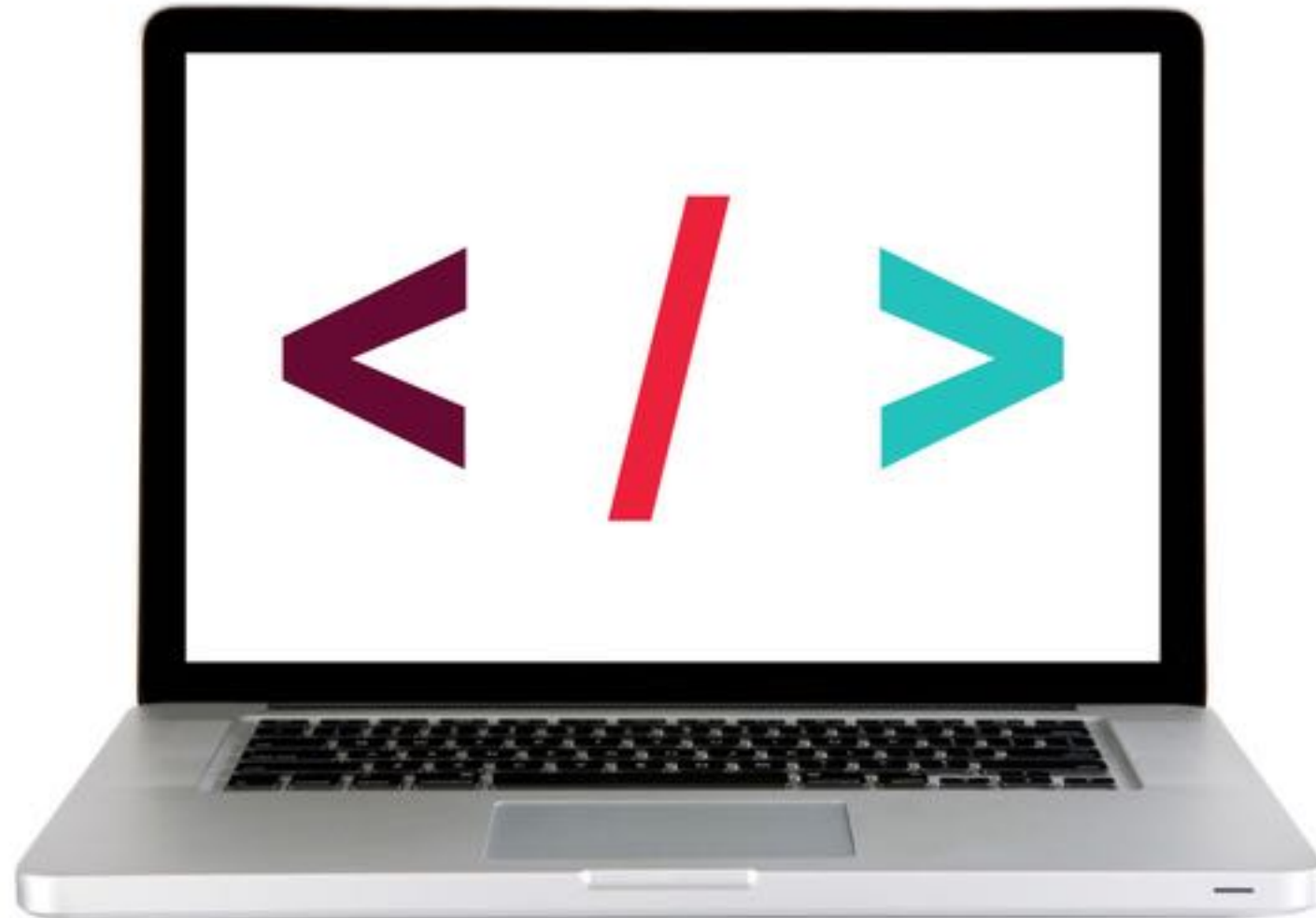
Grocery List

- Pepper Jack Cheese
- Hot Sauce
- Tortilla Chips



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Methods | Getting/Setting Content</title>
    <link rel="stylesheet" href="css/style.css">
  </head>
  <body>
    <h1>Grocery List</h1>
    <ul>
      <li>Pepper Jack Cheese</li>
      <li>Hot Sauce</li>
      <li>Tortilla Chips</li>
    </ul>
    </script>
  </body>
</html>
```

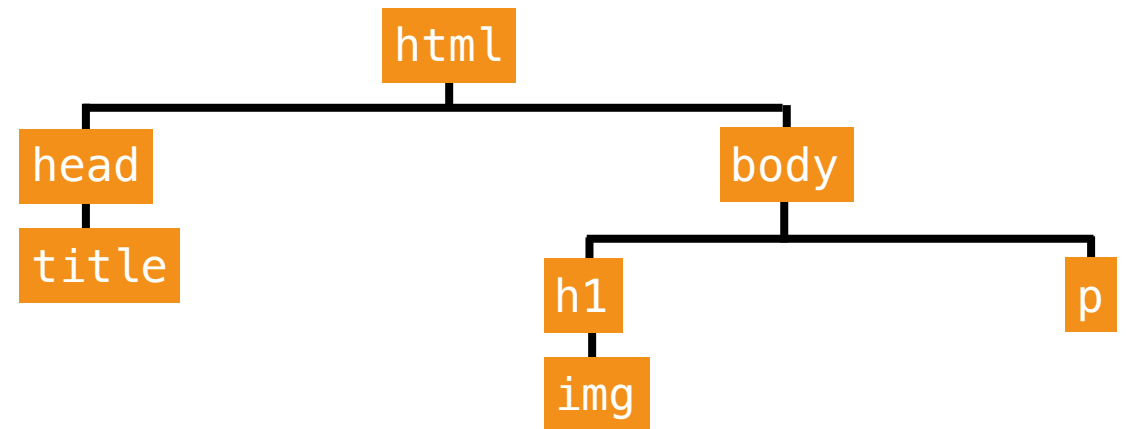
LET'S TAKE A LOOK



Web page elements

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

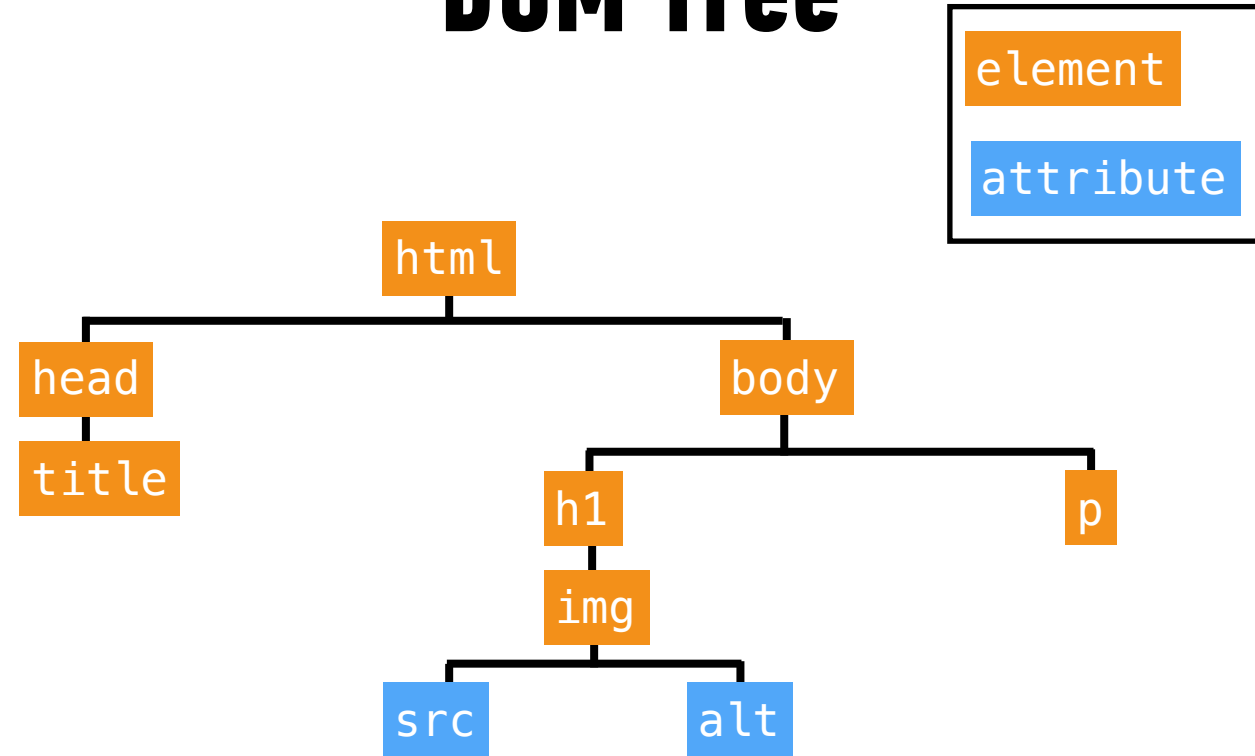
DOM Tree



Web page elements

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

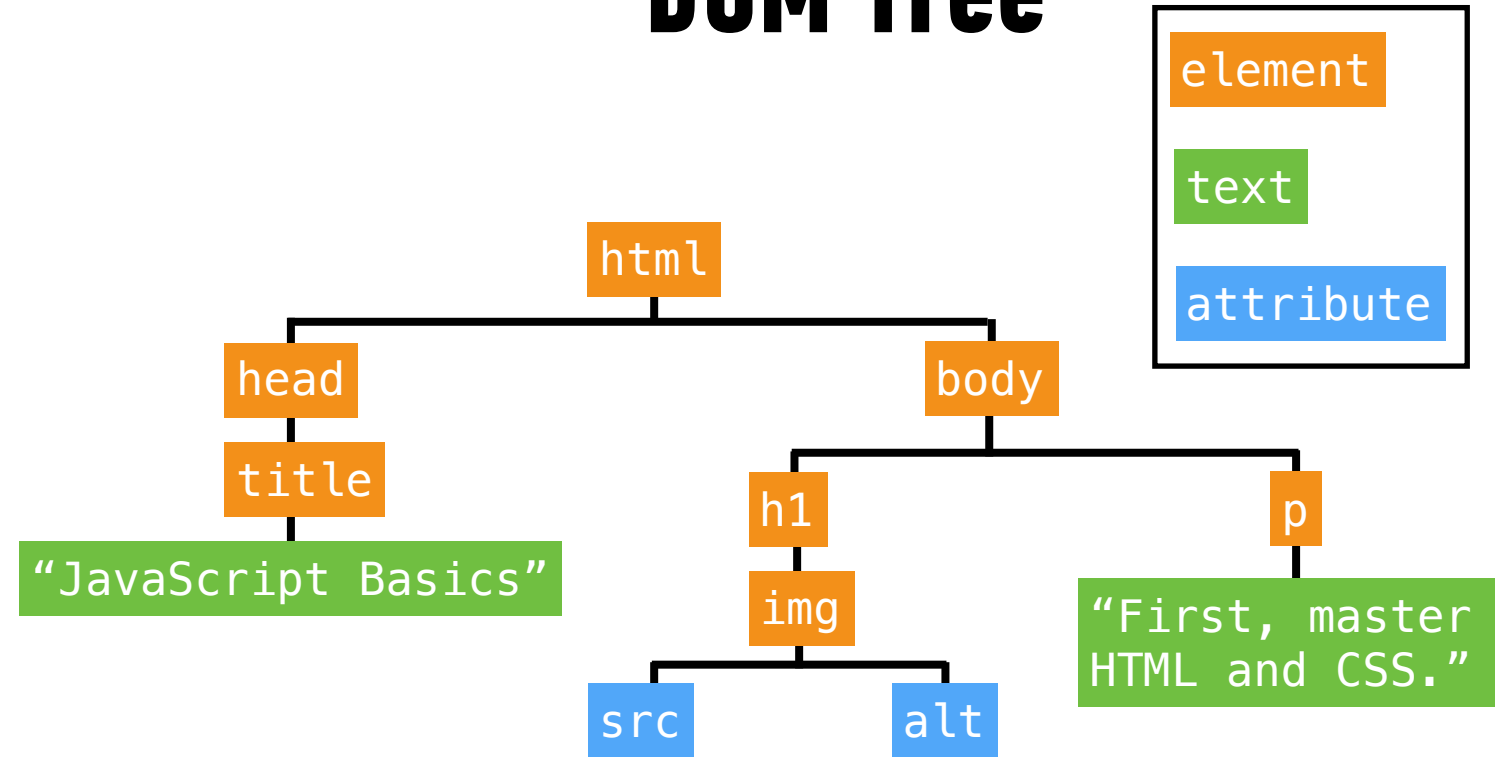
DOM Tree



Web page elements

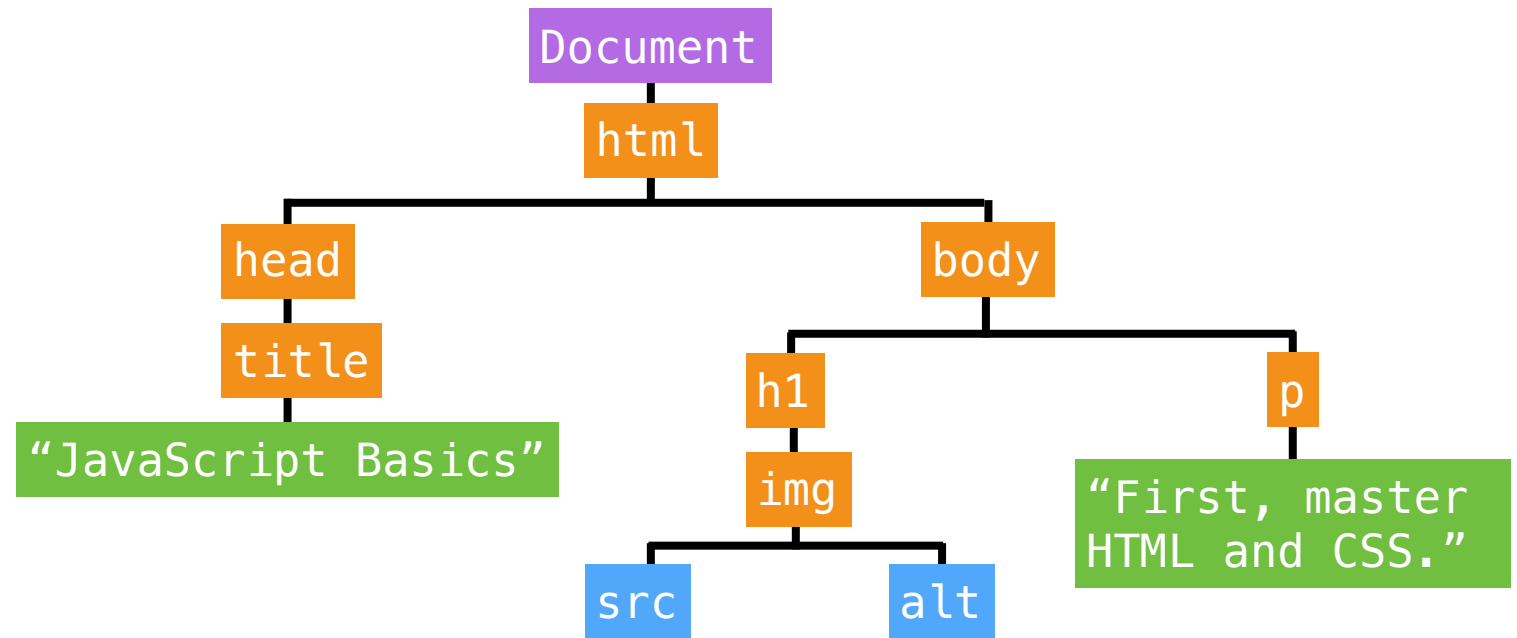
```
<html>
  <head>
    <title>JavaScript Basics</title>
  </head>
  <body>
    <h1>
      
    </h1>
    <p>First, master HTML and CSS.</p>
  </body>
</html>
```

DOM Tree



The Document object

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



EXERCISE



EXERCISE

KEY OBJECTIVE

- Identify differences between the DOM and HTML

TYPE OF EXERCISE

- Pairs

TIMING

2 min

1. Discuss how the DOM is different from a page's HTML

DOM MANIPULATION

SELECTING DOM ELEMENTS

Selecting an element in the DOM

- `getElementById()`
- `getElementsByClassName()`
- `getElementsByTagName()`
- `querySelector()`
- `querySelectorAll()`

Let us select DOM elements
using CSS selector syntax



CSS SELECTORS

CSS SELECTORS REVIEW

Selecting all instances of an element

the name of the element as a string

'div'

Selecting all instances of multiple elements

the names of the elements separated by commas, as a single string

'div, span'

Selecting the element with a specified ID value

the hash symbol (#) followed by the name of the ID value

' #top '

Selecting all elements with a specified class value

a period (.) followed by the name of the class value

' .pullquote '

Selecting all elements with a given relationship

the hierarchical relationship, ending with the element to select

'div p a'

Selecting all elements with a given relationship

the hierarchical relationship, ending with the element to select

'div p .pullquote'

CSS SELECTORS QUIZ

Select all paragraph elements and list item elements

'p li'

Select all paragraph elements and list item elements

'p, li'

Select unordered lists within divs

'ul div'

Select unordered lists within divs

'div ul'

Select the element with the ID value 'main'

'main'

Select the element with the ID value 'main'

' #main '

Select the elements with the class value 'col3'

' : col3 '

Select the elements with the class value 'col3'

' .col3 '

**Select the elements with the class value 'col3'
nested within div elements**

'div.col3'

**Select the elements with the class value 'col3'
nested within div elements**

'div .col3'

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');
```

WORKING WITH SELECTED ELEMENTS

What can we do with a selected element?

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

textContent

- Gets the existing text content of an element

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

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


textContent

- Sets new text content in an element

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

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

innerHTML

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

```
let item = document.querySelector('li');  
  
console.log(item.innerHTML)  
// Gets value: "<strong>Lorem ipsum</strong>"
```

innerHTML

- Sets new content in an element, including nested HTML tags

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

```
item.innerHTML = '<em>Apples</em>'  
// Sets value: '<em>Apples</em>'
```

className property

- Gets an element's `class` attribute value

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

```
console.log(item.className) // Gets value: 'default'
```

className property

- › 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.

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

```
item.className = 'selected'  
// Sets value: 'selected'
```

classList.add method

- Adds specific class attribute value(s)

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

```
item.classList.add('selected');
```

```
// Adds value: 'selected'
```

classList.remove method

- Removes specific class attribute value(s)

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

```
item.classList.remove('default');
```

```
// Removes value: 'default'
```

classList.toggle method

- Removes existing or adds missing class attribute value(s)

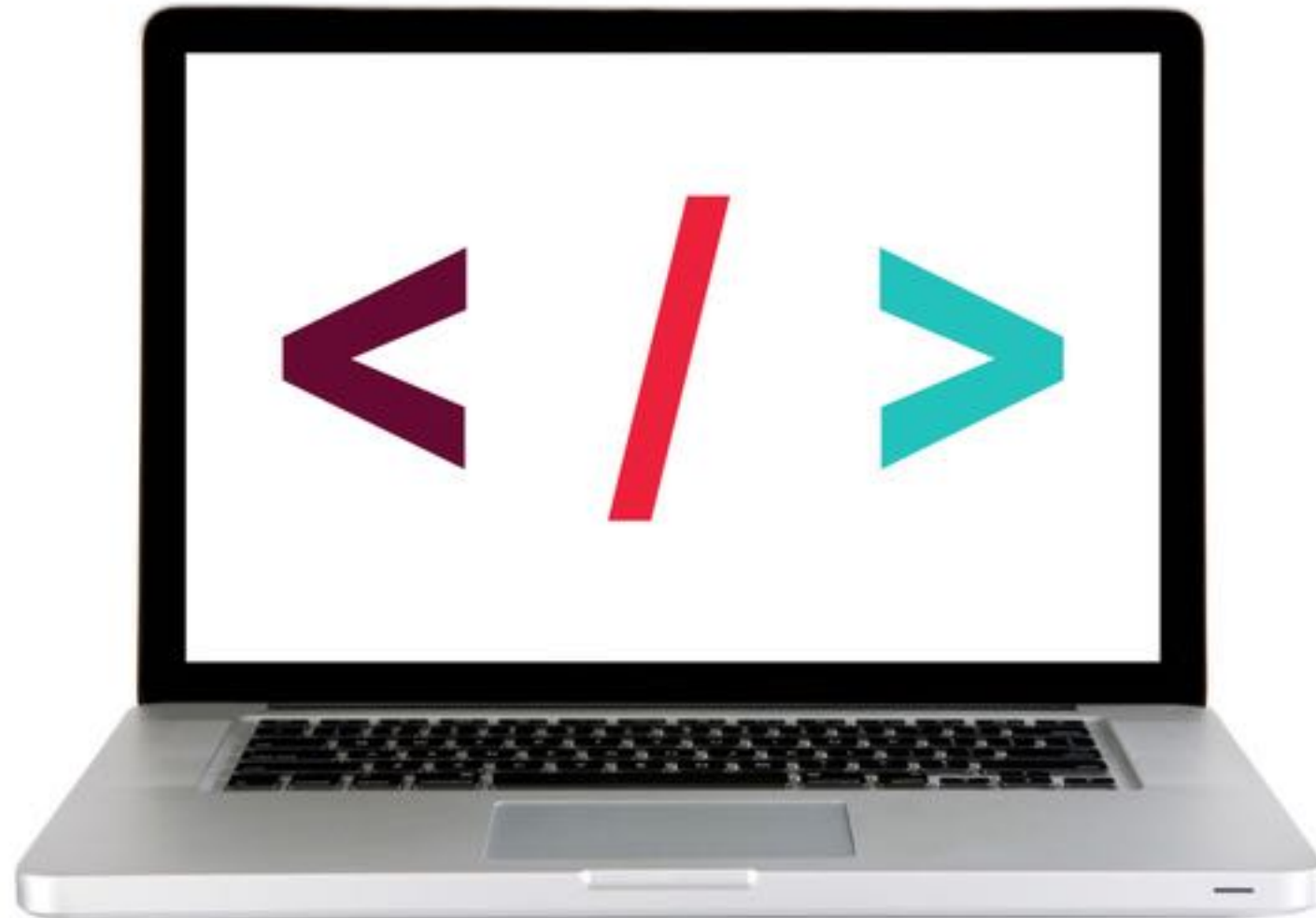
```
let item = document.querySelector('li');
```

```
item.classList.toggle('selected');
```

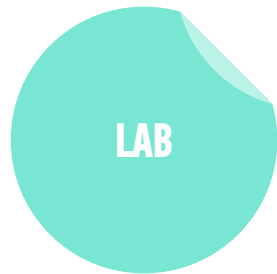
```
// Adds value 'selected' if not already in collection
```

```
// Removes value 'selected' if already in collection
```

LET'S TAKE A LOOK



LAB — DOM



KEY OBJECTIVE

- ▶ Use vanilla JavaScript methods and properties to create and modify DOM nodes.

TYPE OF EXERCISE

- ▶ Individual or pair

TIMING

5 min

1. Open `starter-code > 4-dom-exercise > app.js` in your editor.
2. Follow the instructions to write code that selects and modifies the indicated elements and content.

LAB — DOM



KEY OBJECTIVE

- ▶ Use vanilla JavaScript methods and properties to create and modify DOM nodes.

TYPE OF EXERCISE

- ▶ Individual or pair

TIMING

5 min

1. Open `starter-code > 5-dom-attributes-exercise > app.js` in your editor.
2. Follow the instructions to write code that selects and modifies the indicated elements and content.

CREATING DOM ELEMENTS

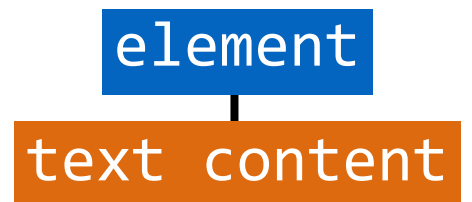
Adding content to the DOM

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

element

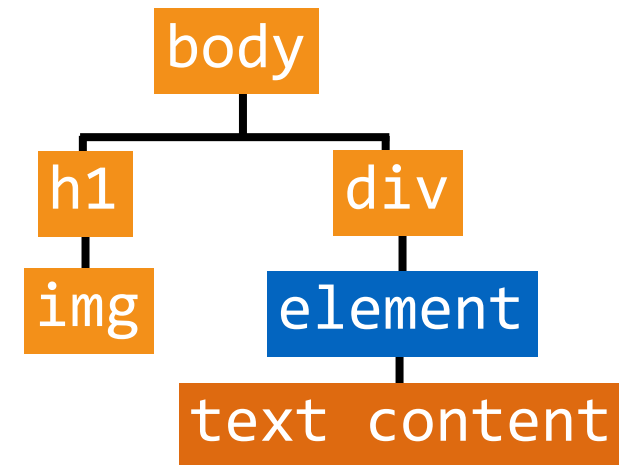
Adding content to the DOM

1. create a new element with `document.createElement()`
2. add text content to that element with the `textContent` or `innerHTML` property



Adding content to the DOM

1. create a new element with `document.createElement()`
2. add text content to that element with the `textContent` or `innerHTML` property
3. 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');
```


textContent or innerHTML

- Specifies content of an element

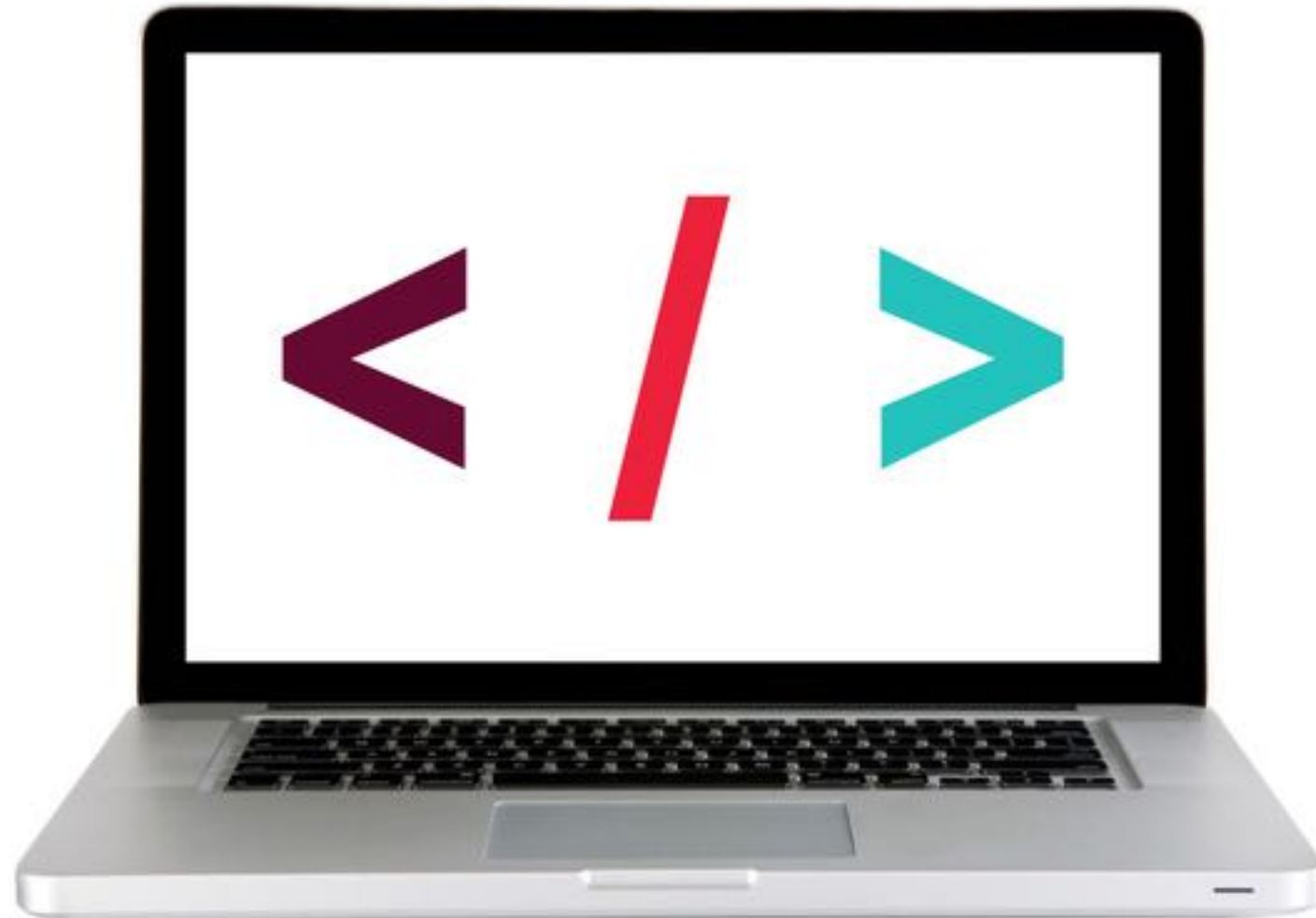
```
item1.textContent = 'banana';  
item2.textContent = 'apple';
```

appendChild()

- 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
```

LET'S TAKE A LOOK



EXERCISE – ADD CONTENT TO A WEB PAGE USING JAVASCRIPT



EXERCISE

LOCATION

► starter-code > 7-create-append-lab

TIMING

until 9:20

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.

Exit Tickets!

(Class #6)

LEARNING OBJECTIVES – REVIEW

- Implement and work with JSON data
- Identify differences between the DOM and HTML.
- Use vanilla JavaScript methods and properties to create and modify DOM nodes.

NEXT CLASS PREVIEW

Intro to jQuery

- Select DOM elements and properties using jQuery.
- Manipulate the DOM by using jQuery selectors and functions.
- Create DOM event handlers using jQuery.

Q&A