JS 20 Calendar OOP

Sometimes, we make a typo and need to **enter edit mode** to make the corrections. Other times, things change and corrections need to be made to reflect the updated data. **One important feature of a calendar is the ability to edit existing data.**

In this JS session, we focus on giving the user the ability to make changes on the fly and then submit the changes to take effect immediately. **This session is important because we are finding a way for JS to collaborate with HTML and this collaboration implements administrative control**. **This JS session combines JS 6 – 19.**

1. HTML gives data to JS 2. Admin control

==========

1. **Write The HTML Code In Between** <body> </body>. **ONLY** write the code in **BLUE, GREEN, PURPLE, and ORANGE** and position it under the code in **BLACK**.

```
<button
id      = "nEventB"
onclick = "make_obj('mar_events')"
>
Mar
</button>


<button
id      = "modEB"
onclick = "mod_eObj()"
>
Mod
</button>
```

Explanation

We are creating a button with the id of "**modEB**". Next, we attach an "**onclick**" event to the button. When the user clicks on this button, the JS function definition "**mod_eObj()"** is called.

This button will be a simple button to accept changes made by the user when making corrections. The JS function and other actions will be written next.

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

```
var oLevel     = 2;
var pLevel     = 4;
var rLevel     = 6;

var chosenTicket = 0;
```

Explanation

We created a variable named "**chosenTicket**" and then used the assignment operator (single equal sign) to load the data 0 into "**chosenTicket**".

In a typical calendar website, there will be hundreds of objects (remember that objects are clones of classes and we customize objects with any values we want). **How do we know which object was clicked on?** We have to store the object into a variable so we can use it later on.

If I make an error and want to enter edit mode, I first have to **remember** what event object needs to be fixed. So, I make a variable called "**chosenTicket**" to store the **HTML id** of the object that needs to be corrected. Since I stored the **HTML id into a JS variable**, I can then use `document.getElementById()` to allow JS to link up with HTML.

Continue to the next page

3. **Write The JS Code In Between <script> </script>. ONLY** write the code in **GREY** and **GOLD** and position it **ABOVE** the code in **BLACK.**

```
// --
function mod_eObj()
{
      //alert( "inside  mod_eObj() " );

      //this.tabId + "_" + this.day + "_" + this.eTicket;

      // var objToMod = document.getElementById( chosenEId );
      eArr[chosenTicket].update();

}

// -- used to show number of events in the month of jan
function inc_jan_e_count()
{
```

Explanation

The code in **GREY** and **GOLD** is the JS function definition that matches the "**onclick**" from chapter 1 above. When the user clicks on the "mod" button, the changes will be accepted and the website is updated with new values.

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

```
show_pop_up()
{
      // -- save the event that was clicked on
      chosenTicket = this.eTicket;
      console.log( "chosenTicket:" + chosenTicket );

      this.pop_form();
```

Explanation – this time, no need to put closing curly braces

Do we want edit mode to **ALWAYS** be active? Probably not. So, the question now is, when will edit mode be activated? The code in gold and green tells us that edit mode will be activated when the pop up banner appears. The pop up banner appears **ONLY** when we click on an event block.

We are double dipping here. When we click on an event block
1. The pop up banner appears
2. Edit mode is activated

5. **Write The JS Code In Between <script> </script>. ONLY** write the code in **GREEN**, **GOLD**, and **GREY** and position it under the code in **BLACK.**

```
toggle_pop_up()
{
	this.popUpSt = !this.popUpSt;

	if( this.popUpSt )
	{
		this.show_pop_up();
	}
	else
	{
		this.hide_pop_up();
	}
}

pop_form()
{
	document.getElementById("nEDay").value   = this.day;
	document.getElementById("nEInput").value = this.msg;
	document.getElementById("nETitle").value = this.eTitle;

	var uPri    = document.getElementById("uPri");
	var hPri    = document.getElementById("hPri");

	if( this.eColor == "#ff0000" )
	{
		uPri.checked = true;
	}
	else if ( this.eColor == "#483D8B" )
	{
		hPri.checked = true;
	}
	else
	{
		uPri.checked = false;
		hPri.checked = false;
	}

}
```

Explanation

The code in **GREEN** is a **private function** of class EventProfile. How do we know if something is a private or public function? See next page for a hint.

```
// -- public function DOES HAVE the word “function”
function get_sum( num1, num2 )
{
	return num1 + num2;

}

// -- private function DOES NOT have the word “function”
get_remainder( num1 )
{
	return num1 % 2;
}
```

6 . **Write The JS Code In Between <script> </script>. ONLY** write the code in **GREEN** and position it **ABOVE** the code in **BLACK.**

```
update()
{
	//alert( "inside private fx update()" );

	// --
	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;

	this.eColor = this.get_cc( uPri, hPri );
	this.msg    = msg;
	this.day    = day;
	this.eTitle = title;

  var eObj = document.getElementById( this.tabId + "_" + this.day + "_" + this.eTicket );

	//alert( "this.eColor: " + this.eColor );
	console.log( "this.eColor: " + this.eColor );
	eObj[0].style.backgroundColor = this.eColor;

	eObj.innerHTML = this.eTitle;

}

set_event_listener( eventTitle )
{
	this.eTitle   = eventTitle;
```

Explanation

The code in **GREEN** is a **private function** of class EventProfile. We are **inside** a private function and this means that we can use `this.` as a short cut. **Time to trick you guys …**
Continue to Classes & Objects Challenge

Classes & Objects Challenge

Look at the code below and see that we have two variables named **this.msg** and **msg**.

```
this.msg    = msg;
```

Which variable is private and which variable is global? The hint is to find the member access operator. To the **left** of the member access operator **is the owner** and to the **right** of the member access operator is **what is being owned**.

If a variable **DOES NOT** have a member access operator, is it private or public?
1. If it is private, then who is the owner?

Classes & Objects Challenge 2 – Let's Make Code Combat!!!
a. Finish all questions below by writing the code in Notepad or a Word document

1. Write a class definition and name it GameChar

2. Write a constructor that takes in 4 inputs. Each input is assigned to a private variable of the same name
    a. costTB
    b. atkAmt
    c. currHealth
    d. gState

3. Write a **private function** of GameChar named "**check_currHealth()**". Inside this **private function**, do the following
    a. check if **currHealth** is less than 0. If true, then assign "sleep" to the **private** variable **gState**
    b. return the private variable **gState**

4. Write a **private function** of GameChar named "**sub_atkAmt()**" and it takes in an input named "**damgAmt**". Inside this **private function**, do the following
    a. subtract "**damgAmt**" from the private variable **currHealth**

5. Outside of the class definition, do the following
    a. Create an object of the class GameChar and name it "**Ogre**"
    b. Initialize "**Ogre**" with the following data.
        i. 10
        ii. 5
        iii. 100
        iv. "idle"

6. Outside of the class definition, do the following
    a. Create an object of the class GameChar and name it "**Brute**"
    b. Initialize "**Brute**" with the following data.
        i. 50
        ii. 25
        iii. 100
        iv. "idle"

7. Outside of the class definition, do the following
    a. Create an object of the class GameChar and name it "**Archer**"
    b. Initialize "**Archer**" with the following data.
        i. 20
        ii. 12
        iii. 100
        iv. "idle"

8. Underneath 5 – 7, do the following
    a. Use "**Ogre**". Next, call Ogre's private function **sub_atkAmt()** and give **Archer**'s **atkAmt** as an input parameter to **Ogre's sub_atkAmt()**.
    b. Next, use "**Ogre**" and call **Ogre**'s **check_currHealth().**
    c. Catch the return data into a variable called "**enemyHealth**"
    d. Make the check
        i. if enemyHealth is less than 0, then send an alert that says "Ogre taken out"