## Exercise Preview


## Exercise Overview

In this exercise, you'll build a postcard generator. We'll have a single page where the user will choose from a list of U.S. states and the page will dynamically generate a postcard for that state. You'll learn about pulling values from a menu list, and use those values to change the text and image on the page.

---

## Getting Started

1. Open your code editor if it isn't already open.
2. Close any files you may have open.
3. For this exercise we'll be working with the **Postcard-Generator** folder located in **Desktop > Class Files > yourname-JavaScript jQuery Class**. You may want to open that folder in your code editor if it allows you to (like Sublime Text does).
4. Open **index.html** from the **Postcard-Generator** folder.
5. Preview **index.html** in Chrome (we'll be using its DevTools later).

   Notice the **Choose a State** menu on the left. When the user selects a state, we want to use JavaScript to generate a postcard with that state's name and a photo (showing the shape of the state).
6. Leave the page open in Chrome so we can come back to it later.
7. Let's take a look at the provided files. On your Desktop, navigate into **Class Files > yourname-JavaScript jQuery Class > Postcard-Generator** folder.

8. Open the **img** folder inside the **Postcard-Generator** folder.

   Within this folder there are 52 images: one for each of the 50 states, one for D.C., and one for the default "empty-state". We've named each state's image with the appropriate 2-letter abbreviation for that state.

9. Switch back to **index.html** in your code editor.

10. On line 21, find the **select** tag.

    The **select** tag creates the **Choose a State** menu. Notice it has an ID of **states**.

11. Within the **select** tag, notice there are **option** tags for each state.

    Each **option** tag has a **value** attribute, which we have assigned a 2-letter state abbreviation. These abbreviations match our corresponding image filenames. To generate the postcard, we will get the specific value from the option that the user chooses. We'll then use that value to know which image to display.

12. On line 78, locate the **div** tag with an ID of **postcard**. We'll be assigning the selected state's image (as a background-image) to this div.

13. On line 80, locate the **p** tag with an ID of **state-heading**. We'll be changing this text to the state's name.

---

## Getting Input From the Menu

Now that we have an idea of how the menu will work, we can use JavaScript to figure out which state the user has selected.

1. Add the following bold code before the closing **</body>** tag (around line 90):

```
</footer>
<script>

</script>
</body>
```

2. To make it easier to refer to the states select menu, let's store it in a variable. Type the following bold code:

```
<script>
   var states = document.getElementById('states');
</script>
```

3. When a user selects a new state from the menu, the menu fires an **onchange** event which we can listen for. We'll add a function that will only be triggered when the user selects a new state. Add the following code shown in bold:

```
var states = document.getElementById('states');
states.onchange = function() {

}
```

Unlike in previous exercises, notice that we are using an **onchange** event handler directly in JavaScript. This way we don't have to add it to the HTML.

Secondly, notice we did not give the function a name. These are called **anonymous functions**. Because it's a function, it will prevent the code we're going to put inside it from being executed right away. But we don't need to give this function a name, because it will be automatically executed when the states menu changes.

> **Unobtrusive JavaScript**
>
> Previously, whenever we wanted to trigger a function (when the user clicked on, focused on, or blurred off of an element), we'd add an event like **onclick** on the HTML element which calls a JavaScript function.
>
> **Unobtrusive JavaScript** is an approach that separates JavaScript and HTML code. By keeping all the JavaScript together, it's easier to find and edit.

4. The states menu contains all of the **option** values, which are also stored in an **options** array. We can figure out which **option** (state) the user selected from the menu, based on its position in the array (its "index").

We can use **states.selectedIndex** to get the position (index) of the selected menu option within the array. Add the following bold code:

```
states.onchange = function() {
   console.log('state index: ' + states.selectedIndex);
}
```

5. Let's test this out. Save the file.
6. Preview **index.html** in Chrome.
7. Open Chrome's Console by hitting **Cmd–Opt–J** (Mac) or **Ctrl–Shift–J** (Windows).
8. Select **Alabama** from the menu.

It should return **state index: 1**. Alabama has an index of 1 because the default "empty-state" (United States of America) is the first option (0) in the array.

9. Choose a different state to see that the Console returns a different index number for that selected state.

10. Return to **index.html** in your code editor.
11. Let's see if we can get a state's name. Add the following bold code:

```
states.onchange = function() {
   console.log('state index: ' + states.selectedIndex);
   console.log('state name: ' + states.options[0].text);
}
```

12. Save the file.
13. Switch to **index.html** in Chrome and reload the page.
14. From the menu, choose a state. Then choose a second state.

    Notice that the Console will show the appropriate **state index** of the selected state, but **state name** will always show **United States of America**. This is because we told the **options** array to only look for the first value (index 0).

15. Return to **index.html** in your code editor.
16. Instead of 0, let's feed **states.selectedIndex** into the **options[]** array so we'll get the name of the selected state. Add the following bold code:

```
states.onchange = function() {
   console.log('state index: ' + states.selectedIndex);
   console.log('state name: ' + states.options[states.selectedIndex].text);
}
```

17. Save the file, switch to Chrome, and reload the page.
18. Choose a state from the menu. The Console now returns both the appropriate **state index** and **state name**. Now we know how to get the values we need, we can continue on!
19. Switch back to your code editor.
20. We also need an image for the postcard, so we'll get the path to the image for the selected state. Add the following bold code:

```
states.onchange = function() {
   console.log('state index: ' + states.selectedIndex);
   console.log('state name: ' + states.options[states.selectedIndex].text);
   console.log('img/' + states.value + '@2x.jpg');
}
```

21. Save the file, switch to Chrome, and reload the page.
22. Choose a state from the menu to see that the Console now also returns the image path to the corresponding image for the selected state. For example, if you choose **Alabama** you should see **`img/al@2x.jpg`** in the Console.
23. Switch back to your code editor.

24. Now that we know the image path works, let's use it to change the background-image of the **postcard** element. Add the following bold code (keep it on one line):

```
states.onchange = function() {
   console.log('state index: ' + states.selectedIndex);
   console.log('state name: ' + states.options[states.selectedIndex].text);
   console.log('img/' + states.value + '@2x.jpg');
   document.getElementById('postcard').style.backgroundImage = 'url(img/' +
states.value + '@2x.jpg)';
}
```

25. Save the file, switch to Chrome, and reload the page.

26. Choose a few states, and notice on the postcard, that the background image behind **Choose a State** should change to the state you selected.

27. Switch back to your code editor.

28. We want the name of the selected state to appear on the postcard. We've already seen how to get the value from the selected menu option, so now we can use this to change the **state-heading** text. Add the following bold code (keep it on one line):

```
   document.getElementById('postcard').style.backgroundImage = 'url(img/' +
states.value + '.jpg)';
   document.getElementById('state-heading').innerHTML =
states.options[states.selectedIndex].text;
}
```

    NOTE: We use **innerHTML** to refer to the contents inside an HTML element.

29. Save the file, switch to Chrome, and reload the page.

30. Choose a few states, and notice on the postcard, that **Choose a State** should change to the name of the state you selected.

31. Switch back to your code editor.

32. Finally, we'll finish off the postcard by changing the **greeting** element's text to **Greetings from**. Add the following bold code:

```
document.getElementById('postcard').style.backgroundImage = 'url(img/' +
states.value + '.jpg)';
document.getElementById('greeting').innerHTML = 'Greetings from';
document.getElementById('state-heading').innerHTML =
states.options[states.selectedIndex].text;
```

33. Save the file, switch to Chrome, and reload the page.

34. Choose a state, and notice that **To Generate a Postcard** should change to **Greetings from**.

    Everything should be working now. Every time you choose a state, a complete postcard should be generated.

35. Switch back to your code editor.

36. We're done testing, so comment out the three **console.log()** lines by adding **//** to the beginning of each line.

    TIP: In Sublime Text you can select the lines and hit **Cmd–/** (Mac) or **Ctrl–/** (Windows) to comment them out.

    NOTE: When you're done troubleshooting, it's a best practice to either delete the Console code or comment it out.

37. Save the file.

    NOTE: If you want to refer to our final code example, go to **Desktop > Class Files > yourname-JavaScript jQuery Class > Done-Files > Postcard-Generator**.

---