**FINAL JAVASCRIPT**

**HINTS & TIPS**

Planning & Structure

Before beginning code, you need to plan exactly what the purpose of the code is e.g.

|  |  |
| --- | --- |
| Add event listener | C |
| Get input values | UI |
| Add the new item to our data structure | Data |
| Add the new item to the UI | UI |
| Calculate budget | Data |
| Update the UI | UI |

*UI = User Interface module, C = controller module, Data = data module*

Modules allow us to break up our code and our code allows these modules to interact with each other. They also allow us to encapsulate some data into privacy and expose other data publicly.

With module pattern, we return all of the objects that we want to make public i.e. the functions that we want to give the outside scope access to.

Implementing the module pattern

Var budgetController = (function(){ //this controller works independently to the UIController

//private functions here

e.g. var Income = function(id, desc,val) {

this.id = id;

this.desc = desc;

this.val = val;

//public functions to be accessed – in an object

Return {  
 logName: function() {  
 console.log(‘Ali’);

}

}; //to access this look at the controller module

})();

Var UIController = (function(){

//private functions

Var DOMStrings = {  
 type: document.querySelector(‘.type’),

Btn: document.querySelector(‘.btn’)

};

//public functions  
 return {  
 getDOMStrings: function() {  
 return DOMStrings;

}

};

})();

Var controller = (function(budgetCtrl, UICtrl){

Var DOMStrings = UICtrl.getDOMStrings(); //get access to DOM strings

budgetCtrl.logName(); //this logs name to console – look in budget controller module

})(budgetController, UIController); //The 2 above controllers added as arguments

**Other helpful tips**

insertAdjacentHTML

insertAdjacentHTML(position, text);

position =

‘beforebegin’ – before the element itself (sibling)

‘afterbegin’ – just inside the element, before its first child (child)

‘beforeend’ – just inside the element, after its last child (child)

‘afterend’ – after the element itself (sibling)

e.g. el.insertAdjacentHTML(‘beforeend’,’hello’);

Encapsulation

One of the main concepts in OOP. It allows an object to group both private and public members (properties & methods), under a single name.

forEach()

the forEach() method executes a provided function once for each array element. Similar to a for loop – the function provided has access to the value, index and full array, of that array.

Var fields = document.querySelectorAll(‘.add\_\_description, .add\_\_value’); //node list

Var fieldsArr = Array.prototype.slice.call(fields); //borrow slice method

fieldsArr.forEach(function(cur, i, arr) {

cur.value = ‘’;

});

Cur = current item selected, i = index number, arr = full array

e.g.2 – var arr = [‘one’,’two’,’three’,’four’,’five’];

arr.forEach(function(cur,i,arr) {

console.log(cur+’ ‘+i+’ ‘+arr);

});

Console => ‘one’ 0 one, two, three, four, five

‘two’ 1 one, two, three, four, five //etc…

Map()

Map() is very similar to the forEach() method, except map() returns a brand new array. Use this when you want to do something *to* an array, and use forEach when you want to do something *with* an array.

The map() method creates a new array with the results of calling a provided function in every element in the calling array.

Var months = [‘jan’,’feb’,’mar’,’apr’,’may’];

Var months = months.map(function (cur) {  
 return ‘month: ‘+cur;

};

=> [‘month: jan’,’month: feb’,’month: mar’,’month: apr’,’month: may’];

The difference between map and forEach is; forEach iterates over a list and applies some operation with side effects to each list member – e.g. saving every list item to the database. Map iterates over a list, transforms each member of that list, and returns another list of the same size with the transformed members – e.g. transforming a list of strings to uppercase.

parseFloat & parseInt

these methods convert string numbers to number primitive type and can also be used on numbers.

parseFloat() – decimal

parseInt() – integar (whole number)

isNaN()

isNaN() is a function which checks if a number is not a number (NaN)

e.g. var a = 21.156;

if(!isNaN(a)) {  
 console.log(‘is a number’);

}; // returns is a number

Event delegation

Event delegation refers to the process of using event propagation (bubbling) to handle events at a higher level in the DOM than the element on which the event originated.

It allows us to attach a single event listener for elements that exist now or in the future

HTML <ul id=’parent’>

<li id=’one’>one</li>

<li id=’two’>Two</li>

</ul>

JS function which(e) {  
 e.target.remove();

};

Document.querySelector(‘.parent’).addEventListener(‘click’,which);

Uses for event delegation:

1. When we have an element with lots of child elements that we are interested in
2. When we want an event handler attached to an element that is not yet in the DOM when our page is loaded

Splice()

The splice() method changes the contents of an array by removing existing elements and/or adding new elements.

Array.splice(start, deleteCount, item1, item2,…);

Start = position you want to start deleting

deleteCount = the number of array elements to remove (optional)

item1,item2,… = new array elements (optional)

var a = [‘ali’,’shahin’,’afshin’,’bob’];

a.splice(2,2); => [‘ali’,’shahin’];

indexOf()

indexOf() returns the position of the first occurrence of a specified value in a string.

Returns -1 if false

Var a = ‘hello’;

Console.log(a.indexOf(‘e’)); =>1

Var arr = [‘one’,’two’,’three’];

Console.log(arr.indexOf(‘three’)); =>2

Substr()

The substring method (substr()) is a method of the string prototype which returns a subset of a string between one index and another.

Takes 2 parameters:

1. Index start
2. Index end (optional) – if not selected then selects last index

Var a = 15315.16516;

A = a.substr(0, a.length-3) + ’,’ + a.substr(a.length-3); // =>15,315.16516

**TEST**

1. Give a small example of modular pattern including how to make functions private and public, as well as normalising the DOM tree the same across modules
2. Define encapsulation
3. How do you use the insertAdjacentHTML method
4. What does the forEach() method do and give example
5. What do the map() method do and give example
6. What is the difference between forEach() and map()
7. Give example of parseFloat() and parseInt()
8. What does isNaN() do
9. Define event delegation and give a brief example of how it can be used
10. Give an example of:
11. Splice()
12. Split()
13. Subsrt()
14. indexOf()