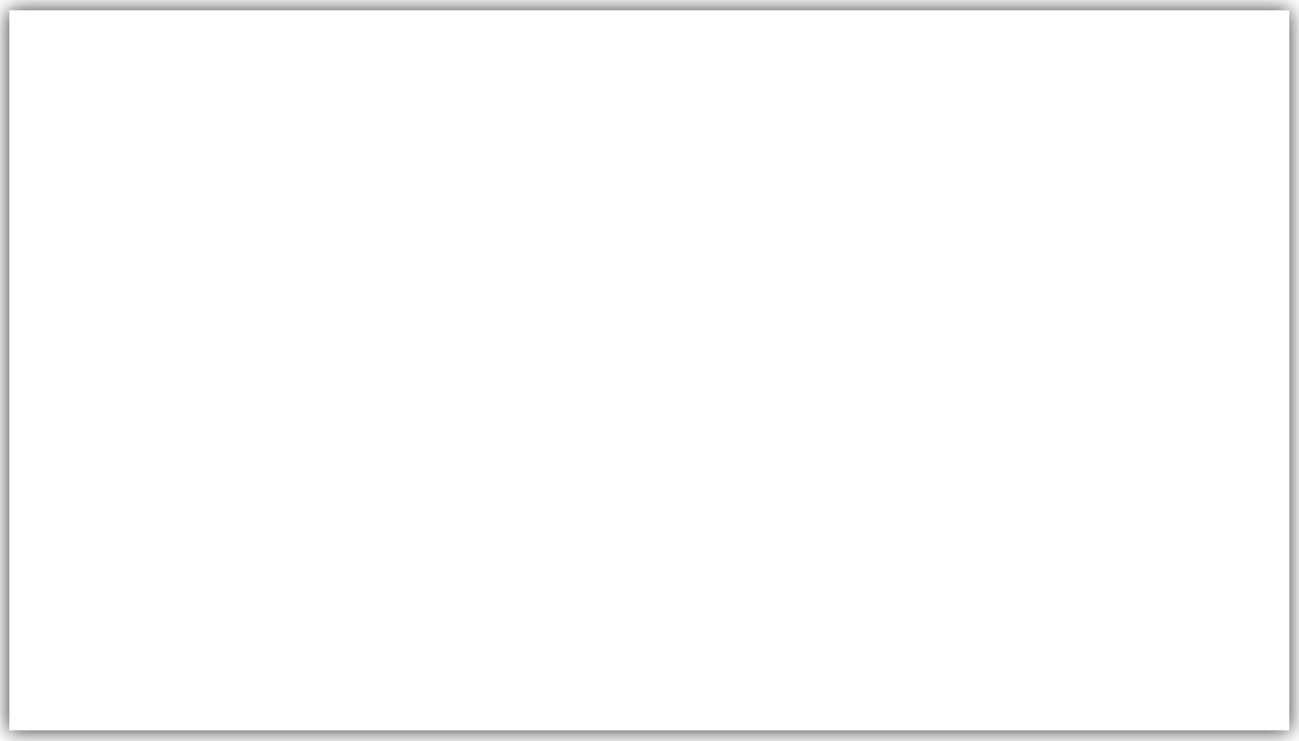
11.5.2

Add forEach to Your Table

Now Dana is ready to start adding data to her table. To do so, she'll need to create another function specifically for building the table. Data from the `data.js` file will be inserted into the table, row by row. This sounds like iterating through an array using a `for loopforEach`, doesn't it?

This time, we'll use a `forEach` function, which loops through the array in the same way as a `for` loop. The difference is that `forEach` works *only* with arrays. Another benefit is that `forEach` can be combined with an arrow function, once again making the code more concise and easy to read.

In the next step, we'll incorporate a `forEach` function that loops through our data array, and then adds rows of data to the table. The following video provides an overview of how `forEach` functions work.



Add the forEach Function

This function works in the same way as a `for` loop. In your code editor, type the following:

```
data.forEach((dataRow) => {

 });
```

Notice the fat arrow? We're using an arrow function here because it's a cleaner way to write a `forEach` loop. There's nothing wrong with using a traditional `for` loop—the code would behave in the exact same manner—but it is neater and easier to read.

With this new function, we have essentially chained a `for` loop to our data. We also added an argument (`dataRow`) that will represent each row of the data as we iterate through the array. Now we want to create a variable that will append a row to the table body. Within this forEach function, add the following code:

```
  let row = tbody.append("tr");
```

NOTE

Notice that we're using `let` instead of `var` to declare the `row` variable. That's because this variable is limited to just this block of code. It's more appropriate to use `var` when we want the variable to be available globally, or throughout all of the code.

This single line of code is doing a lot. It tells JavaScript to find the `<tbody>` tag within the HTML and add a table row (`"tr"`).

We'll get back to HTML when it's time to display our table, but for now keep in mind that the `<tr>` tags are used for each row in a table. Each object, or UFO sighting, in the array will be wrapped in a `<tr>` tag.

Loop Through Data Rows

Next, we'll add code to loop through each field in the `dataRow` argument. These fields will become table data and will be wrapped in `<td>` tags when they're appended to the HTML table. It gets a little confusing here, but we're going to set up another function within our original function for the `forEach` loop.

Below the line where we appended table rows, we'll set up another function:

```
  Object.values(dataRow).forEach((val) => {
});
```

We're already working with an array of objects, where each object is a UFO sighting. By starting our line of code with `Object.values`, we're telling JavaScript to reference one object from the array of UFO sightings. By adding `(dataRow)` as the argument, we are saying that we want the values to go into the dataRow. We've added `forEach((val)` to specify that we want one object per row.

Let's think of it this way: we're telling our code put each sighting onto its own row of data. The `val` argument represents each item in the object, such as the location, shape, or duration.

In the next two lines of code, we'll append each value of the object to a cell in the table. In our editor, the next few lines of code will go inside our new function. Let's first create a variable to append data to a table:

```
  let cell = row.append("td");
```

With this line, we've set up the action of appending data into a table data tag (`<td>`). Now, in the next line we'll add the values.

```
cell.text(val);
```

Let's take a look at our fully assembled function:

```
data.forEach((dataRow) => {
  let row = tbody.append("tr");
```

```
    Object.values(dataRow).forEach((val) => {
      let cell = row.append("td");
      cell.text(val);
      }
    );
  });
```

With this function, we have done the following:

- Looped through each object in the array
- Appended a row to the HTML table
- Added each value from the object into a cell

For example, this is the very first object in our array:

```
{
  datetime: "1/1/2010",
  city: "benton",
  state: "ar",
  country: "us",
  shape: "circle",
  durationMinutes: "5 mins.",
  comments: "4 bright green circles high in the sky going in circles then one bright green light at my fron
},
```

The code we just wrote will turn this object into a clean table like the one below.

Date	City	State	Country	Shape	Duration	Comments
1/1/2010	benton	ar	us	circle	5 mins	4 bright green circles high in the sky going in circles then one bright green light at my front door.

Also, because we've added `forEach`, every object in the array will be added to its own row in the table. We'll be able to test it fully soon, but first we need to add filters to the table. (We'll get to this step soon.)

The complete `buildTable` function we created should match the one below. It's also a good idea to add comments to help us keep track of what our code is doing.

```
function buildTable(data) {
  // First, clear out any existing data
  tbody.html("");

  // Next, loop through each object in the data
  // and append a row and cells for each value in the row
  data.forEach((dataRow) => {
    // Append a row to the table body
    let row = tbody.append("tr");

    // Loop through each field in the dataRow and add
    // each value as a table cell (td)
    Object.values(dataRow).forEach((val) => {
      let cell = row.append("td");
      cell.text(val);
      }
    );
  });
}
```

The first step of our code is complete: we've created a table!

ADD/COMMIT/PUSH

Be sure to save your work and add, commit, and push it to your repo!