Lesson Monday

Introduction to Programming (/introduction-to-programming) / Arrays and Looping (/introduction-to-programming/arrays-and-looping) / Gathering Data with Checkboxes

Text Cheat sheet

We've learned how to use a variety of HTML form inputs (https://www.learnhowtoprogram.com/introduction-to-programming/javascript-and-web-browsers/form-input-types). We held off on learning about checkboxes because we needed to learn about arrays first. Now that we have a basic understanding of arrays, let's learn how to use checkboxes. In this lesson, we'll also learn how to use `Array.prototype.forEach()` on an array of checkbox elements.

Transportation Survey

In this lesson, we'll build a small website that contains a survey. Our survey will ask users which methods of transportation they have used to travel to work or school in the past year. For now, the site will simply display a user's responses back to them after the form has been submitted.

Checkboxes differ from radio buttons because they allow *multiple* options to be selected at once while radio buttons are for selecting a *single* option. Since many people have likely used more than one

mode of transportation in the past year, we'll use checkboxes to collect our survey answers. This will allow users to select *all* answers that apply to them.

Here's the HTML for our form:

transportation_survey/index.html

```
<!DOCTYPE html>
<html lang="en-US">
<head>
  <title>Transportation Survey</title>
  <link href="css/styles.css" rel="stylesheet" type="text/c
ss">
  <script src="js/scripts.js"></script>
</head>
<body>
  <h2>Transportation Survey</h2>
  <form id="transportation_survey">
    <p>In the past year, I have used the following modes of
transportation to travel to work or school:</p>
    <label>
      <input type="checkbox" name="transportation-option" v
alue="bike">
      Riding a bike.
    </label><br />
    <label>
      <input type="checkbox" name="transportation-option" v
alue="car">
      Driving a car.
    </label><br />
    <label>
      <input type="checkbox" name="transportation-option" v
alue="carpool">
      Carpooling with others.
    </label><br />
    <label>
      <input type="checkbox" name="transportation-option" v
alue="walk">
      Walking.
    </label><br />
    <label>
      <input type="checkbox" name="transportation-option" v
alue="bus">
      Riding the bus.
    </label><br />
    <label>
      <input type="checkbox" name="transportation-option" v
alue="train">
```

```
      Riding the train.
    </label><br />
    <label>
      <input type="checkbox" name="transportation-option" v
alue="streetcar">
      Riding the streetcar.
    </label><br />
    <label>
      <input type="checkbox" name="transportation-option" v
alue="taxi">
      Taking a taxi.
    </label><br />
    <label>
      <input type="checkbox" name="transportation-option" v
alue="rideshare">
      Using a rideshare service like Lyft or Uber.
    </label><br />
    <label>
      <input type="checkbox" name="transportation-option" v
alue="skateboard">
      Skateboarding.
    </label><br />
    <label>
      <input type="checkbox" name="transportation-option" v
alue="rollerblade">
      Rollerblading.
    </label><br />
    <label>
      <input type="checkbox" name="transportation-option" v
alue="scooter">
      Riding a scooter.
    </label><br />
    <label>
      <input type="checkbox" name="transportation-option" v
alue="other">
      Another mode of transportation not listed here.
    </label><br />
    <button type="submit">Submit Survey</button>
  </form>
</body>
</html>
```

We've already linked a custom stylesheet and a `scripts.js` file in the `<head>` tag. If you want to add Bootstrap, remember that you'll have to either download it or use a CDN, and then link to Bootstrap in the head.

Just like with radio buttons, checkboxes all should share the same value for its `name` attribute. This relates every checkbox to one another and ensures that every checkbox gets submitted to the server when the form gets submitted. Right now, we're not submitting our checkbox values to any server, so this isn't vital, but instead good practice.

Also, just like with radio buttons, we can nest the input inside of the label. This makes it so we can click on the label to check the box, as well as the checkbox. It's not required, but it sure is nice from a user experience point of view. We can also enable this feature without nesting the HTML elements by using the setting the same value for the `for` attribute on the `<label>` and the `id` attribute on the `<input>`:

```
<label for="bike">Riding a bike.</label><br />
<input type="checkbox" name="transportation-option" value
="car" id="car">
<label for="car">Driving a car.</label><br />
<input type="checkbox" name="transportation-option" value
="carpool" id="carpool">
<label for="carpool">Carpooling with others.</label><br />
<input type="checkbox" name="transportation-option" value
="walk" id="walk">
<label for="walk">Walking.</label><br />
<input type="checkbox" name="transportation-option" value
="bus" id="bus">
<label for="bus">Riding the bus.</label><br />
<input type="checkbox" name="transportation-option" value
="train" id="train">
<label for="train">Riding the train.</label><br />
<input type="checkbox" name="transportation-option" value
="streetcar" id="streetcar">
<label for="streetcar">Riding the streetcar.</label><br />
<input type="checkbox" name="transportation-option" value
="taxi" id="taxi">
<label for="taxi">Taking a taxi.</label><br />
<input type="checkbox" name="transportation-option" value
="rideshare" id="rideshare">
<label for="rideshare">Using a rideshare service like Lyft
or Uber.</label><br />
<input type="checkbox" name="transportation-option" value
="skateboard" id="skateboard">
<label for="skateboard">Skateboarding.</label><br />
<input type="checkbox" name="transportation-option" value
="rollerblade" id="rollerblade">
<label for="rollerblade">Rollerblading.</label><br />
<input type="checkbox" name="transportation-option" value
="scooter" id="scooter">
<label for="scooter">Riding a scooter.</label><br />
<input type="checkbox" name="transportation-option" value
="other" id="other">
<label for="other">Another mode of transportation not liste
d here.</label><br />
```

Right now there's no obvious area in our HTML where we'll show the results of the transportation survey. That's because we'll use the `document.createElement()` and `Element.append()` methods to add the results of the survey from our scripts!

Retrieving Checkbox Values

In order to determine which checkboxes the user has selected, we can fetch the form data in our user interface logic like this:

transportation_survey/js/scripts.js

```
// User Interface Logic
function handleForm(event) {
  event.preventDefault();
  const userSelections = document.querySelectorAll("input[n
ame=transportation-option]:checked");
  // we'll add more code to display results here!
}

window.addEventListener("load", function() {
  document.querySelector("form#transportation_survey").addE
ventListener("submit", handleForm);
});
```

Towards the bottom of our scripts we have the familiar load event handler attached to the `window` object, and directly inside of it we target the form element and attach an event listener for the 'submit' event, passing in the callback `handleForm`.

The `handleForm()` function is at the top of our scripts, and this function handles getting the form values and displaying the results on the webpage. Right now we have two lines of code:

- `event.preventDefault();` is added to prevent the 'submit' event's default action of refreshing the page.

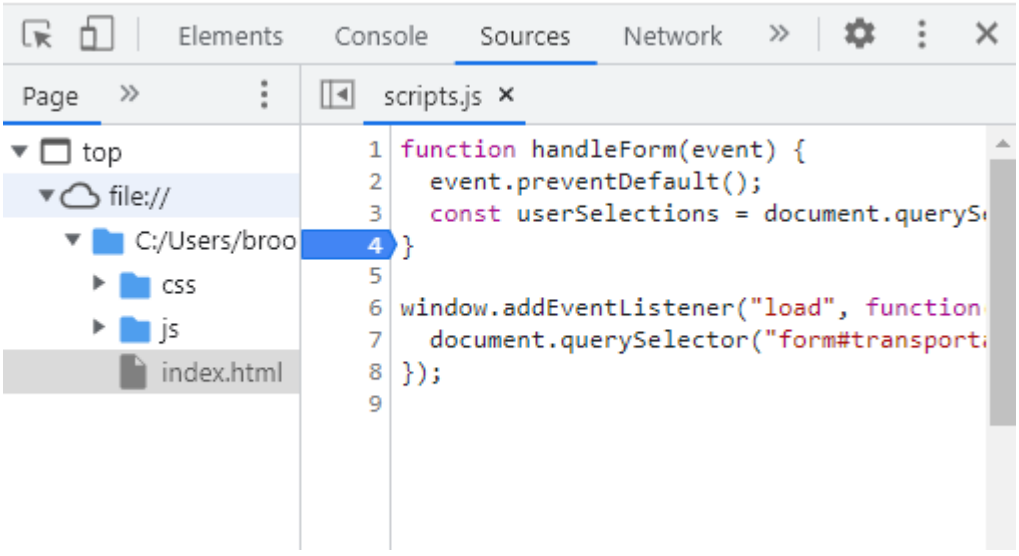
- `const userSelections = document.querySelectorAll("input[name=transportation-option]:checked");` is added to get all of the checkbox inputs that have been checked. We could instead pass in these arguments to the `document.querySelectorAll()` method:
  - `"[name=transportation-option]:checked"`
  - `"input:checked"`
  - Whatever you choose really depends on what's easiest to read so that your code is easy to understand, and what other inputs and forms you have in your webpage.

Next, let's add code that will print all of the user's selections to the webpage. To do this, we'll need to loop through all of the checkbox selections and print each one as a result on the page.

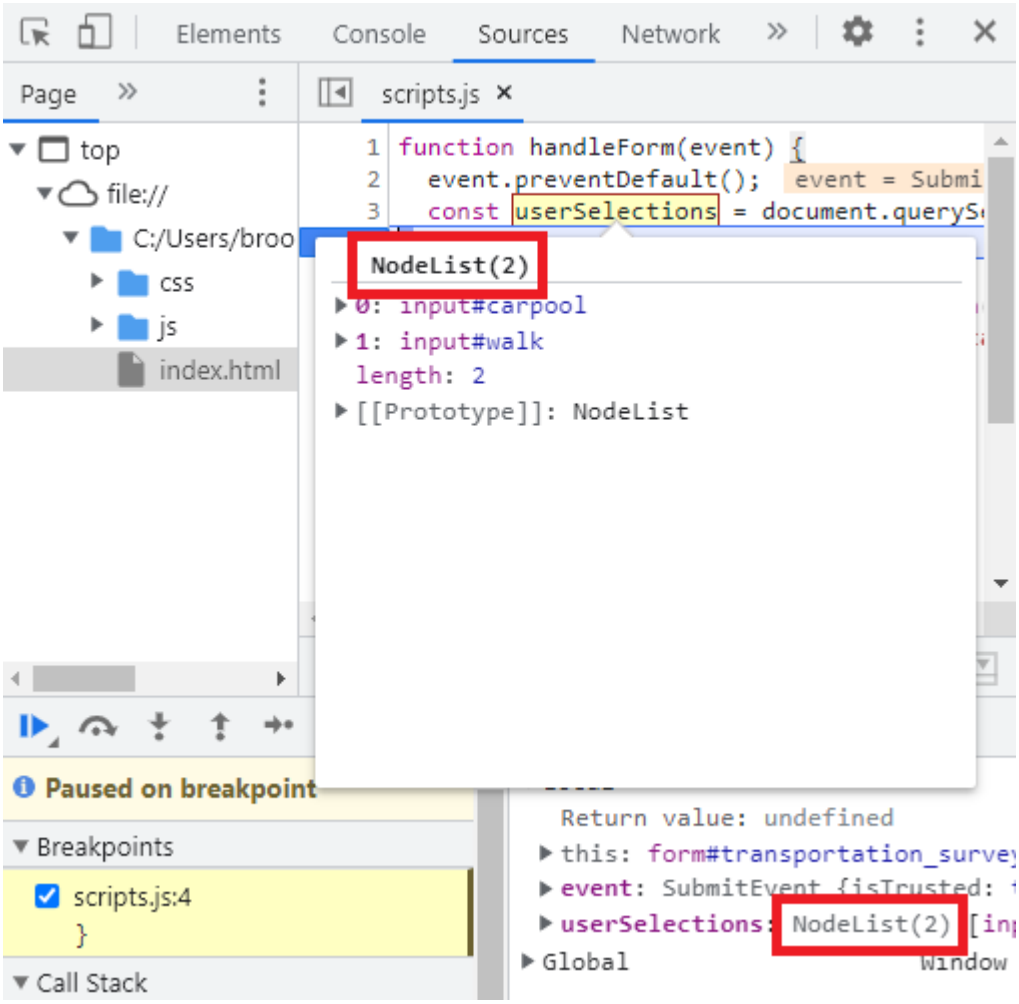
We'll use `Array.prototype.forEach()` to loop through the checkbox selections. But is `userSelections` an array? This is unclear. Let's add a breakpoint to our code to verify what data type `userSelections` is.

Using Breakpoints To Verify `userSelections` Data Type

To check the type of the `userSelections` variable, we should add a breakpoint to the line right after it. This is actually the last line of the `handleForm()` function.



Then, let's submit the form. We'll see the "Paused in debugger" message, and we should be able to hover over the `userSelections` variable to determine the type.



As the image shows, we can tell we're dealing with a `NodeList` object. Every element in the `NodeList` object is an `HTMLInputElement` object. The number of elements inside of the `NodeList` object will vary based on how many checkbox inputs we've selected before submitting the form. In this case, I've selected two transportation options: carpooling and walking.

Instead of hovering over the `userSelections` variable in the Sources tab, we could also just review the list of variables in the "Scope" window to the bottom right of the Sources tab. Either way works to verify the information we need!

Now that we know that we're dealing with a `NodeList` object, we know that we have to turn this object into an array if we want to call array methods on it. Let's do that next, and let's also add the code to loop through the checkbox selections and print each to the webpage.

Looping through Checkbox Selections

transportation_survey/js/scripts.js

```
// User Interface Logic
function handleForm(event) {
  event.preventDefault();
  const userSelections = document.querySelectorAll("input[n
ame=transportation-option]:checked");
  const userSelectionsArray = Array.from(userSelections);

  userSelectionsArray.forEach(function(element) {
    const paragraph = document.createElement("p");
    paragraph.append(element.value);
    document.body.append(paragraph);
  });
}

window.addEventListener("load", function() {
  document.querySelector("form#transportation_survey").addE
ventListener("submit", handleForm);
});
```

Let's break down the new code we've added. First, I begin by changing the `userSelections` variable into an array with the following code:

```
  const userSelectionsArray = Array.from(userSelections);
```

`Array.from()` is a **static** array method. This is in contrast to **instance** methods, like `Array.prototype.forEach()`.

Then, I loop through the `userSelectionsArray` variable with the `Array.prototype.forEach()` method:

```
userSelectionsArray.forEach(/*argument goes here*/);
```

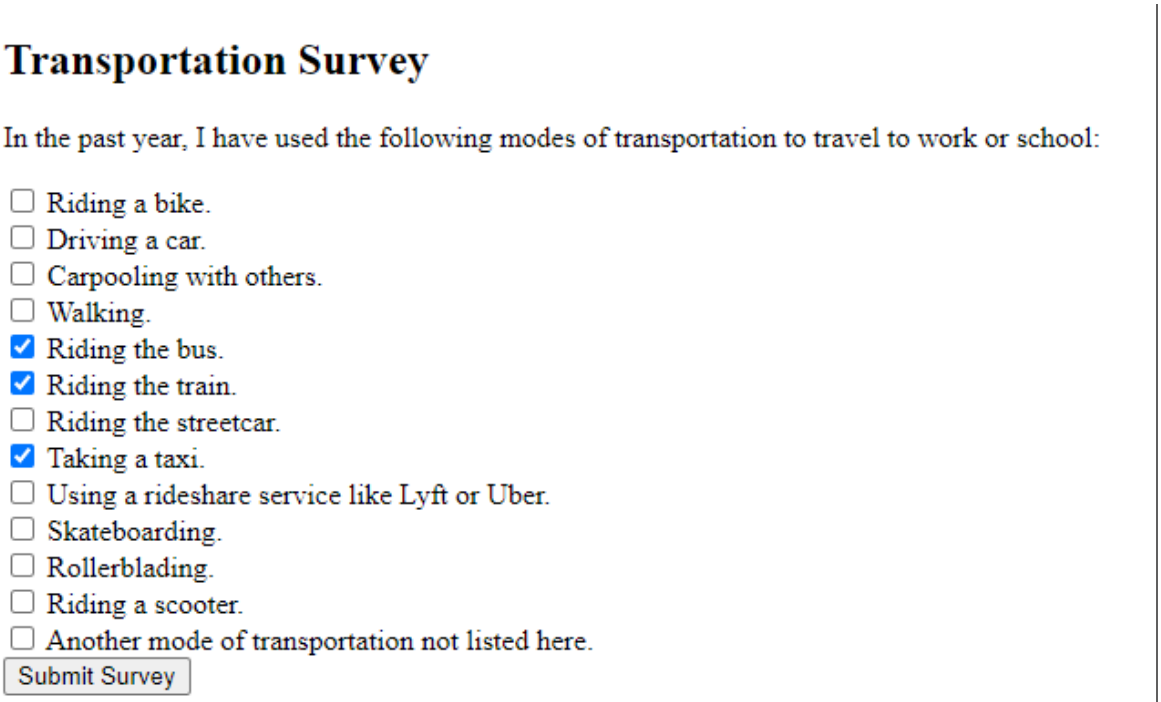
As the argument to this method, I pass in the following callback function (specifically an anonymous function expression that is being used as a callback):

```
function(element) {
  const paragraph = document.createElement("p");
  paragraph.append(element.value);
  document.body.append(paragraph);
}
```

The above callback function will be run for every element in the array. The `element` parameter is the placeholder for the actual array element. Each element in our array is an `HTMLInputElement` with a `value` property that gets the value of the input. So, for every element in our array, we:

- Create a new HTML paragraph element, with `const paragraph = document.createElement("p");`.
- Add the input's value (`element.value`) as the text to that paragraph element, with `paragraph.append(element.value);`.
- Add the new paragraph to the inside/end of our document's body tag with `document.body.append(paragraph);`.

If we now save our project, refresh the browser, and resubmit the form, we'll see our selections at the bottom of the page! Cool!



Why Use a Loop?

You may be wondering why we don't just access the `NodeList` object with bracket notation to print to the screen. Something like this:

```
function handleForm(event) {
  event.preventDefault();
  const userSelections = document.querySelectorAll("input[n
ame=transportation-option]:checked");

  const paragraph = document.createElement("p");
  // using bracket notation to access the first element in
userSelections
  paragraph.append(userSelections[0].value);
  document.body.append(paragraph);

  const paragraph2 = document.createElement("p");
  // using bracket notation to access the second element in
userSelections
  paragraph.append(userSelections[1].value);
  document.body.append(paragraph2);
}
```

Well, we don't know how many selections a user will make. Right now the `handleForm()` function is set up to handle 2 checkbox selections. What if the user makes just one selection? We'll get an error! Anytime we try to access the `userSelections` object with bracket notation to access a property that doesn't exist, we'll get an error.

This is where looping comes in handy! A loop like `Array.prototype.forEach()` will only run as long as the length of the array it is called on. This means there's no chance of error — the length loop will grow and shrink with the length of the array.

Adding a Heading to the Survey Results

Let's polish up our webpage a bit and add an H2 heading element to identify the survey results section. There's a few ways we could organize this code. We'll do it by creating the heading before printing the user selections to the page:

```
function handleForm(event) {
  event.preventDefault();
  const userSelections = document.querySelectorAll("input:c
hecked");
  const userSelectionsArray = Array.from(userSelections);

  // create results heading
  const resultsHeading = document.createElement("h2");
  resultsHeading.append("You use the following methods of t
ransportation to travel to work or school:");
  document.body.append(resultsHeading);

  userSelectionsArray.forEach(function(element) {
    const paragraph = document.createElement("p");
    paragraph.append(element.value);
    document.body.append(paragraph);
  });
}

window.addEventListener("load", function() {
  document.querySelector("form#transportation_survey").addE
ventListener("submit", handleForm);
});
```

Previous (/introduction-to-programming/arrays-and-looping/practice-looping)
Next (/introduction-to-programming/arrays-and-looping/practice-foreach-loops)

Lesson 20 of 50
Last updated March 24, 2023

disable dark mode

(http://www.epicodus.com)