createElement() appendChild()

Math.random()

for loops

functions called by button click events

IIFE (Immediately Invoked Function Expression)

evaluating checkboxes with conditional logic

string concatenation

dynamic text selection

dynamic copying of text to the clipboard

In this lesson, we will make a **Password Generator** application that makes random passwords of specified lengths using character sets chosen by the suer. The interace works like so:

- the user chooses a password length from the menu
- the user unchecks any boxes of character types to exclude from the password
- the user clicks the **generate password** button to call the **generatePassword** function
- the function sees which checkboxes are checked and uses them to make a big string of eligible characters.
- then the function generates random integers in the range of the string length, using the number to get a character by index
- the function charactconcates a password from the random selections.
- the password is outputted to a text box
- the user can click the **copy** button to copy the password to the clipboard

1. Open **06.03-Password-Generator.html** from the **06.03-Password-Generator** folder, and preview the file in the browser.

2. Take it for a spin: choose a password length from the menu and click the **generate password** button. A randomly generated password appears in the text box.

3. Click **copy** to copy the password to the clipboard.

4. To confirm that the password has been copied, empty the input box and paste in the password.

5. Have a look at the html code. There are a few things to note:

- The select menu is empty. This is because we will use a for loop to generate the numbered options from 8 to 25, dynamically.
- The form elements (select, input, label, button) are not wrapped in a form tag. Since we are not sending the data for processing, so we don't need a form.

tooltip

There is a thing called a "tooltip" near the end of the code. This is a cool feature where you get a popup (tooltip) when you mouse over something, in this case the copy button. A tooltip typically looks like a cartoon dialogue balloon, complete with little pointer arrow. The CSS to render a tooltip is fairly tricky, and you would do well to spend some time studying that, although CSS is not the focus of this lesson (or of this course, for that matter).

Now that we have taken the app for a spin and checked out the HTML, let's turn our attention to recreating **FINAL.js** from scratch.

6. Switch from using **FINAL.js** to **START.js** amd open **06.03-Password-Generator-START.js.**.

We start by declaring strings of the characters that go with each checkbox. If a checkbox is checked, its associated characters are eligible for inclusion in the random password. As we will not be editing these strings, they are declared uppercase, with **const**

7. Declare consts for the letters of the alphabet, the digits 0-9 and some special character symbols:

```
const LOWERS = "abcdefghijklmnopqrstuvwxyz";
const UPPERS = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
const NUMBRS = "0123456789";
const SYMBLS = "@#$%^&*_+?!";
```

8. Get the select menu:

```
const selectPswdLen = document.getElementById("select-pswd-len");
```

9. Get the four checkboxes:

```
const lowcaseCB = document.getElementById("lowcase-cb");
const uprcaseCB = document.getElementById("upcase-cb");
const numbersCB = document.getElementById("num-cb");
const symbolsCB = document.getElementById("symbol-cb");
```

10. Get the two buttons:

- one button calls the function that generates the password
- the other button calls the function that copies the password

```
```

```
const genPswdBtn = document.getElementById("gen-pswd-btn");
const copyPswdBtn = document.getElementById("copy-btn");
```

11. Have the buttons call their respective functions on click:

```
genPswdBtn.addEventListener('click', generatePassword);
copyPswdBtn..addEventListener('click', copyPassword);
```

12. Get the text input element for displaying the new password:

```
const newPswdOutput = document.getElementById("new-pswd-output");
```

13. Run a loop that makes an **option** tag with each iteration of the loop. The loop counter, **i**, runs from 8 to 25:

```
for (let i = 8; i <= 25; i++) {
}
```

14. Inside the loop, make the option tags:

- Use **createElement()** to make the option.
- Assign **value** and **text** properties equal to the counter.
- Use **appendChild()** to add the option to the menu:

```
for (let i = 8; i <= 25; i++) {
    const option = document.createElement('option');
    option.text = i;
    option.value = i;
    selectPswdLen.appendChild(option);
}
```

15. Set the default choice to 15 using conditional logic: if the value of the counter is 15, add a selected property to the option object:

```
  for (let i = 8; i <= 25; i++) {
    const option = document.createElement('option');
    option.text = i;
    option.value = i;
    if(i == 15) option.selected = 'selected';
    selectPswdLen.appendChild(option);
  }
```

16. Run the page. The menu should be populated with choices from 8-25.

function generatePassword()

Now for the **generate password** function.

17. Define the function and declare two string variables:

- **charSet** stores the eligible characters, per the checkboxes
- **randomPswd** stores the generated password.

```
function generatePassword() {
  let charSet = ""; // the eligible characters
  let randPswd = ""; // the random password
}
```

evaluating the checkboxes

If a checkbox is checked, its associated string is concatenated onto the **charSet**. Since the if-statements are just one line of logic each, the curly braces may be omitted.

18. Concatenate the **charSet** variable:

```
function generatePassword() {
  let charSet = ""; // the eligible characters
  let randPswd = ""; // the randomly generated password
  if(lowcaseCB.checked) charSet += LOWERS;
  if(uprcaseCB.checked) charSet += UPPERS;
  if(numbersCB.checked) charSet += NUMBRS;
  if(symbolsCB.checked) charSet += SYMBLS;
}
```

generate the random password

Now that all eligible characters are in the **charSet** string, we can proceed to make the actual password:

- run a **for loop** as many times as there are characters in the password
- each time through the loop, generate a random integer in the range of the **charSet**
- use the **charAt()** method to select the character that occurs at the random index
- concatenate the randomly chosen **charSet** character onto **randPswd**
- repeat the loop until the password reaches the target length
- output **randPswd** to the "Your new password" field

19. Still in the function, run a for loop that iterates once for each character of the password:

```
  for (let i = 0; i < selectPswdLen.value; i++) {
  }
```

charAt(index)

the **charAt(index)** method is called on a string and takes an index as its argument. It returns that character at that index:

20. Generate a random integer in the range of the charSet string and then get the character at the random index using **charSet()** method:

```
    let r = Math.floor(Math.random() * charSet.length);
```

21. Look up that character at that index using **charAt(index)** and concatenate that character onto the **randPswe** string. Finally, output the password:

```
function generatePassword() {
    let charSet = ""; // the eligible characters
    let randPswd = ""; // the randomly generated password
    if(lowcaseCB.checked) charSet += LOWERS;
    if(uprcaseCB.checked) charSet += UPPERS;
    if(numbersCB.checked) charSet += NUMBRS;
    if(symbolsCB.checked) charSet += SYMBLS;
    for (let i = 0; i < selectPswdLen.value; i++) {
      r = Math.floor(Math.random() * charSet.length);
      randPswd += charSet.charAt(r);
    } // end loop

    newPswdOutput.value = randPswd;

  } // end function generatePassword()
```

Once the password is generated, we want to make it easy for the user to copy it.

copying text to the clipboard

Clicking the **copy** button calls the **copyText()** function, which does the following:

- selects the generated password from the text box
- uses **navigator.clipboard.writeText()** to copy the password
- shows a tooltip with a "Password copied" message

22. Define the **copyPassword** function:

```
function copyPassword() {
}
```

23. Use the **select()** and **navigator.clipboard.writeText()** methods to copy the password from the output field:

- **select()** method selects (highlights) the text in the target element
- **navigator.clipboard.writeText(text)** copies its argument text to the clipboard

```
  function copyPassword() {
    newPswdOutput.select();
    navigator.clipboard.writeText(newPswdOutput.value);
  }
```

24. Provide feedback that the password has been copied:

```
  function copyPassword() {
    newPswdOutput.select();
    navigator.clipboard.writeText(newPswdOutput.value);
    document.querySelector(".tooltiptext").innerHTML = "Password copied: "
+ newPswdOutput.value;
  }
```

25. Save all and run the page. The application should work.

26. Choose a password length from the menu and click the GENERATE PASSWORD button. A random password should appear in the output box.

27. Click the **copy** button, the password in the box is highlighted (selected) and also copied to the clipboaard.

Some nifty css makes the cartoon speech balloon-like tooltip. You can check out the css on your own, although it is outside the scope of this lesson.

- **END: Lesson 06.03**
- **NEXT: Lesson 06.04**