GENERAL ASSEMBLY

JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

HELLO!

1. Pull changes from the `svodnik/JS-SF-12-resources` repo to your computer
2. Open the `08-events-jquery > starter-code` folder in your code editor

JAVASCRIPT DEVELOPMENT

EVENTS & JQUERY

LEARNING OBJECTIVES

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

- Manipulate the DOM using jQuery selectors and methods.
- Use event delegation to manage dynamic content.
- Use implicit iteration to update elements of a jQuery selection

AGENDA

- Creating and appending DOM nodes
- Event delegation
- Implicit iteration

WEEKLY OVERVIEW

WEEK 5	Events & jQuery / Ajax & APIs
WEEK 6	Asynchronous JS & callbacks / Advanced APIs
WEEK 7	Project 2 Lab / Prototypal inheritance

EXIT TICKET QUESTIONS

1. Is DOM the basis for react?
2. I would like to know how the lesson contents would be applicable for a would be developer
3. Suggestions:
   - As much vanilla javascript as possible :)
   - set amount of time allotted for the break to last ?

HOMEWORK REVIEW

HOMEWORK — GROUP DISCUSSION

EXERCISE

TYPE OF EXERCISE

- Groups of 3

TIMING

4 min

1. Share your solutions for the homework.
2. 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 2-3

TIMING

5 min

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

DOM & JQUERY: REVIEW

JQUERY METHODS — EVENTS!

CREATE EVENT LISTENERS

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

JQUERY METHODS — EVENTS!

CREATE EVENT LISTENERS

$$\overbrace{\texttt{\$('li')}}^{\text{selector}}$$

```
$('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	KEYBOARD	FORM	DOCUMENT
click	keypress	submit	resize
dblclick	keydown	change	scroll
mouseenter	keyup	focus	
mouseleave		blur	

```
$('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

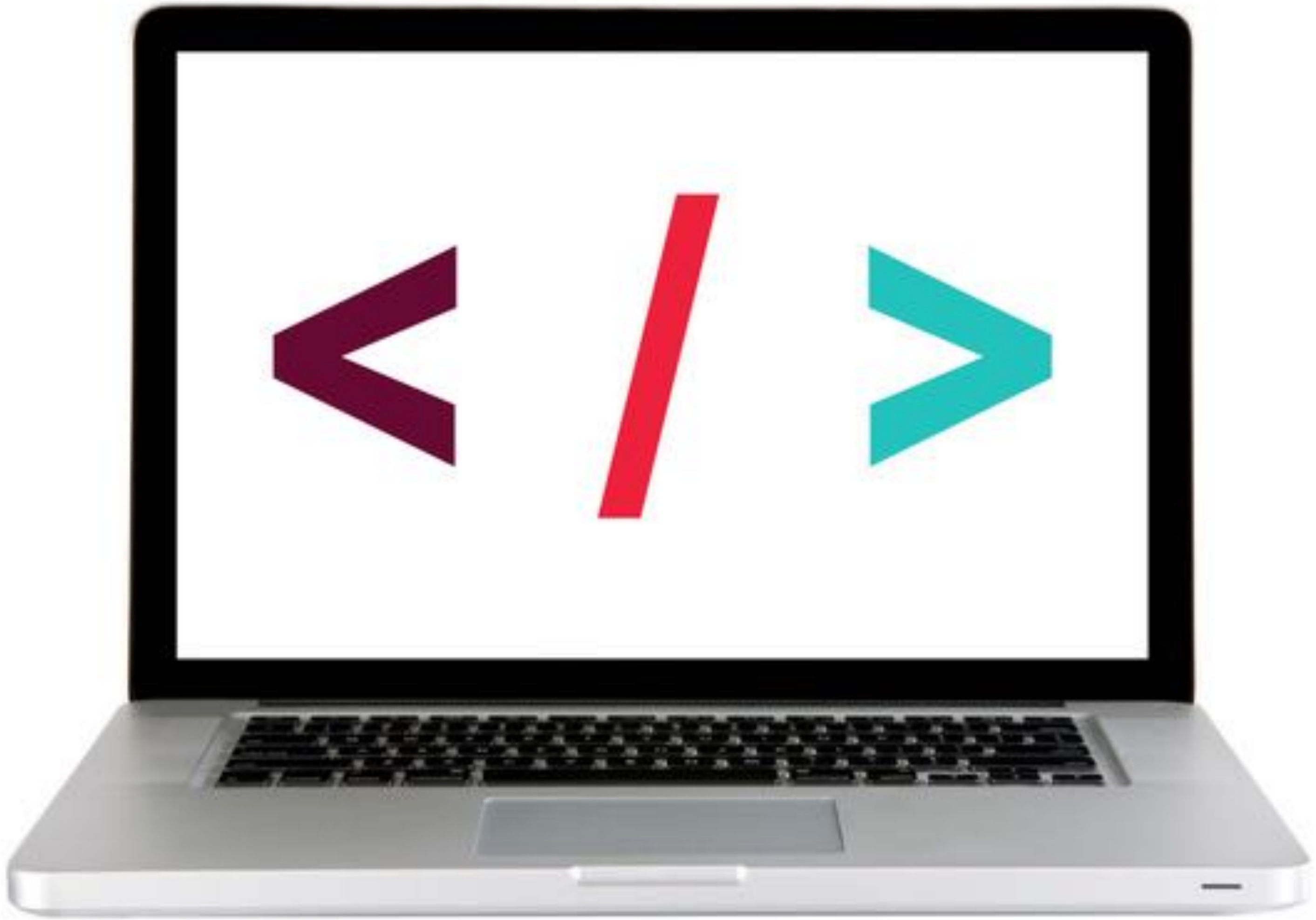
method for all events

type of event

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

function to run when event is triggered

LET'S TAKE A LOOK



ACTIVITY

EXERCISE

KEY OBJECTIVE

- Create DOM event handlers to respond to user actions

TYPE OF EXERCISE

- Individual/Partner

AS A CLASS

10 min

Exercise is in `1-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 handout/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.

CREATING & APPENDING DOM NODES

document.ready()

- specifies code to run only after the DOM has finished loading
- Syntax:

```
$(document).ready(function() {
  // code goes here
});
```

- Shorthand version (best practice):

```
$(function() {
  // code goes here
});
```

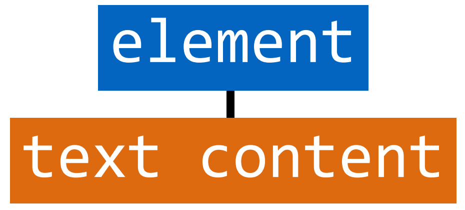
Adding content to the DOM

1. create a new element with `$(‘<element>’)`

`element`

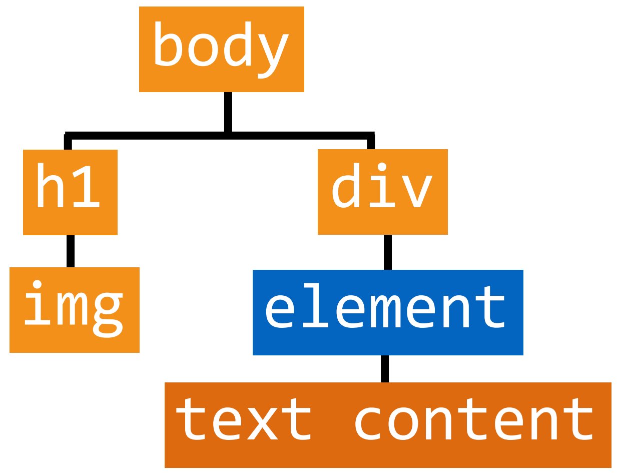
Adding content to the DOM

1. create a new element with `$('<element>')`
2. add new content to that element with `.text()` or `.html()`



Adding content to the DOM

1. create a new element with `$('<element>')`
2. add new content to that element with `.text()` or `.html()`
3. attach the new element to the DOM with `.append()`



$(‘<element>’)

- Creates a new element

```
$(‘<li>’); // creates an li element
```

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

```
let item1 = $(‘<li>’);
let item2 = $(‘<li>’);
```

`.text()` or `.html()`

- Creates and adds text content as the child of an element
- Easiest to add method to same statement that creates element

```
let item1 = $(‘<li>’).text(‘banana’);
let item2 = $(‘<li>’).text(‘apple’);
```

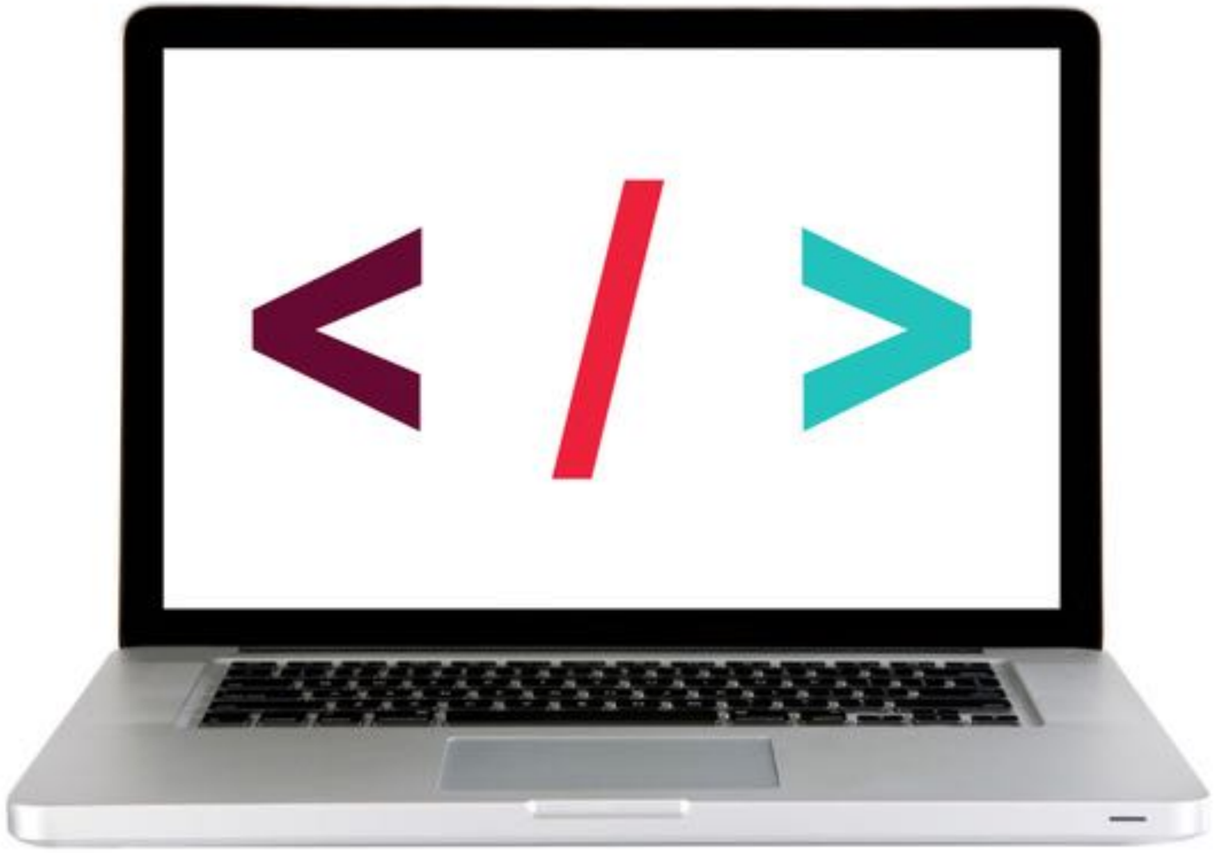
```
let item1 = $(‘<li>’).html(‘<strong>Every</strong> dinosaur’);
let item2 = $(‘<li>’).html(‘Books (<em>not</em> ebooks)’);
```

.append()

- Attaches element or node as child of specified element
  - » Attaching to a DOM element makes it part of the DOM
- Syntax:
  `$(`*`parent`*`).append(`*`child`*`);`

```
const list = $(‘ul’);  // selects ul element
list.append(item1);    // adds item1 li to list ul
list.append(item2);    // adds item2 li to list ul
```

EVENTS & JQUERY



LET'S TAKE A CLOSER LOOK

EXERCISE - ADD CONTENT TO A WEB PAGE USING JQUERY

EXERCISE

LOCATION

- `starter-code > 3-create-append—exercise`

TIMING

10 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.

WORKING WITH EVENT OBJECTS

`preventDefault()`

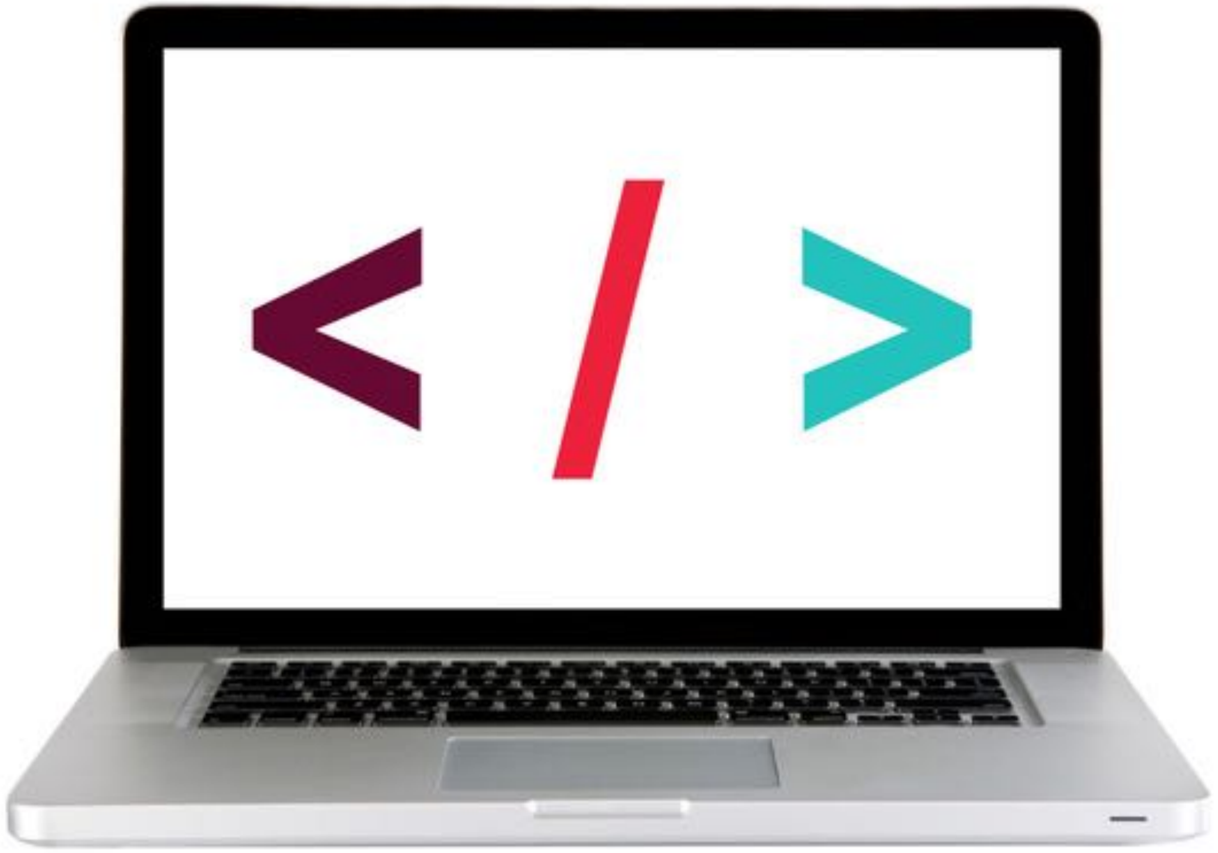
- Prevents element from executing default behavior in response to an event

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();
  ...
}
```

EVENTS & JQUERY



LET'S TAKE A CLOSER LOOK

EXERCISE

EXERCISE

LOCATION

- `starter-code > 5-event object—exercise`

TIMING

2 min

1. Update the code to prevent the form from submitting when the button is clicked.
2. Test your code in the browser and check the URL to verify that the form is not being submitted.

BEST PRACTICES

METHOD CHAINING

CHAINING

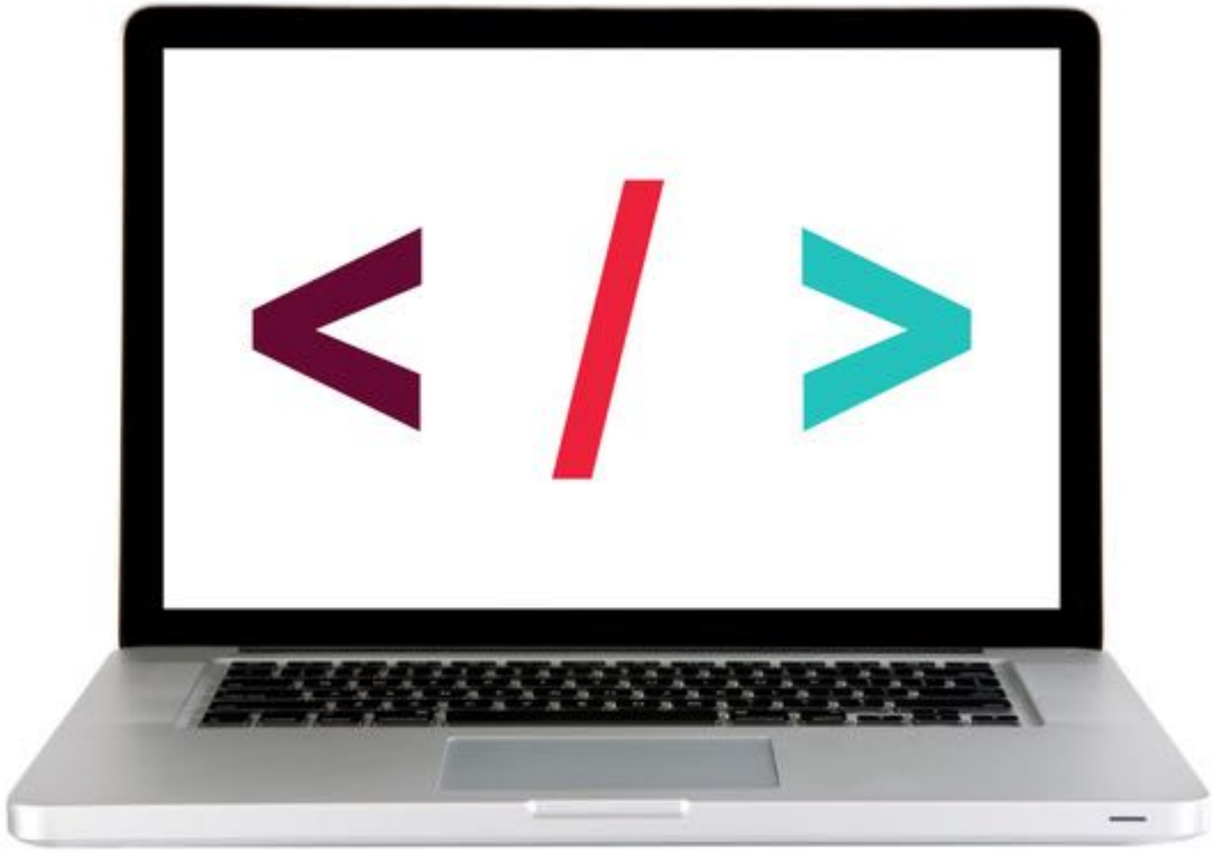
without chaining:

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

with chaining:

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

EVENTS & JQUERY



LET'S TAKE A CLOSER LOOK

EXERCISE - CHAINING

EXERCISE

OBJECTIVE

- Use chaining to place methods on selectors.

LOCATION

- `starter-code > 7-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

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

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

✗ not necessary for element collections

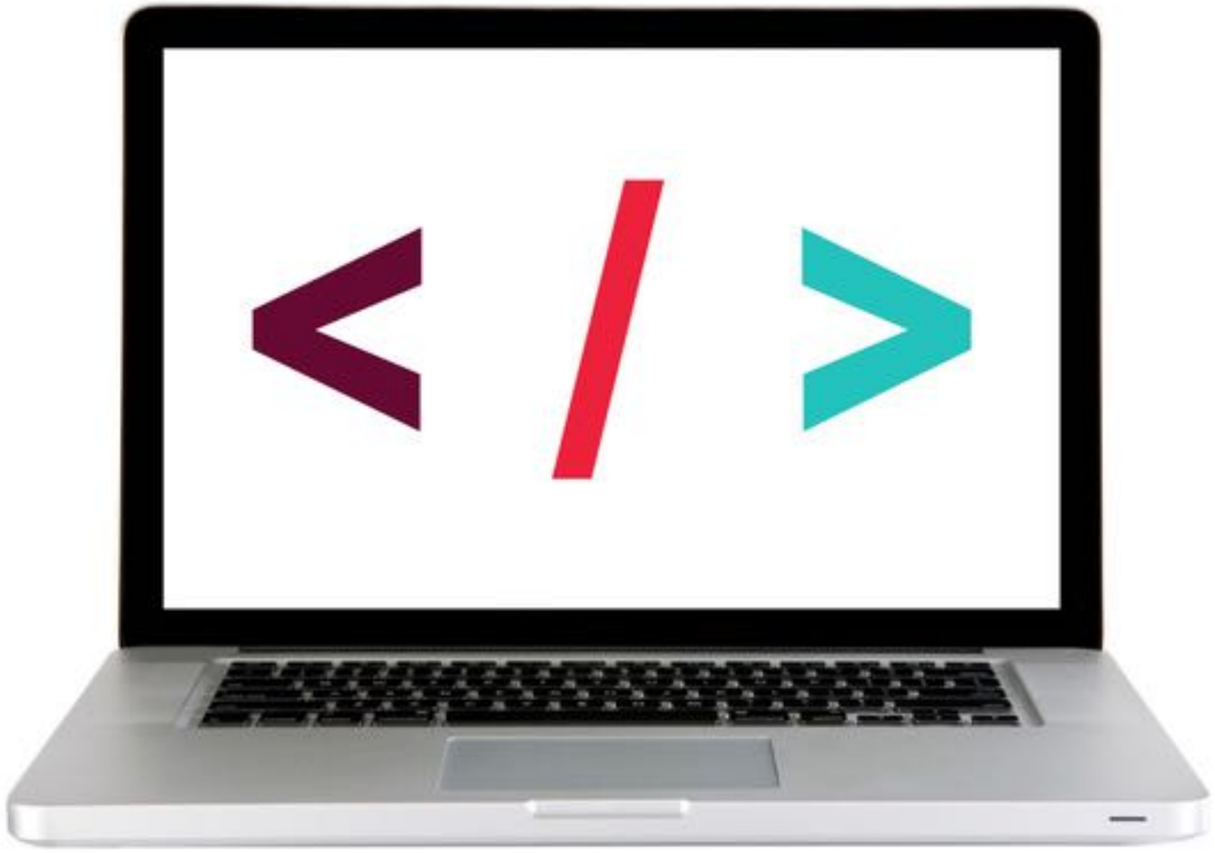
implicit iteration

```
$(‘li’).removeClass(‘current’);
```

applying any method to a jQuery collection iterates through each element!

✓ less code = best practice!

EVENTS & JQUERY



LET'S TAKE A CLOSER LOOK

EXERCISE - IMPLICIT ITERATION

EXERCISE

OBJECTIVE

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

LOCATION

- `starter-code > 3-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

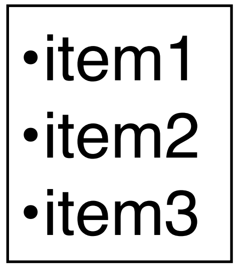
add an event listener to each li in the DOM

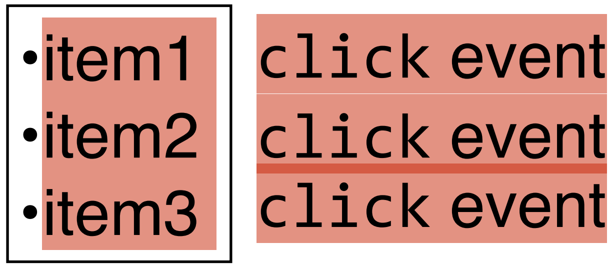
1. load page

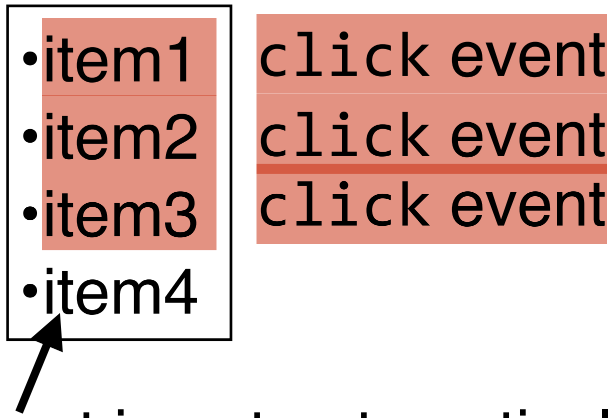
2. set event listener on list items

3. add a new list item

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



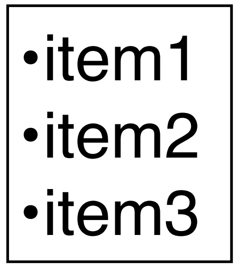




click event is not automatically applied to the new li element

WITH EVENT DELEGATION

1. load page



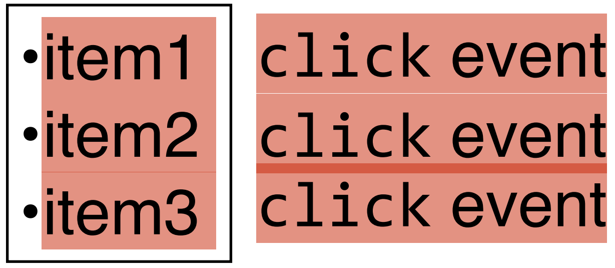
2. set event listener on parent of list items

selector changed from ‘li’ to ‘ul’

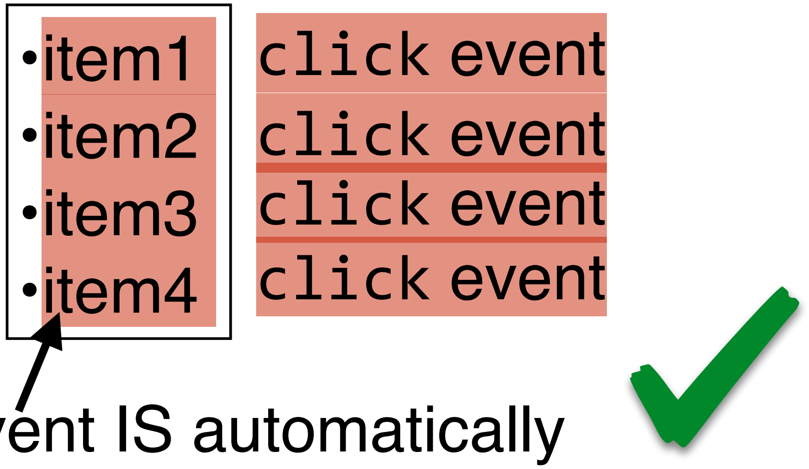
new argument ‘li’ added to on() method

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

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

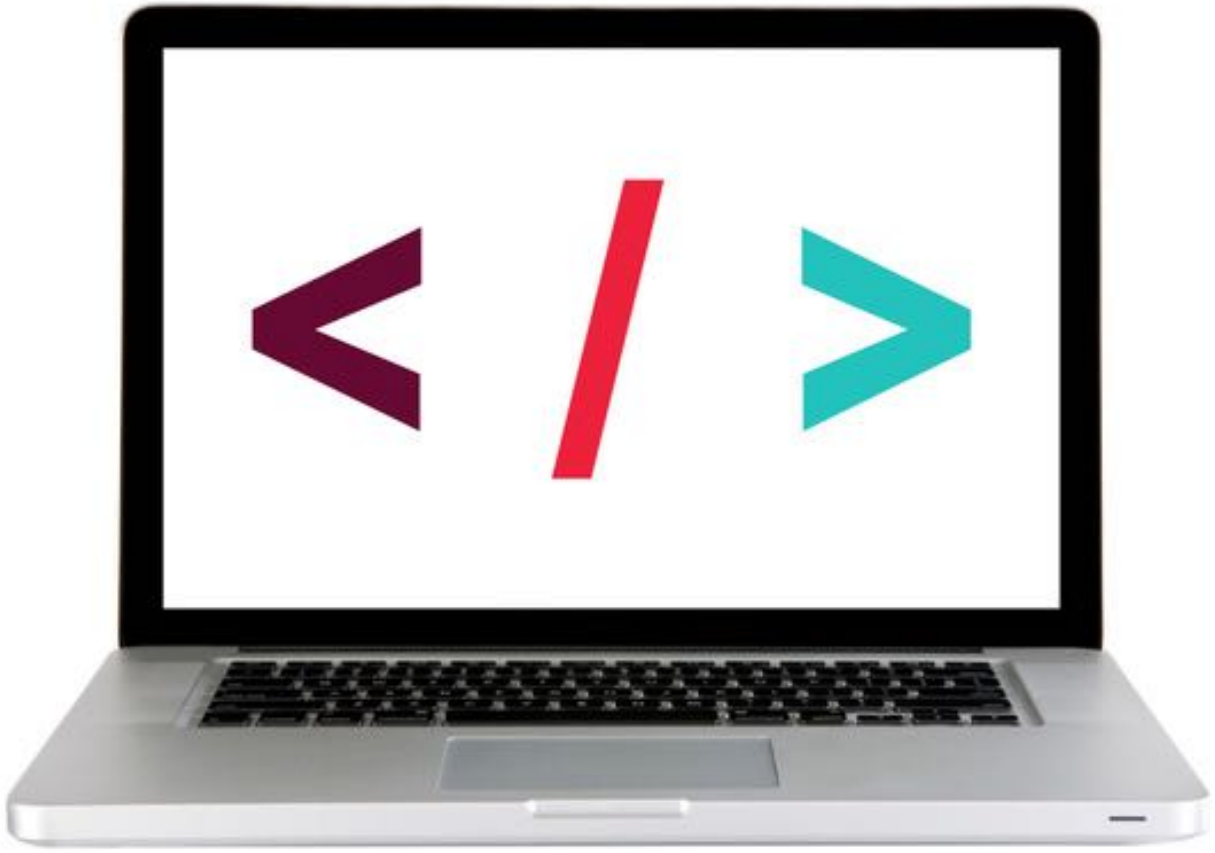


3. add a new list item



click event IS automatically applied to the new li element!

EVENTS & JQUERY



LET'S TAKE A CLOSER LOOK

EXERCISE - EVENT DELEGATION

EXERCISE

OBJECTIVE

- Use event delegation to manage dynamic content.

LOCATION

- `starter-code > 3-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

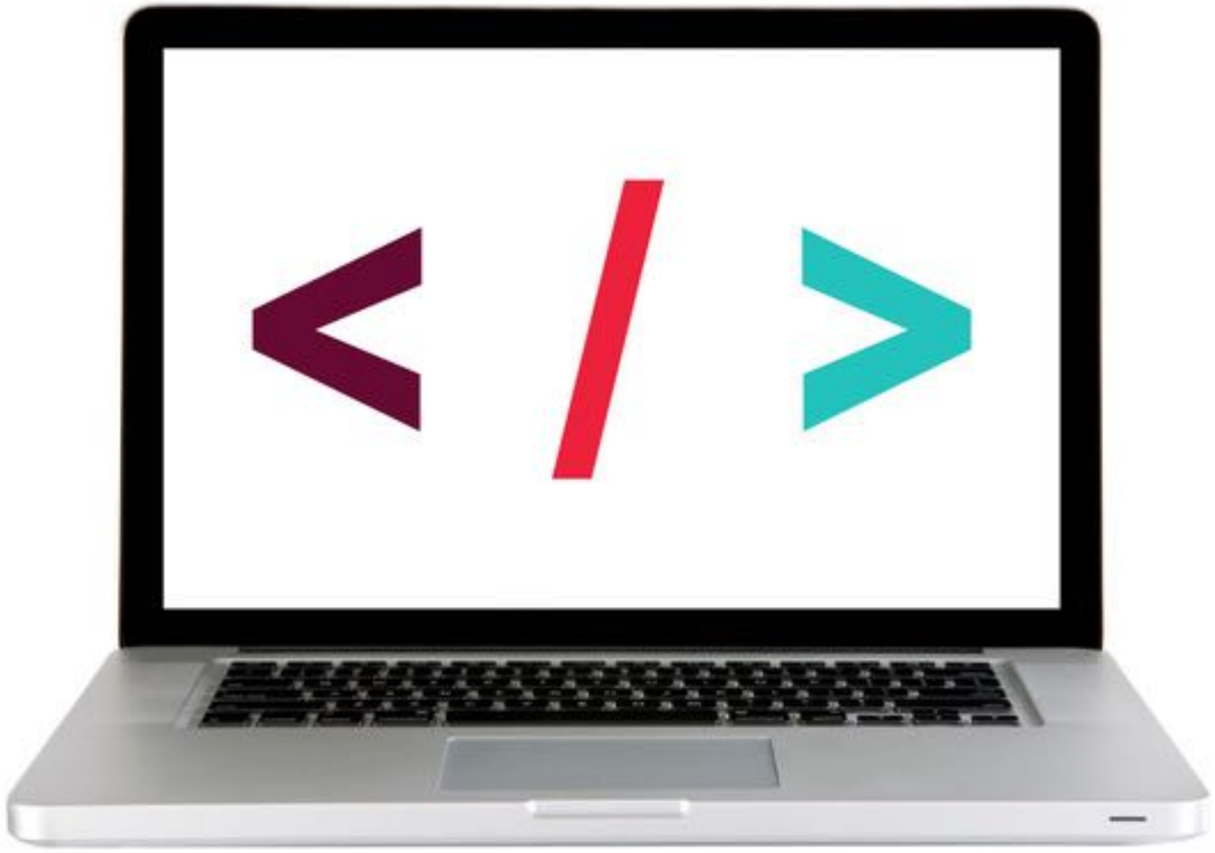
- We could write a separate `.on()` statement for each event on an element:

```
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

```
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');
  }
});
```

EVENTS & JQUERY



LET'S TAKE A CLOSER LOOK

EXERCISE - ATTACHING MULTIPLE EVENTS

EXERCISE

LOCATION

▸ `starter-code > 8-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 #8)

LEARNING OBJECTIVES - REVIEW

- Manipulate the DOM using jQuery selectors and methods.
- 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