JS 18 Calendar OOP

In JS 14 – 16, **we experimented with class definition and objects.** We used classes and objects to **create a basic music service**. Our music service had the options to move forward and backward, pick a song randomly, and we used a pop up menu to hide extra information until the user asks for it.

We will now **transition back to our Calendar app** and apply what we learned to continue adding features.

In this session, we focus on adding an **event day and event title**. Next, we will **route the event** to the appropriate day within the chosen month. Finally, we set up the event for our **pop up banner**.

1. Add event listener 2. Absolute positioning 3. Classes and objects

==========

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

```
<div
class = "data_block"
id = "high_events"
>
high priority events
</div>


<div
class = "data_block"
id    = "jan_events"
>
    <table
    >
        <tr>
            <th>
            1
            </th>

            <td
            id = "jan_events_1"
            class = "eTdCell"
            >

            </td>
        </tr>


        <tr>
```

```
    <th>
    2
    </th>

    <td
    id = "jan_events_2"
    >

    </td>
 </tr>


<tr>
    <th>
    3
    </th>

    <td
    id = "jan_events_3"
    >

    </td>
 </tr>



<tr>
    <th>
    4
    </th>

    <td
    id = "jan_events_4"
    >

    </td>
 </tr>



<tr>
    <th>
    5
    </th>

    <td
    id = "jan_events_5"
    >

    </td>
 </tr>


 <tr>
```

```
                <th>
                23
                </th>

                <td
                id = "jan_events_23"
                >

                </td>
            </tr>
        </table>
    </div>
```

Explanation

In JS 7, we were introduced to tables `<table>`, `table row <tr>`, `table header <th>`, and `table data <td>`. Remember that JS 7 was just a long column (ie. vertical bar). The data that we put in the search bar shows up in the long vertical bar. The main problem was that the events were out of order. The code above is the solution.

The code above is an enhancement of JS 7 because we add the days of the month January. Each day is its own row and this is why we have the code below.

```
            <tr>
                <th>
                1
                </th>

                <td
                id = "jan_events_1"
                class = "eTdCell"
                >

                </td>
            </tr>
```

The only different is that the **red number** is changed and that is it. With this enhancement, an event on Jan 3rd will be displayed in the correct row. **Looking back at JS 7**, notice that the code of JS 7 **DOES NOT** have an id for the table data (the code in **ORANGE** is the table data **<td>**). Not having an id means that all events are out of order. However, with the code in **ORANGE** above, we **DO GIVE** an id to the table data and that allows us to organize events based on the day the event is scheduled to take place.

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

```
class EventProf
{
	constructor( mId, eDay, eMsg, uPri, hPri )
	{
		// -- eColor ==> event color
		this.eColor = this.get_cc( uPri, hPri );
		this.msg    = eMsg;
		this.day    = eDay;
		this.tabId  = mId;

		this.eTitle  = " ";
		this.eTicket = " ";

		this.popUpSt = 0;

	}
```

Explanation

The code in **GREEN** are inputs (they are called parameters) to our class constructor and are used to initialize our objects. Initialize means to give it specialized values and this is how we customize our objects.

The second code in GREEN is a new field that we have added. In previous versions of our calendar app, we just had a message. In this version, we have a title that will be displayed and the message will only appear in the pop up.

Notice that the word `this.` appears in the code and it has a special meaning. The word `this.` indicates that it is a private attribute. How do we access a private attribute? We use the member access operator, which is the period ➔ `this.`

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

```
update()
{
      var monthObj = document.getElementById( this.tabId );
      alert( "monthObj: " + monthObj );
      this.tabId.innerHTML += this.day + ", " + this.msg;
}

set_event_listener( eventTitle )
{
      this.eTitle   = eventTitle;

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

      // ============================ using click to toggle
      eObj.addEventListener( "click", this.toggle_pop_up.bind(this) );

      // ============================ using hover below
      // -- pop up
      // eObj.addEventListener( "mouseover", this.show_pop_up.bind(this) );


      // -- hide
      // eObj.addEventListener( "mouseleave", this.hide_pop_up.bind(this) );

}
```

Explanation

How does Javascript search for and find an HTML element? It uses the id and `document.getElementById()` to find the exact match. The first code in **GREY** does that.

The second lines of code in **GREY** is important because it is setting up our action, which is to show the pop up banner. If we look at the code

```
      eObj.addEventListener( "click", this.toggle_pop_up.bind(this) );
```

Notice that our event is "`click`" and our action is "`this.toggle_pop_up`". So, this translates into "when the user **clicks** on an event, the action **this.toggle_pop_up** is run. The word toggle means to do the opposite.

Assume that the website starts and the pop up banner **IS HIDDEN**. We then **click** on an event on our calendar. Remember that the banner **IS HIDDEN**. A click will toggle and toggle means to do the opposite ➔ the banner becomes **VISIBLE**.

4. **Write The HTML Code In Between** <body> </body>. **ONLY** write the code in **BLUE** and position it under the code in **BLACK**.

```
<input
id         = "nEDay"
value      = "1"
minlength  = "2"
maxlength  = "2"
name       = "nEDay"
/>

<input
id         = "nETitle"
value      = "testing title"
minlength  = "20"
maxlength  = "50"
name       = "nETitle"
/>


<input
id         = "nEInput"
value      = "testing eMsg"
minlength  = "20"
maxlength  = "100"
name       = "nEInput"
/>
```

Explanation

In previous versions, we had code that made a search bar and the user entered a message in the search bar.

The code in **blue** is an update to what we had. We now have 3 different fields for the user to enter information about the event

1. Day of the month
2. Title of the event
3. Extra information

The day and title are visible to the user. However, the extra information is not visible until we activate the pop up banner. Remember, we activate the pop up banner on the "**click**" event. See chapter 3 to confirm.

The extra information can be any message that describes the event in more detail. For example, the extra information can be the time of start and end, the address of the event, or any personal messages.

Remember that we made the pop up banner a large entity by styling it (ie. using CSS) with large values for width and height. This allows more information to pop up.

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

```
<input
id        = "nEInput"
value     = "testing eMsg"
minlength = "20"
maxlength = "100"
name      = "nEInput"
/>

<br/><br/>

<div
id = "pUpBanner"
>
  hi
</div>


<!--

onclick = "make_obj(jan_events)"
a. this pass the HTML DOM to the function

onclick = "make_obj('jan_events')"
a. this passes the string data "jan_events" to the function
-->
<button
id      = "nEventB"
onclick = "make_obj('jan_events')"
>
Jan
</button>


<button
id      = "nEventB"
onclick = "make_obj('feb_events')"
>
Feb
</button>


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

Continue to the next page

Explanation

These are our buttons to register the event. We have 3 buttons, one for each month starting on January. Look at the code below

```
<button
id      = "nEventB"
onclick = "make_obj('jan_events')"
>
Jan
</button>
```

We have "onclick" as our event. Next, we have the code in **GOLD** and **SKY BLUE**, which one is the function? The code in **GOLD** is the JS function (ie. action when we click on Jan) and the code in **SKY BLUE** is the parameter (ie. input to the JS function).

We are saying ➔when the user **clicks** on the button **Jan**, make an object with name '**jan_events**'. Scrolls up to the code again and you will see that the code is the same for all 3 buttons except that the parameter is different (ie. the code in **SKY BLUE** is different).

Continue to the next page

6. Write The CSS Code In Between <style> </style>. Next, style it any way you like.

```
#pUpBanner
{
	position	: absolute;
	top		: 150px;
	left		: 150px;

	padding		: 0px 0px 0px 0px;
	margin		: 0px 0px 0px 0px;

	width		: 300px;
	height		: 150px;

	background-color: #00ffbb;
	box-shadow: 7px 5px 2px #f0f0f0;
	border		: 1px solid #a0a0a0;

	text-shadow: 0px 1px 1px #a0a0a0;
	font-weight: bold;
	color		: #ffa500;
	font-family: Arial, Tahoma, Serif;

}
```

Explanation

This is our pop up banner that is used to display extra information. You can style the pop up banner any way you like. **Change the background-color, border, font-size, text-shadow, and etc.**

HTML Challenge 1

1. We already have the HTML table for January. Now, make a table for February
    a. You don't have to change very much …

HTML Challenge 2

1. We already have the HTML table for January. Now, make a table for March
    a. You don't have to change very much …

JS Challenge

1. Keep track of how many events are in the month of January
2. Next, do the following

2.1. check if the number of events in January is greater than 3. If so,
   a. link JS code with the HTML element whose id is "`jan_events`"
   b. change the backgroundColor to #FF8C00

2.2. check if the number of events in January is greater than 5. If so,
   a. link JS code with the HTML element whose id is "`jan_events`"
   b. change the backgroundColor to #FF6347

2.3. check if the number of events in January is greater than 7. If so,
   a. link JS code with the HTML element whose id is "`jan_events`"
   b. change the backgroundColor to #B22222