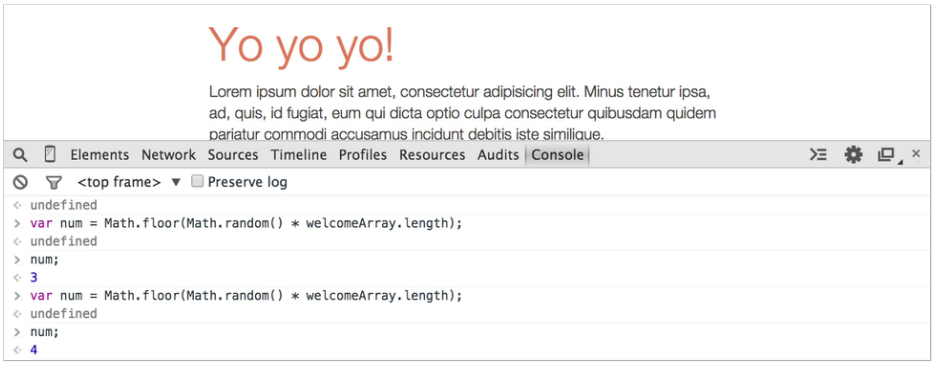
Exercise Preview



Exercise Overview

In this exercise, we will learn about arrays and how to create them. We'll also learn about Math objects and how to use them in conjunction with arrays to display the values we want.

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 **Random-Welcome** 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).

   This folder contains a static HTML file and a basic style sheet.
4. Open **index.html** from the **random-welcome** folder.
5. Preview **index.html** in Chrome (we'll be using its DevTools later).
6. Notice the **Hello!** heading. In this exercise, we'll use JavaScript to randomly show different headings when the page is loaded. In order to accomplish this, we'll be using arrays.

   An **array** is a lot like an object, in that it is a collection of key-value pairs. An array can be a collection of strings, numbers, booleans, functions, objects, or other arrays. You can think of arrays as a filing system that could go on forever. The difference between an array and an object is that for an object, we get to name both the key and the value. When it comes to an array, however, you don't name the keys. Keys are numbered automatically in ascending order starting from 0.

Creating an Array

1. Let's create an array. Open Chrome's Console by hitting **Cmd–Opt–J** (Mac) or **Ctrl–Shift–J** (Windows).
2. In the Console, type the following (but don't hit Return/Enter yet):

   ```
   var myArray = [];
   ```

   NOTE: **[]** denotes an array, while **{}** denotes an object.
3. So far our array is empty. Add some values to the array as shown in bold:

   ```
   var myArray = ['Bob', 'Smith', 23];
   ```
4. Hit **Return** (Mac) or **Enter** (Windows) to apply it.
5. Type **`myArray;`** and hit **Return** (Mac) or **Enter** (Windows). You'll see the array print out in the Console.
6. So how do we get values from this array? To get the first value, type:

   ```
   myArray[0];
   ```

   NOTE: Remember that arrays are **zero-indexed**, which means they start numbering with 0.
7. Hit **Return** (Mac) or **Enter** (Windows) to apply it and the string **`"Bob"`** will print.
8. To get the number value, type:

   ```
   myArray[2];
   ```
9. Hit **Return** (Mac) or **Enter** (Windows) and the number **`23`** will print.

Editing an Array

1. Oops, we forgot that Bob just had his birthday. Fortunately, it's pretty easy to change a value in an array. Type:

   ```
   myArray[2] = 24;
   ```
2. Hit **Return** (Mac) or **Enter** (Windows) and **`24`** will print.
3. Arrays have a variety of methods and we want to test out the most useful, commonly-used methods. Type the following so we can look at these methods:

   ```
   console.dir(myArray);
   ```
4. Hit **Return** (Mac) or **Enter** (Windows).
5. Expand the array's list by clicking the **arrow** to the left of **`Array[3]`**.

6. Click the **arrow** next to **`__proto__: Array[0]`** to see the methods we can use. Take note of `pop()`, `push()`, `shift()`, and `splice()` in particular. Let's try some of these.

7. What if we want to add a value to the array? To do this, type the following code:

   ```
   myArray.push(true);
   ```

   NOTE: The **push()** method is useful when you don't know exactly where you want to add a new value. You can put any value you want to add in the parentheses. Here, we're adding the value of **true** (a boolean) to the array.

8. Hit **Return** (Mac) or **Enter** (Windows). It prints **4** to show how many values are in the array.

9. Type **`myArray;`** and hit **Return** (Mac) or **Enter** (Windows) and it will show the values in the array:

   ```
   ["Bob", "Smith", 24, true]
   ```

10. By default, new values are added at the end of the array. To insert a value at a specific point in the array, we can use a method called **splice()**. Let's add a title for Bob after his last name. Type the following into the console but do not hit enter yet!

    ```
    myArray.splice( );
    ```

    NOTE: The **splice()** method can be used to add, remove, and replace values in an array. We're using it to add a value at a specific index in the array.

11. Between the parentheses, add the following code in bold and hit **Return** (Mac) or **Enter** (Windows):

    ```
    myArray.splice(2, 0, 'Jr.');
    ```

    Let's look further at the parameters for the **splice()** method.

    - **2**: This is the starting point where we want to change the array. We want to add the 'Jr.' title after "Smith" but before 24, which is the index 2 position in the array.
    - **0**: The second parameter is for the number of elements to remove. In this case, we don't want to remove any, so we set it at zero.
    - **'Jr.'**: the value to add to the array.

12. Type **`myArray;`** and hit **Return** (Mac) or **Enter** (Windows) to see what our array looks like now. It should show the following:

    ```
    ["Bob", "Smith", "Jr.", 24, true]
    ```

13. Actually, this array should be about Bob Senior, not Bob Junior. We can use the **splice()** method again. In the Console, type the following code in bold:

    ```
    myArray.splice(2, 1);
    ```

14. Hit **Return** (Mac) or **Enter** (Windows) to apply it.

15. Type **`myArray;`** and hit **Return** (Mac) or **Enter** (Windows) to see that "Jr." was deleted. We told JavaScript to start at index 2 and to delete one element. You should get back:

    ```
    ["Bob", "Smith", 24, true]
    ```

16. To delete the **last** value, use the **`pop()`** method as shown below:

    ```
    myArray.pop();
    ```

17. Hit **Return** (Mac) or **Enter** (Windows) to apply it.

18. Type **`myArray;`** to see that `true` was deleted. You should get back:

    ```
    ["Bob", "Smith", 24]
    ```

19. To delete the **first** value, type the following code using the **shift()** method:

    ```
    myArray.shift();
    ```

    NOTE: This will delete the first value and shift the other values up so that 1 becomes 0, 2 becomes 1, etc.

20. Hit **Return** (Mac) or **Enter** (Windows) to apply it.

21. Type **`myArray;`** to see that `"Bob"` was deleted. You should see:

    ```
    ["Smith", 24]
    ```

22. Actually we didn't really want to delete Bob. To add him back in the first (zero) position, just use the opposite of the shift() method by typing the following:

    ```
    myArray.unshift('Bob');
    ```

23. Hit **Return** (Mac) or **Enter** (Windows) to apply it.

24. Check that it worked by typing **`myArray;`** then hitting **Return** (Mac) or **Enter** (Windows). You should see:

    ```
    ["Bob", "Smith", 24]
    ```

25. Checking how many values are in an array is very useful when writing dynamic code. To test it out, type the following then hit **Return** (Mac) or **Enter** (Windows):

    ```
    myArray.length;
    ```

    That will return the number of values.

26. Leave **index.html** open in Chrome so we can come back to it later.

Creating an Array of Welcome Headings

Now that we've seen a bit of what arrays can do, let's get to work on replacing the static heading with an array of various welcome messages.

1. Switch back to **index.html** in your code editor.
2. Start adding a new array before the closing </body> tag (around line 23):

```
   </div>
   <script>
      var welcomeArray = [];
   </script>
</body>
```

3. Add some welcome messages to the array (feel free to make up your own):

```
<script>
   var welcomeArray = [
      'Pleased to meet you',
      'Hola',
      'Ni hao',
      'Bonjour',
      'Ciao'
   ];
</script>
```

4. Save the file.
5. Switch to **index.html** in Chrome and reload the page.
6. Open the Console if it's not already open.
7. Type **`welcomeArray;`** and hit **Return** (Mac) or **Enter** (Windows). You'll see the welcome messages.
8. Now we need to figure out how to switch out the welcome heading on the page. Go back to **index.html** in your code editor, and around line 11, you'll see that the welcome heading has an ID of **`welcome`**.
9. Let's see if we can change the heading using the Console. Switch back to Chrome.
10. In the Console, add a var reference to the **`welcome`** element as shown:

```
var welcomeMessage = document.getElementById('welcome');
```

11. Hit **Return** (Mac) or **Enter** (Windows) to apply it.
12. Type **`welcomeMessage;`** and hit **Return** (Mac) or **Enter** (Windows). It should print the HTML heading so we know it worked.
13. Specify a new message by typing:

```
welcomeMessage.textContent = welcomeArray[3];
```

14. Hit **Return** (Mac) or **Enter** (Windows) and watch the heading on the page change!

    Now we know how, in theory, we could switch the headings, but we still need to find a practical way to change them without having to manually specify it each time.

The Math Object

The **Math** object contains a lot of very helpful functions for doing various mathematical operations. Let's investigate to see if it can help us.

1. In the Console, type **`Math;`** and hit **Return** (Mac) or **Enter** (Windows).
2. Click the **arrow** next to **`MathConstructor`** (the `Math` object) to expand it. In it, you can see its properties and functions.

   The list starts off with constants (values that are not meant to be changed), such as `PI`. Constants are written in UPPERCASE. So for example, if you ever needed to do a mathematical operation that involves `PI`, type the following and it will give you the value of `PI`: **`Math.PI;`**
3. What we'll be focusing on right now is the **random()** method. Type:

   ```
   Math.random();
   ```
4. Hit **Return** (Mac) or **Enter** (Windows). It'll print a random number between 0 and 1.
5. Type **`Math.random();`** a few more times to see it generate more random numbers.

   So how can we apply this for our purposes? We can have it choose a random number within a certain range that we specify. Then we can apply that number to the array. So if we have five different headings, we'd tell it to pick a number between 0 and 4. We'd then save that number to a variable, drop the variable into the array, and then that's the message that would be featured.
6. To get a random number between 0 and 4, type the following code:

   ```
   Math.random() * 4;
   ```

   NOTE: This multiplies the random number by 4 so instead of getting a number between 0 and 1, it outputs one between 0 and 4.
7. Hit **Return** (Mac) or **Enter** (Windows) to see a random number between 0 and 4.

   The one problem is that right now, it's giving back super long numbers with a lot of decimals. We just want integers.
8. The **Math.floor()** method rounds **down** to the closest whole number. Type:

   ```
   Math.floor(3.7892);
   ```
9. Hit **Return** (Mac) or **Enter** (Windows) and it'll print out **`3`**.
10. The **Math.ceil()** function rounds **up** to the closest whole number. Type:

    ```
    Math.ceil(3.284);
    ```
11. Hit **Return** (Mac) or **Enter** (Windows) and it'll print out **`4`**.

Using the Math Object to Pick Random Headlines

Now that we have an idea of how the **Math** object works, let's figure out how we can customize it for our purposes.

1. Type the following code (but don't hit Return/Enter yet):

   ```
   var num = Math.random() * welcomeArray.length;
   ```

   This says to take a random number between 0 and 1 and multiply it by the **length** of the array. We could specify a certain number to multiply by (such as when we used 4 previously) but it's better to use a dynamic number. That way if the number of items in the array changes, you won't have to rewrite the JavaScript.

2. To put this inside **Math.floor()** to round it down, add the code shown in bold:

   ```
   var num = Math.floor(Math.random() * welcomeArray.length);
   ```

   Now it's going to take the random number, round it down to an integer, then save it to the **num** variable.

3. Hit **Return** (Mac) or **Enter** (Windows) to apply the command.

4. Test it out by typing **num;** then hitting **Return** (Mac) or **Enter** (Windows) to produce a random integer.

5. Perfect! Now we can add it to our JavaScript. Select and **copy** the line we just typed in the Console:

   ```
   var num = Math.floor(Math.random() * welcomeArray.length);
   ```

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

7. In the **script** tag near the bottom of the document and one line under the end of the array, make a new line and **paste** the code, as shown in bold:

   ```
   ];
   var num = Math.floor(Math.random() * welcomeArray.length);
</script>
   ```

8. Next we need to grab the welcome header and switch out the message. Add the following bold code:

   ```
   ];
   var num = Math.floor(Math.random() * welcomeArray.length);
   var header = document.getElementById('welcome');
   header.textContent = welcomeArray[num];
</script>
   ```

9. The last thing we need to do is make sure the page is loaded before we run the part of the JavaScript we just added. Otherwise, it won't be able to find the header because it hasn't been loaded yet. Add the following bold code:

```
    ];
    window.onload = function() {
        var num = Math.floor(Math.random() * welcomeArray.length);
        var header = document.getElementById('welcome');
        header.textContent = welcomeArray[num];
    };
</script>
```

   Now it's going to wait until the window has loaded before firing off the function.

10. Save the file.

11. Preview **index.html** in Chrome. (Reload if it's already open.)

    Cool! The heading should display one of the random messages.

12. Reload the page a few more times to see the heading change every time it's loaded.

    One thing to note… The reason we are leaving the static **Hello!** value in the HTML heading is for SEO purposes, where we wouldn't want an empty header. This is also a graceful fallback in case a visitor has JavaScript turned off.

13. Switch back to your code editor and close any files you may have open.

    NOTE: If you would like to compare your code to our version of the finished code, we have examples of all the completed projects in this workbook in a folder named **Done-Files**. Go to **Desktop > Class Files > yourname-JavaScript jQuery Class > Done-Files > Random-Welcome** to investigate this project, if you wish.
