JS 19 Calendar OOP

We have created a calendar that takes input from the user and uses the inputted data to make a decision.

For example, we input the event day as "**3**" and then click on "**Jan**". The code takes our input and routes our event object to the tab that is "**jan_events**" and then adds it to the day 3 event row.

In this session, we update our tabs to include the number of events in that month.

1. Urgent & High count
2. Events per month count
3. Color code

==========

1. **Write The JS Code In Between `<script> </script>`. WRITE all of it!!! Color code is for explanation.**

```
function update_monthECount( mId )
{
	var mIndex = 0;
	switch( mId )
	{
		// -- january
		case "jan_events":
		//alert( "inside case: jan_events" );
		mIndex = 0;
		break;

		// -- february
		case "feb_events":
		mIndex = 1;
		break;

		// -- march
		case "mar_events":
		mIndex = 2;
		break;

		// -- april
		case "apr_events":
		mIndex = 3;
		break;

		// -- default
		default: mIndex = 0;

	} // -- end switch

	// -- increment count of that month
	ePerMonthArr[mIndex]++;
```

```
	var mObj = document.getElementById( mId + "_ecount" );
	var mTabObj = document.getElementById( mId );

	mObj.innerHTML = "(" + ePerMonthArr[mIndex] + ")";

	if( ePerMonthArr[mIndex] >= oLevel )
	{
		tabColorArr[mIndex] = "#ffa500";
		mTabObj.style.backgroundColor = tabColorArr[mIndex];
		tabArr[mIndex].style.backgroundColor = tabColorArr[mIndex];
		eventArr[mIndex].style.backgroundColor = tabColorArr[mIndex];
		//tabArr[mIndex].style.backgroundColor = tabColorArr[mIndex];
		//tabSel( mIndex );
	}

	else
	{
		//tabColorArr[mIndex] = "#585732";
		//mTabObj.style.backgroundColor = tabColorArr[mIndex];
		//tabArr[mIndex].style.backgroundColor = tabColorArr[mIndex];

	}

	//tabSel( tabCode );

	return mIndex;
}
```

Explanation

We can keep track of the number of monthly events by making a variable for each month. Remember that a variable stores data and that is why variables can be used to keep track of the number of monthly events. **However, this would mean that we would need to make 12 different variables, one variable for each month.** While this is an OK approach, there is a big problem and that problem is remembering the variable names. As you guys notice, we are making **TOO MANY** variable and it will be difficult to keep track of all of those variables.

A better approach is to use arrays, where each position of the array stores the event count for a month.

Look at the code in PURPLE because this is for Jan events. If we add another event to the month of Jan, we want to increment the Jan event count by 1. The code in **GREEN** tells us that Jan is **position 0 of the array**. We store this position 0 into a variable called **mIndex**.

The code in **RED** and **GREEN** uses square brackets to select position 0 ➔ `ePerMonthArr[mIndex]++;` Finally, the **++** at the end adds 1 to the Jan event count.

The code in **ORANGE** is important because it is used to change the color of our tab based on the number of events in that month. The variable `oLevel` has the data 2 stored inside of it. So,

if the event count is greater than `oLevel` (ie. if the event count is greater than 2), then change the background color of our tab to `#ffa500`.

2. Write The JS Code In Between `<script> </script>`. ONLY write the code in **ORANGE** and **GREEN** and position it under the code in **BLACK.**

```
function make_obj( mId )
{

	// -- make the object
	var day = parseInt( document.getElementById("nEDay").value );
	var msg = document.getElementById("nEInput").value;
	var uPri = document.getElementById("uPri");
	var hPri = document.getElementById("hPri");
	var title= document.getElementById("nETitle").value;

	// -- create new Event Profile
	alert( "mId: " + mId );
	console.log( "title: " + title );
	var mEObj = new EventProf( mId, day, msg, uPri, hPri );
	mEObj.make_clone( currTicket );
	mEObj.set_event_listener( title );
	mEObj.update_eTitle();

	eArr.push(mEObj);
	//mEObj.make_clone();
	//alert( "eArr.length: " + eArr.length );

	currTicket++;

	// -- update tab with number of events
	var monthCode = update_monthECount(mId);

	if( mEObj.eColor == "#ff0000" )
	{
		urPerMonthArr[monthCode]++;
		document.getElementById( mId + "_urCount" ).innerHTML = "( U: " + urPerMonthArr[monthCode] + ") ";
	}
	else if( mEObj.eColor == "#483D8B" )
	{
		hiPerMonthArr[monthCode]++;
		document.getElementById( mId + "_hiCount" ).innerHTML = "( H: " + hiPerMonthArr[monthCode] + ") ";
	}
	else
	 {

	 }

	// -- reset radio buttons
	document.getElementById( "uPri" ).checked = false;
	document.getElementById( "hPri" ).checked = false;

}
```

Explanation

Notice that the code in **GREEN** is the JS function definition that you **ALREADY** wrote in Chapter 1. Chapter 1 code had the word "**function**" and it means that it is a **function definition**. The code above only has the name and that makes it a **function call**.

Next, we look at the code in **GREEN** and we start with `urPerMonthArr[monthCode]++;` Notice that we are using square brackets again and square brackets means that we are choosing **ONLY** one position of the array named `urPerMonthArr` (which means urgent count per month array). We select a position and increment the urgent count by 1.

Next, look at the line of code below

```
document.getElementById( mId + "_urCount" ).innerHTML = "( U: " + urPerMonthArr[monthCode] + ") ";
```

The line of code above links JS with HTML by using `document.getElementById()` and then giving a specific id. We then use `.innerHTML` to update the website with the number of urgent events for that specific month.

6. DO NOT Write The CSS Code In Between <style> </style>. Instead, style it any way you like

```
div#jan_events
{
	padding	: 0px 0px 0px 0px;
	margin	: 0px 0px 0px 0px;

	color	: #ffffff;

}

div#feb_events
{
	padding	: 0px 0px 0px 0px;
	margin	: 0px 0px 0px 0px;

 }

div#mar_events
{
	padding	: 0px 0px 0px 0px;
	margin	: 0px 0px 0px 0px;

}
```

Explanation

The code above customizes the div block. You can customize it anyway you like by adding background-color, border, text-size, color, and etc.

JS Challenge 1

We have to filter our data or else our calendar will accept bad data. For example, if we put the day as "**50**" and click on the "**Jan**" button, our calendar code will break. Why? Because the most days a month can have is 31 days. So, for this challenge, do the following

1. go inside the function definition `function make_obj( mId )`
2. check if the "**day**" is greater than 31 **OR** less than 1
   - **2.**1. If true, then alert the user "**Accepted days are 1 – 31**"
   - 2.2. else, accept the data and continue

JS Challenge 2

Currently, our tab and event block changes background color when there are 2 or more events within that month. For example, if you click on the "**Jan**" button 2 times, **ONLY** the "**Jan**" tab and the data block will change color. Your task is to change the background color when there are 4 events or 6 events.

If the number of events in the month is **4 or more**,

1. change the backgroundColor to `#9370DB`

If the number of events in the month is **6 or more,**

2. change the backgroundColor to `#B22222`

HINT: go back to Page 2 and look at `if( ePerMonthArr[mIndex] >= oLevel )`