

# 9.1 Getting started with jQuery

## JavaScript libraries

Writing JavaScript that creates dynamic, interactive webpages that work across all browsers can be very tedious, often requiring many lines of brittle code. Developers typically rely on libraries to ease the burden of writing such code. A **library** is a collection of functions that focus on a related set of tasks. **jQuery** is a popular JavaScript library that focuses on a broad range of tasks, many of them associated with the visual elements of a webpage.

### Library vs. framework

*New JavaScript developers are sometimes confused about the difference between a "library" and a "framework". A **framework** is a suite of libraries designed to offer a more comprehensive platform in which to program. When using a framework, the program's flow is dictated by the framework, not the programmer. Examples of popular JavaScript frameworks include [AngularJS](#), [Ember](#), and [Backbone.js](#).*

Table 9.1.1: Common tasks performed by jQuery.

DOM manipulation	Find, alter, add, or remove DOM elements
User interaction	Respond to mouse clicks, mouse movement, or typing
Animation	Smoothly show, hide, or move webpage elements
Widgets	Display and manage the interaction of complex GUI elements
Ajax	Issue asynchronous HTTP requests and handle responses
Browser quirks	Handle inconsistencies across different browsers

PARTICIPATION  
ACTIVITY

9.1.1: jQuery tasks.



Match the tasks performed by jQuery to the corresponding example.



If unable to drag and drop, refresh the page.

- Ajax
- Animation
- Browser quirks
- DOM manipulation
- User interaction
- Widgets

Respond to the user clicking a button.

Write code that works in older versions of Internet Explorer.

Display a date picker.

Fade an image into view.Interact with a web API.Add a new item to a list.Reset

## Accessing the jQuery library

The jQuery library can be obtained from [jquery.com](https://jquery.com). Version 3 is the latest version. After downloading the library, developers often place the library in a standard location on their web server. Webpages that use jQuery import the library as shown in the figure below. The filename contains a version number (3.5.1), and the ".min" means the code has been minified to download quicker.

Figure 9.1.1: Downloading jQuery library from the local web server.

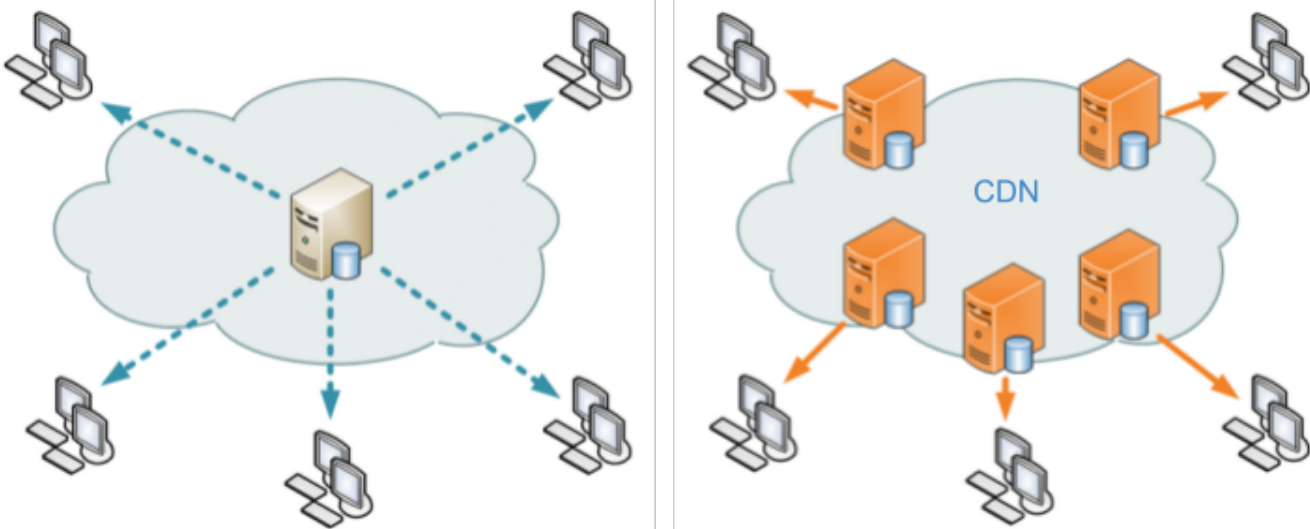
```
<script src="jquery-3.5.1.min.js">
</script>
```

Another option is to write webpages that download the jQuery library from a CDN. A **Content Delivery Network (CDN)** hosts popular web files around the globe and automatically routes requests to the closest server, thus speeding up the delivery of the files. The figure below shows how to import the jQuery library from a CDN. The **integrity** and **crossorigin** attributes are used for **Subresource Integrity (SRI)** checking, which allows web browsers to verify resources hosted on third-party servers have not been altered.

Figure 9.1.2: Downloading jQuery library from the code.jquery.com CDN.

```
<script
  src="https://code.jquery.com/jquery-3.5.1.min.js"
  integrity="sha256-
9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
  crossorigin="anonymous"></script>
```

Figure 9.1.3: Single server distribution vs. CDN distribution.



Source: [Kanoha](#) / [CC BY-SA 3.0](#)

**PARTICIPATION  
ACTIVITY**

9.1.2: Loading the jQuery library.

- 1) Older versions of Query can be downloaded from the jQuery website.

☐ True

☐ False
- 2) The jQuery library will almost always download faster to the browser using a CDN than from a webpage's server.

☐ True

☐ False
- 3) Other CDNs besides code.jquery.com exist.

☐ True

☐ False

## The jQuery() function

The jQuery library defines a primary function called **jQuery()**. The function behaves differently depending on what arguments are passed to `jQuery()`, but the function always returns a `jQuery` object. *Good practice is to use variables that start with "\$" to hold jQuery objects.*

PARTICIPATION  
ACTIVITY

9.1.3: Creating jQuery objects from DOM nodes.



```
<!DOCTYPE HTML>
<html lang="en">
  <title>jQuery Example</title>
  <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
  <body>
    <p id="hello">Hello, jQuery!</p>
    <p>This stuff is great!</p>
    <script>
      let helloPar = document.getElementById("hello");
      let $helloPar = jQuery(helloPar);
      let allParas = document.getElementsByTagName("p");
      let $allParas = jQuery(allParas);
    </script>
  </body>
</html>
```

helloPar	<p>	DOM node
\$helloPar	[ <p> ]	jQuery obj
allParas	[<p>,<p>]	DOM node array
\$allParas	[<p>,<p>]	jQuery obj

Animation content:

The following code block is displayed.

```
<!DOCTYPE HTML>
<html lang="en">
  <title>jQuery Example</title>
  <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
  <body>
    <p id="hello">Hello, jQuery!</p>
    <p>This stuff is great!</p>
    <script>
```

```
let helloPar = document.getElementById("hello");
let $helloPar = jQuery(helloPar);
let allParas = document.getElementsByTagName("p");
let $allParas = jQuery(allParas);
</script>
</body>
</html>
```

Step 1: The line of code reading "<script src='\"https://code.jquery.com/jquery-3.5.1.min.js\"'></script>" is highlighted.

Step 2: The line of code reading "let helloPar = document.getElementById(\"hello\");" is highlighted. Next, the line reading "<p id='\"hello\"'>Hello, jQuery!</p>" is highlighted, and "<p>" is recorded in a table below as the corresponding DOM node for helloPar.

Step 3: First, the line of code reading "let \$helloPar = jQuery(helloPar);" is highlighted. "[<p>]" is added to the table as the jQuery obj corresponding to \$helloPar.

Step 4: A line of code reading "let allParas = document.getElementsByTagName(\"p\");" is highlighted. Next, both lines of code containing "<p>" are highlighted, and "[<p>,<p>]" is added to the table as the corresponding DOM node array for allParas.

Step 5: The line of code reading "let \$allParas = jQuery(allParas);" is highlighted. "[<p>,<p>]" is added to the table as the corresponding jQuery obj array for \$allParas.

### Animation captions:

1. The <script> tag downloads the jQuery library.
2. getElementById() finds the element with ID "hello" and returns the first <p> DOM node.
3. The jQuery() function creates a jQuery object with the <p> DOM node.
4. getElementsByTagName() finds all paragraphs and returns an array of DOM nodes.
5. The jQuery() function creates a jQuery object from the array of DOM nodes.

The **\$()** function is the same as the **jQuery()** function, which developers often use to type less code.

Figure 9.1.4: Using the `$()` function.

```
let $p = $(document.getElementById("hello"));  
  
// same as  
let $p =  
jQuery(document.getElementById("hello"));
```

**PARTICIPATION  
ACTIVITY**

9.1.4: The `jQuery()` and `$()` functions.

- 1) If the `jQuery` library is not downloaded to the user's browser, using `$ ( )` will result in a syntax error.  
☐ True  
☐ False
- 2) `$(x)` is equivalent to `jQuery(x)`, where `x` contains a DOM node.  
☐ True  
☐ False
- 3) The `jQuery` object contains a collection of DOM nodes.  
☐ True  
☐ False
- 4) The `jQuery` object returned by `$( )` must be stored in a variable that begins with `$`.  
☐ True  
☐ False

Exploring further:

- [jQuery API](#)
- [Subresource Integrity](#) from W3C
- [jQuery\(\) function](#)

## 9.2 Selectors

### Basic selectors

The `$ ( )` function is used to select DOM elements using selectors. A **selector** is a string that is crafted to match specific DOM elements. Once an element is selected, other jQuery methods can be used to add and remove CSS classes or properties from the selected element, show or hide the selected element, etc. The jQuery method **`addClass()`** is used to add a CSS class to selected elements as illustrated in the animation below.

#### PARTICIPATION ACTIVITY

9.2.1: Adding classes to all paragraphs.



```
<!DOCTYPE html>
<html lang="en">
<title>jQuery Example</title>
<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
<style>
    .important { color: red; }
</style>

<p id="hello" class="important">Hello, jQuery!</p>
<p class="important">This stuff is great!</p>

<script>
    let $allParas = $("p");
    $allParas.addClass("important");
</script>
</html>
```

Hello, jQuery!  
This stuff is gr

\$allParas [`<p>`,<

### Animation content:

The following code block is displayed.



```
<!DOCTYPE html>
<html lang="en">
<title>jQuery Example</title>
<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
<style>
    .important { color: red; }
</style>

<p id="hello">Hello, jQuery!</p>
<p>This stuff is great!</p>

<script>
    let $allParas = $("p");
    $allParas.addClass("important");
</script>
</html>
```

Step 1: The two lines reading "`<p id="hello">Hello, jQuery!</p>`" and "`<p>This stuff is great!</p>`" are highlighted. First, the plain text, "Hello, jQuery! This stuff is great!" is displayed to the right.

Step 2: The line reading "`let $allParas = $("p");`" is highlighted. The two lines, "`<p id="hello">Hello, jQuery!</p>`" and "`<p>This stuff is great!</p>`" are highlighted again in red. "`[<p>,<p>]`" is shown beside "\$allParas" to the side.

Step 3: The line reading "`$allParas.addClass("important");`" is highlighted. The two lines of code surrounded by "`<p>`" tags are changed to read "`<p id="hello" class="important">Hello, jQuery!</p>`" and "`<p class="important">This stuff is great!</p>`". When this new code is executed, the text reading "Hello, jQuery! This stuff is great!" changes to a red font.

## Animation captions:

1. The browser renders the webpage.
2. `$("p")` selects all `<p>` elements.
3. The `addClass("important")` method adds the "important" class to both paragraphs.
4. Changes to the DOM cause the browser to update the rendered webpage, making both paragraphs use a red font.

A web developer can select elements and call jQuery methods to perform operations on the selected elements in a single line of code. The figure below selects and removes the "important" class from all paragraphs using the jQuery method **removeClass()**, which removes a class from a selected element.

Example 9.2.1: Using \$() and removeClass() on a single line.

```
// Select all paragraphs, then remove the "important" class from all of them
$("p").removeClass("important");
```

Three common ways exist to select elements: by element, ID, or class. The jQuery syntax for selecting by element, ID, and class is similar to CSS selector syntax.

Table 9.2.1: Basic jQuery selectors.

Selector Type	Example	Explanation
Element	<code>\$("p")</code>	Selects all <code>&lt;p&gt;</code> elements
ID	<code>\$("#hello")</code>	Selects the element with <code>id="hello"</code>
Class	<code>\$(".important")</code>	Selects all elements with <code>class="important"</code>

PARTICIPATION  
ACTIVITY





9.2.2: Using basic selectors.



1) Which line of code selects all `<h2>` elements in a document?



- ☐ `$("<h2>")`
- ☐ `$("#h2")`
- ☐ `$("h2")`

- 2) Which line of code adds the class "crazy" to the element with ID "game"? 
- ☐ `$("#game").addClass("crazy");`
  - ☐ `$(".game").addClass("crazy");`
  - ☐ `$("game").addClass("crazy");`
- 3) Which line of code removes the class "crazy" from all elements currently using the "crazy" class? 
- ☐ `$(".crazy").removeClass(".crazy");`
  - ☐ `$(".crazy").removeClass("crazy");`
  - ☐ `$("div").removeClass("crazy");`
- 4) Which line of code removes the "crazy" class from all <div> elements currently using the "crazy" class and adds the "crazy" class to all <div> elements currently not using the "crazy" class? 
- ☐ `$("div").removeClass("crazy");`
  - ☐ `$("div").addClass("crazy");`
  - ☐ `$("div").toggleClass("crazy");`
- 5) Which line of code adds the classes "crazy" and "completed" in the most efficient manner to all <p> elements? 
- ☐ `$("p").addClass("crazy").addClass("completed");`
  - ☐ `$("p").addClass("crazy + completed");`
  - ☐ `$("p").addClass("crazy");  
$("p").addClass("completed");`

## Additional selectors

Additional jQuery selectors provide more sophisticated selection of DOM elements:

- **Attribute selector** - Selects elements based on an element attribute.
- **Basic filter selector** - Selects elements based on a variety of properties.
- **Child filter selector** - Selects child elements based on location or other properties.

- **Content filter selector** - Selects elements based on an element's contents.
- **Hierarchy selector** - Selects elements based on an element's location within the DOM hierarchy.

The table below shows examples of each additional jQuery selector type.

Table 9.2.2: Additional jQuery selectors.

Selector Type	Example	Explanation
Attribute	<code>\$("span[id]")</code>	Selects all <code>&lt;span&gt;</code> that have an <code>id</code> attribute
Attribute	<code>\$("a[href\$='.pdf']")</code>	Selects all <code>&lt;a&gt;</code> with <code>href</code> attributes ending in <code>.pdf</code>
Basic filter	<code>\$("p:first")</code>	Selects the first <code>&lt;p&gt;</code> element
Basic filter	<code>\$("tr:even")</code>	Selects the first, third, fifth, etc. table rows (zero-indexed)
Basic filter	<code>\$("li:eq(1)")</code>	Selects the second <code>&lt;li&gt;</code> element (index <code>n</code> )
Child filter	<code>\$("li:last-child")</code>	Selects the last <code>&lt;li&gt;</code> in each group
Content filter	<code>\$("p:contains('bye')")</code>	Selects all <code>&lt;p&gt;</code> that contain the word "bye"
Hierarchy	<code>\$("li span")</code>	Selects all <code>&lt;span&gt;</code> that are descendants of <code>&lt;li&gt;</code>

PARTICIPATION  
ACTIVITY

9.2.3: Using additional selectors.



Given the HTML below, what elements are selected with each jQuery selector?

```
<h2>About Me</h2>
<p id="intro">Hello, my name is <strong>Jamie</strong>!</p>
<p class="books">My favorite books:</p>
<ol>
  <li>Animal Farm</li>
  <li>1984</li>
  <li>Oh... I forgot the name!</li>
</ol>
<div>One of my favorite quotes from <strong>Robert Wilensky</strong>:
  <blockquote>
    We've all heard that a million monkeys banging on a million
    typewriters
    will eventually reproduce the entire works of Shakespeare. Now,
    thanks
    to the Internet, we know this is <strong>not true</strong>.
  </blockquote>
</div>
```

1) \$("div strong")

- ☐ <strong>Jamie</strong>
- ☐ and <strong>Robert Wilensky</strong>
- ☐ <strong>Robert Wilensky</strong>
- ☐ <strong>Robert Wilensky</strong> and <strong>not true</strong>

2) \$("li:first")

- ☐ <li>Animal Farm</li>
- ☐ <li>Oh... I forgot the name!</li>
- ☐ All <li> elements

3) \$("li:eq(2)")

- ☐ <li>1984</li>
- ☐ <li>Oh... I forgot the name!</li>
- ☐ All <li> elements

4) `$("div[class]")`

- ☐ `<p class="books">...</p>`
- ☐ `<div>One of...</div>`
- ☐ Nothing

5) `$("p:contains('name')")`

- ☐ Both `<p>` elements
- ☐ `<p id="intro">...</p>`
- ☐ Nothing

**PARTICIPATION  
ACTIVITY**

9.2.4: Selector practice.

The following webpage is rendered using a combination of HTML, CSS, and JavaScript. Try making the following JavaScript modifications so that the rendered webpage matches the expected webpage. Do not modify the HTML or CSS.

1. Modify the jQuery so that only the even `<li>` elements have the "highlight" class.
2. Add a jQuery method call to remove the "border" class from the first `<p>`.
3. Add a jQuery method call to add the "border" class to the second `<p>`.

HTML

CSS

JavaScript

```
1 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
2
3 <h1>Highest-Grossing Films</h1>
4 <p class="border">Films adjusted for inflation as of 2019:</p>
5 <ol>
6   <li><cite>Gone with the Wind</cite> (1939)</li>
7   <li><cite>Avatar</cite> (2009)</li>
8   <li><cite>Titanic</cite> (1997)</li>
9   <li><cite>Star Wars</cite> (1977)</li>
10  <li><cite>Avengers: Endgame</cite> (2019)</li>
11 </ol>
12 <p><cite>Gone with the Wind</cite> is the oldest movie to rank in the top 10 list of highest-grossing
13 <p>Source: <a href="https://en.wikipedia.org/wiki/List_of_highest-grossing_films">https://en.wikipedia.org/wiki/List_of_highest-grossing_films</a>
14
```

Render webpage

Reset code

## Your webpage

## Expected webpage

# Highest-Grossing Films

Films adjusted for inflation as of 2019:

1. *Gone with the Wind* (1939)
2. *Avatar* (2009)
3. *Titanic* (1997)
4. *Star Wars* (1977)
5. *Avengers: Endgame* (2019)

*Gone with the Wind* is the oldest movie to rank in the top 10 list of highest-grossing

# Highest-Grossing Films

Films adjusted for inflation as of 2019:

1. *Gone with the Wind* (1939)
2. *Avatar* (2009)
3. *Titanic* (1997)
4. *Star Wars* (1977)
5. *Avengers: Endgame* (2019)

*Gone with the Wind* is the oldest movie to rank in the top 10 list of highest-grossing

► View solution

CHALLENGE  
ACTIVITY

9.2.1: jQuery selectors.



550544.4142762.qx3zqy7

[Start](#)

Using jQuery, add class "subtle" to the element with an id of "first". [SHOW EXPECTED](#)

HTML

CSS

JavaScript

```
1 <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
2
3 <p id="first">First paragraph</p>
4 <p id="second">Second paragraph</p>
5 <p id="third">Third paragraph</p>
6 <p id="fourth">Fourth paragraph</p>
7 <p id="fifth">Fifth paragraph</p>
8 <p id="sixth">Sixth paragraph</p>
```

©zyBooks 04/15/24 16:43 2071381  
Marco Aguilar  
CIS192\_193\_Spring\_2024

1

2

3

4

[Check](#)[Next](#)

Exploring further:

- [jQuery selectors](#)

## 9.3 Events

### Callback functions



Web developers may use jQuery to write code that executes in response to an event. The `on()` method attaches a callback function to the selected elements of a `jQuery` object. A **callback function** is a function that is executed when an event occurs. A callback function is also called an **event listener** or **event handler**. jQuery code for registering callback functions is more concise and easier to read than using plain JavaScript.

Figure 9.3.1: Registering callback functions in JavaScript and jQuery.

```
// Example callback function
function myCallback() {
    console.log("You clicked the button");
}

// Registering a click callback function with plain JavaScript
document.getElementById("mybutton").addEventListener("click",
myCallback);

// Registering a click callback function with jQuery
$("#mybutton").on("click", myCallback);

// or simply
$("#mybutton").click(myCallback);
```

jQuery also provides a number of shortcut methods for handling common events. Ex: `click()` can be used to register a click callback function instead of the more verbose `on("click")` method.

Table 9.3.1: Mouse events and shortcut methods.

Event	Shortcut	Description
click	<code>click()</code>	Triggered when the mouse clicks on an element.
dblclick	<code>dblclick()</code>	Triggered when the mouse double-clicks on an element.
mouseover / mouseout	<code>mouseover()</code> / <code>mouseout()</code>	Triggered only once when the mouse pointer moves over or leaves an element and any of the element's children.
mouseenter / mouseleave	<code>mouseenter()</code> / <code>mouseleave()</code> <code>hover()</code>	Triggered only once when the mouse pointer moves over or leaves an element. <code>hover()</code> binds a callback function, or functions, for both events
mousemove	<code>mousemove()</code>	Triggered when the mouse pointer moves over an element.

**PARTICIPATION  
ACTIVITY**

## 9.3.1: Mouse events.



Given the following JavaScript, match the equivalent jQuery code segments.

```
function doThis() {  
    console.log("doThis");  
}  
  
function doThat() {  
    console.log("doThat");  
}
```

If unable to drag and drop, refresh the page.

```
$("#p").mousemove(doThis);
```

```
$("#p").hover(doThis, doThat);
```

```
$("#p").on("dblclick", doThis);
```

```
$("#p").dblclick(doThis);
```

```
$("#p").mouseenter(doThis).mouseleave(doThat);
```

```
$("#p").mousemove(function() {  
    console.log("doThis"); });
```

[Reset](#)

## The ready event

One of the most important jQuery events is the **ready** event. The **ready event** is triggered when the browser has finished loading the webpage's DOM. *Good practice is to use jQuery selectors inside the **ready** callback function to ensure the DOM is fully loaded before searching the DOM for elements.*

Figure 9.3.2: Waiting for the ready event.

```
// Wait until the document's DOM is  
ready  
$(document).ready(function() {  
    // DOM is ready to go  
    $("button").addClass("big");  
});
```

Developers use the **ready** event so frequently that the shortcut **ready ( )** method is not needed. The **\$ ( )** function can be supplied a callback function that will only be executed when the **ready** event triggers on the document.

Figure 9.3.3: Registering a ready event callback function with \$().

```
$(function() {  
    // DOM is ready to go  
  
    $("button").addClass("big");  
});
```

**PARTICIPATION  
ACTIVITY**

9.3.2: The ready event.

1) Developers typically use a ready event callback function when the jQuery code is located at the end of the webpage.

- ☐ True  
☐ False

2) The code `$(document).ready(doThis);` is equivalent to `$(doThis);` where `doThis` is a function.

- ☐ True  
☐ False

3) The ready event is triggered after all the page's images, videos, and scripts have finished downloading.

- ☐ True  
☐ False

**PARTICIPATION  
ACTIVITY**

9.3.3: Mouse event practice.

The following HTML uses CSS and JavaScript to render the webpage. Try modifying the JavaScript so that the rendered webpage matches the expected webpage. Do not modify

the HTML or CSS.

1. Make the words "Click me!" become large when the mouse is hovering over the words and return to regular size when the mouse is not hovering over the words. Do this by modifying the call to `hover()` so the `big` class is added or removed to/from the `<div>` at the right time.
2. Make the words "Click me!" alternate between red and blue when the user clicks on the words. Do this by adding a "click" event callback function to the `<div>` and calling the `toggleClass()` method to add or remove the `red` and `blue` classes to the `<div>`.
3. Although the JavaScript code will work without a ready event callback function, put all of the JavaScript code inside a ready event callback function, just for practice.

[HTML](#)[CSS](#)[JavaScript](#)

```
1 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery"
2
3 <div class="red">
4   Click me!
5 </div>
6
```

[Render webpage](#)[Reset code](#)

Your webpage

Click me!

Expected webpage

Click me!

► View solution

Keyboard and form events

jQuery supports keyboard events that are helpful when a webpage needs to react to the user pressing or releasing keys on a keyboard. Ex: Moving a spaceship left or right depending on what key is pressed. jQuery also supports form events that are helpful when the webpage needs to perform data validation. Ex: Ensuring the credit card field contains the correct number of digits when the user presses the submit button.

Table 9.3.2: Keyboard events and shortcuts.

Event	Shortcut	Description
keydown	<code>keydown ( )</code>	Triggered when the user first presses a key on the keyboard.
keyup	<code>keyup ( )</code>	Triggered when the user releases a key from the keyboard.
keypress	<code>keypress ( )</code>	Triggered when the browser registers keyboard input from printable character keys. Non-printing keys like Shift and Esc do not register keypress events.

Table 9.3.3: Form events and shortcuts.

Event	Shortcut	Description
focus / blur	<b>focus()</b> / <b>blur()</b>	Triggered when an element gains or loses focus.
focusin / focusout	<b>focusin()</b> / <b>focusout()</b>	Triggered when an element or any of the element's children gain or lose focus.
change	<b>change()</b>	Triggered when an element's value changes.
select	<b>select()</b>	Triggered when a user selects text in an <code>&lt;input type="text"&gt;</code> or <code>&lt;textarea&gt;</code> element.
submit	<b>submit()</b>	Triggered when the user is attempting to submit a form.

**PARTICIPATION  
ACTIVITY**

## 9.3.4: Keyboard and form events.

- 1) What event is helpful for writing a callback function that ensures the password the user is entering is long enough while the password is being typed?

☐ blur

☐ keyup

☐ change
- 2) What event is helpful for writing a callback function that ensures the user typed a valid phone number after the user moves to another input field?

☐ keypress

☐ submit

☐ blur

3) What event is helpful for writing a callback function that ensures a radio button was selected when the user pressed the form's submit button?

- ☐ submit
- ☐ focusout
- ☐ select

#### PARTICIPATION ACTIVITY

#### 9.3.5: Keyboard event practice.

The following webpage converts Celsius to Fahrenheit and vice versa. The JavaScript code uses the jQuery method `val()` to extract the number that was typed into one of the text input fields when Convert is pressed. The jQuery function `$.isNumeric()` determines if what was typed is a number or not. If a number has not been typed, no conversion is performed. Otherwise `val()` is used to put the converted temperature into the appropriate text field.

Try modifying the JavaScript so the rendered webpage matches the behavior of the expected webpage. Do not modify the HTML or CSS.

1. Make the °F or °C label appear blue when focus is given to the Fahrenheit or Celsius input field, respectively. Use the focus and blur events to add or remove the CSS class `selected` to the labels.
2. Blank-out the °F input field when the user types anything into the °C field and vice versa. Use the keyup event on each of the respective input fields to change the opposing field into an empty string with the `val()` method.
3. Add the `error` class to the °F or °C text fields if the user has typed anything but a number. Remove the `error` class if the user has typed a number. Use the keyup event on each of the respective input fields to test the user input while the input is being typed.

[HTML](#)[CSS](#)[JavaScript](#)



```
1 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery"
2
3 <h1>Temperature Converter</h1>
4 <p>
5     <label id="fahrLabel" for="fahr">&deg;F:</label>
6     <input type="text" id="fahr">
7 </p>
8 <p>
9     <label id="celsLabel" for="cels">&deg;C:</label>
10    <input type="text" id="cels">
11 </p>
12 <p>
13    <input type="button" value="Convert" id="convertBtn">
14 </p>
15
```

[Render webpage](#)[Reset code](#)**Your webpage****Expected webpage**

# Temperature Converter

°F: °C: 

# Temperature Converter

°F: °C: [► View solution](#)**CHALLENGE  
ACTIVITY**

9.3.1: jQuery events.



550544.4142762.qx3zqy7

## Start

Register the `textSize` event handler to handle the `blur` event for the `textarea` tag. Note: The function counts the number of characters in the `textarea`.

HTML

JavaScript

```
1 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery"
2
3 <label for="userName">User name:</label>
4 <textarea id="userName" cols="40" rows="3"></textarea><br>
5 <p id="stringLength">0</p>
6
```

1

2

3

Check

Next

Exploring further:

- [jQuery Events documentation](#)
- [Events - jQuery Learning Center](#)
- [\\$.isNumeric\(\) documentation](#)
- [.val\(\) documentation](#)

## 9.4 Styles and animation

### The `css()` method

jQuery simplifies the process of adding and removing CSS properties to an element's inline style. CSS properties can be added to selected elements with the **`css()`** method as illustrated in the following animation.

#### PARTICIPATION ACTIVITY

9.4.1: Using the `.css()` method.



```
<body style="background-color: peachpuff;
            color: green;
            font-size: 20pt">
  <p>Hello!</p>
</body>
```

```
$( "body" ).css( "background-color", "peachpuff" );

$( "body" ).css( {
  color: "green",
  "font-size": "20pt"
} );
```



Hello!

### Animation content:

Step 1: Two boxes of code are shown. The first reads:

```
<body><p>Hello!</p>
</body>
```

The second reads:

```
$( "body" ).css( "background-color", "peachpuff" );
$( "body" ).css( {
  color: "green",
  "font-size": "20pt"
} );
```

"Hello!" is highlighted and displayed to the right.

Step 2: The first block of code is changed to read

```
"<body style="background-color: peachpuff "> <p>Hello!</p> </body>".
```

"Hello!" is highlighted and displayed to the right on top of a peach colored background.

Step 3: The first block of code is changed to read

```
<body style="background-color: peachpuff;  
color: green;  
font-size: 20pt">
```

```
<p>Hello!</p> </body>
```

"Hello!" is displayed to the right in a large green font over a peach colored background.

### Animation captions:

1. Browser renders the webpage.
2. `css()` method adds the "background-color" CSS property to the `<body>`.
3. `css()` method adds two CSS properties to `<body>` using an object literal.

jQuery converts each camel-cased property name to a string that is equivalent to the CSS property name. Ex: `backgroundColor` is converted to `"background-color"`. However, `"background-color"` cannot be used without quotes since the dash is not a legal character in a JavaScript identifier.

#### PARTICIPATION ACTIVITY

#### 9.4.2: Using the `.css()` method.



Given the following HTML, what is the effect of each jQuery code segment?

```
<h1 style="border-style: solid">To Do</h1>  
<span id="important">Study for exam</span> and  
<span>feed the fish</span>
```

1) `$("#important").css("display", "none");`

- ☐ "Study for exam" is replaced with a blank area.
- ☐ "Study for exam" is no longer visible, and the browser does not leave room for the missing words.
- ☐ Nothing because no elements use the class "important".

2) `$("#h1").css({ "text-decoration": "underline", "border-style": "dotted" });`

- ☐ "To Do" is underlined with a dotted border.
- ☐ "To Do" is underlined, but the solid border remains.
- ☐ Nothing because there is a syntax error in the object literal.

3) `$("#span").css({ textDecoration: "underline", border-style: "dotted" });`

- ☐ Both span elements are underlined with a dotted border.
- ☐ Nothing because quotes are needed around textDecoration.
- ☐ Nothing because there is a syntax error in the object literal.

4) `let x =  
$( "h1" ).css( "border-  
style" );`

- ☐ The border around "To Do" is removed.
- ☐ x is set to an empty string.
- ☐ x is set to "solid".

5) `$( "h1" ).css( "border-style",  
" " );`

- ☐ "To Do" continues to have a solid border.
- ☐ Nothing because an empty string cannot be used as a property value.
- ☐ The solid border around "To Do" is removed.

## jQuery effects

jQuery has three sets of animation methods for showing and hiding webpage elements. The animation methods in the table below alter various CSS properties of selected elements to produce animations. The methods take a speed argument that determines the duration of the animation. jQuery animation methods are also called **jQuery effects**.

Table 9.4.1: Animation methods.

Methods	Example	Description
<code>show()</code> <code>hide()</code> <code>toggle()</code>	<code>\$("#h1").show("slow");</code> <code>\$("#h1").hide("slow");</code> <code>\$("#h1").toggle("slow");</code>	Alters width, height, and opacity all at once
<code>fadeIn()</code> <code>fadeOut()</code> <code>fadeToggle()</code>	<code>\$("#h1").fadeIn("normal");</code> <code>\$("#h1").fadeOut("normal");</code> <code>\$("#h1").fadeToggle("normal");</code>	Alters opacity only
<code>slideDown()</code> <code>slideUp()</code> <code>slideToggle()</code>	<code>\$("#h1").slideDown("fast");</code> <code>\$("#h1").slideUp("fast");</code> <code>\$("#h1").slideToggle("fast");</code>	Alters height only

Table 9.4.2: The speed argument for animation methods.

Argument	Example	Explanation
"slow"	<code>\$("#p").show("slow");</code>	0.6 seconds to show the paragraph
"normal"	<code>\$("#p").show("normal");</code>	0.4 seconds to show the paragraph
"fast"	<code>\$("#p").show("fast");</code>	0.2 seconds to show the paragraph
milliseconds	<code>\$("#p").show(1500);</code>	1.5 seconds to show the paragraph

PARTICIPATION  
ACTIVITY

9.4.3: Animation methods.



The following webpage is using jQuery animation methods to show and hide the cat

image. Click on each button to see the effect on the image. Then try modifying the JavaScript so that when the image is clicked two things will happen:

1. A solid red border will immediately appear around the image. Use the `css()` method to add the appropriate CSS properties.
2. The image will quickly hide, display, fade out, fade in, slide up, and slide down. Call each of the animation methods to produce the series of animations.

HTML

CSS

JavaScript

```
1 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquer
2
3 <div>
4   <button id="show-btn">show</button>
5   <button id="hide-btn">hide</button>
6   <button id="toggle-btn">toggle</button>
7 </div>
8 <div>
9   <button id="fadein-btn">fadeIn</button>
10  <button id="fadeout-btn">fadeOut</button>
11  <button id="fadetoggle-btn">fadeToggle</button>
12 </div>
13 <div>
14  <button id="slidedown-btn">slideDown</button>
15  <button id="slideup-btn">slideUp</button>
16  <button id="slidetoaale-btn">slideToaale</button>
```

Render webpage

Reset code



## Your webpage

show

hide

toggle

fadeIn

fadeOut

fadeToggle

slideDown

slideUp

slideToggle

[► View solution](#)

### PARTICIPATION ACTIVITY

#### 9.4.4: jQuery effects.

- 1) The `hide()` method does not animate opacity.  
☐ True  
☐ False
- 2) If the `show()` method is not given any arguments, the method does not show an animated transition.  
☐ True  
☐ False
- 3) The call `$("h1").toggle(2)` will show or hide all `<h1>` elements during a 2 second interval.  
☐ True  
☐ False

4) The `slideDown()` method will slide the element down until the element is no longer visible.

- ☐ True
- ☐ False

## Controlling animation order

Sometimes developers want to perform an operation after an effect completes. Ex: Display a "Thank you" message after the user has typed information into a form that gradually disappears. However, code that is written after a call to an animation method will execute before the animation is complete. To execute code after the animation is complete, the jQuery animation methods can be passed a callback function.

### PARTICIPATION ACTIVITY

9.4.5: Waiting for an animation to complete.

```
<body>
  <h1>jQuery effects</h1>
  <h2>Good stuff!</h2>
</body>
```

```
$( "h1" ).fadeOut( "slow" );
$( "h2" ).fadeOut( "slow" );

$( "h1" ).fadeIn( "slow", function() {
  $( "h2" ).fadeIn( "slow" );
});
```

jQuery effects  
Good stuff!

### Animation content:

The following code blocks are displayed.

```
<body>
  <h1>jQuery effects</h1>
  <h2>Good stuff!</h2>
</body>
```

```
$("#h1").fadeOut("slow");  
$("#h2").fadeOut("slow");
```

```
$("#h1").fadeIn("slow", function() {  
    $("#h2").fadeIn("slow");  
});
```

Step 1: jQuery effects

Good stuff!" is initially displayed to the right. "\$("#h1").fadeOut("slow");  
\$("#h2").fadeOut("slow");" is highlighted, and the two lines on the right gradually fade away at the same time.

Step 2:

The lines of code reading "\$("#h1").fadeIn("slow", function() {  
 \$("#h2").fadeIn("slow");  
});" are highlighted, and the two lines "jQuery effects" and "Good stuff!" gradually appear one after the other.

### Animation captions:

1. The fadeOut() methods cause <h1> and <h2> to fade out at the same time.
2. The first fadeIn() completes before the second fadeIn() is called, so <h2> fades in after <h1> has finished fading in.

When two animations are applied to different elements, the animations occur at the same time. However, when two animations are applied to the same element, the first animation completes before starting the second animation. jQuery queues animations applied to the same element so that each animation finishes before the next begins.

Figure 9.4.1: Queuing animations on the same element.

```
// Both animations occur at the same time  
$("#h1").fadeOut("slow");  
$("#h2").fadeOut("slow");  
  
// Animations on the same element are  
queued  
$("#h1").fadeIn("slow");  
$("#h1").slideUp(1000);
```

Non-animation method calls are not placed in the animation queue.

Figure 9.4.2: Only animation methods are queued.

```
$("#h1").hide("slow")  
  .addClass("important") // Called immediately, not waiting for hide to  
  complete  
  .slideDown(1000);      // Begins after hide completes
```

jQuery provides a **queue()** method to aid in queuing code that should be executed after the previous animations complete. The **queue()** method can take the place of the callback function used with the animation methods. The **queue()** method takes a function argument that is passed a function parameter called **next**. The **next** function must be called so the next animation in the queue can be processed. *Good practice is to use the **queue()** method instead of callback functions when developing elaborate animations. Using many nested callback functions can lead to overly complicated code.*

Figure 9.4.3: Using the .queue() method.

```
$("#h1").hide("slow")  
  .queue(function(next) {  
    // Add the class after the <h1> hide  
    completes  
    $(this).addClass("important");  
  
    // Process the next animation in the queue  
    next();  
  })  
  .slideDown(1000);
```

#### PARTICIPATION ACTIVITY

#### 9.4.6: Animation order.



Refer to the jQuery code below:

```
$( "h1" ).show(1000)
  .queue(function(next) {
    $(this).removeClass("important");
    next();
  })
  .css("color", "red")
  .delay(1000)
  .hide("fast", function() {
    $(this).css("color", "green");
  });
```

- 1) The show animation completes before the hide animation begins.  
☐ True  
☐ False
- 2) `$(this)` in the code above is equivalent to `$( "h1" )`.  
☐ True  
☐ False
- 3) The show animation completes before the "important" class is removed from `<h1>`.  
☐ True  
☐ False
- 4) The show animation completes before the CSS color property is set to red.  
☐ True  
☐ False
- 5) The `delay()` method delays executing the hide animation for one second.  
☐ True  
☐ False

6) The hide animation completes before the CSS color property is set to green.

- ☐ True
- ☐ False

## The animate() method

### PARTICIPATION ACTIVITY

#### 9.4.7: The animate() method.

```
<body>
  
</body>
```

```
$( "img" ).animate({
  left: "200px",
  height: "-=60"
}, "slow", "linear");

$( "img" ).animate({
  opacity: 0.1,
  top: "+=50"
}, 2000);

$( "img" ).animate({
  opacity: 1,
  width: "200px"
}, function() {
  $(this).css("border", "2px solid red");
});
```



### Animation content:

The following code blocks are displayed.

```
<body>
  
```

```
</body>
```

```
$("#img").animate({
left: "200px",
height: "-=60"
}, "slow", "linear");
```

```
$("#img").animate({
opacity: 0.1,
top: "+=50"
}, 2000);
```

```
$("#img").animate({
opacity: 1,
width: "200px"
}, function() {
$(this).css("border", "2px solid red");
});
```

Step 1: The lines of code reading

```
"$("#img").animate({
left: "200px",
height: "-=60"
```

}, "slow", "linear");" are highlighted. A photograph of a kitten on the right is distorted according to the new left and height values.

Step 2: The lines of code reading

```
"$("#img").animate({
opacity: 0.1,
top: "+=50"
```

}, 2000);" are highlighted. The photograph of the kitten shifts down on the screen and becomes slightly opaque.

Step 3: The lines of code reading

```
"$("img").animate({  
  opacity: 1,  
  width: "200px"
```

}, function() {" are highlighted. The photograph of the kitten is stretched wide and becomes less opaque.

Step 4: The line of code reading "\$ (this).css("border", "2px solid red");" is highlighted. A solid red border is added around the photograph of the kitten.

### Animation captions:

1. animate() makes image's left become 200px and height become  $100\text{px} - 60 = 40\text{px}$  with linear easing over 0.6 secs (length of "slow").
2. animate() makes image's opacity change to 0.1, top to  $0 + 50 = 50\text{px}$  over 2 seconds.
3. animate() makes image's opacity change to 1 and width to 200px over 0.4 seconds (default length of time).
4. css() sets image's border to solid red after animation completes.

#### PARTICIPATION ACTIVITY

#### 9.4.8: The animate() method.

1) In the **animate( )** property:value map, what does the property value "+=20" do?

- ☐ Sets the property to the number 20.
- ☐ Adds 20px to the property's value.
- ☐ Subtracts 20px from the property's value.



2) What easing function does the code segment below use?



```
$( "img" ).animate({  
    opacity: 1,  
    width: "200px"  
}, function() {  
    $(this).css( "border", "2px  
solid red" );  
});
```

- ☐ linear
- ☐ swing
- ☐ No easing function was specified, so no easing was used.

3) Can `animate()` animate the transition from one font color to another?



- ☐ Yes
- ☐ No

#### PARTICIPATION ACTIVITY

9.4.9: Practice with `animate()` and `queue()`.



The following webpage displays some help information when the "?" is clicked. The help information is hidden when "?" is clicked a second time.

Make the following modifications so the rendered webpage acts like the expected webpage:

1. Modify the `animate()` call to make the width of "?" increase by 15px and font-size increase by 10px when the "?" is first clicked. Make the width decrease by 15px and font-size decrease by 10px when "?" is clicked again.
2. Use the `queue()` method to change the background of the "?" to orange after the "?" has finished moving to the right. Change the background back to light blue after the "?" has finished moving back to the left.
3. Use the `queue()` method to fade in the help text after the "?" has finished moving to

the right. Fade out the help text when the "?" has been clicked a second time but before the "?" moves back to the left.

HTML

CSS

JavaScript

```
1 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquer
2
3
4 <div id="help" title="Help">
5   ?
6 </div>
7 <div id="help-text">
8   To print the document, select the <b>Print</b> option from the <b>
9 </div>
10
```

Render webpage

Reset code

Your webpage



Expected webpage



► View solution

Exploring further:

- [jQuery Effects documentation](#)
- [Queue and Dequeue Explained](#)

## 9.5 DOM manipulation

### Accessing element attributes

jQuery provides many methods for DOM manipulations that allow developers to dynamically add, remove, or modify content on a webpage.

The **attr()** method gets and sets attribute values of a DOM element.

Figure 9.5.1: Changing image attributes with attr().

```
// Change all images to 
$("img").attr("src", "star.png");

// Change all images to 
$("img").attr({
  src: "star.png",
  alt: "Bright star",
});
```

#### PARTICIPATION ACTIVITY

#### 9.5.1: jQuery attr() method.

1) `$("li").attr("style", "color:red")` is equivalent to `$("li").css("color", "red")`.

- ☐ True
- ☐ False

- 2) If an image has a src attribute of "moon.png", calling `$("img").attr("src")` returns "moon.png".
- ☐

 True
- ☐

 False



Adding DOM nodes

The `$()` function creates new DOM nodes when given an HTML string. Ex: `$("<span>I'm a new node!</span>");` creates a span node. However, the new node is not visible until the node is added to the DOM.

Table 9.5.1: Methods for adding DOM nodes.

Methods	Example	Before	After
<code>prependTo()</code> <code>prepend()</code>	<pre>\$("&lt;li&gt;New first&lt;/li&gt;").prependTo("ol");  // same as \$("ol").prepend("&lt;li&gt;New first&lt;/li&gt;");</pre>	<pre>&lt;ol&gt;   &lt;li&gt;A&lt;/li&gt;   &lt;li&gt;B&lt;/li&gt; &lt;/ol&gt;</pre>	<pre>&lt;ol&gt;   &lt;li&gt;New first&lt;/li&gt;   &lt;li&gt;A   &lt;li&gt;B &lt;/ol&gt;</pre>
<code>appendTo()</code> <code>append()</code>	<pre>\$("&lt;li&gt;New last&lt;/li&gt;").appendTo("ol");  // same as \$("ol").append("&lt;li&gt;New last&lt;/li&gt;");</pre>	<pre>&lt;ol&gt;   &lt;li&gt;A&lt;/li&gt;   &lt;li&gt;B&lt;/li&gt; &lt;/ol&gt;</pre>	<pre>&lt;ol&gt;   &lt;li&gt;A   &lt;li&gt;B   &lt;li&gt;New last&lt;/li&gt; &lt;/ol&gt;</pre>
<code>insertBefore()</code> <code>before()</code>	<pre>\$("&lt;p&gt;Before&lt;/p&gt;").insertBefore("h2");  // same as \$("h2").before("&lt;p&gt;Before&lt;/p&gt;");</pre>	<pre>&lt;h2&gt;Test&lt;/h2&gt;</pre>	<pre>&lt;p&gt;Before&lt;/p&gt; &lt;h2&gt;Test</pre>

insertAfter() after()	\$( " <p>After</p>" ).insertAfter( "h2" );  // same as \$( "h2" ).after( "<p>After</p>" );	<h2>Test</h2>	<h2>Test <p>After
wrap()	\$( "p" ).wrap( "<div></div>" );	<p>A</p> <p>B</p>	<div> <p>A< </div> <div> <p>B< </div>
wrapAll()	\$( "p" ).wrapAll( "<div></div>" );	<p>A</p> <p>B</p>	<div> <p>A< <p>B< </div>
wrapInner()	\$( "p" ).wrapInner( "<div></div>" );	<p>A</p> <p>B</p>	<p>  <div>A</div> <p>  <div>B</div>

# PARTICIPATION ACTIVITY

## 9.5.2: Adding to the DOM.



Given the HTML below, match the jQuery code to the resulting DOM transformation.

```
<p>  
  Test <i>this</i>  
</p>
```

If unable to drag and drop, refresh the page.

```
$("#p").after(
  "<span>x</span>");
```

```
$("#p").append(
  "<span>x</span>");
```

```
$("#<span>x</span>")
  .prependTo("#p");
```

```
$("#i").wrap(
  "<span></span>");
```

```
$("#<span>x</span>")
  .insertBefore("i");
```

```
<p><span>x</span>Test <i>this</i></p>
```

```
<p>Test <i>this</i><span>x</span></p>
```

```
<p>Test <span>x</span><i>this</i></p>
```

```
<p>Test <i>this</i></p><span>x</span>
```

```
<p>Test <span><i>this</i></span></p>
```

Reset

## Removing DOM nodes and manipulating HTML text

The jQuery methods ***remove()*** and ***detach()*** remove DOM nodes. Both methods are identical except ***detach()*** returns the removed nodes to the caller as a **jQuery** object in case the developer wants to use the nodes for other purposes.

Table 9.5.2: Methods for removing DOM nodes.

Methods	Example	Before	After
<code>remove()</code>	<code>\$(<b>"li"</b>).remove();</code>	<pre> &lt;ol&gt; &lt;li&gt;A&lt;/li&gt; &lt;li&gt;B&lt;/li&gt; &lt;/ol&gt; </pre>	<pre> &lt;ol&gt; &lt;/ol&gt; </pre>
<code>detach()</code>	<pre> let \$listElems = \$(<b>"li"</b>).detach(); </pre>	<pre> &lt;ol&gt; &lt;li&gt;A&lt;/li&gt; &lt;li&gt;B&lt;/li&gt; &lt;/ol&gt; </pre>	<pre> &lt;ol&gt; &lt;/ol&gt; </pre>

jQuery has two methods for getting and setting the HTML or textual content in a webpage: **html()** and **text()**. The `text()` method works like `html()` except `text()` strips out any HTML tags.



Table 9.5.3: Methods for modifying DOM text.

Methods	Example	Before	After
html()	<pre>let s = \$("p").html(); \$("div").html(s);</pre>	<pre>&lt;p&gt; A&lt;b&gt;B&lt;/b&gt;C &lt;/p&gt; &lt;div&gt; &lt;/div&gt;</pre>	<pre>&lt;p&gt; A&lt;b&gt;B&lt;/b&gt;C &lt;/p&gt; &lt;div&gt; A&lt;b&gt;B&lt;/b&gt;C &lt;/div&gt;</pre>
text()	<pre>let s = \$("p").text(); \$("div").text(s);</pre>	<pre>&lt;p&gt; A&lt;b&gt;B&lt;/b&gt;C &lt;/p&gt; &lt;div&gt; &lt;/div&gt;</pre>	<pre>&lt;p&gt; A&lt;b&gt;B&lt;/b&gt;C &lt;/p&gt; &lt;div&gt; ABC &lt;/div&gt;</pre>

PARTICIPATION  
ACTIVITY

9.5.3: Altering the DOM.



Given the HTML below, match the jQuery code to the resulting DOM transformation.

```
<p>  
Check <i>this</i> out!  
</p>
```

If unable to drag and drop, refresh the page.

\$("p").text("<b>Check!</b>");

\$("p").html(\$(".i").detach());

\$("p").html(\$(".p").text());

\$(".i").remove();

<p>Check out!</p>

<p><i>this</i></p>

<p>Check this out!</p>

<p>&lt;b&gt;Check!&lt;/b&gt;</p>

Reset

**PARTICIPATION  
ACTIVITY**

9.5.4: DOM manipulation practice.



The following webpage displays a poem. Use the jQuery DOM modification methods to alter the poem in the following ways when the Scramble button is pressed:

1. Swap the words between the **<strong>** tags from the first and third lines.
2. Detach the last line, and place the line immediately after the first line.
3. Place a single **<div>** around the entire poem that changes the font color to blue using CSS.

Finally, use jQuery to change the link's href attribute to point to <https://www.quora.com/What-is-the-origin-of-the-roses-are-red-violets-are-blue-poem>

Note that the Scramble button is disabled when pressed. Render the webpage again to re-enable the button.

HTML

CSS

JavaScript

```
1 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery"
2
3 <p>Roses are <strong>red</strong>,</p>
4 <p>Violets are <em>blue</em>.</p>
5 <p>Sometimes short <strong>poems</strong></p>
6 <p>Just don't <em>rhyme</em>.</p>
7
8 <div>
9   <button>Scramble</button>
10 </div>
11
12 <a href="https://en.wikipedia.org/wiki/Roses_are_Red" target="blank">
13
```

Render webpage

Reset code

## Your webpage

Roses are **red**,  
Violets are *blue*.  
Sometimes short **poems**  
Just don't *rhyme*.

Scramble

[Origins of the poem](https://en.wikipedia.org/wiki/Roses_are_Red)

## Expected webpage

Roses are **red**,  
Violets are *blue*.  
Sometimes short **poems**  
Just don't *rhyme*.

Scramble

[Origins of the poem](https://en.wikipedia.org/wiki/Roses_are_Red)

► View solution

CHALLENGE  
ACTIVITY

9.5.1: jQuery DOM manipulation.



550544.4142762.qx3zqy7

[Start](#)

For the <img> tag, set src to

<https://resources.zybooks.com/WebProgramming/dice2v1.png> [SHOW EXPECTED](#)

HTML

JavaScript

```
1 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery
2
3 <img>
```

1

2

3

4

5

[Check](#)[Next](#)

Exploring further:

- [jQuery DOM manipulation documentation](#)
- [Manipulating Elements](#)

## 9.6 Ajax

## Loading HTML snippets

Writing Ajax code with plain JavaScript can be a daunting task, so jQuery provides several functions to make writing Ajax code easier. All the methods examined in this section are wrappers around the `XMLHttpRequest` object. The **`load()`** method is used for asynchronously loading a short snippet of HTML that is inserted into a webpage.

### PARTICIPATION ACTIVITY

9.6.1: Using `load()` to load content asynchronously.

```
<body>
  <h1>Movie Information</h1>
  <p id="movieinfo">
    <cite>Star Wars</cite>: Rated PG, released in 1977
  </p>
</body>
```

```
$("#movieinfo").load("starwars.html");
```

HTTP  
request

HTTP  
response

Web server

starwars.html

```
<cite>Star Wars</cite>:
Rated PG, released in 1977
```

### Movie Information

*Star Wars*: Rated PG,  
released in 1977

### Animation content:

Step 1: The following code blocks are displayed.

```
<body>
  <h3>Movie Information</h3>
  <p id="movieinfo">
  </p>
</body>
```

```
$("#movieinfo").load("starwars.html");
```

The text "Movie Information" is displayed on the screen to the right.

Step 2: A HTTP request is sent to the web server. Text reading "starwars.html" is displayed below the code with contents reading "<cite>Star Wars</cite>:"

Rated PG, released in 1977."

Step 3: The text and contents of "starwars.html" are brought to the line of code reading "\$("#movieinfo").load("starwars.html");".

Step 4: The first block of code is changed to read:

```
<body>
  <h3>Movie Information</h3>
  <p id="movieinfo">
    <cite>Star Wars</cite>: Rated PG, released in 1977
  </p>
</body>
```

Step 5: The text displayed on the right is updated to read "Movie Information <i>Star Wars</i>: Rated PG, released in 1977."

### Animation captions:

1. Browser renders the webpage with no movie information.
2. The load() method sends an HTTP request for starwars.html from the web server.
3. The web server responds with starwars.html in the HTTP response.
4. The content of starwars.html is placed in the paragraph with ID movieinfo.
5. The browser updates when the DOM is altered, displaying the movie information.

#### PARTICIPATION ACTIVITY

#### 9.6.2: The load() method.



1) The `load()` method can only download HTML.



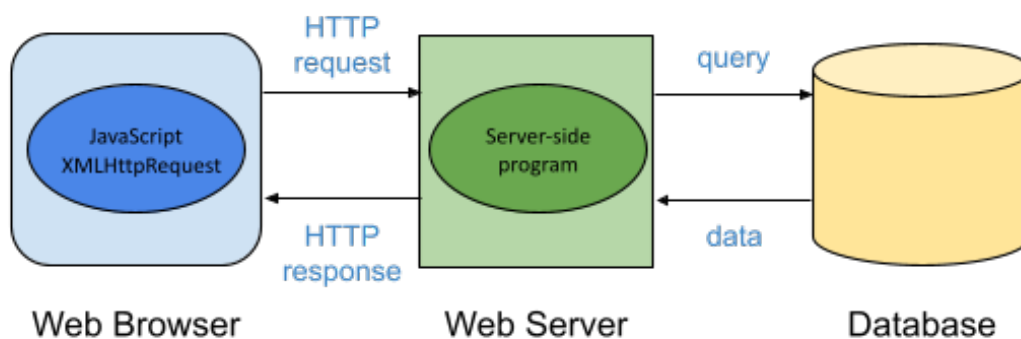
- ☐ True
- ☐ False

- 2) The `load()` method can execute code when the load completes.
- ☐ True
- ☐ False
- 3) The `load()` method uses the `XMLHttpRequest` object internally to make an HTTP request.
- ☐ True
- ☐ False

## GET and POST requests

Most web applications use Ajax to interact with programs that run on a web server. The programs may be created using a number of server-side technologies including PHP, Node.js, ASP.NET, Python, and Java. Regardless of what technology is being used on the web server, the server-side program generally accepts data from an Ajax request, uses the data to query a database, and sends a response back to the web browser.

Figure 9.6.1: Server-side program interacting with web browser and database.



The web server's response may contain HTML or plain text, but frequently the response contains data in XML or JSON format. The jQuery library has functions for parsing XML data, but JSON data is parsed automatically and is generally easier for web developers to work with. JSON is more popular than XML and will therefore be the focus of this section.

## XML vs. JSON

The two most common ways to transport Ajax data is with XML and JSON. XML was once quite popular, hence the "x" in Ajax and "XML" in the XMLHttpRequest object name. However, JSON has become the de facto standard over the last several years primarily because JSON has a more compact structure than XML, making more efficient use of network bandwidth.

XML	JSON
<pre>&lt;movie&gt;   &lt;title&gt;Star Wars&lt;/title&gt;   &lt;rating&gt;PG&lt;/rating&gt;   &lt;year&gt;1977&lt;/year&gt; &lt;/movie&gt;</pre>	<pre>{   "title": "Star Wars",   "rating": "PG",   "year": "1977" }</pre>

The jQuery methods **\$.get()** and **\$.post()** are general-purpose functions for sending asynchronous HTTP GET and POST requests to the web server. Both **\$.get()** and **\$.post()** are methods of the global **jQuery** object, unlike methods like **load()** that work on instances of **jQuery** objects.

The **\$.get()** and **\$.post()** methods send HTTP requests and supply a callback function that executes when the full response is received from the web server. The callback function has a **data** parameter, which contains the server's response.

### PARTICIPATION ACTIVITY

#### 9.6.3: Using \$.get() to load JSON asynchronously.

```
<body>
  <p>Title: <input type="text" id="title"></p>
  <button id="search">Search</button>
  <p id="movieinfo">
    <cite>Star Wars</cite>: Rated PG, released in 1977
  </p>
</body>
```

```
$("#search").click(function() {
  let requestData = { title: $("#title").val() };
  $.get("lookup.php", requestData, function(data) {
    $("#movieinfo").html("<cite>" + data.title +
```

Title:

Search

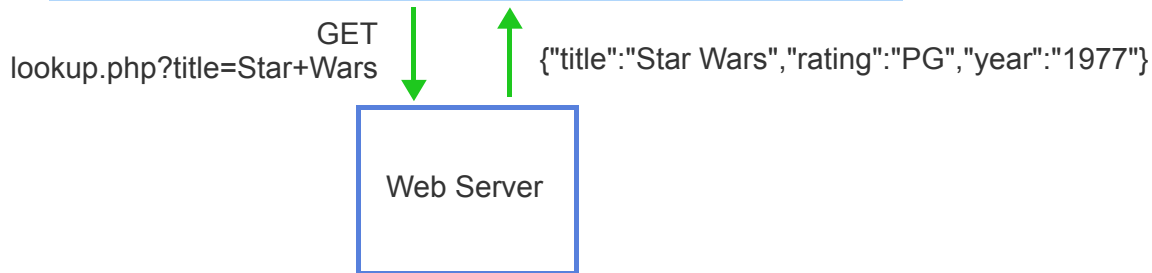
Star Wars: Rated PG,  
released in 1977



```

    "</cite>: Rated " + data.rating +
    ", released in " + data.year);
  }, "json");
});

```



## Animation content:

The following code blocks are displayed.

```
<body>
```

```
  Title: <input type="text" id="title"><br>
```

```
  <button id="search">Search</button>
```

```
  <div id="movieinfo">
```

```
<cite>Star Wars</cite>: Rated PG, released in 1977
```

```
  </div>
```

```
</body>
```

```

$("#search").click(function() {
  let requestData = { title: $("#title").val() };
  $.get("lookup.php", requestData, function(data) {
    $("#movieinfo").html("<cite>" + data.title +
      "</cite>: Rated " + data.rating +
      ", released in " + data.year);
  }, "json");
});

```

Step 1: "Star Wars" is typed in the search bar to the right.

Step 2: The line of code reading "let requestData = { title: \$("#title").val() };" is highlighted.

Step 3: The line of code reading "\$.get("lookup.php", requestData, function(data) {" is highlighted. The GET request sent to the web server is represented by "lookup.php?title=Star+Wars"

Step 4: The server's response is represented by "{"title":"Star Wars","rating":"PG","year":"1977"}"

Step 5: The lines of code reading `"$("#movieinfo").html("<cite>" + data.title + "</cite>: Rated " + data.rating + ", released in " + data.year);"` are highlighted. The first block of code is edited to read as follows.

```
<body>
```

```
Title: <input type="text" id="title"><br>
```

```
<button id="search">Search</button>
```

```
<div id="movieinfo">
```

```
<cite>Star Wars</cite>: Rated PG, released in 1977
```

```
</div>
```

```
</body>
```

The text reading "`<i>Star Wars:</i> Rated PG, releasing in 1977`" is outputted to the user.

### Animation captions:

1. The user types the movie title "Star Wars" and presses the Search button.
2. The requestData object literal is assigned the "Star Wars" value from the text box.
3. `$.get()` makes an HTTP GET request for lookup.php. The data in requestData is sent in the URL's query string.
4. The web server looks up "Star Wars" in a database and sends back a JSON response.
5. jQuery converts JSON into a JavaScript object, which is assigned to the data parameter. The `html()` method displays the movie data in the paragraph.

#### PARTICIPATION ACTIVITY

#### 9.6.4: Sending GET and POST requests.

1) Which method sends data to a web server in an HTTP POST request and may receive back a JSON response?

- ☐ `load()`
- ☐ `$.get()`
- ☐ `$.post()`

2) When is the callback function passed to `$.get()` executed?

- ☐ As soon as the GET request is sent.
- ☐ As soon as the response from the server is received.

3) The code from the animation above was modified to use a named callback function instead of an anonymous function. What is missing from the code?

```
$("#search").click(function()  
{  
    let requestData = { title:  
$("#title").val() };  
    $.get("lookup.php",  
requestData, ____, "json");  
});  
  
function responseHandler(data)  
{  
    $("#movieinfo").html("<cite>" + data.title +  
        "</cite>: Rated " +  
data.rating +  
        ", released in " +  
data.year);  
}
```

- ☐ `responseHandler(data)`
- ☐ `responseHandler()`
- ☐ `responseHandler`

- 4) What does the URL-encoded string that is sent to the web server look like for the following method call?



```
$.post("create.php", { title: "War Games", year: 1983 });
```

- ☐ {"title": "War Games", "year": "1983"}
- ☐ title=War+Games&year=1983
- 5) What other data type arguments could be used instead of "json" in a call to `$.get()` or `$.post()`?
- ☐ "xml"
- ☐ "image"
- ☐ "xml", "script", "html"



## The \$.ajax() method

The **\$.ajax()** method is a general purpose method for making Ajax requests. The **\$.ajax()** method can send GET and POST requests like **\$.get()** and **\$.post()** but can also send PUT and DELETE requests. The **\$.ajax()** method's settings parameter is an object with a number of optional properties, including:

- **url** - URL to request
- **method** - HTTP request method "GET", "POST", etc.
- **data** - Object that is converted into key/value pairs for a query string and sent in the request
- **dataType** - Type of data expecting back, like "html", "json", "text", "xml", etc.

The **\$.ajax()**, **\$.get()**, and **\$.post()** methods all return a **jqXHR** object, which is an abbreviation for **jQuery XMLHttpRequest** object. The **jqXHR** object adds additional capabilities to the **XMLHttpRequest** object the browser uses to make Ajax requests. The **jqXHR** object has a callback method **done()** that is called when a successful HTTP response is received.

Figure 9.6.2: Example call to \$.ajax().

```
// Data to send in request
let requestData = { title: "Star Wars" };

// Make GET request for Star Wars
$.get("lookup.php", requestData, function(data)
{
    console.log("success"); }, "json");

// Same thing as $.get()
$.ajax({
    url: "lookup.php",
    method: "GET",
    data: requestData,
    dataType: "json"
})
.done(function(data) {
    console.log("success");
});
```

## Promises

The *jQueryXHR* implements the Promise interface, so the object has all the properties and methods of a Promise. Promises are covered in detail elsewhere in this material.

### PARTICIPATION ACTIVITY

#### 9.6.5: The \$.ajax() method.



Refer to the code segment below that uses search.php to find book information.

```
$.ajax({
    url: "search.php",
    method: __A__,
    data: __B__,
    dataType: __C__
})
.done(function(data) {
    $("#bookPrice").html("$" + data.price);
});
```

- 1) What replaces A to create a POST request?
- ☐ "POST"
  - ☐ POST
  - ☐ "GET"
- 2) What replaces B to search for a used "Harry Potter" book?
- ☐ [ "Harry Potter", "used" ]
  - ☐ { "Harry Potter", "used" }
  - ☐ { title: "Harry Potter", type: "used" }
- 3) What replaces C to specify the server is returning JSON?
- ☐ json
  - ☐ "json"
  - ☐ JSON



## Cross-origin HTTP request

*For security reasons, browsers limit Ajax requests to the web server from which the JavaScript was downloaded. Ex: JavaScript downloaded from <http://instagram.com> may only make Ajax requests to [instagram.com](http://instagram.com). A **cross-origin HTTP request** is a request made to another domain. Ex: An Ajax request from JavaScript downloaded from [instagram.com](http://instagram.com) to [yahoo.com](http://yahoo.com) is a cross-origin HTTP request. Browsers can make cross-origin HTTP requests using a number of techniques including proxy servers, Cross-Origin Resource Sharing (CORS), and JSON with Padding (JSONP).*

## Handling errors

Developers should write Ajax code that is robust and properly handles error conditions. Two of the most common error conditions are:

1. The web server returns an empty response or a response with an error message because the requested data was not found.
2. The web server returns a 40x or 500 response code.

The `jqXHR` object has a callback method **`fail()`** that is called when a 40x or 500 response code is received from the Ajax request.

**PARTICIPATION  
ACTIVITY**

9.6.6: Robust Ajax code.



Match the code segments below with the type of error that each is designed to catch.

```
// (a)
$.get("lookup.php", requestData, function(data) {
    if (data.title) {
        $("#movieinfo").html("<cite>" + data.title + "</cite>: Rated " +
            data.rating + ", released in " + data.year);
    }
    else {
        $("#movieinfo").html("The movie title could not be found.");
    }
}, "json");
```

```
// (b)
$.get("lookup.php", requestData, function(data) {
    $("#movieinfo").html("<cite>" + data.title + "</cite>: Rated " +
        data.rating + ", released in " + data.year);
}, "json").fail(function(jqXHR) {
    $("#movieinfo").html("There was a problem contacting the server: " +
        jqXHR.status + " " + jqXHR.responseText);
});
```

If unable to drag and drop, refresh the page.

(b)

(a)

Empty response was returned from the web server.

40x or 500 status code was returned from the web server.

Reset

PARTICIPATION  
ACTIVITY

9.6.7: Ajax practice.



The webpage below prompts the user for a ZIP code. When the Search button is pressed, the JavaScript displays the "high" values from the test response, regardless of what ZIP code is typed.

Modify the JavaScript code to make an Ajax request to the web server and display the weather forecast returned by the web server in an ordered list. Use `$.get()` or `$.ajax()` to send the ZIP code in a request to the `weather.php` script located at `https://wp.zybooks.com/weather.php`.

The `weather.php` script looks for the ZIP code in the query string. Ex: `zip=80110`. The server responds with a 200 status code and a JSON object containing a Boolean `success` indicating if the request was successful. If the request is successful, the JSON object also contains `forecast` with an array of forecast details. If the request is not successful, the JSON object contains `error` with a string describing the error.

Display the error message in the webpage if the request is unsuccessful.

Successful request	Unsuccessful request
<pre>{   "success": true,   "forecast": [     { "high": 90, "low": 72, "desc": "sunny" },     { "high": 92, "low": 73, "desc": "partly sunny" },     { "high": 87, "low": 64, "desc": "rain" },     { "high": 88, "low": 65, "desc": "cloudy" },     { "high": 90, "low": 68, "desc": "partly cloudy" }   ] }</pre>	<pre>{   "success": false,   "error": "ZIP code not found" }</pre>



HTML

JavaScript

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jc
5 </head>
6 <body>
7   <p>
8     <label for="zip">ZIP code:</label>
9     <input type="text" id="zip" maxlength="5" size="5">
10    <button id="search">Search</button>
11    <div id="forecast"></div>
12  </p>
13 </body>
14 </html>
15
```

Render webpage

Reset code

Your webpage

Expected webpage

ZIP code:  ZIP code:  

► View solution

## The user interface and Ajax

*Developers sometimes overlook the importance of keeping the user knowledgeable about what the software is doing. Ex: In the weather webpage above, an Ajax request is made in response to a button click, and the webpage is updated with the weather when the response arrives. Usually the weather update appears immediately after the button click, but if the web server is busy or the network is slow, the update could take a few seconds to appear. If the user interface (UI) does not immediately update after the button click, the user may not be aware that the software is busy processing the button click, and the user may press the button again.*

*Good practice is to display a "waiting" message or progress bar immediately when the button is pressed and hide the message or progress bar when the Ajax response is received. When the UI acknowledges the button click immediately, users will normally be patient before trying to click the button again.*

### CHALLENGE ACTIVITY

#### 9.6.1: jQuery Ajax.



550544.4142762.qx3zqy7

## Start

Make a request to <https://wp.zybooks.com/weather.php> with the `requestData` variable, handle the response by passing the `ajaxHandler` function, and a data type of `json`. Hint: A function can be passed as an argument just using the function name, without parentheses. Ex: `ajaxHandler`

```
1 function ajaxHandler(data) {  
2     console.log("Data handled");  
3 }  
4  
5 let requestData = { "zip": "90210" };  
6 $.get(  
7  
8     /* Your solution goes here */  
9  
10 );
```

1

2

3

Check

Next

Exploring further:

- [jQuery Ajax documentation](#)
- [Ajax - jQuery Learning Center](#)
- [HTTP access control \(CORS\)](#)
- [Working with JSONP](#)

## 9.7 Using third-party web APIs (jQuery)

### Introduction

Many organizations have created public web APIs that provide access to the organization's data or the user's data that is stored by the organization. Ex: The Google Maps API provides applications information about geographic locations, and the Instagram API allows applications access to photos shared on Instagram. [Public APIs](#) on GitHub.com lists thousands of free, public web APIs.

A **third-party web API** is a public web API used by a web application to access data provided by a third party. "Third-party" refers to a person or organization that is neither the web application using the API nor the user using the web application, which are the "first" and "second" parties. Websites rely on third-party web APIs to integrate with social media, obtain maps and weather data, or access collections of data.

To use a third-party web API, a developer usually registers with the third party to obtain an **API key**. Third parties require API keys for several reasons:

- The API key identifies who or what application is using the web API.
- The API key helps the third party limit the number of requests made to the API in a fixed time period or may be used to charge a developer a fee for additional requests.
- To obtain an API key, developers must agree to restrictions the third party places on data obtained from the web API.

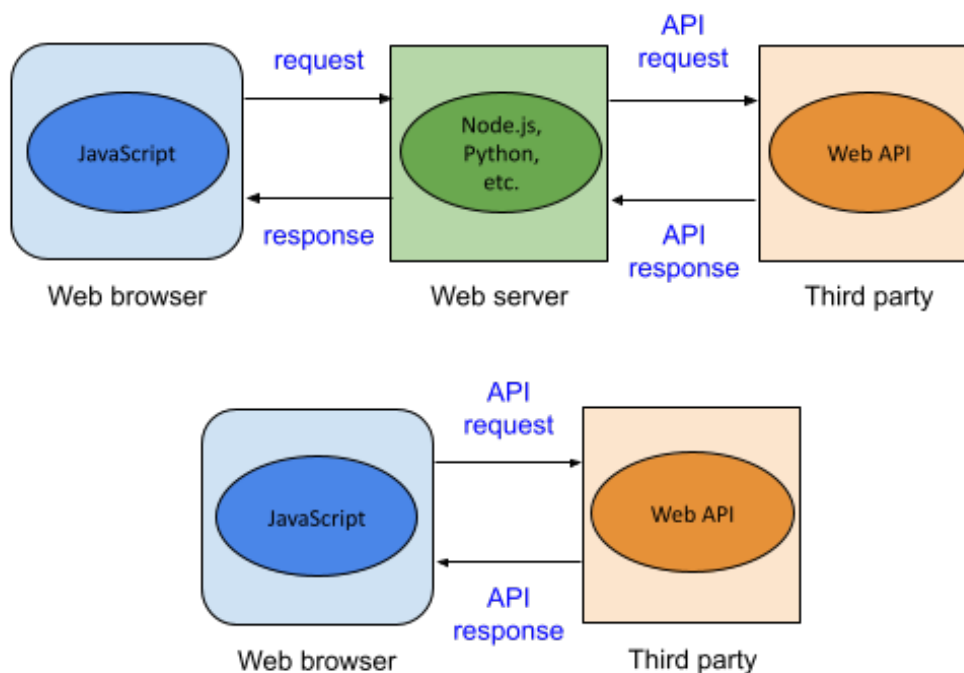
Most third-party web APIs are RESTful. A **RESTful web API** is a web API that is called with a URL that specifies API parameters and returns JSON or XML containing the API data. Ex: The URL `http://linkedin.com/api/article?id=123` specifies the article ID 123, so the article would be returned formatted in JSON.

Third-party web APIs may be called from the web server or the web browser. This material shows how to call web APIs from the web browser using JavaScript.

## SOAP

A **SOAP-based web API** is another type of web API that relies heavily on XML and is in general more complex to use than RESTful web APIs. See the "Exploring further" section for more information on SOAP.

Figure 9.7.1: Calling third-party web API from the web server or web browser.



### PARTICIPATION ACTIVITY

#### 9.7.1: Third-party web APIs.



- 1) Information from a third-party web API reaches the browser faster if the browser calls the web API directly instead of the web server calling the web API.  
☐ True  
☐ False
- 2) For a third-party web API requiring an API key, the API key must be transmitted with every API request.  
☐ True  
☐ False
- 3) When the browser makes an API request to a third-party web API, the web API key can be kept secret from prying eyes.  
☐ True  
☐ False
- 4) Many web APIs charge a fee to the developer after a limited number of requests have been made in a 24-hour period.  
☐ True  
☐ False
- 5) RESTful web APIs only return XML.  
☐ True  
☐ False



## Weather API

OpenWeatherMap provides a free [Weather API](#) providing current weather data, forecasts, and historical data. Developers must register at [openweathermap.org](https://openweathermap.org) for an API key that must be

transmitted in all API requests.

The OpenWeatherMap website provides documentation explaining how to use the Weather API using GET requests with various query string parameters. The API endpoint `http://api.openweathermap.org/data/2.5/weather` returns the current weather based on the following query string parameters:

- zip - Five digit US ZIP code
- units - Standard, metric, or imperial units to use for measurements like temperature and wind speed
- appid - Developer's API key

Other parameters are documented in the OpenWeatherMap website. The Weather API returns weather data in JSON format by default.

Figure 9.7.2: GET request to obtain the current weather for ZIP 90210.

[http://api.openweathermap.org/data/2.5/weather?](http://api.openweathermap.org/data/2.5/weather?zip=90210&units=imperial&appid=APIKEY)

[zip=90210](#)[&units=imperial](#)[&appid=APIKEY](#)

```
{
  "coord":{
    "lon":-118.4,
    "lat":34.07
  },
  "weather":[
    {
      "id":800,
      "main":"Clear",
      "description":"clear sky",
      "icon":"01d"
    }
  ],
  "base":"cmc stations",
  "main":{
    "temp":75.61,
    "pressure":1017,
    "humidity":14,
    "temp_min":60.8,
    "temp_max":82.4
  },
  "wind":{
    "speed":3.36
  },
  "clouds":{
    "all":1
  },
  "id":5328041,
  "name":"Beverly Hills",
  "cod":200
}
```

City's geo location

Overall description

Degrees Fahrenheit

Percent humidity

Minimum and maximum temps at the moment

Miles per hour

Percent cloudy

City



## Try 9.7.1: Try OpenWeatherMap's API in your web browser.

1. Go to [openweathermap.org](https://openweathermap.org).
2. Sign up for an account to obtain an API key.
3. When your API key is ready, try the link:  
<http://api.openweathermap.org/data/2.5/weather?zip=90210&units=imperial&appid=APIKEY> to make an API request for the weather with ZIP 90210. The page should indicate an invalid API key was used.
4. Replace APIKEY in the URL's query string with your API key, and reload the webpage. The JSON-encoded weather information for 90210 should be displayed.
5. Change the ZIP code in the URL's query string to your ZIP code, and reload the URL to see the weather in your ZIP code.

### PARTICIPATION ACTIVITY

#### 9.7.2: The Weather API.

- 1) What does the Weather API return when an invalid API key is used in a request?
  - ☐ A blank webpage
  - ☐ Weather for the 90210 ZIP
  - ☐ An error message formatted in JSON
- 2) In the figure above, what does the Weather API return as the current humidity in the 90210 ZIP code?
  - ☐ 75.61
  - ☐ 14
  - ☐ 3.36

- 3) What "units" parameter value would make the Weather API return the temperature in Celsius?
- ☐ imperial
- ☐ metric
- ☐ standard
- 4) Does the Weather API support finding the current weather by city name?
- ☐ Yes
- ☐ No

## Cross-origin requests

Calling a third-party web API from the web browser requires a cross-origin HTTP request, since the web API is not hosted on the local website's web server. Two main techniques are used to make cross-origin requests:

- **Cross-Origin Resource Sharing (CORS)** is a W3C specification for how web browsers and web servers should communicate when making cross-origin requests.
- **JSON with Padding (JSONP)** is a technique to circumvent cross-origin restrictions by injecting `<script>` elements dynamically into a webpage. Script elements have no cross-origin restrictions.

CORS is the more common of the two techniques and, for the web API user, the easiest to use. CORS requires the web browser to send an **Origin** header in a web API request to indicate the scheme and domain making the API request. If the API accepts the request, the API responds with an **Access-Control-Allow-Origin** header indicating the same value in the **Origin** request header or **"\*"**, which indicates that requests are allowed from any origin. CORS uses other headers that begin with **Access-Control-\*** to support other interactions with the API.

CORS allows the browser to send GET, POST, PUT, and DELETE requests. JSONP limits the browser to sending only GET requests.

Figure 9.7.3: Making a request to the Weather API with CORS.

HTTP request	HTTP response
<pre>GET /data/2.5/weather? zip=90210&amp;units=imperial&amp;appid=APIKEY HTTP/1.1 Host: api.openweathermap.org Origin: http://mywebsite.com User-Agent: Mozilla/5.0 Chrome/48.0.2564</pre>	<pre>HTTP/1.1 200 OK Access-Control-Allow-Origin: * Content-Type: application/json; charset=utf-8 Content-Length: 431 Date: Mon, 28 Mar 2016 16:09:48 GMT Server: openresty  {"coord": {"lon":-118.4,"lat":34.07},"weather": [{"id":500, "main":"Rain","description":"light rain","icon":"10d"}], etc...}</pre>

**PARTICIPATION  
ACTIVITY**

## 9.7.3: Cross-origin requests.

- 1) What HTTP header must the web browser send in every CORS request?

☐ Access-Control-Allow-Origin

☐ Origin

☐ User-Agent
- 2) The web browser knows to send the **Origin** header in the HTTP request when the requested URL's domain name and the requesting script's domain name are \_\_\_\_\_.

☐ the same

☐ different

3) When a third-party web API does not support CORS, what is Access-Control-Allow-Origin set to in the web API's response?

- ☐ Access-Control-Allow-Origin is set to \*.
- ☐ Access-Control-Allow-Origin is set to the Origin value.
- ☐ Access-Control-Allow-Origin is not present.

4) Does JSONP support POST or PUT request methods?

- ☐ Yes
- ☐ No

## Weather API request with jQuery

The jQuery `$.ajax()` method is commonly used to make Ajax requests to third-party web APIs from a webpage. The OpenWeatherMap implements CORS, and API requests can come from any origin.

### PARTICIPATION ACTIVITY

9.7.4: Calling the Weather API from the web browser using CORS.

The webpage below allows the user to enter a ZIP code. When the Find button is clicked, the jQuery code uses `$.ajax()` to send a CORS request to the Weather API.

- If the API returns a 200 status code, the `done()` callback function executes with the API response in the `data` parameter. The callback function displays the current temperature in the webpage.
- If the API returns a 401 or 404 status code, the `fail()` function callback executes. A 401 indicates an invalid API key was used. A 404 indicates the ZIP code could not be found. The callback function displays a general error message.

To make the webpage operational, replace "**APIKEY**" with your API key. Then render the webpage, type in a five digit ZIP code, and press Find. The current temperature for the ZIP

code should display.

Modify the webpage to display the description and humidity by adding HTML code under the "Current temperature" paragraph and by adding JavaScript code in the `done ( )` callback function.

HTML

JavaScript

CSS

```
1 <script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
2
3 <label for="zip">ZIP code:</label>
4 <input type="text" id="zip" maxlength="5" size="5">
5 <input type="button" value="Find" id="findBtn">
6 <p id="working" style="display:none">Retreiving the current weather..
7 <p id="error"></p>
8
9 <div id="results" style="display:none">
10   <p>
11     Current temperature: <strong id="currentTemp">
12       </strong> &deg;F
13   </p>
14 </div>
15
```

Render webpage

Reset code

## Your webpage

ZIP code:

► View solution

## Using JSONP

If a third-party web API does not support CORS, the API may support JSON with Padding (JSONP). The `$.ajax()` method uses the JSONP technique to call a third-party web API when the following parameters are set:

- `dataType` - Set to "jsonp".
- `jsonp` - Set to "callback" or whatever value the third-party web API is expecting.

### PARTICIPATION ACTIVITY

#### 9.7.5: Calling the Weather API using JSONP.



### Web browser

localhost:3000/weather.ht

#### Weather for 9021

Current temp: 53.2 °F  
Description: clear sky  
Humidity: 71%

```
// weather.js
$(function() {
  let zip = 90210;

  $.ajax({
    url: "http://api.openweathermap.org/data/2.5/weather",
    jsonp: "callback",
    dataType: "jsonp",
    data: { zip: zip, units: "imperial", appid: "APIKEY" }
  }).done(function(data) {
```

```

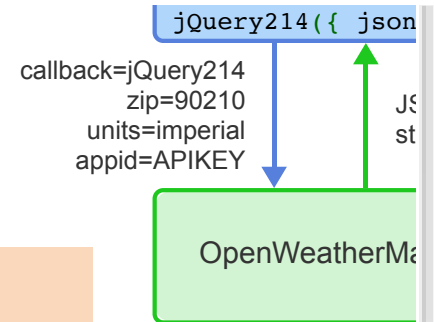
$("#zip").html(zip);
$("#currentTemp").html(data.main.temp);
$("#description").html(data.weather[0].description);
$("#humidity").html(data.main.humidity);
}).fail(function(jqXHR) {
  $("#error").html("Error retrieving the weather.");
});
});

```

```

<!DOCTYPE html>
<html>
  <title>Weather</title>
  <script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
  <script src="scripts/weather.js"></script>
  <script src="http://api.openweathermap.org/data/2.5/weather?
    callback=jQuery214&zip=90210&units=imperial&appid=APIKEY">
  </script>
  <body>
    <h1>Weather for <span id="zip"></span></h1>
    <p id="error" style="color:red"></p>
    <p>Current temp: <span id="currentTemp"></span> &deg;F</p>
    <p>Description: <span id="description"></span></p>
    <p>Humidity: <span id="humidity"></span>%</p>
  </body>
</html>

```



## Animation content:

The following code blocks are displayed.

// weather.js

```

$(function() {
  let zip = 90210;
  $.ajax({
    url: "http://api.openweathermap.org/data/2.5/weather",
    jsonp: "callback",
    dataType: "jsonp",
    data: { zip: zip, units: "imperial", appid: "APIKEY" }
  }).done(function(data) {
    $("#zip").html(zip);
    $("#currentTemp").html(data.main.temp);
    $("#description").html(data.weather[0].description);
    $("#humidity").html(data.main.humidity);
  }).fail(function(jqXHR) {
    $("#error").html("Error retrieving the weather.");
  });
});

```

```
<!DOCTYPE html>
<html>
  <title>Weather</title>
  <script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
  <script src="scripts/weather.js"></script>
  <script src="http://api.openweathermap.org/data/2.5/weather?
    callback=jQuery214&zip=90210&units=imperial&appid=APIKEY">
  </script>
  <body>
    <h1>Weather for <span id="zip"></span></h1>
    <p id="error" style="color:red"></p>
    <p>Current temp: <span id="currentTemp"></span> °F</p>
    <p>Description: <span id="description"></span></p>
    <p>Humidity: <span id="humidity"></span>%</p>
  </body>
</html>
```

Step 1: The browser request is shown as "localhost:3000/weather.html". The html is rendered to read

"Weather for  
Current temp: °F  
Description:  
Humidity: %".

Step 2: The lines of code reading "\$.ajax({  
 url: "http://api.openweathermap.org/data/2.5/weather",  
 jsonp: "callback",  
 dataType: "jsonp",  
 data: { zip: zip, units: "imperial", appid: "APIKEY" }  
}).done(function(data) {" are highlighted in the first block.

The lines of code in the second block reading "<script  
src='http://api.openweathermap.org/data/2.5/weather?<br>callback=jQuery214&zip=90210&units=imperial&appid=APIKEY'><br></script>" are also highlighted.

Step 3: The weather data requested from OpenWeatherApp from the browser is represented to the side as "callback=jQuery214  
zip=90210



```
units=imperial  
appid=APIKEY"
```

Step 4: The JSONP string returned by the browser is represented by "jQuery214({ json })".

Step 5: "jQuery214({ json })" is highlighted to indicate that the weather data is turned into JavaScript objects.

Step 6: The lines of code reading `}).done(function(data) {  
 $("#zip").html(zip);  
 $("#currentTemp").html(data.main.temp);  
 $("#description").html(data.weather[0].description);  
 $("#humidity").html(data.main.humidity);` are highlighted.

The output is updated to read

```
"Weather for 90210  
Current temp: 53.2 °F  
Description: clear sky  
Humidity: 71%"
```

### Animation captions:

1. Web browser requests weather.html from the Express web server and renders the HTML.
2. \$.ajax() method with dataType parameter "jsonp" creates a script element in the DOM, which makes a request to the Weather API using a "callback" parameter.
3. Browser requests weather data from OpenWeatherMap.
4. OpenWeatherMap recognizes the JSONP request because of "callback" in the query string and returns a string containing the JSON weather data wrapped in a function call.
5. Web browser executes the jQuery function, which converts JSON weather data into JavaScript objects.
6. done() function callback is executed with the "data" parameter containing the weather data, and jQuery html() method updates the webpage with the weather data.

#### PARTICIPATION ACTIVITY

9.7.6: Calling the Weather API from the web browser.



1) Does the `$.ajax()` method use the `XMLHttpRequest` object to make an HTTP request when the `dataType` parameter is set to "jsonp"?

- ☐ Yes  
☐ No

2) Does the web browser allow the `<script>` element to make cross-origin HTTP requests?

- ☐ Yes  
☐ No

3) What query string parameter tells OpenWeatherMap that the request is a JSONP request?

- ☐ appid  
☐ zip  
☐ callback

4) In the animation above, what is the name of the function the jQuery library wants the browser to execute when the browser receives the API response containing JSON weather data?

- ☐ jQuery214  
☐ callback  
☐ imperial

Exploring further:

- [Public APIs](#) on GitHub.com
- [Understanding SOAP and REST Basics And Differences](#)
- [jQuery: Working with JSONP](#)

# 9.8 Plugins

## Introduction to jQuery plugins

jQuery's plugin architecture allows developers to add additional functionality like new selectors and user interface (UI) widgets. The [jQuery Plugin Registry](#) organizes plugins into categories and provides links to download and use the plugins. An [active forum](#) exists for developers seeking help creating or using plugins.

PARTICIPATION  
ACTIVITY

9.8.1: Popular jQuery plugins.

Match the jQuery plugin with the plugin's description.

If unable to drag and drop, refresh the page.

- payform
- Cycle2
- Tiny Colorpicker
- menu-aim
- Tooltipster
- animatedModal.js

	Creating slideshows with various transition effects.
	Producing dropdown menus.
	Displaying tooltips.
	Displaying fullscreen modal dialog boxes.
	Allows users to select a color.

Creating credit card forms and validating inputs.

Reset

## Cycle2 plugin

The [Cycle2 plugin](#) is a popular plugin for showing slideshows. Like many plugins, Cycle2 requires little JavaScript coding. The plugin uses special HTML attributes prefixed with "data-cycle" to apply various plugin settings. A [webpage](#) documents how to use Cycle2. Like all jQuery plugins, the plugin library must be downloaded to the browser along with the jQuery library.

### PARTICIPATION ACTIVITY

#### 9.8.2: Cycle2 plugin.



The Cycle2 library is used in the rendered webpage below to cycle through three images. The `<div>` uses "data-cycle" attributes to make the images scroll horizontally in 200 milliseconds with a delay of 2 seconds before the next image is displayed. The JavaScript code uses the `cycle()` method to pause or resume the slideshow when the slideshow is clicked.

Try to make the following modifications:

1. Replace the "scrollHorz" transition with "fadeout", and change the delay between images to 3 seconds.
2. Add Next and Previous buttons that appear below the slideshow.
3. Add the necessary JavaScript so pressing the Next and Previous buttons stops the slide show by calling `cycle("stop")`. Then each button should display the next or previous images by calling `cycle("next")` or `cycle("prev")`.

HTML

JavaScript

```
1 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
2 <script src="https://malsup.github.io/jquery.cycle2.js"></script>
3
4 <div class="cycle-slideshow"
5     data-cycle-fx="scrollHorz"
6     data-cycle-speed="200"
7     data-cycle-timeout="2000">
8
9     
10    
11    
12 </div>
13
```

[Render webpage](#)[Reset code](#)

### Your webpage

[▶ View solution](#)

## jQuery UI plugin

jQuery UI is a popular collection of jQuery plugins that are grouped into four categories:

1. Interactions - support for dragging, dropping, sorting, selecting, and resizing
2. Widgets - buttons, accordions, date pickers, dialog boxes, sliders, etc.
3. Effects - color animation, easing functions, and animation effects
4. Utilities - position elements on the screen, widget factory

Developers can obtain the jQuery UI library from [jqueryui.com](https://jqueryui.com) and place the library on their web server, or developers can use a CDN, such as [Google Hosted Libraries](https://www.google.com/hostedlibraries/), to download the library to the browser. The library works with an external stylesheet, which defines numerous classes needed by the library. The jQuery UI stylesheet must be downloaded to the browser along with the jQuery UI library.

#### PARTICIPATION ACTIVITY

#### 9.8.3: jQuery UI interactions.



The rendered webpage below demonstrates two jQuery UI interactions:

1. Resizing: Drag the right or bottom edges of the rectangle to resize the rectangle. The `resizable()` method enables the resizing of the `<div>` with ID "resizable".
2. Selecting: Click on each of the bulleted superheroes to select a superhero. Dragging while selecting a superhero, or holding down the Control key while selecting superheroes, will enable multiple heroes to be selected. The `selectable()` method enables selecting items in the `<ul>` with ID "selectable".

Add the following code to display the selected superheroes to the JavaScript console:

```
$("#selectable").on("selectableselecting", function(event, ui) {  
    console.log(ui.selecting.innerHTML);  
});
```

Verify the code works by opening the JavaScript console, then click on some superheroes to see the superheroes appear in the console.

[HTML](#)[CSS](#)[JavaScript](#)

```
1 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
2 <script src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.4/jquery-ui.min.js"></script>
3
4 <link rel="stylesheet" href="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.4/themes/smoothness/jquery-ui.css">
5
6 <div id="resizable" class="ui-widget-content">
7   <h3>Resize Me!</h3>
8   <p>This box can be resized.</p>
9 </div>
10
11 <ul id="selectable">
12   <li>Superman</li>
13   <li>Batman</li>
14   <li>Wonder Woman</li>
15   <li>Aquaman</li>
16   <li>The Flash</li>
```

©zyBooks 04/15/24 16:43 2071381  
Marco Aguilar  
CIS192\_193\_Spring\_2024

Render webpage

Reset code

#### Your webpage

### Resize Me!

This box can be  
resized.

- Superman
- Batman
- Wonder Woman
- Aquaman
- The Flash

► View solution

## jQuery UI animations

jQuery UI provides the ability to animate color transitions and the adding and removing of CSS

classes. jQuery UI also adds some fun animation effects.

**PARTICIPATION  
ACTIVITY**

9.8.4: jQuery UI animations.



The rendered webpage below demonstrates jQuery UI animations. Place the mouse over each square to see the animation effects. Examine the JavaScript and note the following:

- The **animate()** method animates the transition from one color to another by specifying the target color.
- The **addClass()** and **removeClass()** methods animate the transition when adding and removing a class to an element by specifying a time argument.
- The **effect()** method uses "shake" and "bounce" animation effects to grab the user's attention. jQuery UI supplies fifteen different animation effects, including drop, explode, and puff.

Add another **<div>** to create a fourth box with the text "TEST". When the mouse hovers over the new box, the following should happen:

1. Use **animate()** to change the border color to some other color.
2. Create a new CSS class that uses a large font size. Use **addClass()** and **removeClass()** with a time argument to add and remove the new class so the box's font changes size.
3. Use **effect()** with the "explode" animation effect to make the square disappear.

HTML

CSS

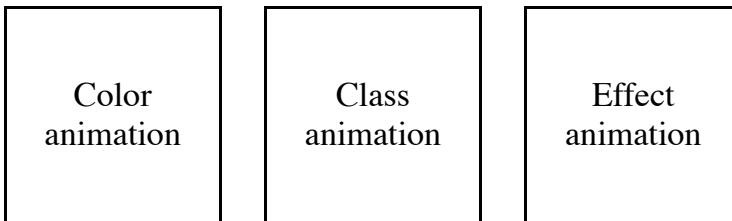
JavaScript



```
1 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
2 <script src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.4/jquery-ui.min.js"></script>
3
4 <link rel="stylesheet" href="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.4/themes/smoothness/jquery-ui.css">
5
6 <div id="colorAnim">
7   Color animation
8 </div>
9
10 <div id="classAnim">
11   Class animation
12 </div>
13
14 <div id="effectAnim">
15   Effect animation
16 </div>
```

[Render webpage](#)[Reset code](#)

#### Your webpage

[► View solution](#)

## jQuery UI widgets

The jQuery UI plugin provides numerous widgets, including a progress bar, autocomplete, and

slider. Some jQuery UI plugin widgets are similar to their HTML counter-part, but other widgets like the accordion and tabs do not have an HTML equivalent. The datepicker widget is especially useful since all browsers do not currently support the HTML datepicker widget using `<input type="date">`.

**PARTICIPATION  
ACTIVITY**

## 9.8.5: jQuery UI widgets.



The rendered webpage below demonstrates three jQuery UI widgets:

- Datepicker: Click the date text field to display the datepicker widget. The `datepicker( )` method adds a click event callback function to the text field to display the datepicker when the text field is clicked.
- Dialog: Click the "Show dialog box" button to display the dialog box. The `dialog( )` method displays the dialog box. The dialog box remains visible until the user clicks the "X" to dismiss the dialog box. The [online documentation](#) shows how to create dialog boxes with customized buttons.
- Tabs: Click each tab to reveal the tab's contents. The `tab( )` method converts the `<div>` with nested tags into a set of tabs. Each tab's title is specified in an unordered list, and each tab's content is specified within a `<div>` with the tab's id.

Try to modify the JavaScript so the dialog box displays the date that is currently selected. If no date is selected, the dialog box should say "No date was selected."

HTML

JavaScript

```
1 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
2 <script src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.4/jquery-ui.min.js"></script>
3
4 <link rel="stylesheet" href="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.4/themes/smoothness/jquery-ui.css">
5
6 <p>
7 Registration date: <input type="text" id="datepicker">
8 </p>
9
10 <p>
11 <button id="showDialog">Show dialog box</button>
12 </p>
13
14 <div id="dialog" title="User error">
15   <p>I'm sorry, Dave. I'm afraid I can't do that.</p>
16 </div>
```

[Render webpage](#)[Reset code](#)

### Your webpage

Registration date:

Show dialog box

Breakfast

Lunch

Dinner

Pancakes and sausage.

► [View solution](#)



1) jQuery UI is well adapted for mobile devices.

- ☐ True  
☐ False

2) The jQuery ThemeRoller allows developers to customize the look and feel of jQuery UI widgets.

- ☐ True  
☐ False

3) Developing jQuery plugins is a difficult undertaking.

- ☐ True  
☐ False

Exploring further:

- [jQuery Plugin Registry](#)
- Plugins: [Cycle2](#), [menu-aim](#), [Tooltipster](#), [animatedModal.js](#), [Tiny Colorpicker](#), [payform](#)
- [jQuery UI API documentation](#)
- [jQuery UI - jQuery Learning Center](#)
- [How to Create a Basic Plugin](#)
- [jQuery Mobile](#)
- [Google Hosted Libraries](#)

## 9.9 Example: Weather Comparison (jQuery)

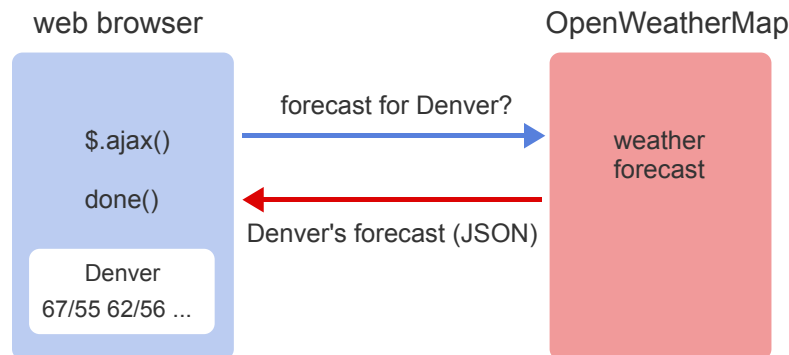
### Weather Comparison page HTML and CSS

This section presents a jQuery implementation of the Weather Comparison web app from an earlier

section. The weather app compares the weather forecast between two cities using the OpenWeatherMap's free [5 day weather forecast API](#).

**PARTICIPATION  
ACTIVITY**

9.9.1: Using the OpenWeatherMap web API to retrieve a weather forecast.

**Animation content:**

Step 1: The web browser sends a request labeled "forecast for Denver?" from the browser's `$.ajax()` function to OpenWeatherMap's weather forecast.

Step 2: OpenWeatherMap's response is labeled "Denver's forecast (JSON)".

Step 3: The web browser's `done()` function receives OpenWeatherMap's response, and a preview of the five day forecast is displayed as "Denver 67/55 62/56".

**Animation captions:**

1. The jQuery `$.ajax()` method requests a city's forecast from the OpenWeatherMap's web API.
2. OpenWeatherMap responds with JSON that includes the city's 5 day forecast.
3. The `done()` callback receives the JSON response and displays the forecast in the browser.

## HTML and CSS for Weather Comparison app.

The HTML for the page has two text inputs and a Compare button so the user can enter the two

cities to compare. The 5 day forecast will be displayed below in two tables. The webpage uses an external stylesheet styles.css for styling the page. The weather.js file will be implemented further below in this section.

weather.html

styles.css

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>Weather Comparison</title>
6     <link rel="stylesheet" href="styles.css">
7   </head>
8   <body>
9     <h1>Weather Comparison</h1>
10    <section id="weather-input">
11      <div>
12        <label for="city1">City 1:</label>
13        <input type="text" id="city1">
14        <span class="error-msg hidden" id="error-value-city1">Enter a ci
15      </div>
16    </div>
```

Render webpage

Reset code

Your webpage

# Weather Comparison

City 1: City 2:

**PARTICIPATION  
ACTIVITY**

## 9.9.2: Weather Comparison HTML and CSS.



Refer to the Participation Activity above.

- 1) All the page contents are visible in the rendered webpage.

☐ True

☐ False
- 2) Removing the **hidden** class from the forecast **<section>** displays two "Loading..." messages and two empty tables.

☐ True

☐ False
- 3) An error message displays if Compare is pressed without first entering two city names.

☐ True

☐ False



## Adding event handling

Callback functions must be implemented for several events:

- ready - DOM is ready, so register other event callbacks.
- click - Compare button is clicked, so forecast should be fetched if city names are provided.
- input - Text box input is changed and must be validated.

The figure below waits until the jQuery ready event is triggered to register callback functions **compareBtnClick()** for the click event and **cityInput()** for input events. Both methods show error messages if the user does not enter text in both text boxes.

Figure 9.9.1: Event callback functions.

```
// Wait until DOM is ready to register callbacks
$(function() {
    $("#compareBtn").click(compareBtnClick);
    $("#city1").on("input", cityInput);
    $("#city2").on("input", cityInput);
});

// Called when city input values change
function cityInput(e) {
    // Extract the text from city input that triggered the
    callback
    const cityId = e.target.id;
    const city = $("#${cityId}`).val().trim();

    // Only show error message if no city
    if (city.length === 0) {
        showElement("error-value-" + cityId);
    }
    else {
        hideElement("error-value-" + cityId);
    }
}

// Compare button is clicked
function compareBtnClick() {
    // Get user input
    const city1 = $("#city1").val().trim();
    const city2 = $("#city2").val().trim();

    // Show error messages if city fields left blank
    if (city1.length === 0) {
        showElement("error-value-city1");
    }
    if (city2.length === 0) {
        showElement("error-value-city2");
    }

    // Ensure both city names provided
    if (city1.length > 0 && city2.length > 0) {
        showElement("forecast");
        hideElement("error-loading-city1");
        hideElement("error-loading-city2");
        showElement("loading-city1");
        showText("loading-city1", `Loading ${city1}...`);
        showElement("loading-city2");
        showText("loading-city2", `Loading ${city2}...`);
        hideElement("results-city1");
        hideElement("results-city2");

        // Fetch forecasts
        getWeatherForecast(city1, "city1");
        getWeatherForecast(city2, "city2");
    }
}
```



## Event callbacks implemented.

Two `<script>` elements are added to the HTML to import the jQuery library and weather.js.

Press Compare before entering a city name to see error messages. Then enter two city names and press Compare. The `getWeatherForecast()` function has not been implemented yet, so no forecasts are displayed.

[weather.html](#)[styles.css](#)[weather.js](#)

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>Weather Comparison</title>
6     <link rel="stylesheet" href="styles.css">
7     <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery">
8     <script src="weather.js"></script>
9   </head>
10
11   <body>
12     <h1>Weather Comparison</h1>
13     <section id="weather-input">
14       <div>
15         <label for="city1">City 1:</label>
16         <input type="text" id="city1">
```

[Render webpage](#)[Reset code](#)

## Your webpage

# Weather Comparison

City 1: City 2: **PARTICIPATION  
ACTIVITY**

## 9.9.3: Handling events.

- 1) What displays if the user enters a space in the City 1 text box?
- ☐ Nothing.
  - ☐ Error message is displayed next to the text box.
  - ☐ Error messages are displayed next to both text boxes.
- 2) Which jQuery method causes the error message to appear?
- ☐ `display()`
  - ☐ `css()`
  - ☐ `show()`

3) Which jQuery method displays a "Loading" message?

- ☐ text()
- ☐ html()
- ☐ show()

## Requesting the forecast

The `getWeatherForecast()` function uses the `$.ajax()` method to request the forecast from the forecast web API. If the API successfully returns the forecast, `displayForecast()` is called to populate the table with the forecast data. If the API returns an error response (perhaps the city name was not found), `displayError()` is called to display an appropriate error message.

Figure 9.9.2: `getWeatherForecast()` function.

```
// Request this city's forecast
function getWeatherForecast(city, cityId) {
    // Must supply API key
    const apiKey = "API key goes here";

    // Make GET request to API for the given city's forecast
    $.ajax({
        url:
            "https://api.openweathermap.org/data/2.5/forecast",
        method: "GET",
        data: { q: city, units: "imperial", appid: apiKey },
        dataType: "json"
    })
    .done(function(data) {
        displayForecast(data, cityId, city);
    })
    .fail(function() {
        displayError(cityId, city);
    });
}
```

## Complete Weather Comparison app.

The complete implementation is provided below. Enter two city names like "Denver" and "Seattle" and press Compare to see the forecasts compared.

The implementation uses zyBook's API key. You may need to replace the API key in `getWeatherForecast()` with your own key due to API query limits.

weather.html

styles.css

weather.js

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>Weather Comparison</title>
6     <link rel="stylesheet" href="styles.css">
7     <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.
8     <script src="weather.js"></script>
9   </head>
10
11   <body>
12     <h1>Weather Comparison</h1>
13     <section id="weather-input">
14       <div>
15         <label for="city1">City 1:</label>
16         <input type="text" id="city1">
```

Render webpage

Reset code

Your webpage

# Weather Comparison

City 1: City 2:

**PARTICIPATION  
ACTIVITY**

## 9.9.4: Requesting the forecast.



Refer to the Participation Activity above.

- 1) The `$.ajax()` method makes a POST request.

☐ True

☐ False
- 2) The `$.get()` method could be used instead of `$.ajax()`.

☐ True

☐ False
- 3) What type of data does `$.ajax()` expect to receive?

☐ XML

☐ JSON
- 4) The `done()` callback is called when the API returns a 404 status code.

☐ True

☐ False
- 5) The `data` parameter passed to the `done()` callback contains an object with the parsed JSON response.

☐ True

☐ False



## 9.10 LAB: Currency Conversion (jQuery)