Challenge 3 / Javascript Password Generator

https://stephcambria.github.io/Javascript-Random-Password-Generator/

In order to avoid as much confusion as possible, my first course of action was to outline and re-word all of the tasks I would need to complete in order to get this application to run:

```
//step 1 - click button to generate password
//step 2 - window prompt for password criteria
//step 3 - select criteria
//step 4 - window prompt for length of password
//step 5 - user chooses length of 8 - 128 characters
//step 6 - window prompt for character types to include
//step 7 - user confirms whether or not to include lowercase, uppercase, numeric, and
/ or special characters
//step 8 - when user answers each prompt, their input should be validated and at least
one character type should be selected
//step 9 - when all prompts are answered, a password is generated that matches the
selected criteria
//step 10 - when the password is generated, it is either displayed in an alert or
written to the page
```

I chose to do this in the comments of my javascript file because it is primarily where I would be working, so when I hit a wall, I could just scroll up in my file and get re-situated. I've learned from the previous challenges to always break up the tasks I'm given into little bite-sized pieces. However, despite this, I was still unsure of where to start, and decided that I would benefit from reviewing the core concepts of javascript as I worked out a plan. Up until now, I have been able to google my way out of every coding obstacle, but this time around I found youtube to be a bigger help. Personally, I am terrible with numbers, and just reading anything that looks like an equation turns my brain to soup, so turning javascript into something visual where I can see how and why things are working in real-time was a game changer for me!

A lot of the videos I used for review advised me to set the variables and functions first, and worry about the window alerts later, so this is where I decided to begin actually writing my code. Starting with generating random letters, I used the following syntax:

```
//add functions
//Random lowercase letters
var lower = getRandomLower;
function getRandomLower(){
  var lower = getRandomLower();

   return String.fromCharCode(Math.floor(Math.random() * 26) + 97);
```

```
}
//math.random generates a random number (tied to letters of the alphabet), and
math.floor returns the values to a specified range (in this case 26, because there are
26 letters in the alphabet)

//Random uppercase letters
var upper = getRandomUpper;
function getRandomUpper(){
   var upper = getRandomUpper();

   return String.fromCharCode(Math.floor(Math.random() * 26) + 65);
}
//same syntax, same idea
```

One of the videos I watched mentioned the browser character set, which greatly simplified this process. Using math.random and math.floor as I mentioned in my file comments, I used the equation * 26 + 97 for lowercase letters (there are 26 letters in the alphabet, and 97 is the start of the lowercase letters in the browser character set) and * 26 + 65 for uppercase letters (65 being the start of the uppercase letters in the browser character set). Because this process is somewhat overwhelming for me, I made sure to comment on everything I did as I successfully did it.

For numbers and symbols, I used more or less the same syntax:

```
//Random numbers
var number = getRandomNumber;
function getRandomNumber(){
   var number = getRandomNumber();

   return String.fromCharCode(Math.floor(Math.random() * 10) + 48);
}
//48 through 57 are numbers 0 through 9 in the browser character set, so we need a
string of 10 for our range

//Random symbol
var symbols = '!@#$%^&*(){}[]=<>/,.';
function getRandomSymbol(){
   var symbols = getRandomSymbol();
   return symbols[Math.floor(Math.random() * symbols.length)];
}
//declare the symbols variable string, then use similar math syntax to generate a
random one. Instead of using numbers, multiplying by symbols.length will allow access
to the entire list of symbols
```

After writing all of the functions out, I realized that my starter code was actually missing a function for generatePassword; it was being used in the starter code, but it had never actually been defined:

```
// Assignment Code
var generateBtn = document.querySelector("#generate");

//Here I'm adding the generate password function
function generatePassword() {
   console.log("you did it!");
   return "generated password";
}

// Write password to the #password input
function writePassword() {
 var password = generatePassword();
 var passwordText = document.querySelector("#password");

 passwordText.value = password;

}

// Add event listener to generate button
generateBtn.addEventListener("click", writePassword);
```

I added the console log just to make sure things were working so far (which they were!), and added the return to make sure my final generated password would appear in the box.

Now that I had all of my functions made and ready to be used, I began writing the window prompts. My code got quite messy as I continued to write and tweak things, so I ended up reorganizing it frequently. As I completed each individual task, I took the comment outlining that task away so I was just focusing on what I needed to do next. However, I plan to ideally use this as a reference for future javascript projects, so I made sure to keep all of my functionality comments where they were. When writing out the code for each prompt, I moved whichever function I wanted to call into that section, for example:

```
//Random numbers
var userWantsNumbers = window.confirm("Would you like to include numbers in your
password?")
var getRandomNumber;
function getRandomNumber(){
   window.prompt("Would you like to include numbers in your password?");
   if (getRandomNumber === true) {
       getRandomNumber()
```

```
        return String.fromCharCode(Math.floor(Math.random() * 10) + 48);
    }

}
//48 through 57 are numbers 0 through 9 in the browser character set, so we need a
string of 10 for our range
```

I just added whatever variables I needed to, and moved the functions I already made to be next to them. Under each variable and function, I added if statements to tell the application which random characters to add depending on what the user clicked. All of which were in the brackets for the generatePassword function I added in. However, after doing this, I began to get lost in my own code as I continued to work, so I decided to take a few steps back and simplify things for myself. Putting my functions to the side, I opted for a clean slate and decided to make arrays of all of the characters I would be using:

```
var numberList = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9"]
var symbolList = ["~", "!", "@", "#", "$", "%", "^", "&", "*", "?"]
var lowercaseList = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m",
"n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"]
var uppercaseList = ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M",
"N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z"]
```

While this looks redundant and like extra work, it actually redirected me and made sorting through the code a lot easier. It also made me understand why the initial way I started writing my code wasn't the best– I had begun writing multiple functions within another function. On top of that, I was not properly declaring or calling them anywhere else in my code, so nothing would have worked in the end anyway.

Here is how I rewrote every option I wanted the user to have:

```
//Random numbers
var userWantsNumbers = window.confirm("Would you like to include numbers in your
password?")
    if (userWantsNumbers === true) {
     optionsCart.push(numberList)
    }


//Random lowercase letters
```

```
var userWantsLowercase = window.confirm("Would you like to include lowercase letters
in your password?")
   if (userWantsLowercase === true) {
   optionsCart.push(lowercaseList)
   }


//Random uppercase letters
var userWantsUppercase = window.confirm("Would you like to include uppercase letters
in your password?")
   if (userWantsUppercase === true) {
       optionsCart.push(uppercaseList)
   }



//Random symbol
var userWantsSymbols = window.confirm("Would you like to include symbols in your
password?")
   if (userWantsSymbols === true) {
       optionsCart.push(symbolList)
   }



//if the user selects nothing
   if (optionsCart.length === 0) {
       window.alert("Invalid password")
       return
   }
```

Prior to this, I had added the options cart variable:

```
//here is where the user-selected characters will go
var optionsCart = []
```

I found this idea while I was digging around online to better understand how to randomize a selection in javascript. Later, when I went back and added the function to randomize the user

selected choices, this is where all of those choices would go. The way it was explained to me was that it's like a shopping cart at a supermarket.

```
//local variable for the final output
   var generatedPassword = ""
```

Now I had created the variable for the randomly generated password, but had no means to actually randomize it. This was extremely tricky for me to figure out, as, like I mentioned, numbers and math are not strong suits for me. Luckily, with enough research, I found an equation somebody online recommended that worked perfectly:

```
//here is where the user-selected characters will go
var optionsCart = []

//and here is the function to randomize them
function randomInt(min, max) {
   //if 'max' is not defined, assume we want range from 0 to min
   if (!max) {
       max = min
       min = 0
   }

   //for random value
   var rand = Math.random()
   return Math.floor(min + (1 - rand) + rand*max)
}

function getRandomItem(list) {
   return list[randomInt(list.length)]
}
```

It turned out there was a lot less actual math involved than I thought, which was relieving. All I really had to do was define my minimum base, and then use Math.random and Math.floor to randomize and drop the decimals in the result.

```
//local variable for the final output
   var generatedPassword = ""
```

```
//the calculation for each randomized password
   for (var i = 0; i < passwordLength; i++) {
       var randomList = getRandomItem(optionsCart)
       var randomChar = getRandomItem(randomList)
       generatedPassword += randomChar
   }

//returning the variable without calling the function will only show the generated
password in the console log, not in an alert or on the page
return generatedPassword
}
```

And from there, I ran a for loop to tell the function how to calculate each randomized password based on all of the user inputs. Returning the generated password without calling the function will only show the result in the console log, not in an alert or on the page. Thankfully, the starter code I was given already had that part done:

```
// Write password to the #password input
function writePassword() {
   //call the generate password function so it will actually work
   var password = generatePassword();
   var passwordText = document.querySelector("#password");
    //this will enable the generated password to appear in the text box on the page
   passwordText.value = password;
  }
  // Add event listener to generate button
 generateBtn.addEventListener("click", writePassword);
```

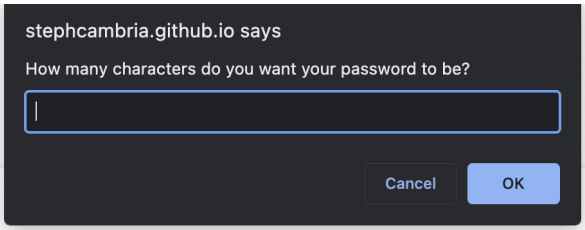
I'm sure I overlooked a few details but overall, I got the application to successfully run! The flow / structure / language of javascript makes a lot of sense to me now, which I'm grateful for and excited about.

Password Generator

Generate a Password

Your Secure Password

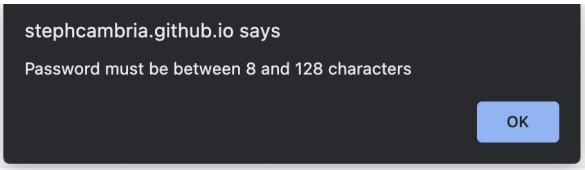
Generate Password



Generate a Password

Your Secure Password

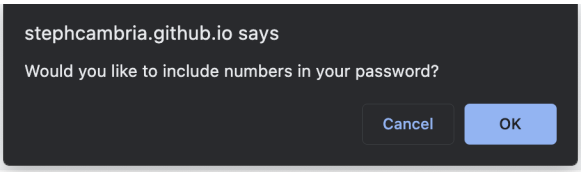
Generate Password



Generate a Password

Your Secure Password

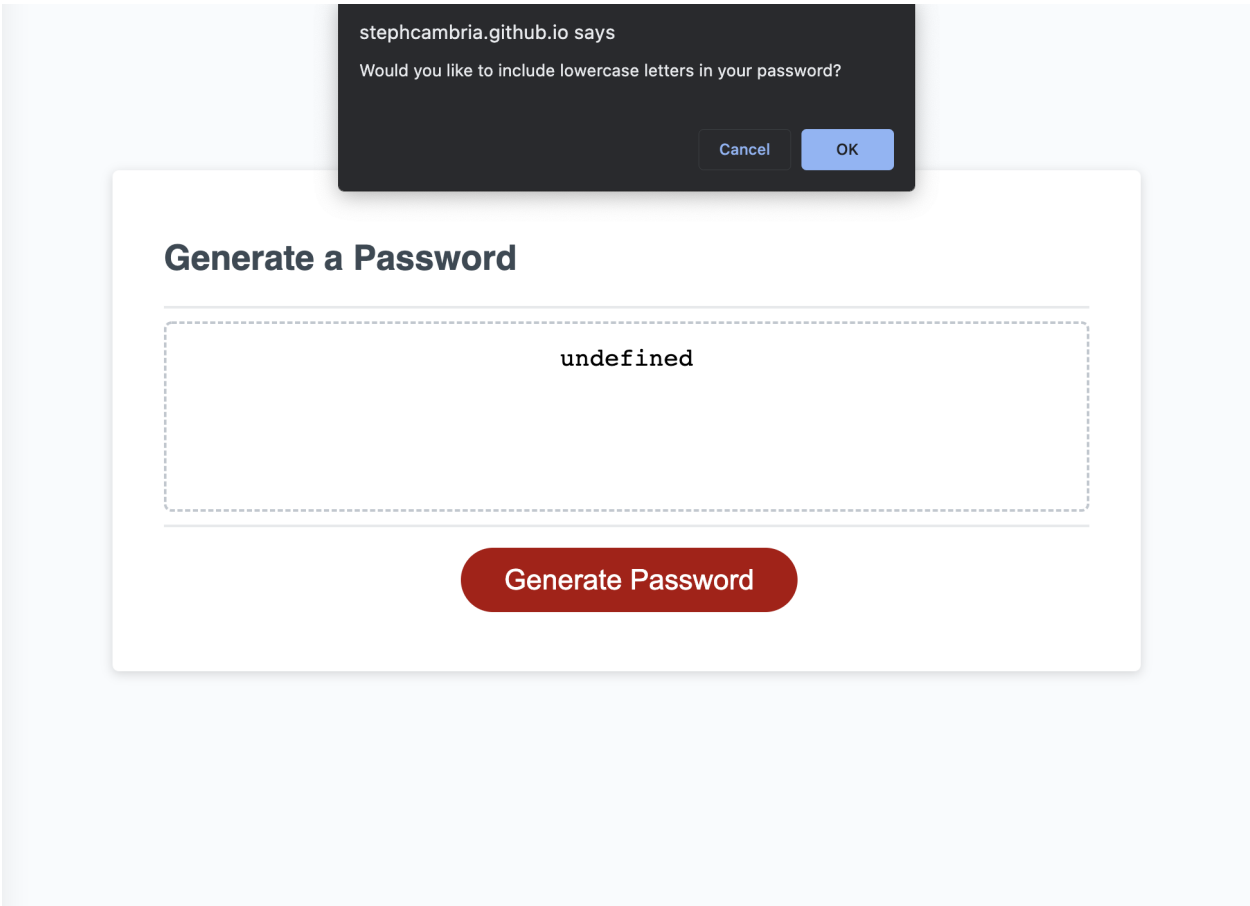
Generate Password

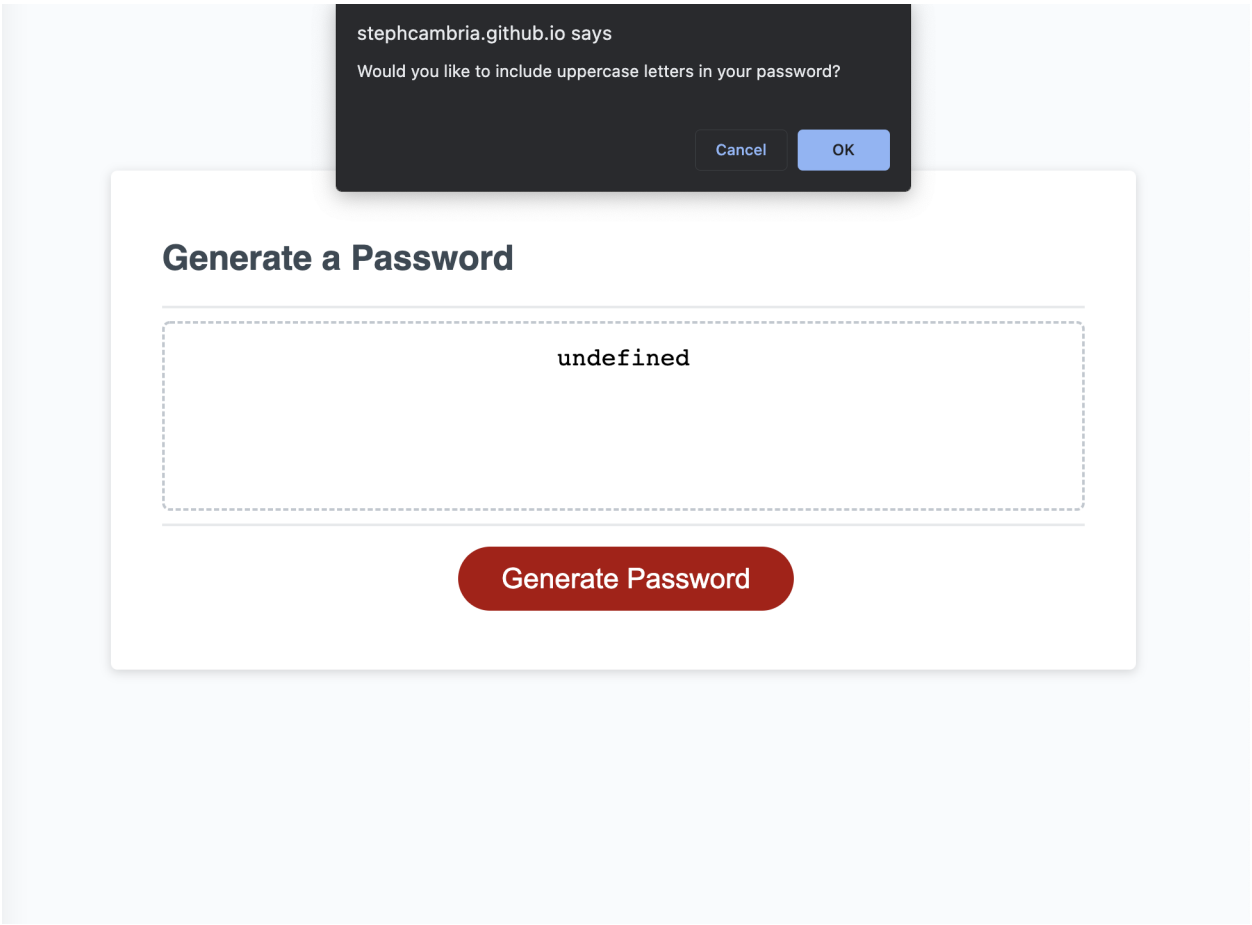


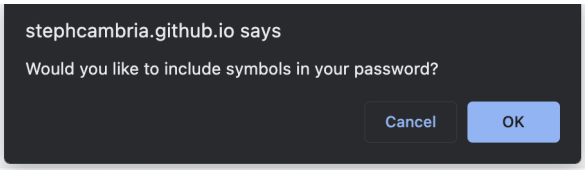
Generate a Password

undefined

Generate Password


stephcambria.github.io says
Would you like to include lowercase letters in your password?
Cancel
OK
Generate a Password
undefined
Generate Password


stephcambria.github.io says
Would you like to include uppercase letters in your password?
Cancel
OK
Generate a Password
undefined
Generate Password



Generate a Password

undefined

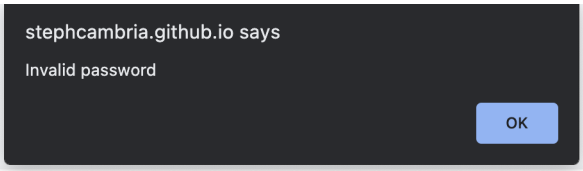
Generate Password

Password Generator

Generate a Password

BNQhFcvLY!ZYceUPRUB@FH

Generate Password



Generate a Password

Your Secure Password

Generate Password