# JS 8

In Session 6 & 7, we used an HTML input to let the user type in event data and then redirect to the correct month. Buttons were used to indicate the month to associate with the event data.

What about priority? What if we have multiple events in January and some events are "Urgent" whiel others are "High" priority. Since there are of different priority level, we need to use colors to let the user know the difference.

In this session, we focus on function definition that returns a value. This return value is then used to set the color of "Urgent" and "High" priority.

1. Color priority calculation  2. function definition, call and return value

==========

## 1. Write The HTML Code In Between <body> </body>.

```
<input type="radio" id="uPri" name="pLevel" value="urgent">
<label for="urgentCB"> Urgent </label>


<input type="radio" id="hPri" name="pLevel" value="high">
<label for="highCB"> High </label>


<input id = "eT" type = "text" name = "eT" value = "..." minlength = "10" maxlength = "20" />
```

### Explanation

This may seem similar to JS 6 but it is actually different, very different. In JS 6, we used an <input/> tag. However, the type was "**input**", which created a search bar.

In this HTML code, we **ALSO** use an <input/> tag but **the type is different**. This time, the type is "**radio**", which creates a radio button for single selection. With a radio button, we can **ONLY SELECT ONE OPTION**. This option is "**checked**" and we will use "**checked**" later on in JS code.

How does JS link up with HTML? Notice that both <input/> have an id and this id allows JS to link up with HTML.

## 2. Write The JS Code In Between <script> </script>

```
function get_cc( uRB, hRB )
{
	var cc = "#606060";

	if( uRB.checked == true )
	{
		cc = "#ff0000";
	}
	else if( hRB.checked == true )
	{
		cc = "#483D8B";
	}
	else
	{
		cc = "#a0a0a0";
	}

	return cc;

}
```

### Explanation 1 - Radio Box

Notice that we have the code

```
	if( uRB.checked == true )
```

This is the radio button that we gave to the function definition. The radio button has a special attribute called "**checked**".

If this value is true, then the user clicked on it.

However, if this value is false, then it is blank and the user did not click on it.

If the user clicked on "**urgent**", then we give a color code of **"#ff0000",** which is red.

If the user clicked on "**high**", then we give a color code of **"#483D8B"**, which is purple.

If the user didn't click on anything, then it is a normal event and we give a color code of "#a0a0a0", which is grey.

**Explanation 2 - Function Definition, Parameters, and Return Value**

In between the curly brace { } is the code that calculates the color of the event ( ie. urgent or high priority ). What if the user clicks on the "**Add New Event**" button multiple times. This means that we have to recompute the same code again since the user clicked on it again. How can we reuse the code inside the curly braces?

If we give code a name, we can call the name of the code to use it again. A function definition is the process of giving code a name. In the above, a function definition starts with the word "**function**" and then the name of the code comes after. So the name of the code is **get_cc()**.

In between the open and closed parenthesis are parameters. Parameters are inputs that we give to the function definition. In the code above, we gave 2 parameters to the function definition and the two parameters are named `uRB, hRB`.

How do we give data back? We use the word "**return**". Based on the priority, whether it's urgent or high priority, we give back the color code to mark that event as a special color.

We gave code a name and if we want to run that code again, we call its name again. See below.

### 3. Write The JS Code In Between <script> </script> ONLY TYPE THE CODE IN GREEN in the correct location

```
function get_cc( uRB, hRB )
{
	var cc = "#606060";

	if( uRB.checked == true )
	{
		cc = "#ff0000";
	}
	else if( hRB.checked == true )
	{
		cc = "#483D8B";
	}
	else
	{
		cc = "#a0a0a0";
	}

	return cc;

}

// -- eCount means entry count
var eCount = 1;

function newEvent()
{
	// -- linkn JS with radio box
	var uObj = document.getElementById( "uPri" );
	var hObj = document.getElementById( "hPri" );

	// -- link JS with input box
	var sBar = document.getElementById( "eT" );
	var sBarData = sBar.value;
	alert( "user input: " + sBar );

	var baseObj = document.getElementById( "eb_" );

	// -- make a clone of the <div id = "eb_">
	var clObj   = baseObj.cloneNode(true);

	// -- set the attribute using setAttribute( )
	clObj.setAttribute( "id", "eb_" + eCount );
	clObj.innerHTML = sBarData;

	// -- change color
	clObj.style.backgroundColor = get_cc( uObj, hObj);
```

```
        // -- put at the end
        baseObj.appendChild( clObj );

        eCount++;


}
```

**Explanation - Function Definition, Parameters, and Return Value**

Look at the line of code

```
        clObj.style.backgroundColor = get_cc( uObj, hObj);
```

Notice that **get_cc()** is actually above and you already wrote it. Go back to Chapter 2 to refresh our memory. Remember that the function **get_cc()** returns a color code based on if the user clicked on "urgent" or "high" priority.
1. if we return something, we have to store it into a variable or else the data is lost.

In this case, we use the color code as the background value and this allows the event block to have a unique color.

**HTML Challenge**
1. create another <h1> tag with an id of "**uEC**". The id of "**uEC**" stands for "**urgent event count**"

**JS Challenge**
1. link JS code with the tag with id "**uEC**"
2. keep track of the number of "**urgent**" events and then display the result to the tag with id "**uEC**"

**JS Super Challenge**
1. link JS code with the tag with the id of "**bTag**"
2. if the number of urgent events is greater than 2, then change the backgroundColor of "**bTag**" to "`#ff0000`";