

# JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

# HELLO!

1. Pull changes from the `svodnik/JS-SF-9-resources` repo to your computer
2. Open the `08-jquery > starter-code` folder in your code editor

---

**JAVASCRIPT DEVELOPMENT**

---

# **INTRO TO JQUERY**

# **LEARNING OBJECTIVES**

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

- Explain and use JavaScript methods for DOM manipulation.
- Create DOM event handlers to respond to user actions
- Manipulate the DOM by using jQuery selectors and functions.
- Register and trigger event handlers for jQuery events.

# **AGENDA**

- DOM manipulation
- JavaScript events
- jQuery intro
- jQuery events

---

## INTRO TO JQUERY

---

# WEEKLY OVERVIEW

### WEEK 5

Intro to the DOM / Intro to jQuery

### WEEK 6

Advanced jQuery / Ajax & APIs

### WEEK 7

Asynchronous JavaScript & Callbacks / Advanced APIs

# **EXIT TICKET QUESTIONS**

1. What is a "node" in reference to tonight's content?
2. Can DOM manipulation be used for animation in websites?
3. Event listeners!

---

## EXERCISE — CONTINUED FROM LAST TIME

---



### EXERCISE

#### LOCATION

---

- In the start files from the **previous class**,  
2-dom-attributes-exercise  
(or if you want to start fresh, in tonight's start files,  
00-dom-attributes-exercise)

#### TIMING

---

*5 min*

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



# ADDING CONTENT TO THE DOM

# Adding content to the DOM

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

element

# Adding content to the DOM

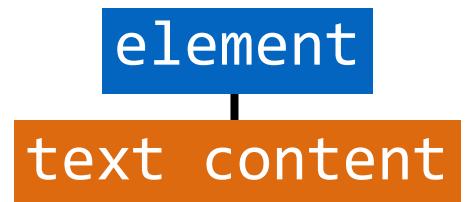
1. create a new element with  
`document.createElement()`
2. create new content for that element  
with `document.createTextNode()`

element

text content

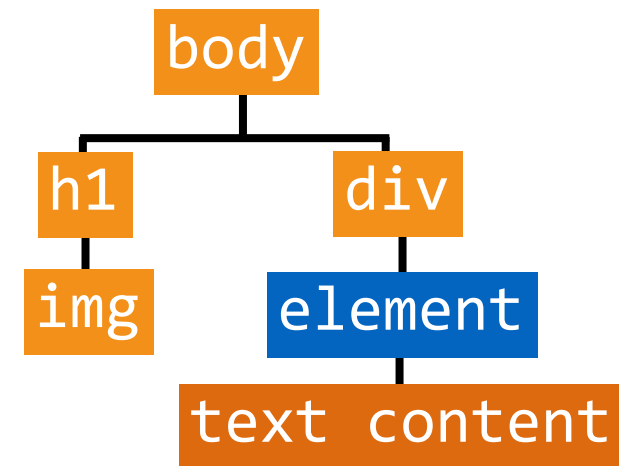
# Adding content to the DOM

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
```



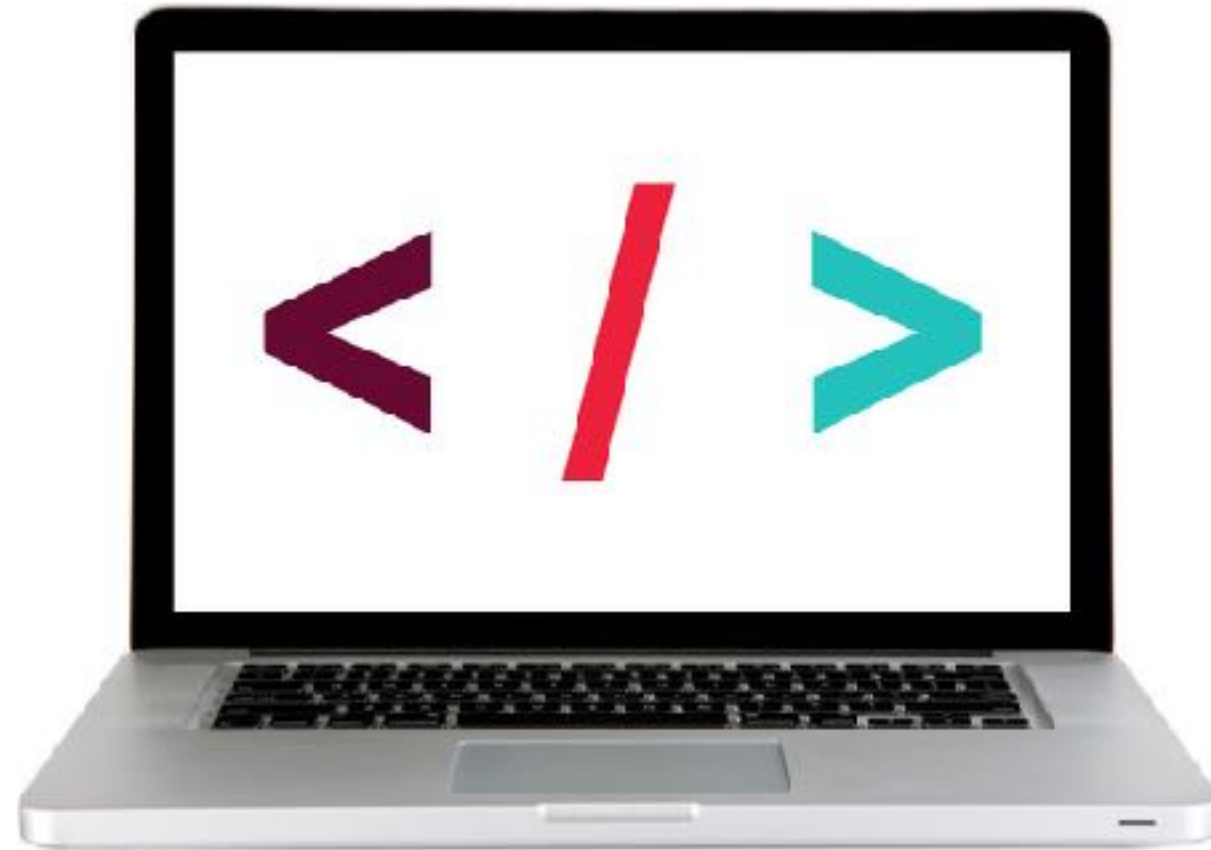
## appendChild ( ) (continued)

- 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
```

# INTRO TO JQUERY

---



**LET'S TAKE A CLOSER LOOK**

# EXERCISE

---



## 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

---



## EXERCISE

### LOCATION

---

► starter-code > 02-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.

# **DOM EVENTS**

# EXERCISE

---



## TYPE OF EXERCISE

---

► Pairs

## TIMING

---

*2 min*

1. Make a list of events you've seen, experienced, or imagined that can trigger a change in a web page
2. Compare your list with your partner, and ask about any events on your partner's list that you're unfamiliar with

# EVENTS



### MOUSE

click  
dblclick  
mouseenter  
mouseleave



### KEYBOARD

keypress  
keydown  
keyup



### FORM

submit  
change  
focus  
blur



### DOCUMENT

resize  
scroll

# EVENT HANDLER

- › **Event handler:** code that responds to an event

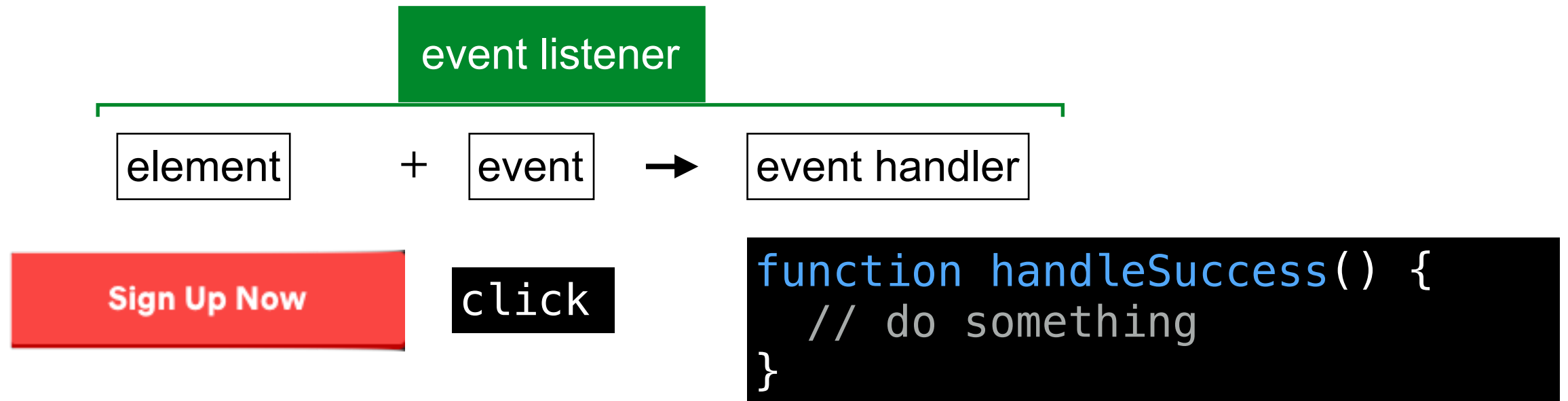
event handler

```
function handleSuccess() {  
    // do something  
}
```



# EVENT LISTENER

- › **Event listener:** code that associates an element, an event, and an event handler



# **addEventListener()**

- Method used to add an event listener to an element

# 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

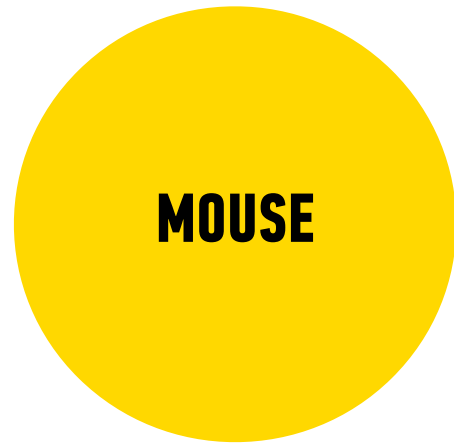
```
button.addEventListener('click', function() {  
    // your code here  
}, false);
```

# EVENT LISTENERS

```
let button = document.querySelector('.submitBtn');
```

type of event

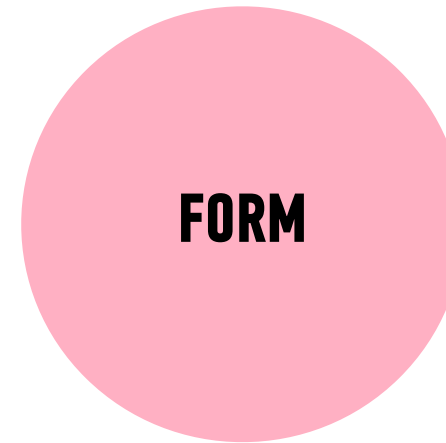
```
button.addEventListener('click', function() {  
    // your code here  
}, false);
```



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

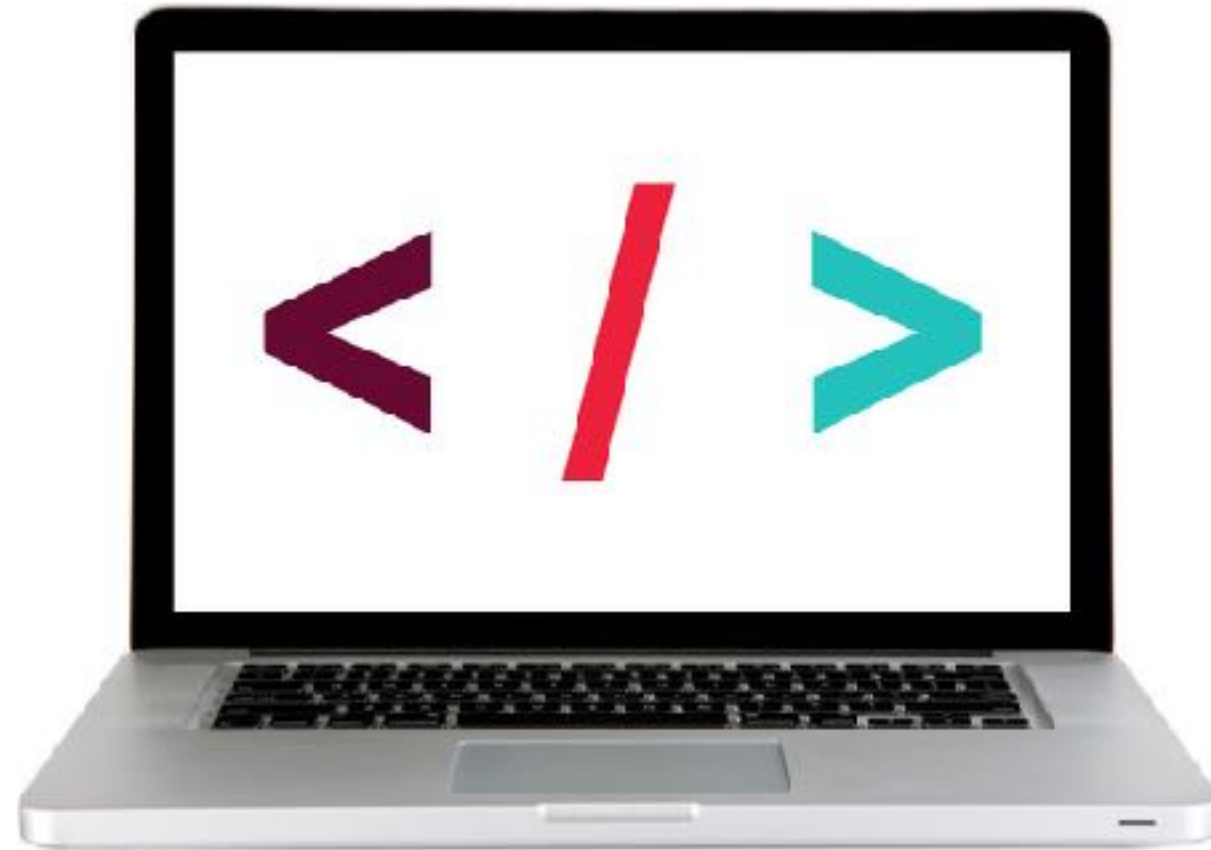
```
button.addEventListener('click', function() {  
    // your code here  
}, false);
```

function  
to run  
when  
event is  
triggered

final boolean parameter  
for backward compatibility

# INTRO TO JQUERY

---



**LET'S TAKE A CLOSER LOOK**

# ACTIVITY

---



## EXERCISE

### KEY OBJECTIVE

---

- ▶ Create event handlers to respond to user actions

### TYPE OF EXERCISE

---

- ▶ Individual/Partner

### TIMING

---

*10 min*

Exercise is in 04-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

reference to  
parameter  
name we  
chose

```
submitButton.addEventListener('click', function(event) {  
    event.preventDefault();  
    ...  
}, false);
```

parameter name

# Referencing an event

- We could choose a different parameter name and our code would still work the same

reference to  
parameter  
name we  
chose

```
submitButton.addEventListener('click', function(evt) {  
    ...  
}, false);
```

parameter name

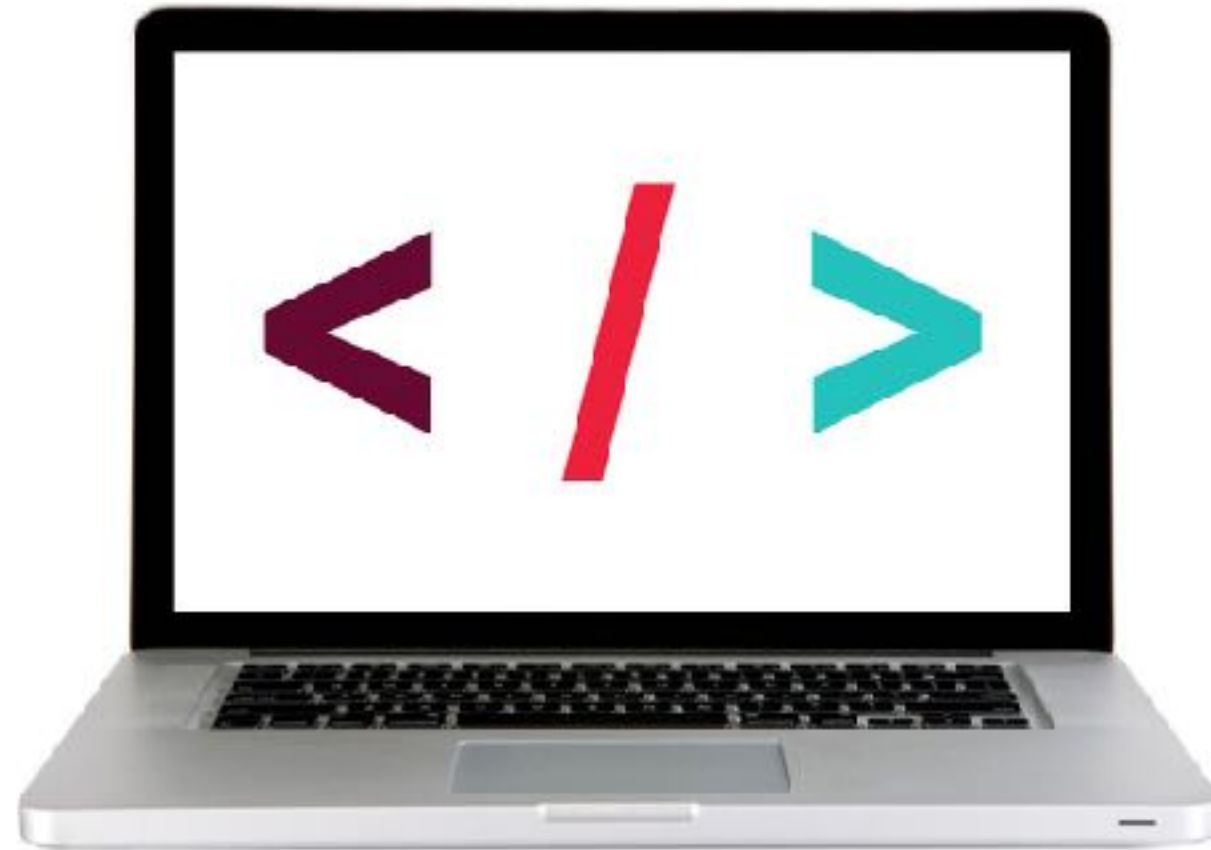
parameter name

reference to  
parameter  
name we  
chose

```
submitButton.addEventListener('click', function(e) {  
    ...  
}, false);
```

# INTRO TO JQUERY

---



**LET'S TAKE A CLOSER LOOK**

# EXERCISE

---



## EXERCISE

### LOCATION

---

► starter-code > 06-js-dom-exercise

### TIMING

---

*10 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.)



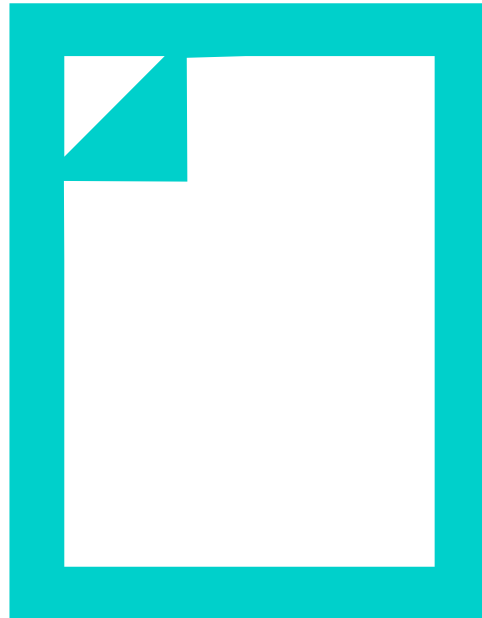
# JQUERY

---

# INTRO TO JQUERY — YOUR NEW BEST FRIEND!

---

jQuery is a JavaScript library you include in your pages.



---

## JQUERY VS. JAVASCRIPT

---

jQuery allows us to keep using the CSS-style selectors that we know and love — but more concisely! Yay!

**JS:**



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



```
document.querySelector('#about')
```



**JQUERY:**

```
$('.ul li')
```



```
$('#about')
```



---

## JQUERY VS. JAVASCRIPT

---

jQuery statements for DOM manipulation are also more concise!

**JS:**

```
document.querySelector('#heading').innerHTML = "Your Name";
```



**JQUERY:**

```
$('#heading').text('Your Name');
```



*\*\*You could do everything jQuery does with plain-old vanilla Javascript\*\**

---

## JQUERY VS. JAVASCRIPT — A COMPARISON OF BENEFITS

---

### **JQUERY**

- Write way less code to achieve the same tasks

### **PURE JAVASCRIPT**

- Better performance
- Faster

---

**JQUERY**

---

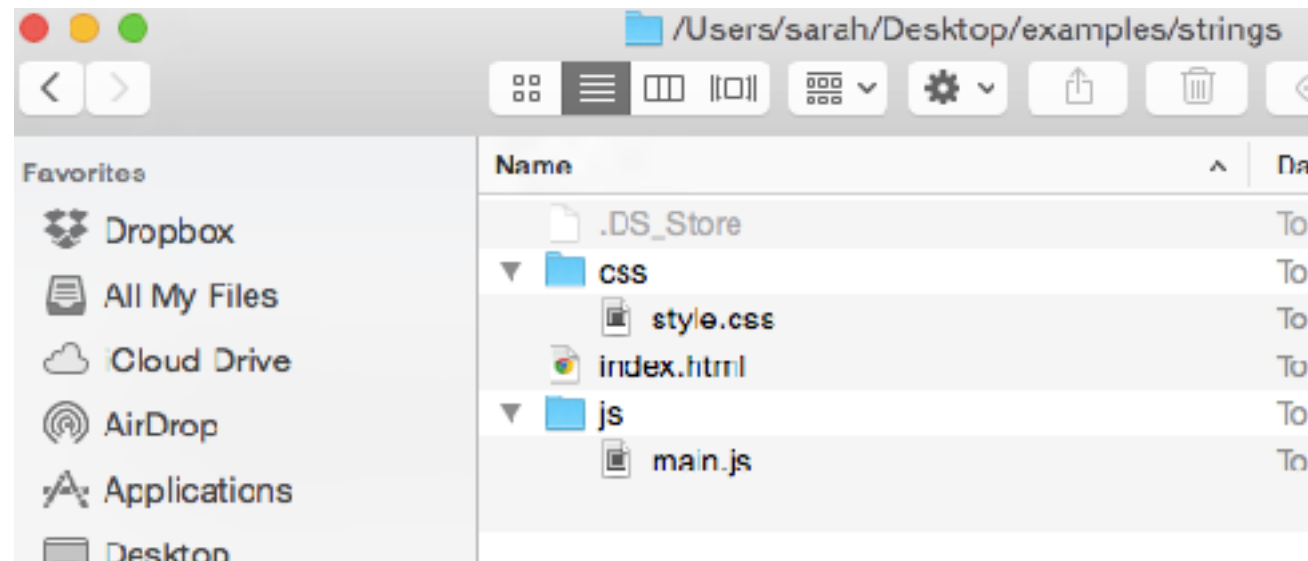
# ADDING JQUERY TO YOUR PROJECT

---

## KEEP IT ON THE UP AND UP!

---

- It is considered **best practice** to keep Javascript files organized in one folder.
- Usually people name this folder *scripts*, *js*, or *javascript*.



Remember - use an underscore or dash between words in folder names instead of a space. And try to avoid characters/symbols in file names (*really\_cool\_page.html* or *really-cool-page.html*).

---

## STEP 1: ADD JQUERY TO YOUR WEBSITE

---

1. Download the [jQuery](#) script (version 3.x, compressed).
2. Add a js folder to your project
3. Move the jQuery file you downloaded to the js folder
4. Use a `<script>` tag to include the jQuery file after your HTML content and before any other JavaScript files that use it.

```
<body>  
  <!-- HTML content here -->  
  <script src="js/jquery-3.2.1.min.js"></script>  
  <script src="js/main.js"></script>  
</body>
```



---

## STEP 2: ADD A JAVASCRIPT FILE

---

1. Create your custom JavaScript file with a .js extension (example: main.js)
2. Link to the JavaScript file from your HTML page using the `<script>` element. Add this **right before the closing `</body>` tag and after the `<script>` element for your jQuery file.**

```
<body>  
  <!-- HTML content here -->  
  <script src="js/jquery-3.2.1.min.js"></script>  
  <script src="js/main.js"></script>  
</body>
```



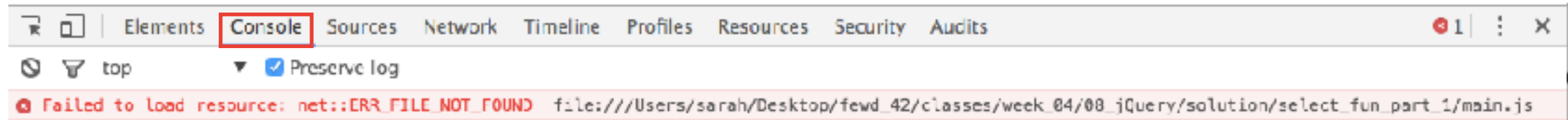
**ORDER IS IMPORTANT!!!!**

---

## MAKE SURE YOUR JS IS HOOKED UP PROPERLY

---

- ▶ Open the page in Chrome, then open the console (command + option + J [Mac] or Ctrl + Alt + J [Win]) and make sure there are no errors.



*This error means the file can't be found. Check your url in your `<script>` tag. Make sure the file exists.*

---

**JQUERY**

---

# **PART 1 — SELECT AN ELEMENT**

# A JQUERY STATEMENT INVOLVES 2 PARTS

**1**

Select an element/elements

**2**

Work with those elements

---

## INTRO TO JQUERY

---

**1**

Select an element/elements

**2**

Work with those elements

---

## JQUERY — SELECTING ELEMENTS

---

Selector



```
$('li').addClass('selected');
```

---

# JQUERY OBJECTS — FINDING ELEMENTS: SOME EXAMPLES

---

	CSS	JQUERY
ELEMENT	<code>a { color: blue; }</code>	<code>\$('a')</code>
ID	<code>#special { color: blue; }</code>	<code>\$('#special')</code>
CLASS	<code>.info { color: blue; }</code>	<code>\$('.info')</code>
NESTED SELECTOR	<code>div span { color: blue; }</code>	<code>\$('div span')</code>

---

---

```
<button id="form-submit">Submit</button>
```

```
<li class="circle">One</li>
```

```
<h1>Color Scheme Switcher</h1>
```



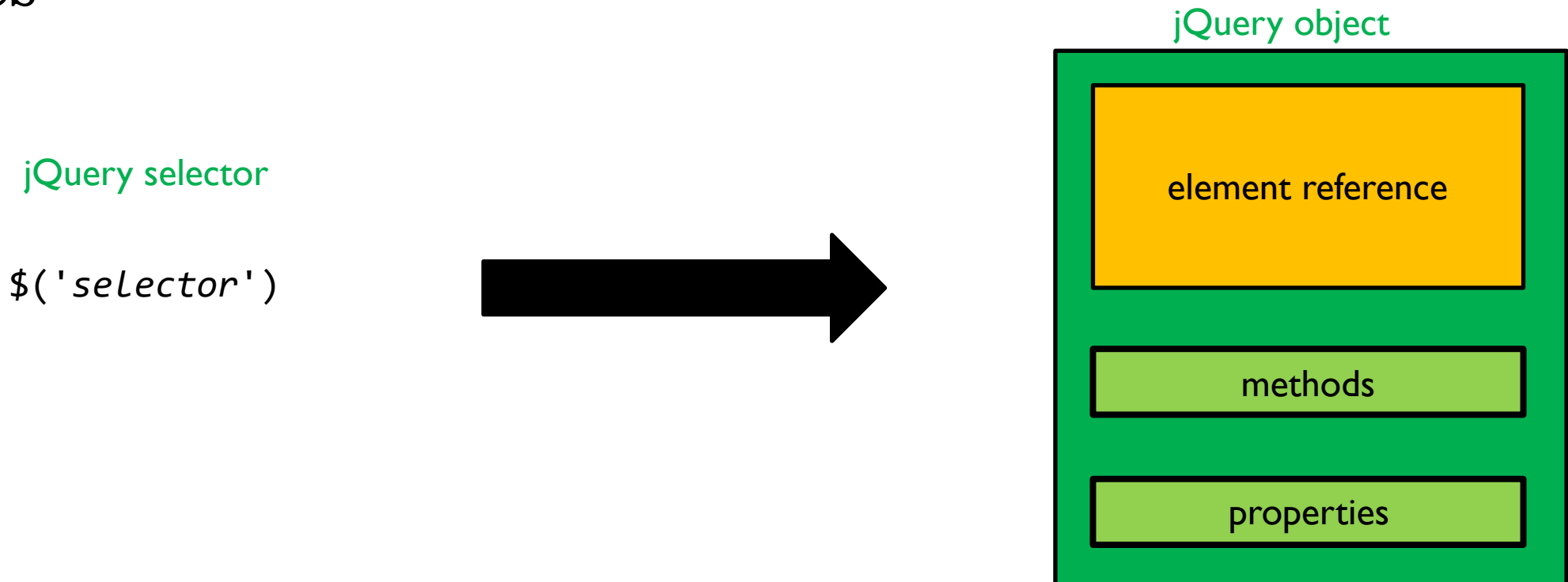
# JQUERY OBJECTS

- Selecting elements with vanilla JavaScript returns an element reference (`querySelector()`) or a collection of element references (`querySelectorAll()`)



# JQUERY OBJECTS

- Selecting elements with jQuery returns a **jQuery object**, which is one or more element references packaged with jQuery methods and properties



# NAMING VARIABLES WHEN USING JQUERY

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

\$ included at start of variable name to indicate that its value is a jQuery object

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



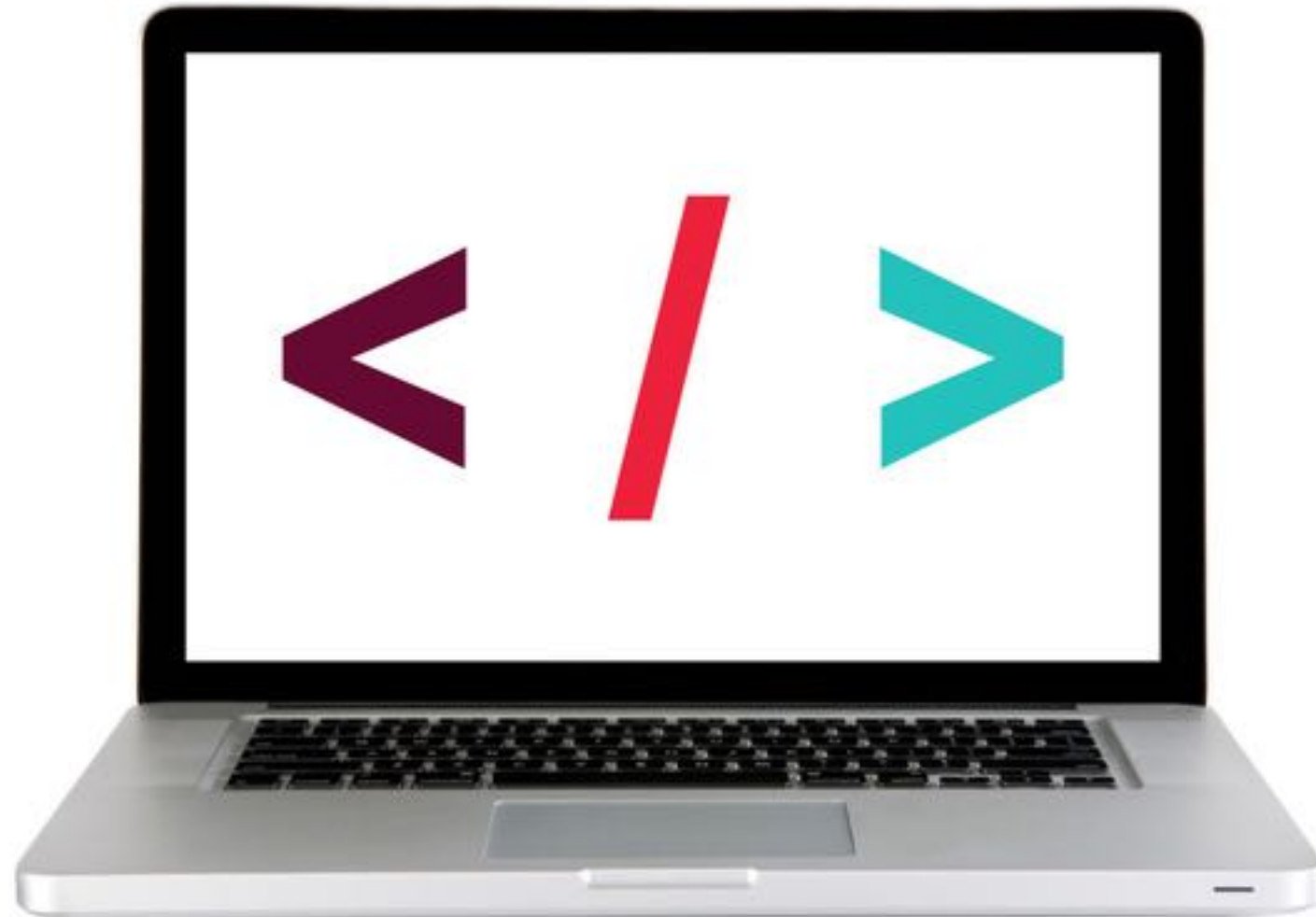
it's not an error to name the variable with out the \$ — it just wouldn't give us as much information

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

---

## LET'S TAKE A CLOSER LOOK

---



---

**JQUERY**

---

# **PART 2 — ADD A METHOD**

---

## USING JQUERY TO MANIPULATE THE DOM

---

**1**

Select an element/elements

**2**

Work with those elements

---

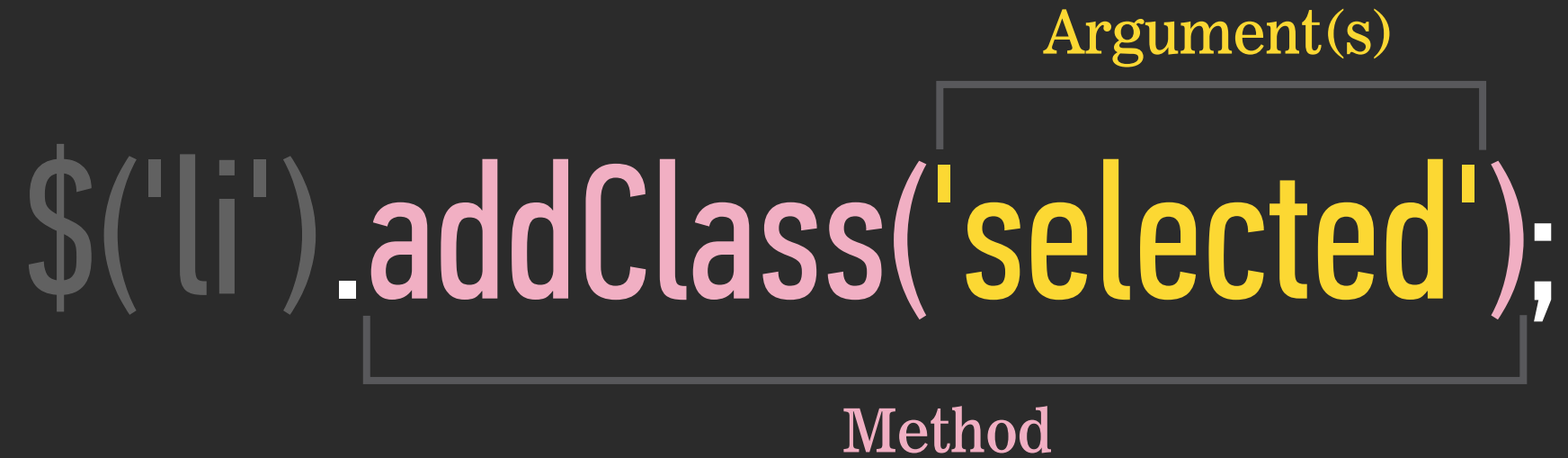
## JQUERY — WORKING WITH THOSE ELEMENTS

---

Argument(s)

```
$('li').addClass('selected');
```

Method

The image shows the jQuery code snippet `$('li').addClass('selected');` with two annotations. A bracket above the string `'selected'` is labeled "Argument(s)" in yellow text. A bracket below the `addClass` method name is labeled "Method" in pink text. The `$('li')` part of the code is rendered in a lighter gray color.

---

## JQUERY METHODS

---

**Be forewarned!**

**There are a lot of methods!**

*Do not feel like you need to sit down and memorize these.* The important things is knowing that they're there and **being able to look them up** in the documentation.

**[api.jquery.com](https://api.jquery.com)**



---

## JQUERY METHODS — WORKING WITH THOSE ELEMENTS

---

After we've selected elements, we can use jQuery methods to:

**FIND  
ELEMENTS**

**GET/SET  
CONTENT**

**ADD  
EFFECTS/  
ANIMATION**

**CREATE  
EVENT  
LISTENERS**

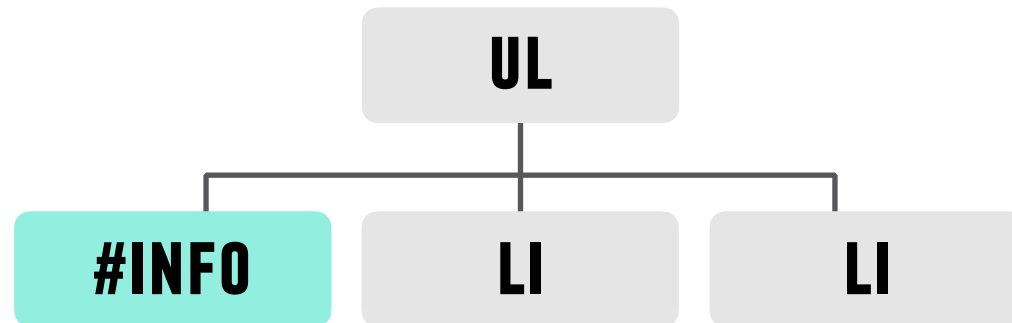


*See your handout or the [jQuery docs](#) for list!*

## TRAVERSING THE DOM?

---

```
$('#info').parent();
```

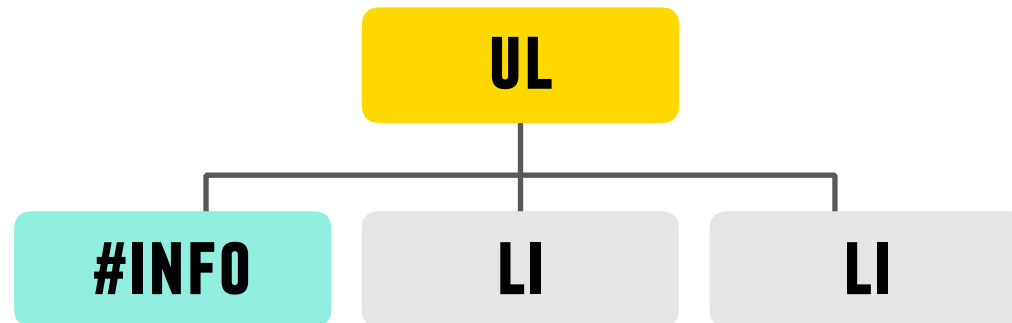


---

## TRAVERSING THE DOM?

---

```
$(' #info' ).parent();
```



## JQUERY METHODS — TRAVERSING THE DOM

### TRAVERSE THE DOM

- ▶ Think of these as filters, or part of the selection process.
- ▶ They must come *directly after another selection*

METHODS	EXAMPLES
<code>.find()</code> <i>finds all descendants</i>	<code>\$('h1').find('a');</code>
<code>.parent()</code>	<code>\$('#box1').parent();</code>
<code>.siblings()</code>	<code>\$('p').siblings('.important');</code>
<code>.children()</code>	<code>\$('ul').children('li');</code>

What goes in the parentheses?  
**A css-style selector**

---

## JQUERY METHODS — WORKING WITH THOSE ELEMENTS

---

After we've selected elements, we can use jQuery methods to:

**FIND  
ELEMENTS**

**GET/SET  
CONTENT**

**ADD  
EFFECTS/  
ANIMATION**

**CREATE  
EVENT  
LISTENERS**



*See your handout or the [jQuery docs](#) for list!*

Get/change content of elements and attributes

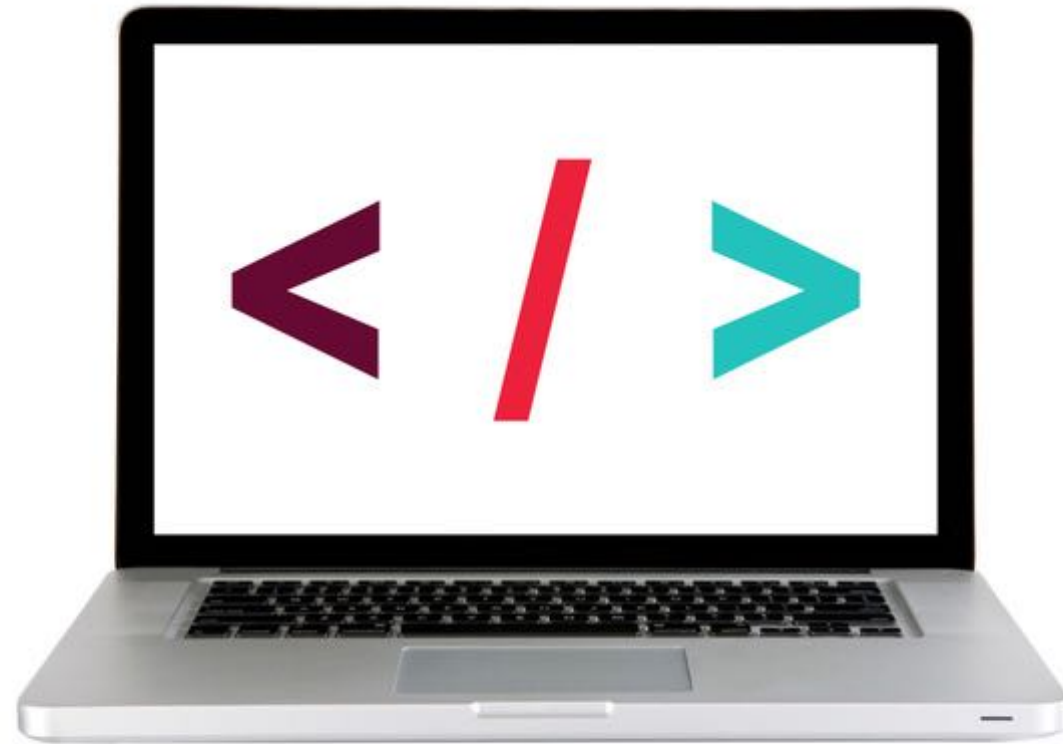
METHODS	EXAMPLES
<code>.html()</code>	<code>\$('#h1').html('Content to insert goes here');</code>
<code>.attr()</code>	<code>\$('#img').attr('src', 'images/bike.png');</code>

What goes in the parentheses?  
The **html** you want to change.

---

## LET'S TAKE A CLOSER LOOK

---



Get/change content of elements and attributes

METHODS	EXAMPLES
<code>.addClass()</code>	<code>\$('.p').addClass('success');</code>
<code>.removeClass()</code>	<code>\$('.p').removeClass('my-class-here');</code>
<code>.toggleClass()</code>	<code>\$('.p').toggleClass('special');</code>

What goes in the parentheses?  
The **classes** you want to change.



`$('li').addClass('selected');`

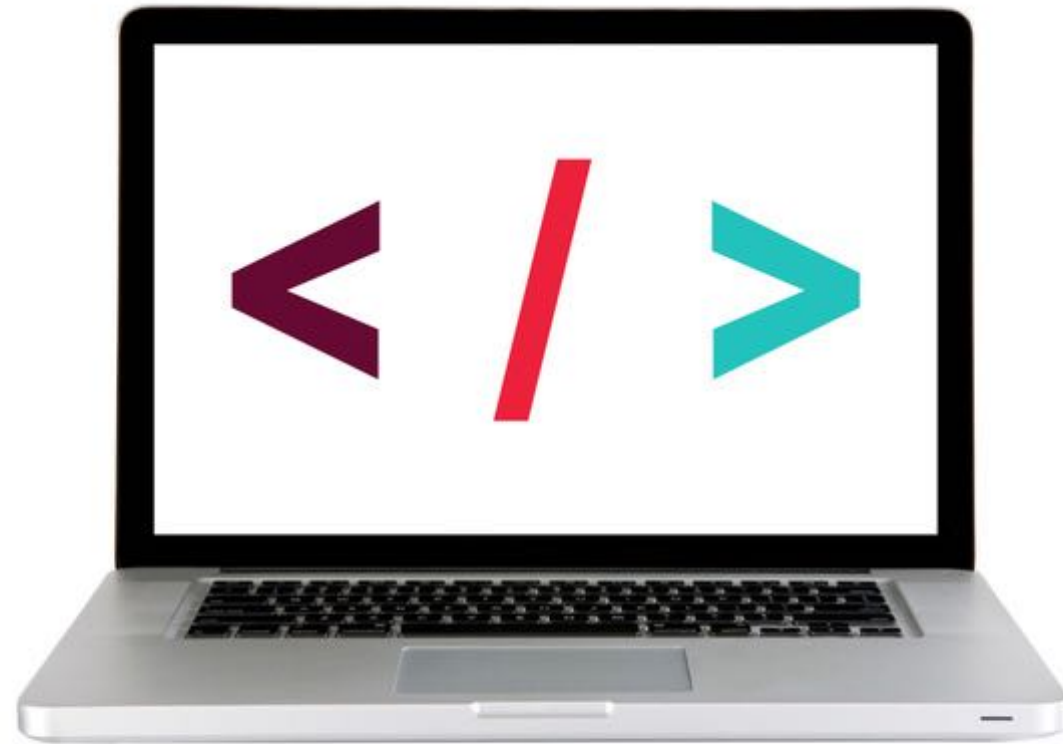


**NO PERIOD!!!**

---

## LET'S TAKE A CLOSER LOOK

---



# ACTIVITY

---



## EXERCISE

### KEY OBJECTIVE

---

- ▶ Utilize jQuery to access and manipulate DOM elements.

### TYPE OF EXERCISE

---

- ▶ Individual/Partner

### TIMING

---

*5 min*

Exercise is in 08-jquery-exercise

1. Follow the instructions under part 1 in main.js
2. Use cheat sheet/slides as a guide for syntax

---

## JQUERY METHODS — WORKING WITH THOSE ELEMENTS

---

After we've selected elements, we can use jQuery methods to:

**FIND  
ELEMENTS**

**GET/SET  
CONTENT**

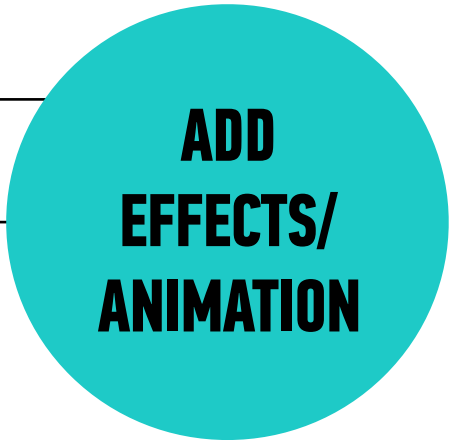
**ADD  
EFFECTS/  
ANIMATION**

**CREATE  
EVENT  
LISTENERS**



*See your handout or the [jQuery docs](#) for list!*

# JQUERY METHODS — EFFECTS/ANIMATION



Add effects and animation to parts of the page

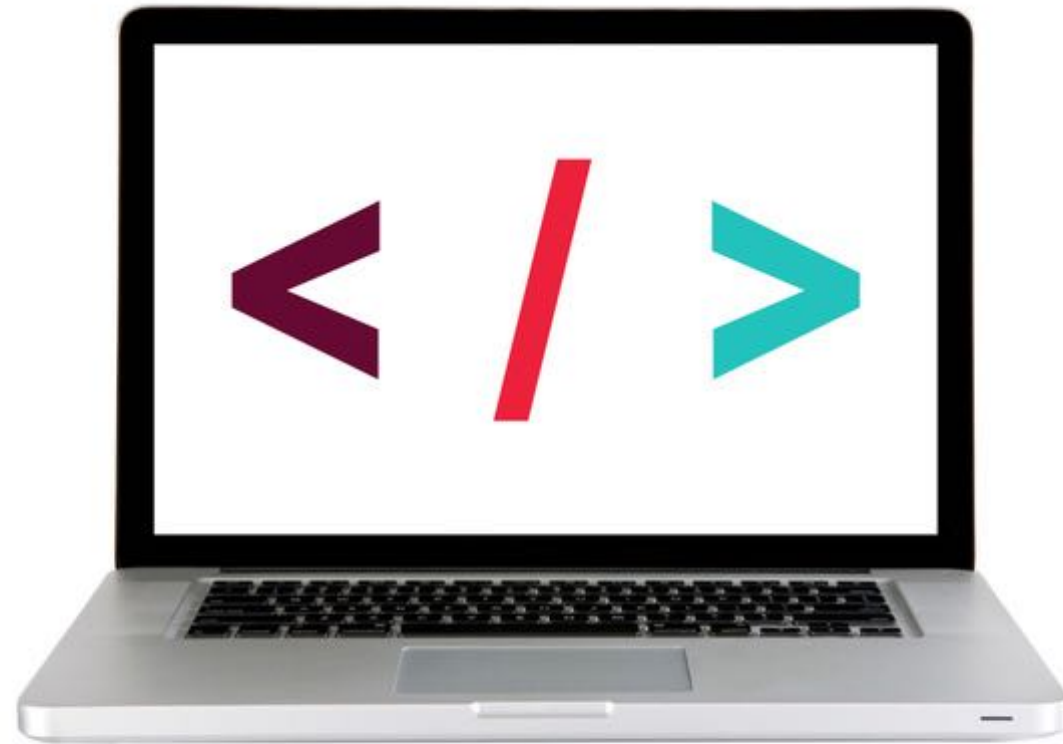
METHODS	EXAMPLES
<code>.show()</code>	<code>\$('h1').show();</code>
<code>.hide()</code>	<code>\$('ul').hide();</code>
<code>.fadeIn()</code>	<code>\$('h1').fadeIn(300);</code>
<code>.fadeOut()</code>	<code>\$('.special').fadeOut('fast');</code>
<code>.slideUp()</code>	<code>\$('div').slideUp();</code>
<code>.slideDown()</code>	<code>\$('#box1').slideDown('slow');</code>
<code>.slideToggle()</code>	<code>\$('p').slideToggle(300);</code>

What goes in the parenthesis?  
**An animation speed**

---

## LET'S TAKE A CLOSER LOOK

---



---

## JQUERY METHODS — WORKING WITH THOSE ELEMENTS

---

After we've selected elements, we can use jQuery methods to:

**FIND  
ELEMENTS**

**GET/SET  
CONTENT**

**ADD  
EFFECTS/  
ANIMATION**

**CREATE  
EVENT  
LISTENERS**



*See your handout or the [jQuery docs](#) for list!*

---

## JQUERY METHODS — EVENTS!

---

A red circle graphic containing the text "CREATE EVENT LISTENERS" in bold, black, uppercase letters.

**CREATE  
EVENT  
LISTENERS**

We can use the `on()` method to handle all events in jQuery.



---

## JQUERY METHODS — EVENTS!

---

**CREATE  
EVENT  
LISTENERS**

selector

```
$('li').on('click', function() {  
    // your code here  
});
```

## JQUERY METHODS — EVENTS!

---

**CREATE  
EVENT  
LISTENERS**

method for all events

```
$( 'li' ).on( 'click', function() {  
    // your code here  
} );
```

---

## JQUERY METHODS — EVENTS!

---

**CREATE  
EVENT  
LISTENERS**

type of event

```
$( 'li' ).on( 'click', function() {  
    // your code here  
});
```

## MOUSE

click  
dblclick  
mouseenter  
mouseleave

## KEYBOARD

keypress  
keydown  
keyup

## FORM

submit  
change  
focus  
blur

## DOCUMENT

resize  
scroll



```
$('#li').on('eventGoesHere', function() {  
  // your code here  
});
```

---

## JQUERY METHODS — EVENTS!

---



### CREATE EVENT LISTENERS

```
$('.li').on('click', function() {  
    // your code here  
});
```

function to run  
when event is  
triggered

## JQUERY METHODS — EVENTS!

### CREATE EVENT LISTENERS

selector      method for  
all events      type of  
event

```
$( 'li' ).on( 'click', function() {  
    // your code here  
});
```

function to run  
when event is  
triggered

# ACTIVITY

---



## EXERCISE

### KEY OBJECTIVE

---

- ▶ Utilize jQuery to access and manipulate DOM elements.

### TYPE OF EXERCISE

---

- ▶ Individual/Partner

### TIMING

---

*5 min*

Continue with 08-jquery-exercise

1. Follow the instructions under Part 2 in main.js
2. Use cheat sheet/slides as a guide for syntax

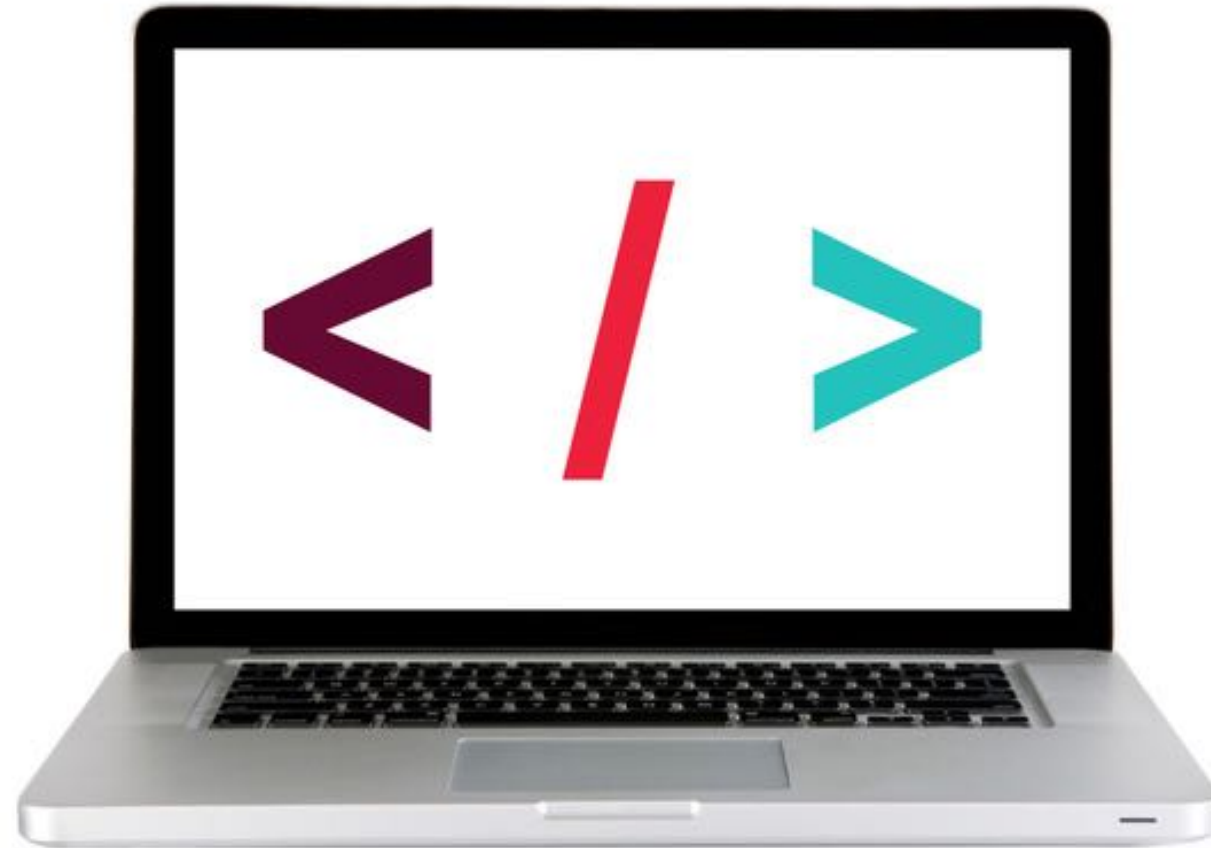
# REFACTORING

- **Refactoring** is the process of rewriting code to make it more efficient, or to incorporate new coding practices
- Rewriting code to replace vanilla JavaScript with jQuery methods is an example of refactoring



# INTRO TO JQUERY

---



**LET'S TAKE A CLOSER LOOK**

# EXERCISE

---



## EXERCISE

### OBJECTIVE

---

- ▶ Manipulate the DOM by using jQuery selectors and functions.

### LOCATION

---

- ▶ `starter-code > 10-jquery-todo-list`

### TIMING

---

*until 9:20*

1. The HTML document contains an empty unordered list. It also contains a text input box and a Create button. Write jQuery to enable users to add elements to the to do list.
2. BONUS: Use jQuery to add a "complete task" link at the end of each to-do item when it is added to the list.

# **Exit Tickets!**

**(Class #8)**

---

# **LEARNING OBJECTIVES – REVIEW**

- Explain and use JavaScript methods for DOM manipulation.
- Create DOM event handlers to respond to user actions
- Manipulate the DOM by using jQuery selectors and functions.
- Register and trigger event handlers for jQuery events.

# **NEXT CLASS PREVIEW**

## **Advanced jQuery**

- Use event delegation to manage dynamic content.
- Use implicit iteration to update elements of a jQuery selection

# **Q&A**