

Coding & Cocktails Session 11: Programming Fundamentals



Overview

Tonight we discussed some common programming concepts found in the JavaScript programming language. Many of these concepts are shared across programming languages but in a different programming language the syntax (the format for how you write or utilize a specific concept) may differ slightly. You will utilize these concepts often as you write code so plenty of practice with them will help a lot!

If there is a term you don't know check out our glossary here: bit.ly/CnCgloss

If you don't find what you're looking for in our glossary let us know and we'll get it added! You can either submit a github issue on the repository or send us an email to codingandcocktails@kcwomenintech.org.

Prep Work

1. Utilize our Tools worksheet (bit.ly/CCKCTools) to install our recommended tools prior to the session. You will need a text editor like Sublime Text 2, a command line tool such as Git Bash or iTerm2, a GitHub account, and Git for version control.


Project

We are going to utilize the drink order app that some of you may have seen during our March version control session or our June jQuery session. We're going to make it interactive and utilize the concepts we discussed tonight.

Note: If you need a refresher on Git version control refer to the March worksheet here: bit.ly/CnCGit. A refresher on the Command line can be found here: bit.ly/CnCCmdLn

Part 1: Variables, Data Types, Functions, Comments, Console, Mathematical Operators, Conditionals, Equality, Addition and Concatenation

Note: The answer key is linked in step 11 if you'd like to reference it as you're working. Try your hardest to make an attempt and talk things over with a mentor before using it though!

1. First we need to get to our starting point utilizing git and our GitHub accounts.
 - a. **If you've never been to Coding & Cocktails before** follow these steps:
 - i. In google chrome using your GitHub account, fork the DrinkOrderApp repository (bit.ly/CnCPFStrt) to your own account by pressing the Fork  20 button () in the upper right.
 - ii. On the GitHub site in Google Chrome, grab the https URL from GitHub (**click on the clipboard icon on the right side to copy the URL**):

- iii. In your command line tool (Git Bash on windows and iTerm2 on macs), change directory into the CodingAndCocktails directory you created during the prep work if you're not there yet: `cd CodingAndCocktails`
- iv. Still in the command line tool run the following command to clone the repository to your local device:
`git clone` <paste URL copied from GitHub above (*Hint: use shift + enter on Windows*) do not include angle brackets>
- v. Finally, checkout the branch we will use as our starting point for this evening with the following command in the command line tool:
`git checkout programming-fundamentals-master`

b. **If you've been to Coding & Cocktails previously and worked in the DrinkOrderApp repository** follow these steps in your command line tool (Git Bash on windows or iTerm2 on macs) to get to the right starting point

- i. Make sure you've navigated into the DrinkOrderApp directory in your command line tool. (*Hint: `cd` is the command for "change directory"*)
- ii. Add your upstream remote (the repository that you forked) by running the following command:
`git remote add upstream`
`https://github.com/KansasCityWomeninTechnology/DrinkOrderApp.git`
- iii. Fetch any upstream changes or new branches that have been made since you last worked in the repository:
`git fetch upstream`
- iv. Checkout your new branch to start from:
`git checkout programming-fundamentals-master`

2. Open up the index.html file in Google Chrome to view our starting point.

3. First we need to add a function that will handle our orders. We will make these changes to the end of the `my-scripts.js` file in Sublime Text. Open your text editor and open the folder for the Drink order app. Find the `my-scripts.js` file and open it. In order to give our function a name create a variable set equal to the function like this:

```
var submitOrder = function () {}
```

4. Next, let's get some more practice using variables. We'll gather some input from the user and save it off.

Note: we'll be using some jQuery (a JavaScript library) to help simplify some of the coding but we'll supply

that portion of the code for you. Instead of copying and pasting from this document try typing out what you see.

- a. Create a variable called `orderName` inside the submitOrder function. *Remember to start with the var keyword! Your function code goes inside the function's curly braces - remember the parentheses are where we could pass parameters to the function but we won't need those for our project.*

- b. Set the variable's value to the following: `$("#order-form-input").val();`

This code is finding the HTML element that has the id order-form-input (which is our input box on

the page) and grabbing the value that has been entered there.

5. Now that you've saved that string into a variable, let's make sure we get a value and try our console.log statements!

- a. On the line underneath your variable add a `console.log();` statement, placing your variable name inside the parentheses.
 - b. In order to see what we've accomplished so far the function has to be called. Inside the click event handler for the order button add a call to the "submitOrder" function. To call a function write the name of the function followed by open and close parentheses. Inside the parentheses is where you would pass parameters to the function but since ours does not require parameters we just leave them empty. Example:
`myFunction();`
 - c. Save your work, reload your page in Google Chrome and open up the developer tools or console (push F12 on a windows machine or cmd+opt+i on a mac)
 - d. Enter your name in the input box and push the Order button.
 - e. Your name should appear in the Google Chrome console! Check your function name or grab a mentor if it is not showing up!

6. We also want to save off the drink name to display it with the order so add this code to your submitOrder function after your orderName variable (but still inside the function's curly braces!):

```
var drinkName = $("input[type='radio']:checked").val();
```

This code is looking for the value of the checked radio input. It is storing that text in a variable called drinkName.

7. Finally we want to display the orders that are coming in! Add this code snippet to your function after both variables. Try concatenating your variables and a string together replacing the <your code goes here> (including the angle brackets) with your string concatenation. Try to make the message say what would come out as "Jane would like a Strong Lady" Keep in mind your name is saved in the orderName variable and the drink name is saved in the drinkName variable. If you're struggling grab a mentor or take a peek at the answer key here: bit.ly/CnC1stKey (note: The answer key will look slightly different than what your code should look like at this point since it is the answer key for the entire part 1, not just until this point so grab a mentor if you're confused!)

```
$("#order-details").append("<h1>" + <your code goes here> + "</h1>");
```

This code is finding the HTML element with the id of order-details and appending to (or placing inside of) that element a new h1 element with the text that we specify using the append method.

Note: These orders are only based on what is being submitted on your device in your current browser session. Since we don't have a server connected in, the data is only persisted as long as you don't refresh your browser. Once you run a refresh, the variables and page content are reset starting over. In order to clear out your order details, just refresh your browser window.

8. Save your file, go to your chrome browser and reload the page. You should now be able to select a drink, enter your name, hit the order button and have your order text display in the light purple order detail area on your webpage!

You've added drink ordering to your website! Sounds like it's time to grab another!

9. We'll want to make sure we don't overwhelm our bartender with 50 drink orders at one time so let's make sure to count how many drinks have been ordered.
 - a. Back in *my-scripts.js* in Sublime Text add an `orderCount` variable to the top of your file and initialize it to 0. Initializing is setting the starting value of a variable.
 - b. Inside the `submitOrder` function make sure to add one to the `orderCount` variable so it gets incremented each time we create another order. *Remember there's a shortcut for adding one in programming!*
 - c. This time we're going to add a function that has a parameter passed to it. Add a new function to your file called `updateOrderCount`. Make sure you pass a parameter to this function called `count`. Remember to pass a parameter that you add it inside the parenthesis like this:
`var myFunction = function (myParameter) { }`
 - d. Inside the `updateOrderCount` function add this code:
`$('#drink-count').html("Drinks Ordered: " + count);`
This code is finding the HTML element with the id of `drink-count` and changing the content of that HTML element to be what we pass to the `html` method, here the string "Drinks Ordered:" and our updated order count.
 - e. You'll need to add a call to this function inside the `submitOrder` function (this should go just under the code to add the order display.) *Remember how we call functions like in step 5b above but this time you'll need to pass the `orderCount` variable to the function!*
`myFunction(orderCount);`
 - f. Take a minute to read about alert boxes. We'll utilize one in our next step. You can find information about alert boxes here: bit.ly/CnCAAlert. You can customize the message they display to say whatever you want.
 - g. Add a conditional to check if the `orderCount` is greater than 5. If so display an alert that says "Drink order queue is full. Please try ordering again in a few minutes." Otherwise (else) add the order to the display! Check this out if you need a reminder on what if statements look like: bit.ly/CnCIfElse

Note: These orders are only based on what is being submitted on your device in the current browser session. Since we don't have a server connected in, the data is only persisted as long as you don't refresh your browser. Once you run a refresh the variables and page content are reset starting over so in order to reset your order count, just refresh your browser window!

10. Save your file, reload your page in chrome, try adding 6 orders and see your alert!
11. Check your work with our answer key here: bit.ly/CnC1stKey

You've finished part one! Celebrate with a toast with your neighbor!

Part 2: Loops, Arrays, and Objects

1. Create your first data structure in *my-scripts.js*. We'll take the drinks from our HTML markup and move them into our javascript file to make them easier to change. Since we have multiple drinks we'll use an array (like a list) to hold them. Each drink will be an object and will need to key value pairs, a value that will easily link the label and the radio button and a label that will display nicely. Make sure to add at least 5 drink objects to your menu.

- a. Create a variable called `cocktails` and initialize it to an empty array. An empty array is just an opening and closing square bracket with no internal content. Example array with content of different data types:

```
[“element1” , “element2”, 3, true]
```

- b. Create an object for each drink. Each drink will need two keys - id and label. Each key will need a value paired with it. The “id” key will need a camelCase (*camelCase means the first word starts with a lowercase letter and each subsequent word starts with an uppercase letter without spaces between words!*) value that is the drink's name paired with it. We will use this to connect the label to the radio button input. The “label” key will need the display version of the drink name paired with it. Objects are surrounded by curly braces and contain key-value pairs like this:

```
{  
  “key”: “value”,  
  “anotherKey”: “Another Value”  
}
```

Don't forget the comma between key value pairs!

- c. Make sure to save your work even though we won't see any cool changes just yet!
2. Now we need a function that will loop through the objects and add them to the menu
 - a. Create another function called `loadMenu` in the *my-scripts.js* file in Sublime Text.
 - b. Inside the function add a for loop. The for loop has three pieces, the initializing expression, the condition and the incrementing expression. That means we tell the loop where to start with the initializing expression, how many times we will need to loop with the condition and how much to add to the loop counter after each time through the loop. Here is a for loop example we'll need to change ours to use our array's variable name so we're looping through our array of drink objects.

```
for (var i = 0; i < array.length; i++) { }
```

- c. Inside the for loop (that means inside the curly braces!) add this code:

```
$('.radio-group').append('<label class="radio" for="' + cocktails[i].id +  
'"><input type="radio" id="' + cocktails[i].id + ' " name="drink" value="' +  
cocktails[i].label + '"> ' + cocktails[i].label + '</label>');
```

This code is finding the HTML element that has the class radio-group applied to it. Then we

are appending (or adding inside of that element) our label and radio input elements for the drink items. The label has a “class” attribute of radio for styling purposes. There is also a “**for**” attribute that ties the label to the input element’s “**id**” attribute. This is where we are using the “**id**” key’s value from our drink objects.

The input element is of type “**radio**” so it will be a radio button selector and all of them have the “**name**” attribute of drink. Having the same “**name**” across the radio input elements ensures the user can only select one radio button at a time. The last attribute is the value of the radio button which holds the drink objects “**label**” key’s value and helps us display formatted text.

Finally, inside the label element we are utilizing the drink object’s “**label**” value to display the formatted drink name text.

- d. Now we need to make sure this function is called so the menu gets loaded into the page. Add the function call just inside the `$(document).ready(function() {`
 - i. To call a function write the name of the function followed by open and close parentheses. Inside the parentheses is where you would pass parameters to the function but since ours does not require parameters we just leave them empty. Example:
`myFunction();`
- e. Finally we need to remove the markup from the *index.html* page so it will be loaded with our JavaScript instead of with the markup. Remove all of the label and input elements that are inside the div element with classes input-group and radio-group
`<div class="input-group radio-group">`
- f. Save both index.html and my-scripts.js and reload your page in Google Chrome. It won’t look much different but you’ve utilized an array, objects, a for loop and a function to create different loading behavior!

3. Check your work against the part two answer key here: bit.ly/CnCPF2ndKey

Congratulations! You worked through some hard stuff tonight! Way to stick with it!

Bonus Section

If you’ve made it through all of the above or want to practice a little further at home, try ordering a drink without entering a name. That’s not useful, is it? This is called a bug which is something you’ll run into regularly as a developer. Let’s fix that with another conditional! These changes need to happen in *my-scripts.js* in Sublime Text.

1. Check to see if `orderId` has a value. We just want the conditional to evaluate to true and an undefined value will always evaluate to the boolean value false. That means all we have to do to check whether or not `orderId` has a value is check `if (orderId)`
2. Wrap that if statement around the if statement that is checking our drink count. This is called nesting our if statements. It is bad practice to nest too many if statements since it decreases code readability which makes maintenance more difficult. It can also increase the potential for a bug or defect in your code since keeping track of what if branch you are in is hard when you get too many nested levels.

3. Trigger an alert that happens if the `orderName` does not have a value. Make it say something like “Oops! Please enter your name to order your drink.”
4. Save your changes and refresh the page in Google Chrome. Try placing an order without a name now and see what happens!
5. Compare with the bonus answer key here: bit.ly/CnCPFBonusKey

Homework

The more you practice, the more comfortable you'll feel! Work through the following tutorial on your own time at home to help cement the concepts in your brain. Don't forget you can always talk to us on kcwit.slack.com to get help if you feel stuck or confused.

1. Codecademy learn JavaScript tutorial: bit.ly/CnCJSHmwk