GENERAL ASSEMBLY

JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

HELLO!

1. Submit your homework and create a pull request
2. Pull changes from the `svodnik/JS-SF-9-resources` repo to your computer
3. Open the `09-advanced-jquery > starter-code` folder in your code editor

ADVANCED JQUERY

LEARNING OBJECTIVES

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

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

AGENDA

- jQuery selectors
- jQuery methods
- jQuery events
- jQuery best practices

WEEKLY OVERVIEW

WEEK 6	Advanced jQuery / Ajax & APIs
WEEK 7	Asynchronous JavaScript & Callbacks / Advanced APIs
	HOLIDAY WEEK — NO CLASS!
WEEK 8	Project 2 Lab / Closures & the module pattern

EXIT TICKET QUESTIONS

1. Can you apply a single event listener to multiple classes?
2. Where else would you use event.preventDefault() besides submit buttons and is it best practice to add event.preventDefault() to all addEventListeners?

HOMEWORK REVIEW

HOMEWORK — GROUP DISCUSSION

EXERCISE

TYPE OF EXERCISE

- Groups of 3

TIMING

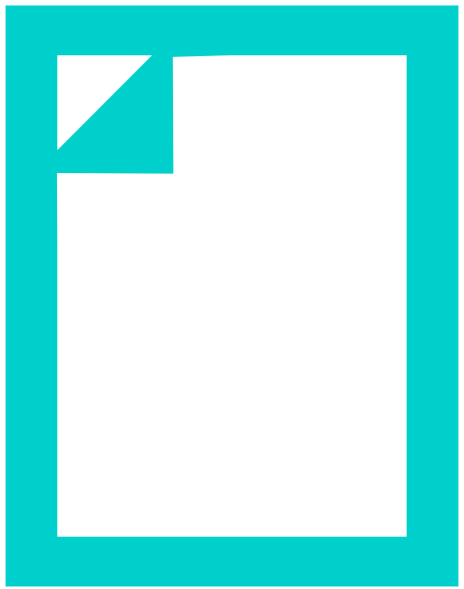
4 min

1. Share your solutions for the DOM 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.

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!

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

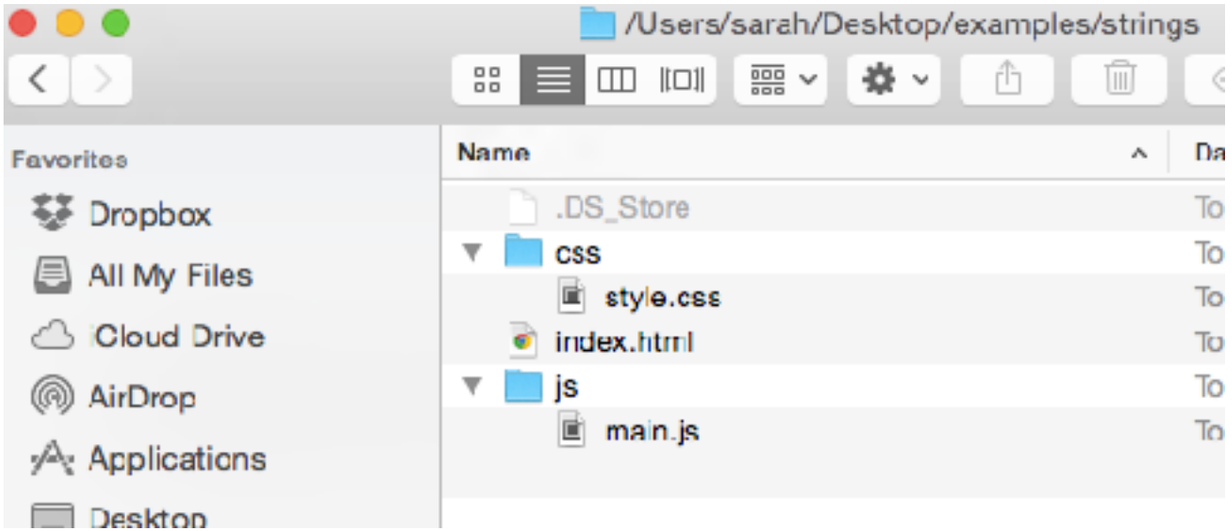
PURE JAVASCRIPT

- Better performance
- Faster

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

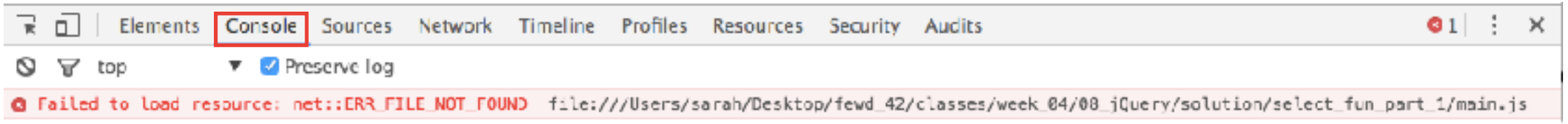
1. Create your custom JavaScript file with a .js extension (example: main.js)
2. Link to the JavaScript file from your HTML page using the `<script>` element. Add this **right before the closing `</body>` tag and after the `<script>` element for your jQuery file.**

```
<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.

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

JQUERY OBJECTS — FINDING ELEMENTS: SOME EXAMPLES

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

```
<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()`)

vanilla JavaScript selector

`querySelector('selector')`

plain element reference

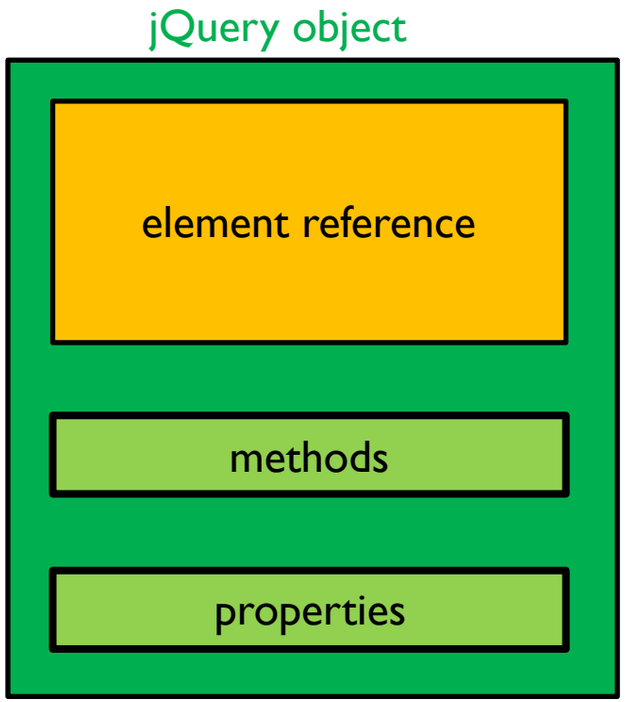
element reference

JQUERY OBJECTS

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

jQuery selector

```
$('selector')
```



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

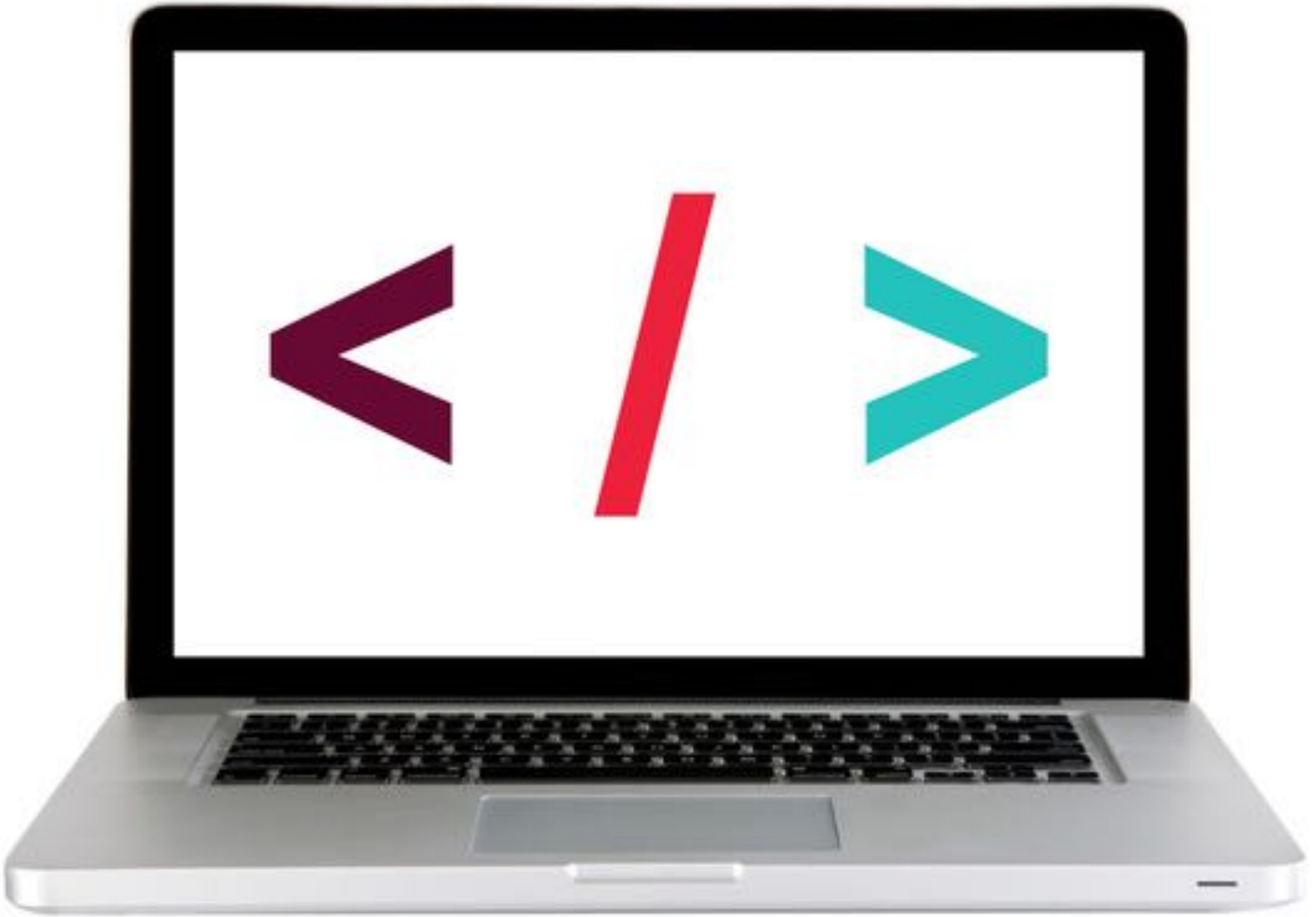
$ included at start of variable name to indicate that its value is a jQuery object

```
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



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

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

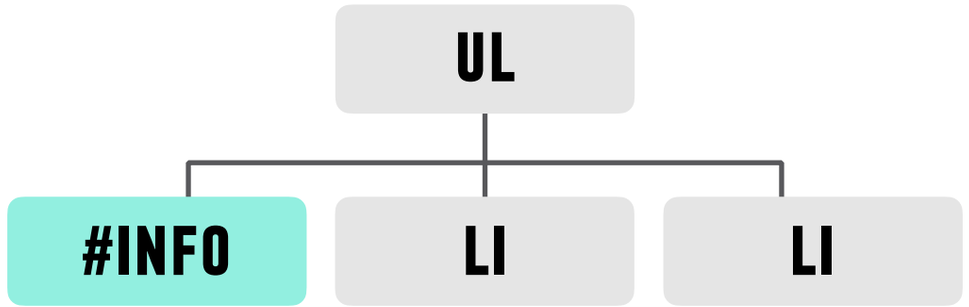
JQUERY METHODS — WORKING WITH THOSE ELEMENTS

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

See your handout or the jQuery docs for list!

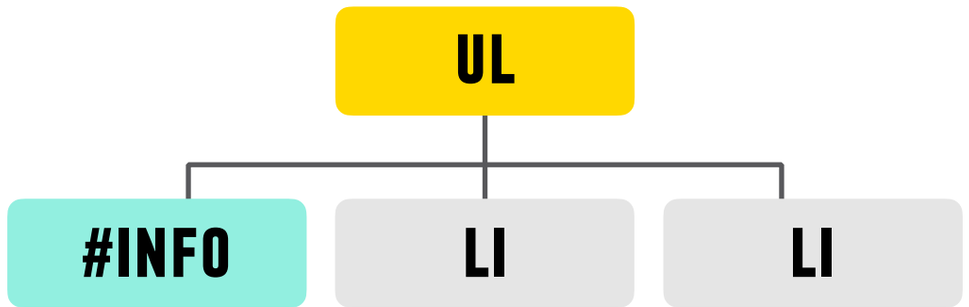
TRAVERSING THE DOM?

```
$('#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
`.find()` *finds all descendants*	`$('h1').find('a');`
`.parent()`	`$('#box1').parent();`
`.siblings()`	`$('p').siblings('.important');`
`.children()`	`$('ul').children('li');`

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:

See your handout or the jQuery docs for list!

GETTING/SETTING CONTENT — PART 1

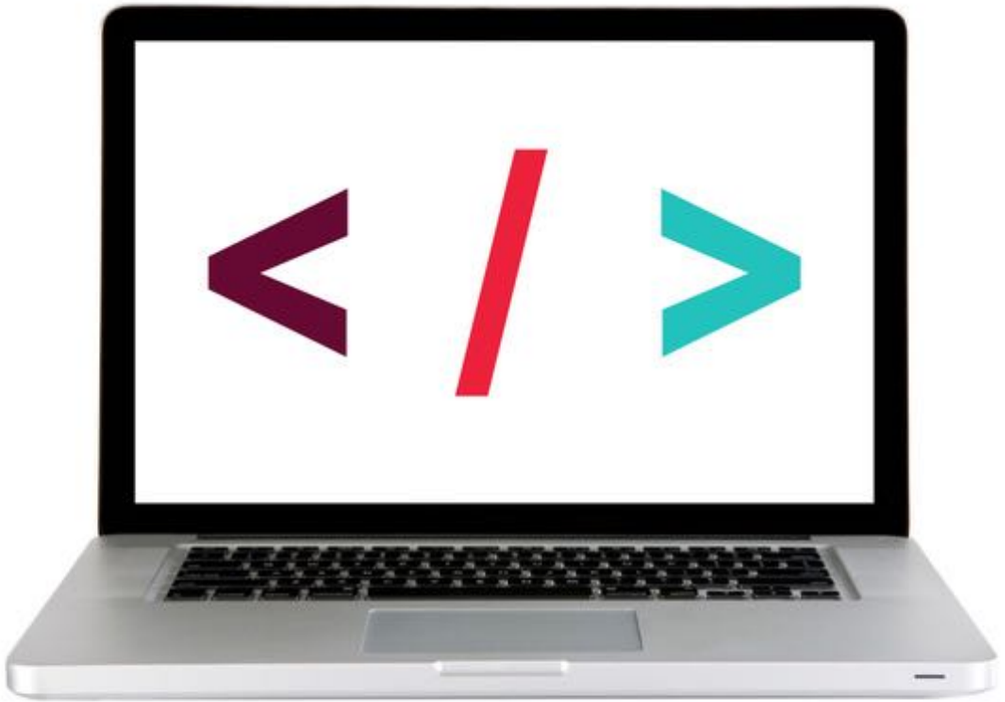
GET/SET CONTENT

Get/change content of elements and attributes

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

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

LET'S TAKE A CLOSER LOOK



GETTING/SETTING CONTENT — PART 2

GET/SET CONTENT

Get/change content of elements and attributes

METHODS	EXAMPLES
`.addClass()`	`$('p').addClass('success');`
`.removeClass()`	`$('p').removeClass('my-class-here');`
`.toggleClass()`	`$('p').toggleClass('special');`

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

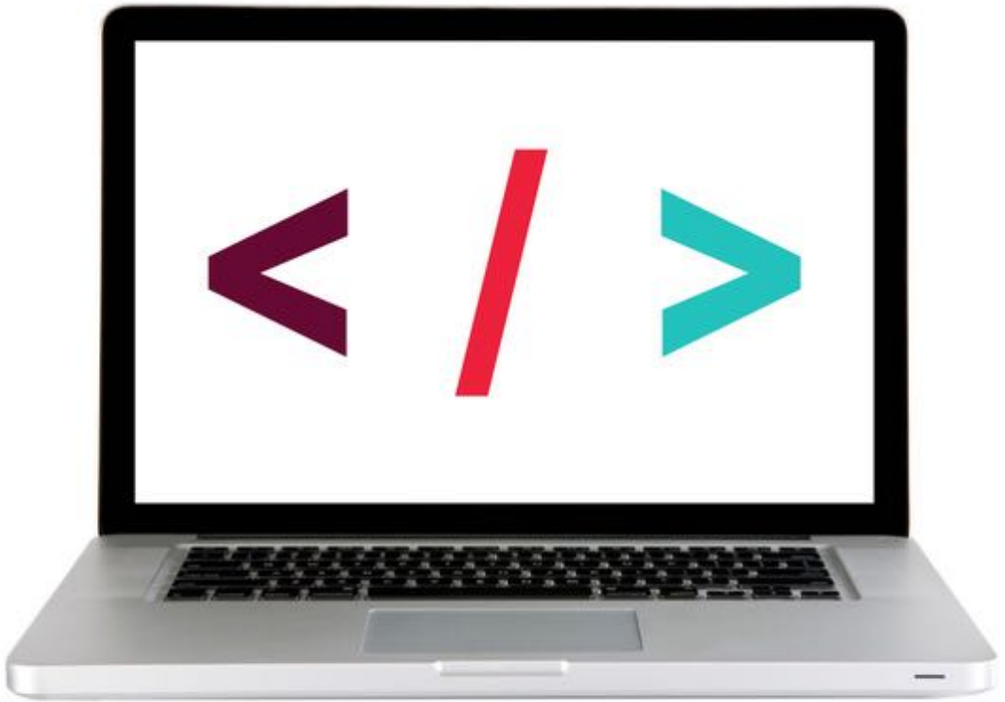
JQUERY METHODS — GETTING/SETTING CONTENT

GET/SET CONTENT

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

NO PERIOD!!!

LET'S TAKE A CLOSER LOOK



ACTIVITY

KEY OBJECTIVE

- Utilize jQuery to access and manipulate DOM elements.

TYPE OF EXERCISE

- Individual/Partner

TIMING

5 min

Exercise is in `1-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:

See your handout or the jQuery docs for list!

JQUERY METHODS — EFFECTS/ANIMATION

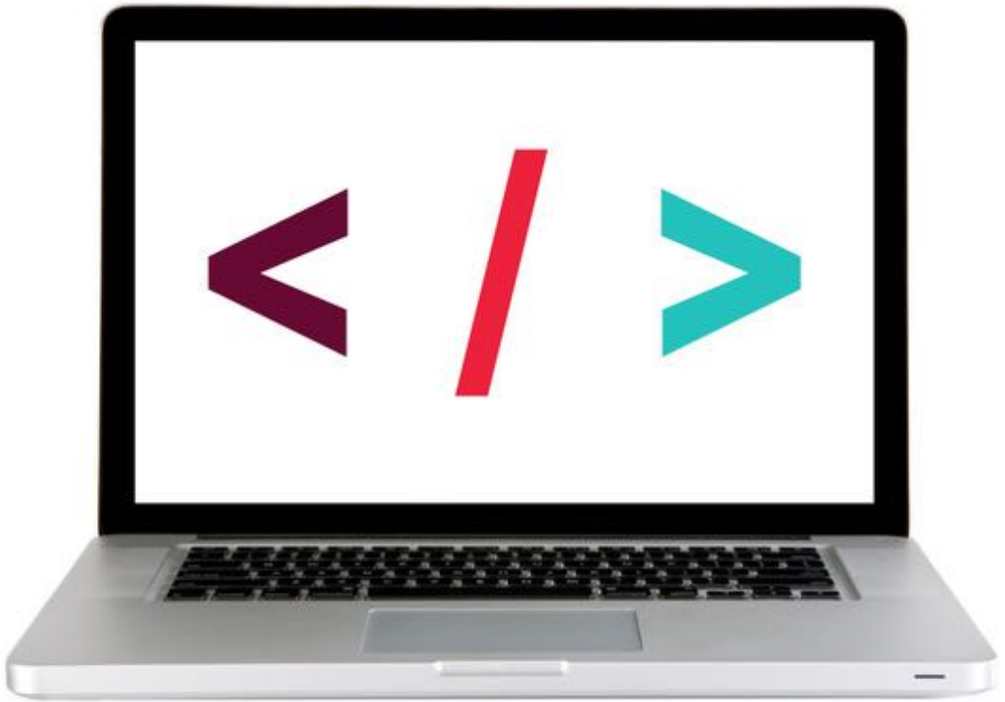
ADD EFFECTS/ ANIMATION

Add effects and animation to parts of the page

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

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:

See your handout or the jQuery docs for list!

JQUERY METHODS — EVENTS!

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

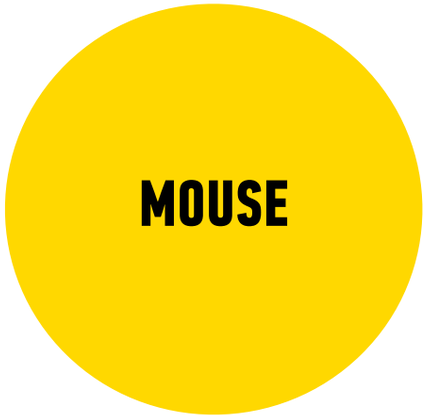
```
$('li').on('click', function() {
  // your code here
});
```

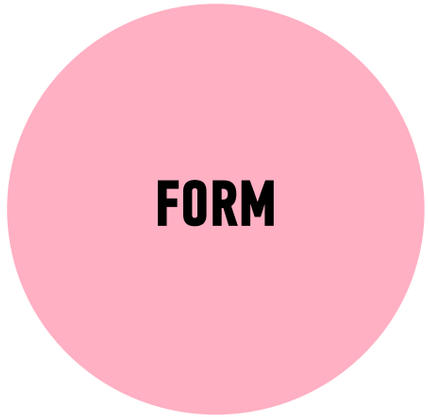
JQUERY METHODS — EVENTS!

CREATE EVENT LISTENERS

type of event

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





click

dblclick

mouseenter

mouseleave

keypress

keydown

keyup

submit

change

focus

blur

resize

scroll

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

JQUERY METHODS — EVENTS!

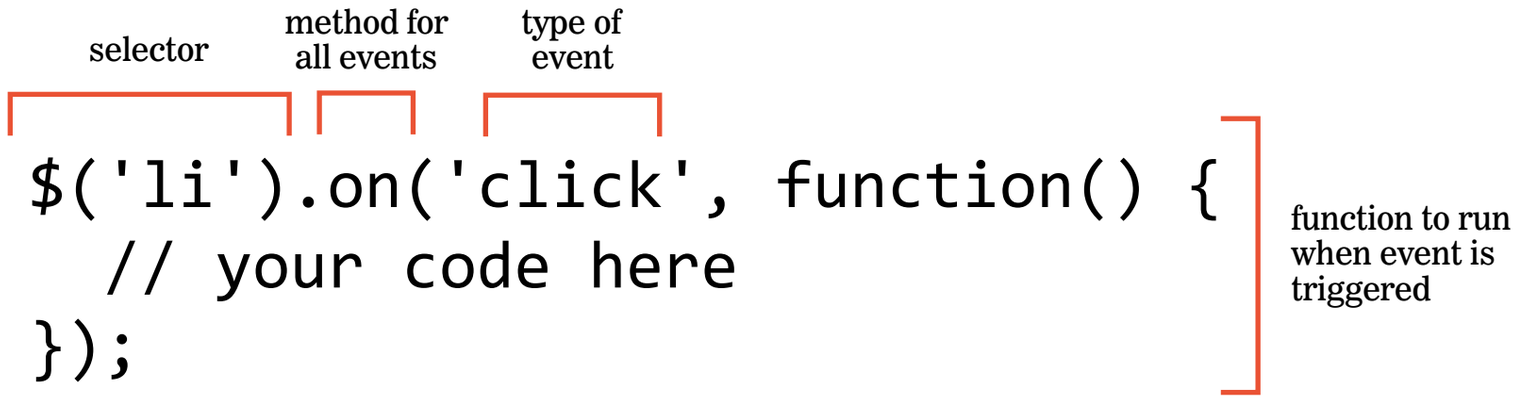
CREATE EVENT LISTENERS

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

function to run when event is triggered

JQUERY METHODS — EVENTS!

CREATE EVENT LISTENERS



ACTIVITY

KEY OBJECTIVE

- Utilize jQuery to access and manipulate DOM elements.

TYPE OF EXERCISE

- Individual/Partner

TIMING

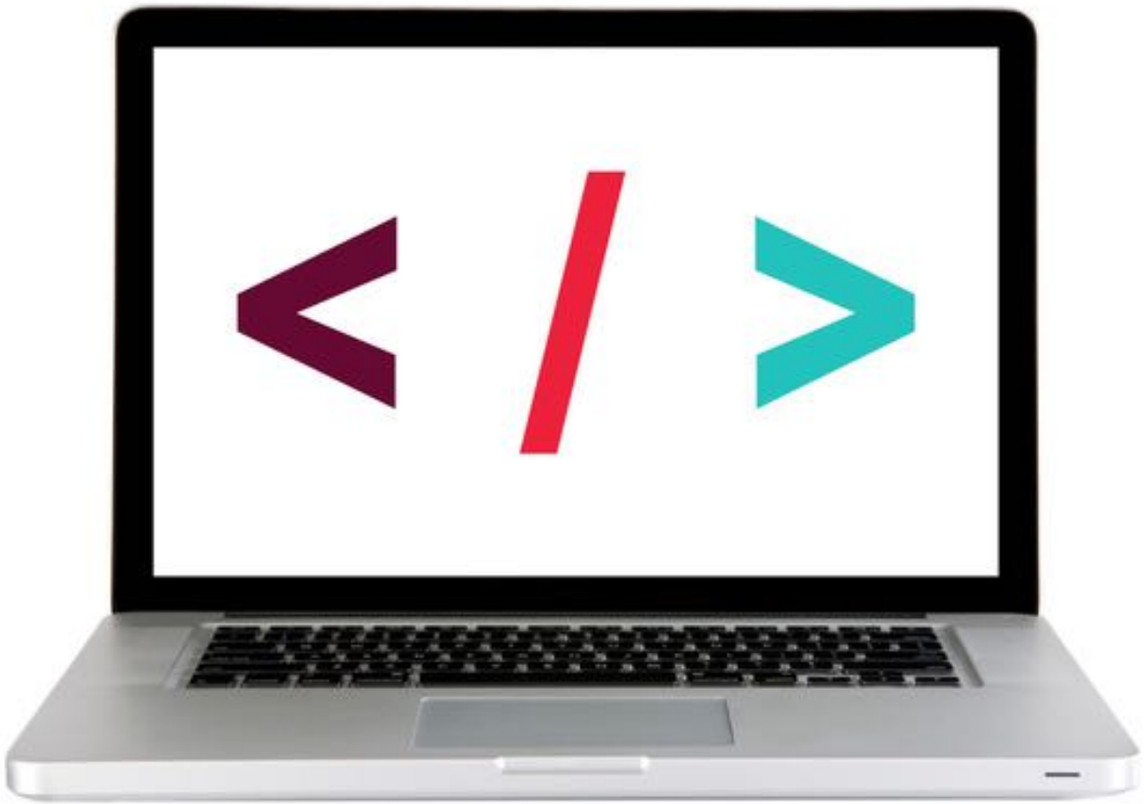
5 min Continue with `1-jquery-exercise`

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

OBJECTIVE

- Manipulate the DOM by using jQuery selectors and functions.

LOCATION

- `starter-code > 3-jquery—reminders-list`

TIMING

10 min

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.

BEST PRACTICES

METHOD CHAINING

CHAINING

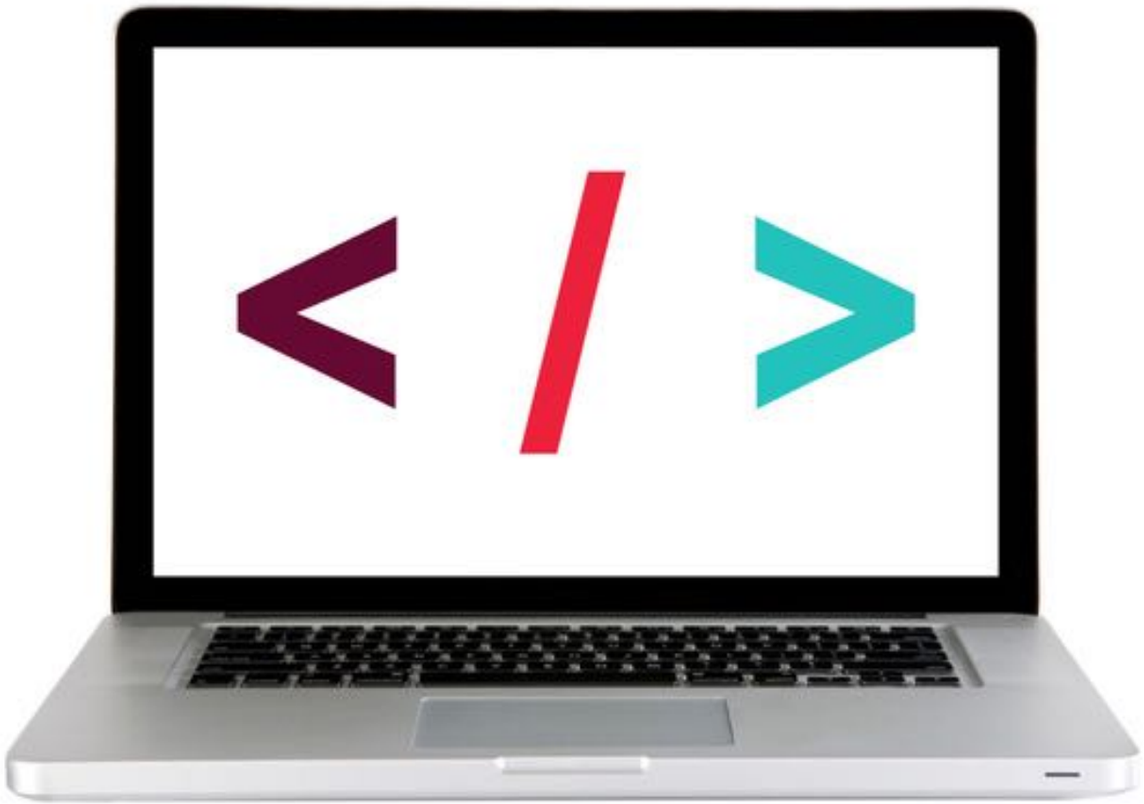
without chaining:

```
var $mainCaption = $('<p>');
var $captionWithText = $mainCaption.html('Today');
var $fullCaption = $captionWithText.addClass('accent');
```

with chaining:

```
var $fullCaption = $('<p>').html('Today').addClass('accent');
```

INTRO TO JQUERY



LET'S TAKE A CLOSER LOOK

EXERCISE - CHAINING

OBJECTIVE

- Use chaining to place methods on selectors.

LOCATION

- `starter-code > 5-best-practices-exercise`

TIMING

3 min

1. In your browser, open index.html and test the functionality.
2. Open main.js in your editor and complete items 1 and 2.
3. In your browser, reload index.html and verify that the functionality is unchanged.

IMPLICIT ITERATION

IMPLICIT ITERATION

explicit iteration

selects a jQuery collection

`.each()` method works like a `forEach` loop

```
$('li').each(function() {
 $(this).removeClass('current');
});
```

not necessary for element collections

implicit iteration

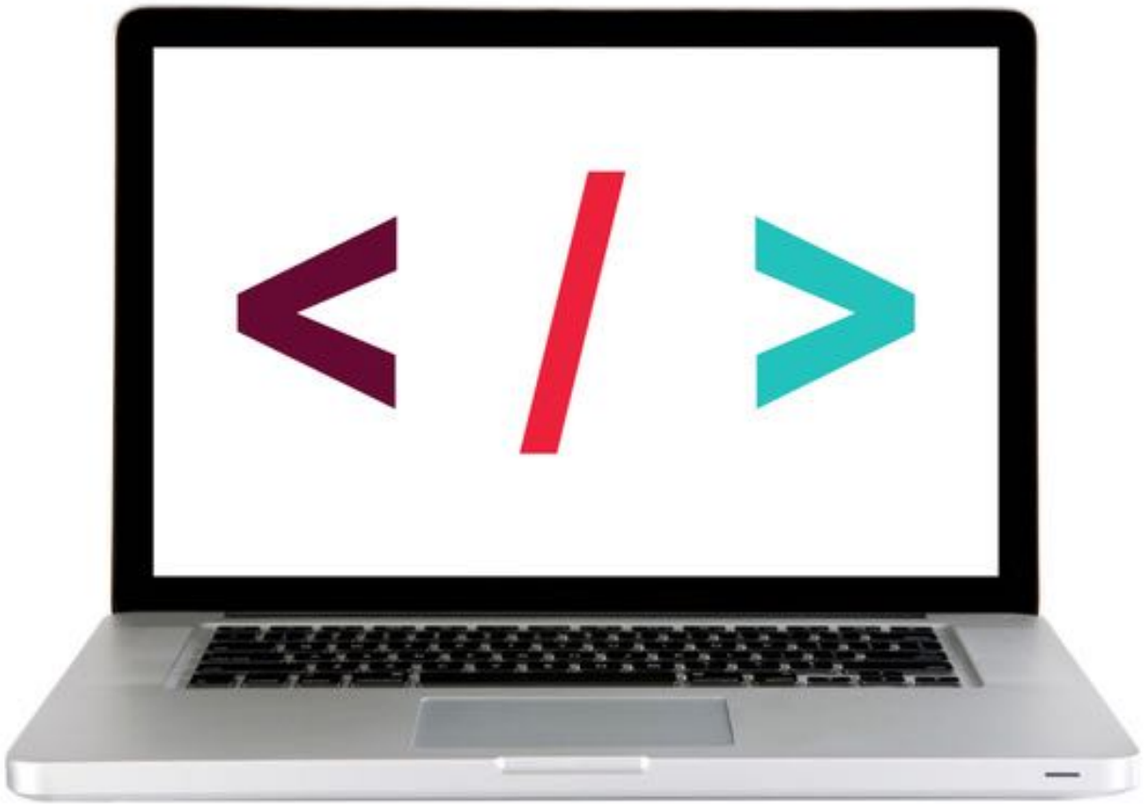
selects a jQuery collection

applying any method to a jQuery collection iterates through each element

```
$('li').removeClass('current');
```

less code = best practice!

INTRO TO JQUERY



LET'S TAKE A CLOSER LOOK

EXERCISE - IMPLICIT ITERATION

OBJECTIVE

- Use implicit iteration to update elements of a jQuery selection.

LOCATION

- `starter-code > 5-best-practices-exercise`

TIMING

5 min

1. Return to main.js in your editor and complete item 3.
2. In your browser, reload index.html and verify that the functionality is unchanged.

EVENT DELEGATION

WITHOUT EVENT DELEGATION

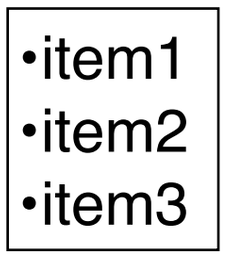
add an event listener to each li in the DOM

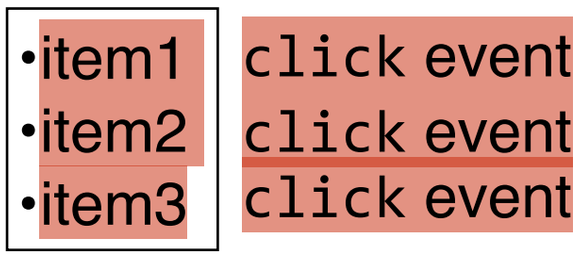
1. load page

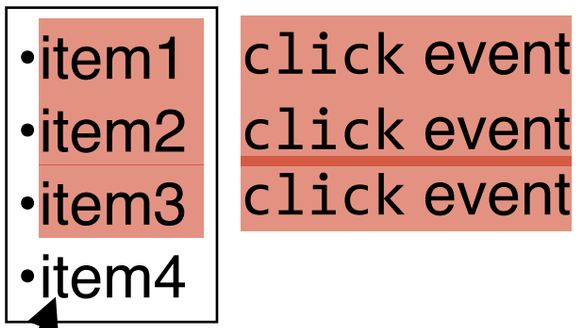
2. set event listener on list items

3. add a new list item

```
$('li').on('click',function(){
  addClass('selected')
});
```







click event is not automatically applied to the new li element

WITH EVENT DELEGATION

1. load page

- item1
- item2
- item3

2. set event listener on parent of list items

selector changed from 'li' to 'ul'

new argument 'li' added to on() method

```
$('ul').on('click', 'li', function(){
  addClass('selected')
});
```

add an event listener to the ul element that applies to all of its li descendants

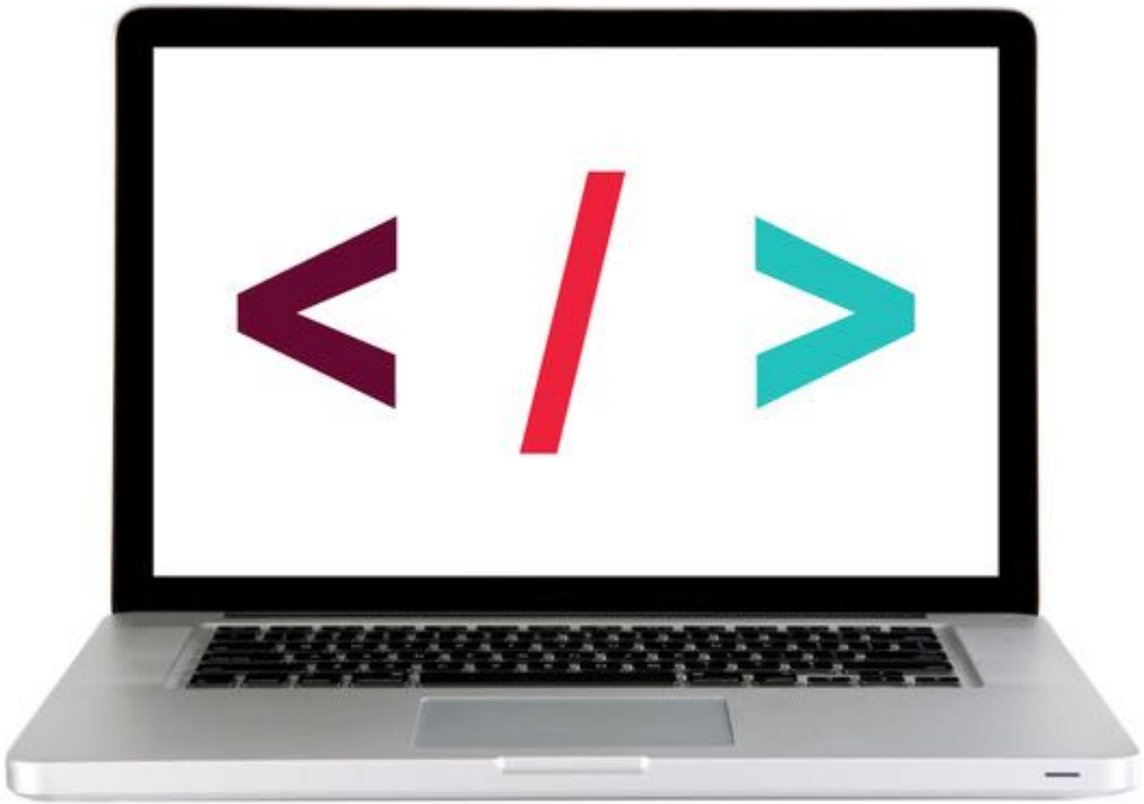
- item1 — click event
- item2 — click event
- item3 — click event

3. add a new list item

- item1 — click event
- item2 — click event
- item3 — click event
- item4 — click event

click event IS automatically applied to the new `li` element!

INTRO TO JQUERY



LET'S TAKE A CLOSER LOOK

EXERCISE - EVENT DELEGATION

OBJECTIVE

- Use event delegation to manage dynamic content.

LOCATION

- `starter-code > 5-best-practices-exercise`

TIMING

10 min

1. Return to main.js in your editor and complete items 4a and 4b.
2. In your browser, reload index.html and verify that when you add a new item to the list, its “cross off” link works.
3. BONUS: When the user mouses over each item, the item should turn grey. Don't use CSS hovering for this.
4. BONUS: Add another link, after each item, that allows you to delete the item.

ATTACHING MULTIPLE EVENTS WITH A SINGLE ON() STATEMENT

ATTACHING MULTIPLE EVENTS WITH A SINGLE .ON() STATEMENT

- We could write a separate `.on()` statement for each event on an element:

```
var $listElement = $(‘#contents-list');

$listElement.on(‘mouseenter', ‘li’, function(event) {
  $(this).siblings().removeClass(‘active’);
  $(this).addClass('active');
});

$listElement.on('mouseleave', ‘li’, function(event) {
  $(this).removeClass('active');
});
```

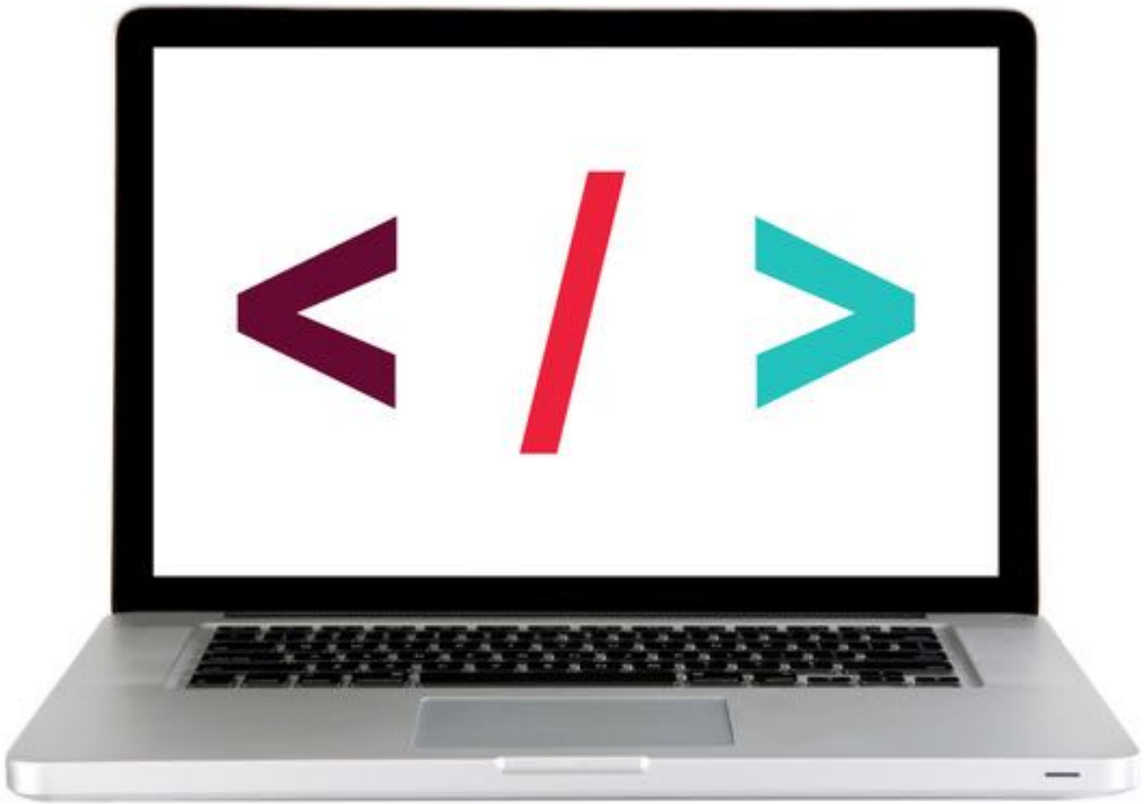
ATTACHING MULTIPLE EVENTS WITH A SINGLE .ON() STATEMENT

- Grouping all the events for an element in a single `.on()` statement makes our code more organized and is faster

```
var $listElement = $('#contents-list');

$listElement.on('mouseenter mouseleave', 'li', function(event) {
  if (event.type === 'mouseenter') {
    $(this).siblings().removeClass('active');
    $(this).addClass('active');
  } else if (event.type === 'mouseleave') {
    $(this).removeClass('active');
  }
});
```

INTRO TO JQUERY



LET'S TAKE A CLOSER LOOK

EXERCISE - ATTACHING MULTIPLE EVENTS

LOCATION

▸ `starter-code > 6-multiple-events-exercise`

TIMING

5 min

1. In your browser, open index.html. Move the mouse over each list item and verify that the sibling items turn gray.
2. In your editor, open main.js and refactor the two event listeners near the bottom of the file into a single event listener for multiple events.
3. In your browser, reload index.html and verify that the functionality is unchanged.

Exit Tickets!

(Class #9)

LEARNING OBJECTIVES - REVIEW

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

NEXT CLASS PREVIEW

Ajax & APIs

- Identify all the HTTP verbs & their uses.
- Describe APIs and how to make calls and consume API data.
- Access public APIs and get information back.
- Implement an Ajax request with Fetch.
- Create an Ajax request using jQuery.
- Reiterate the benefits of separation of concerns – API vs. Client.

Q&A