JS 13 Valid Input Check

Project 10 - 12 focused on array manipulation. Project 13 also focuses on array manipulation. Why is array manipulation so important? Because of how easy it is to apply in a real world situation.

We cannot simply just enter and ask for resources, like access to email, calendar events, music playlist, and etc. There are restrictions and these restrictions are used as security measures. One basic, but important security measure, is **authentication**.

One of the most common ways to authenticate is to pass a test. In our case, the test is to enter valid input. Pass or reject of data is based on rules, such as the data must have at least 8 characters or more. Rule-based computation simplifies our code into logical checks (ie. we can now use "if-else" statements and logical AND or logical OR to decide pass or reject of the data that was entered).

Once we do enter valid input, we are given access to resources, such as email. However, if we enter **in**valid input, then we stay at the same spot.

In this session, we focus on valid input check by creating a login page with front side validation. For simplicity, we will assume that the characters are combined together and there are no spaces in between.

1. form validation 2. array manipulation 3. rule based computation

===

Continue to the next page for instructions

Introduction

User Name Rules

1. loop through and check that the username has

    1.1 at least 8 characters long
    1.2. at least 1 special character of %, ^, or # (keyword is **OR**)
    1.3. user name can only have 11 or less characters. Count of 11 is ok, count of 12 and beyond is invalid number of characters.
    1.4. BUT can't have the character ~

Password Rules

2. loop through and check that the password has

    2.1. at least 5 characters
    2.2. has at least 2 special characters, either # or &

Login Rules

1. if username is valid, then change background color to blue

2. if password is valid, then change background color to green

3. if **BOTH** username and password are valid, then change background color to yellow

4. if point any of the points from 1 - 3 are false, then do the following

    4.1. link JS code to the HTML element whose id is "**msg**"
    4.2. use **.innerHTML** to display the data "**Username or password format needs refinement**"
    4.3. link JS code to the HTML element whose id is "**bTag**". Next, change the style of the **backgroundColor** to **red**

0. Logical AND (&&) compared to Logical OR (||)

Notice that we have use the logical operators AND (**&&**) and OR (||). The result of these logical operators can be one of two values, TRUE or FALSE.

The **logical AND (&&)** is TRUE when **ALL conditions are true**. For example,

```
var keyCode_1 = ‘A’;
var keyCode_2 = ‘C’;

if( keyCode_1 == ‘A’ && keyCode_2 == ‘B’ )
{
        securitySystem.openFrontDoor();
}
```

Explanation
In the above code, we declare a variable named `keyCode_1` and put the value of ‘A’ into it. Next, we declare another variable named `keyCode_2` and put the value of ‘C’ into it. We are doing a simple test of the system’s authentication program.

The security system will open the front door **ONLY WHEN** the `keyCode_1` has a value of ‘A’ **AND** `keyCode_2` has a value of ‘B’. The security system **DOES NOT** open the front door. Why?

Because the “if” statement is false.

1. Does `keyCode_1` hold the value ‘A’? **TRUE**, because of `var keyCode_1 = ‘A’;`
2. Does `keyCode_2` hold the value of ‘B’? **FALSE**, because of `var keyCode_2 = ‘C’;`

The logical AND (**&&**) is **ONLY TRUE** when **ALL** conditions are true. The “if” statement needs both conditions to be **TRUE** **BUT** only one is true. This is why the logical AND (&&) is false and the security system **DOES NOT** open the front door.

The **logical OR (||)** is TRUE when **AT LEAST one condition is true**. The logical OR only needs one condition to be true for the result to be TRUE. For example,

```
/* ====================================
Rules:
   1. if the sensor has a value of 0, then everything is ok.

   2. however, if the sensor has a value of 1, then someone activated the sensor

*/
var frontDS = 1;        // -- frontDS is front door sensor
var backDS = 0;         // -- backDS is back door sensor

if( frontDS == 1 || backD == 1 )
{
      securitySystem.soundAlarm();
}
```

Explanation
In the above code, we declare one variable named `frontDS` and put the value of '1' `into it.`

We also declare another variable named `backDS` and put the value of '0' into it. We are testing if the security system's alarm will work.

The security system will sound the alarm when **AT LEAST** one of the sensors is activated. In the "**if**" statement, the left side of the logical OR (||) is true and we only need one side to be true. As a result, the alarm's audio alert is activated.

1. This is the password check. Write JS code in between <script> </script>. Write ALL OF IT!!!

```
function pw_check()
{
     var pwInput = document.getElementById( "pw" );

    // -- pwSB --> password search bar
    var pwSB = pwInput.value;

    var i          = 0;
    var char5      = 0;
    var hashFound = 0;
    var ampFound  = 0;

    if( pwSB.length >= 5 )
    {
         char5 = 1;
    }

    for( ; i < pwSB.length; i++ )
    {
         if( pwSB[i] == "#" )
        {
         hashFound = 1;
        }

        if( pwSB[i] == "&" )
        {
         ampFound = 1;
        }
    }

    return ( char5 && hashFound && ampFound );

}
```

Explanation

This will look familiar. We are using the `document.getElementById("pw")` to link JS code with the HTML element whose id is `"pw"`.

Remember that the search bar is an `<input /> tag` and the `<input /> tag` allows the user to type in data to be sent to the JS code. How can JS code get the information that was typed into the search bar by the user? All we have to do is put `.value` at the end.

```
// -- pwSB --> password search bar
var pwSB = pwInput.value;
```

Where does `.value` come from? Look at the HTML code in between the <body> </body> and you will see the below

```
<input
id        = "pw"
value     = ""
minlength = "20"
maxlength = "50"
name      = "pw"
/>
```

You already wrote this code, continue to the explanation below

```
if( pwSB.length >= 5 )
{
    char5 = 1;
}

for( ; i < pwSB.length; i++ )
{
    if( pwSB[i] == "#" )
     {
          hashFound = 1;
     }

     if( pwSB[i] == "&" )
     {
          ampFound = 1;
     }
}

return ( char5 && hashFound && ampFound );
```

Explanation

The password must be at least 5 characters in length. The first "**if**" statement checks if the password has 5 or more characters. Once we have a password that is 5 or more characters in length, we set the rule of **char5 to 1** to mark it as complete.

Next, we have to loop through and check the two remaining rules, which are that the password must have a hash tag (#) **AND** the ampersand character (&). **These two special characters can be in ANY POSITION within the array and this is why we must loop from start to finish and run the same checks each time.**

The "**for**" loop uses the variable **i** as the index (ie. the index is the marker of our current position in the line) and then we check if we are at the end of the line with `i < pwSB.length`.

If we **are not** at the end of the line, we continue **INSIDE** the body of the "**for**" loop and run the code in **orange**. Inside the body of the "**for**" loop, we use the index **i** and square brackets to access a **SINGLE** data within the array.

We then use two "**if**" statements to check if our password has a hash tag or an ampersand.

> ***** The hash tag and ampersand can be in ANY POSITION within the array and this is why we must loop through and check *****

Finally, we check if all rules are satisfied by using the logical **AND** operator and then giving back the result by using the "**return**" keyword.

2. Write JS code in between <script> </script>. Write ALL OF IT!!! Color code is for the explanation

```
function login_check()
{

        var pwCheck = pw_check();
        alert( "pwCheck: " + pwCheck );

        if(  pwCheck == 1 )
        {

            if( pwCheck == 1 )
            {
                document.getElementById( "bTag" ).style.backgroundColor = "#006500";

                document.getElementById( "msg" ).innerHTML = "";
            }

        }

        else
        {
              document.getElementById( "msg" ).innerHTML = "Username or password invalid";

                     document.getElementById( "bTag" ).style.backgroundColor = "#B22222";
        }

}
```

Explanation

Notice that the code in **orange** is the name of the function definition that you **ALREADY** wrote in chapter 1. The main difference here is that we don't have the word "**function**" and only have the function name. So, it becomes a function call.

We are calling the function name to check if the user entered valid data. Next, we see two "**if**" statements that are exactly the same, it is the code in **green**. This is partial code that will be completed later on.

Remember that in chapter 1, we see the word "**return**". Scroll back up to chapter 1 to confirm that the word "**return**" is there. The word "**return**" means that the function is giving data back to us. We then compare the data that the function gave to us with the value of 1.

> If the function call gave the data of 1 back to us, then the password satisfied all of the rules and we change the **backgroundColor** to **green**.
>
> Else, the user did not enter data that satisfied all of the password rules. We alert the user by changing the **backgroundColor** to **red** and using **.innerHTML** to update the website to say `"Username or password invalid";`

3. Write HTML code in between <body> </body>. ONLY write the code in SKY BLUE and Purple.
Also, position the code in between the **BLACK code**

```
<label for="pw">Password</label>
<input
id        = "pw"
value     = ""
minlength = "20"
maxlength = "50"
name      = "pw"
/>

<br/><br/>
<button
id      = "nEventB"
onclick = "login_check()"
>
Enter Data
</button>


<p id = "msg">

</p>
```

Explanation

This will look familiar. We are using the `<button> </button>` to create a button. Next, we attach the event "**onclick**" to the button. When we click on the word "**Enter Data**", the JS function named "**login_check()**" is called.

This translate into ==> click on "**Enter Data**" and the action "**login_check()**" is called. Note that the "**login_check()**" is a JS function.

This means that when we click on "**Enter Data**", HTML will send a signal to JS code. The JS code will then execute the action "**login_check()**".

Click on green "Run" button. Go to the right side of the word "Password" and test it by entering the data "vuong#&". Next, press the "Enter Data" button.

The background color should turn green to indicate valid password.

Continue to the next page.

4. **Write JS code in between <script> </script>. Write ALL OF IT!!!** The color code splits it up into readable sections. This is **Username Check**. **Position** the code right under the open **<script>**

The test input for the user name check is: **v5d5n^00**

```
function u_name_check()
{
	var uInput = document.getElementById( "uName" );
	var
	percentFound    = 0,
	carrotFound     = 0,
	hashFound       = 0,
	complementFound = 0;

	uNameArr = uInput.value;

	alert( "user entered: " + uNameArr[1] );

	var char8 = ( uNameArr.length >= 8 ) ? true : false;
	var lim11 = ( uNameArr.length <= 11 ) ? true : false;

	alert( "char8: " + char8 );

	var dat = 0;

	for( var i = 0; i < uNameArr.length; i++ )
	{
		dat = uNameArr[i];

		if( dat == "%" )
		{
			percentFound = 1;
		}

		if( dat == "^" )
		{
			carrotFound = 1;
			alert( "carrot found" );
		}

		if( dat == "#" )
		{
			hashFound = 1;
		}

		if( dat == "~" )
		{
			complementFound = 1;
		}
	}

return ( (percentFound || carrotFound || hashFound ) && !complementFound && char8 && lim11);

}
```

Explanation

This code is very similar to the code that we wrote for password check. The username has more rules and this is why we have more “`if`” statements.

> ****** The percent, carrot, hash tag, and complement special characters can be in ANY POSITION within the array and this is why we must loop through and check ******

At the end, we use the word “**return**” to give data back.

a. The return data is a **0** if the username **DID NOT** satisfy all of the username rules.
b. However, the return data is a **1** if the username **DID** satisfy all of the username rules.

Continue to the next page

5. Go back and MODIFY the function definition "`login_check()`" to have it match the code below. WRITE All of it. Color code is for explanation.

```
function login_check()
{
   var uNameCheck = u_name_check();
   alert( "uNameCheck: " + uNameCheck );

    var pwCheck = pw_check();
    alert( "pwCheck: " + pwCheck );

    if( uNameCheck || pwCheck )
    {
         if( uNameCheck )
         {
              document.getElementById( "bTag" ).style.backgroundColor = "#000055";

              document.getElementById( "msg" ).innerHTML = "";
         }

         if( pwCheck )
         {
              document.getElementById( "bTag" ).style.backgroundColor = "#006500";

              document.getElementById( "msg" ).innerHTML = "";
         }

         if( uNameCheck && pwCheck )
         {
              document.getElementById( "bTag" ).style.backgroundColor = "#DAA520";

              document.getElementById( "msg" ).innerHTML = "Open sesame!!!";
         }


    }

     else
     {
           document.getElementById( "msg" ).innerHTML = "Username or password invalid";

                   document.getElementById( "bTag" ).style.backgroundColor = "#B22222";

     }

}
```

Explanation

This updated function definition of "`login_check()`" includes the username check.

The code in **blue** checks if the username data entered by the user satisfies the username rules. If so, then we change the background color to **blue**.

The code in **green** checks if the password data entered by the user satisfies the password rules. If so, then we change the background color to **green**.

The code in **gold** checks if the password data **AND** the username data entered by the user satisfies all rules (username and password). If so, then we change the background color to **gold** and then use **.innerHTML** to update the website to say `“Open sesame!!!”;`

If ANY of the rules were violated, then the code in **red** executes. It will turn the background color to **red** and then use **.innerHTML** to update the website to say `"Username or password invalid”;`