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-10-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

- Use event delegation to manage dynamic content.
- Use implicit iteration to update elements of a jQuery selection
- Build content programmatically using template literals

AGENDA

- jQuery best practices
- Template literals

WEEKLY OVERVIEW

WEEK 6	Advanced jQuery / Ajax & APIs
WEEK 7	Asynchronous JavaScript & Callbacks / Advanced APIs
WEEK 8	Project 2 Lab / Closures & the module pattern

EXIT TICKET QUESTIONS

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.

EXERCISE — CATCH PHRASE

EXERCISE

TYPE OF EXERCISE

- Groups of 3

TIMING

5 min

1. Describe the term on one of your slips of paper **without saying the term itself**.
2. Take turns so everyone gets a chance to give clues.

BEST PRACTICES

METHOD CHAINING

CHAINING

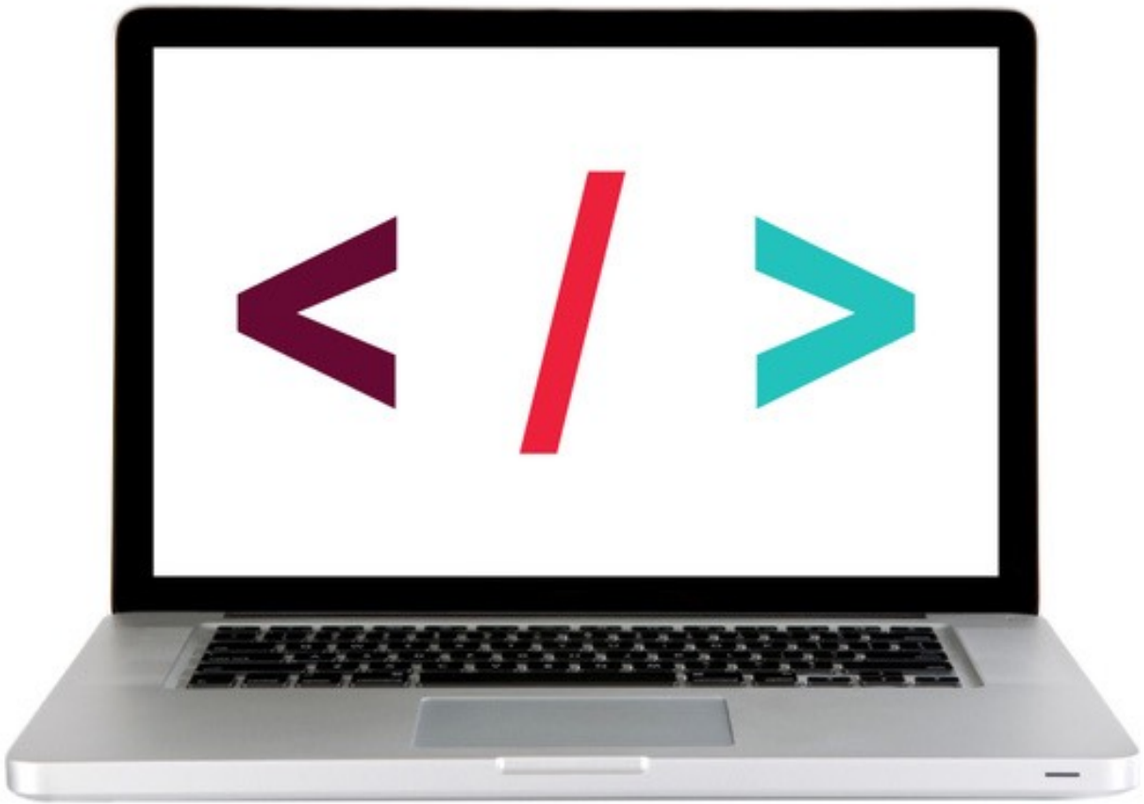
without chaining:

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

with chaining:

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

ADVANCED JQUERY



LET'S TAKE A CLOSER LOOK

EXERCISE - CHAINING

OBJECTIVE

- Use chaining to place methods on selectors.

LOCATION

- `starter-code > 1-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

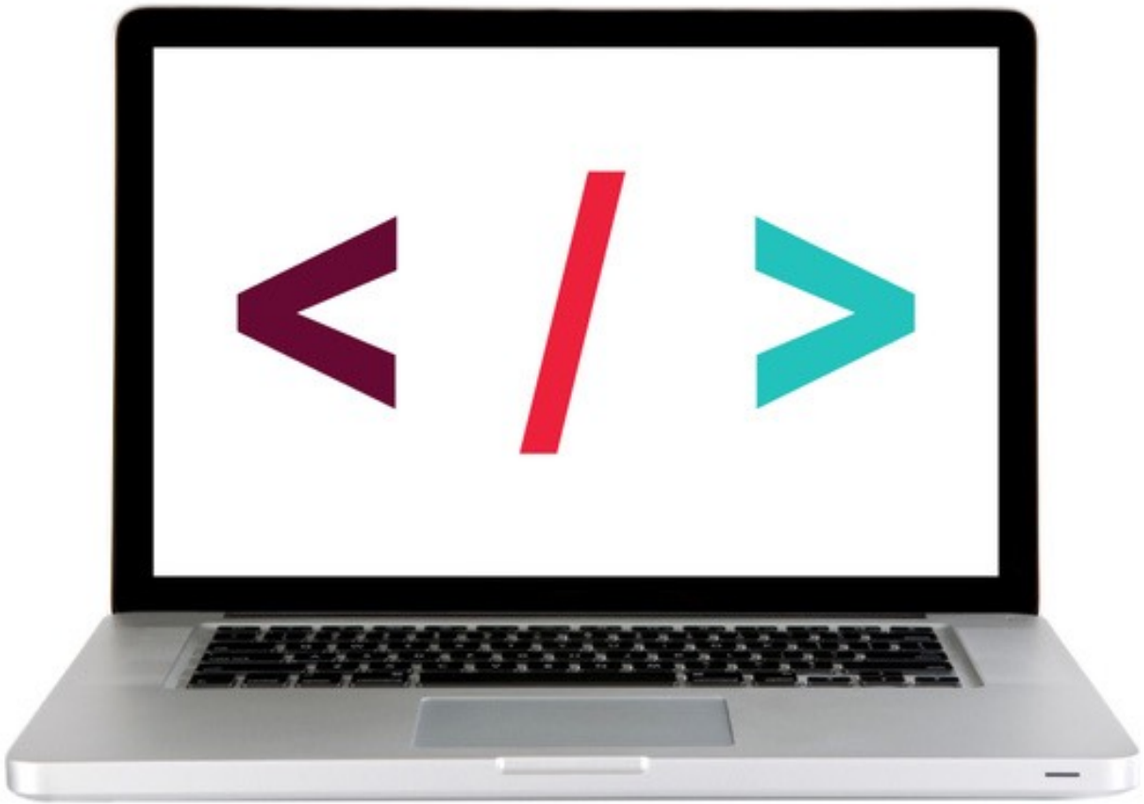
selects a jQuery collection

applying any method to a jQuery collection iterates through each element

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

less code = best practice!

ADVANCED JQUERY



LET'S TAKE A CLOSER LOOK

EXERCISE - IMPLICIT ITERATION

OBJECTIVE

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

LOCATION

- `starter-code > 1-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

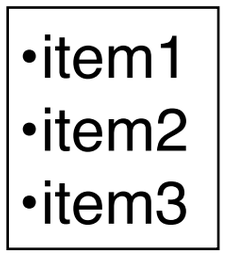
WITHOUT EVENT DELEGATION

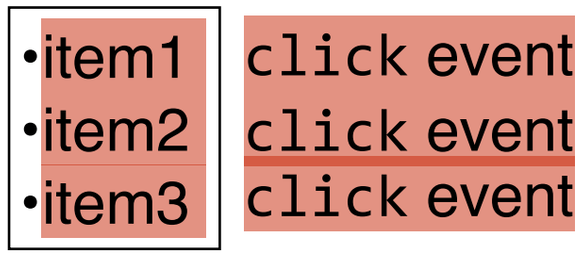
1. load page

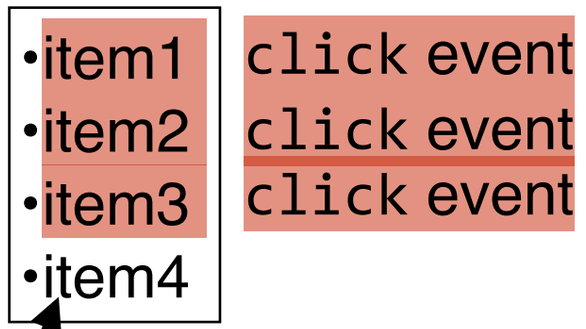
2. set event listener on list items

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

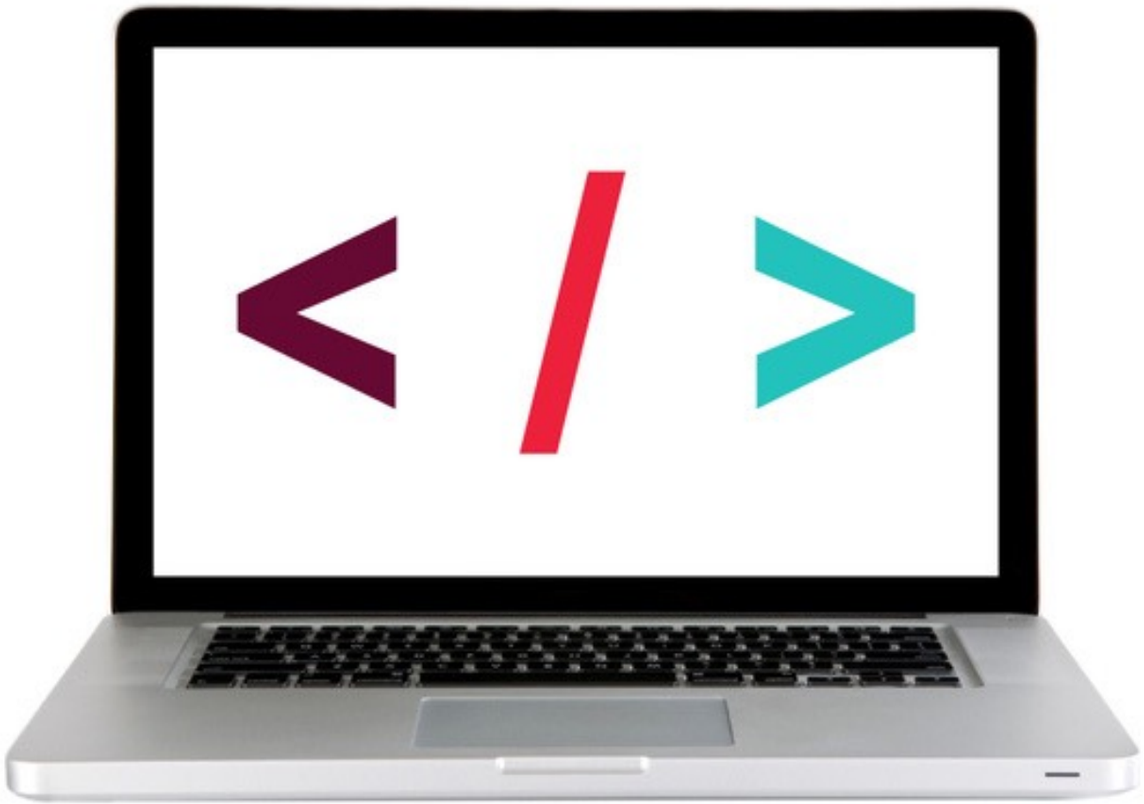
- item1 — click event
- item2 — click event
- item3 — click event

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!

ADVANCED JQUERY



LET'S TAKE A CLOSER LOOK

EXERCISE - EVENT DELEGATION

OBJECTIVE

- Use event delegation to manage dynamic content.

LOCATION

- `starter-code > 1-best-practices-exercise`

TIMING

10 min

1. Return to main.js in your editor and complete item 4.
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 1: When the user mouses over each item, the item should turn grey. Don't use CSS hovering for this.
4. BONUS 2: 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

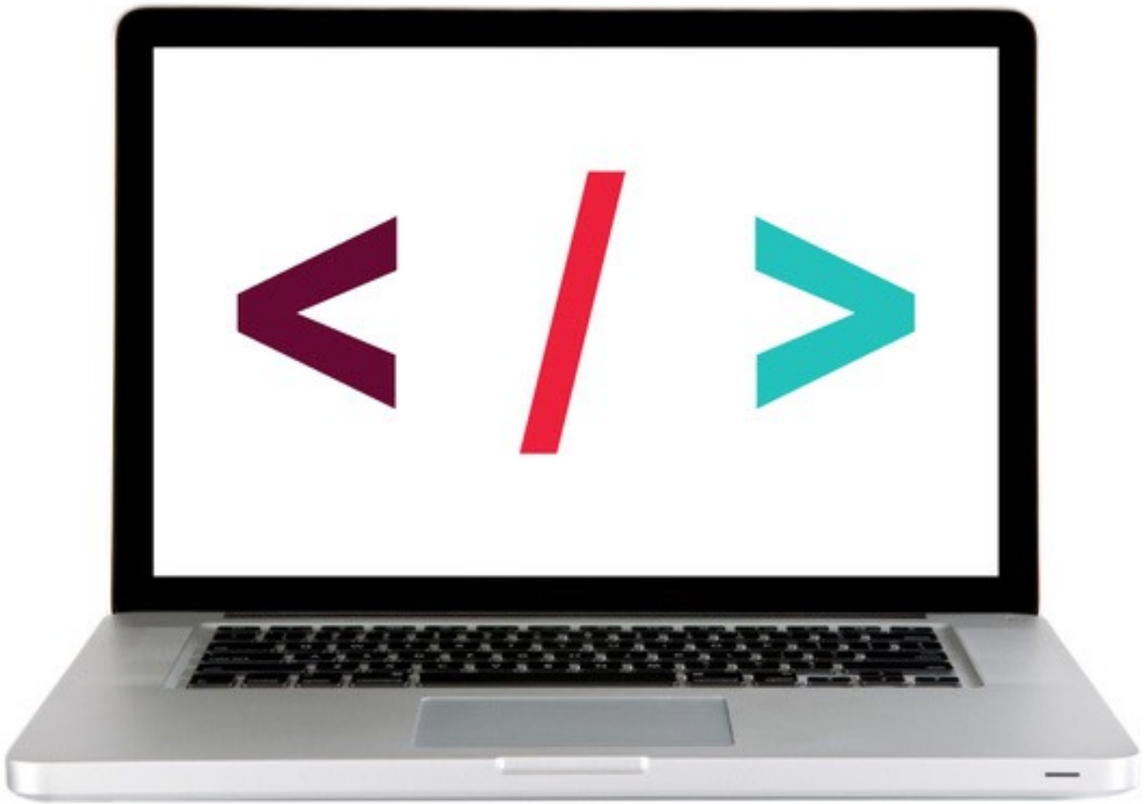
- We could write a separate `.on()` statement for each event on an element:

```
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

```
const $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');
  }
});
```

ADVANCED JQUERY



LET'S TAKE A CLOSER LOOK

EXERCISE - ATTACHING MULTIPLE EVENTS

LOCATION

▸ `starter-code > 2-multiple-events-exercise`

TIMING

5 min

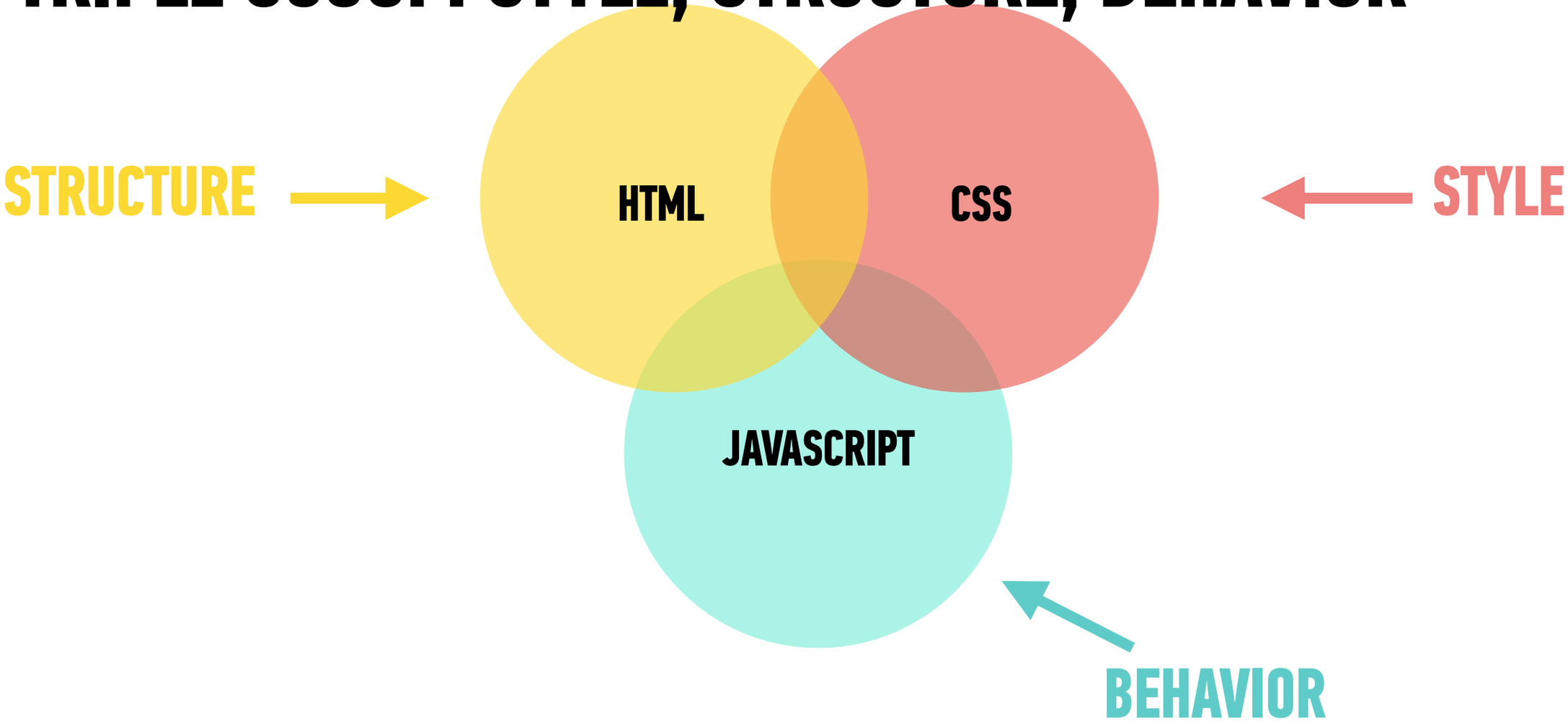
1. In your browser, open index.html. Move the mouse over each list item and verify that the sibling items turn gray.
2. In your editor, open main.js and refactor the two event listeners near the bottom of the file into a single event listener for multiple events.
3. In your browser, reload index.html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

TEMPLATING

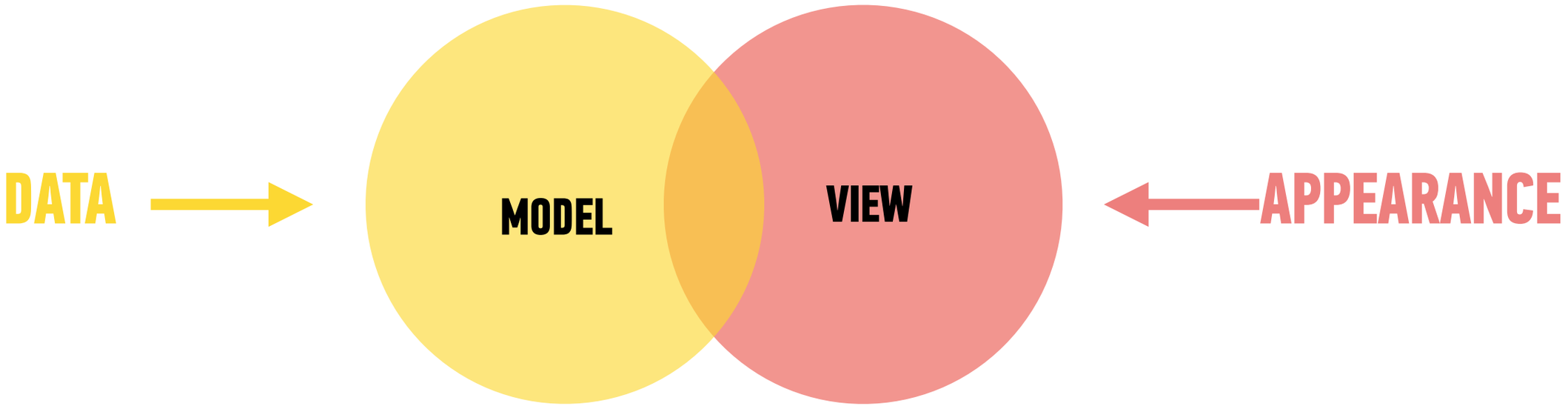
SEPARATION OF CONCERNS

- Programming principle of keeping different aspects (or **concerns**) of an application separate
- Many ways to do this
- One common separation is between data (the information we're presenting) and view (the code that determines how data is presented)
- We should be able to change the code for one concern without affecting the code for the other

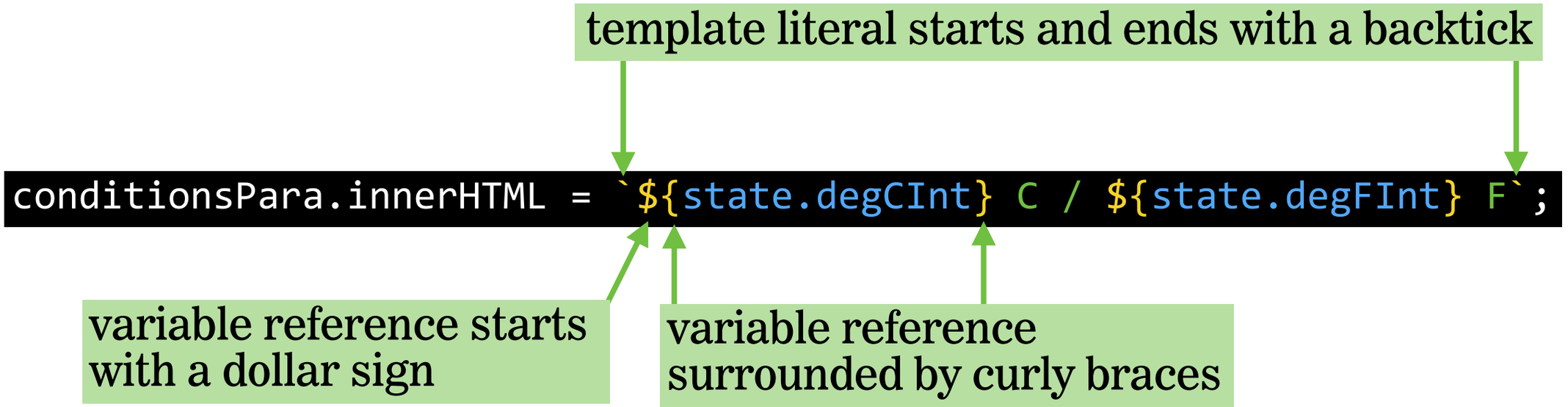
TRIPLE SCOOP: STYLE, STRUCTURE, BEHAVIOR



MODEL VS VIEW



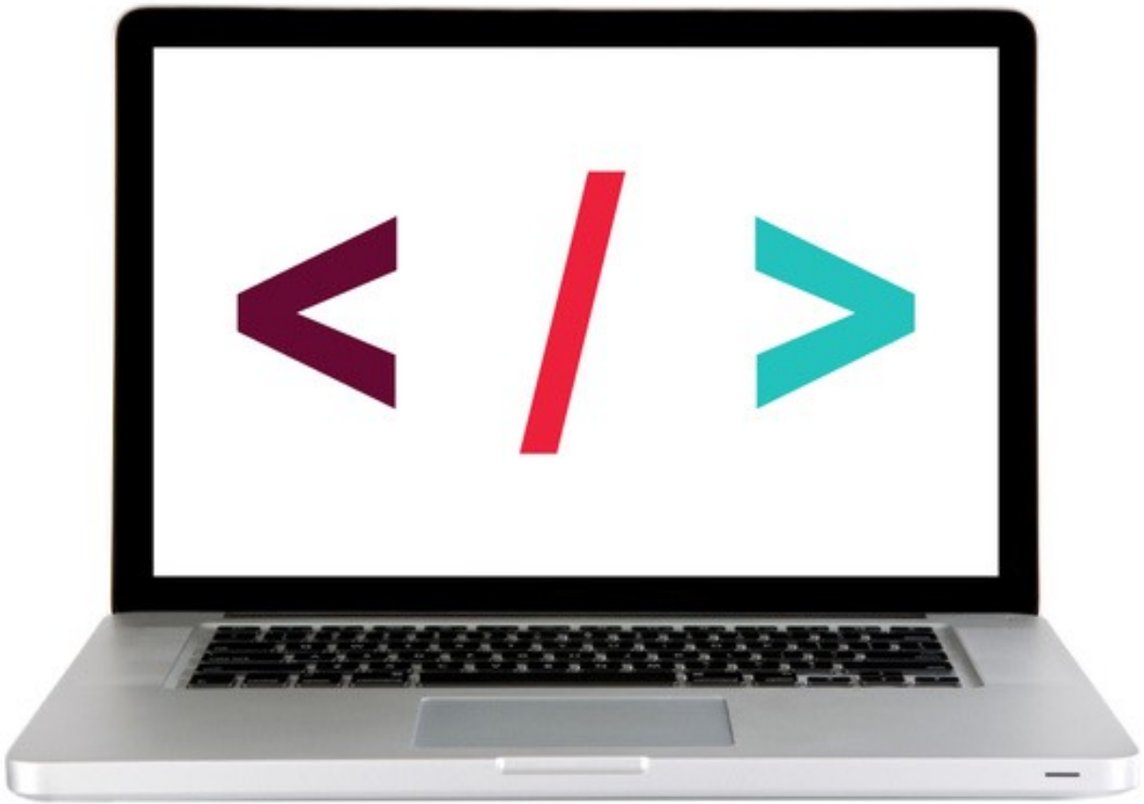
TEMPLATE LITERALS



CREATING A TEMPLATE LITERAL

1. Create or reference an object/array/other variable that stores the content
2. Create the template literal
3. Add the template literal to the DOM

ADVANCED JQUERY



LET'S TAKE A CLOSER LOOK

EXERCISE - TEMPLATING

LOCATION

- `starter-code > 5-templating-lab`

TIMING

10 min

1. Create a template literal and use it to display the data in the `favorite` object.
2. Use the HTML structure shown in main.js.
3. BONUS: create a template literal that displays the contents of the 'favorites' object at the bottom of main.js.

Exit Tickets!

(Class #9)

LEARNING OBJECTIVES - REVIEW

- Use event delegation to manage dynamic content.
- Use implicit iteration to update elements of a jQuery selection
- Build content programmatically using template literals

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