THE COMPLETE JAVASCRIPT COURSE

**FINAL PROJECT**

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.

\*\*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.\*\*

Implementing the module pattern

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

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

Var Expense = function(id, description, value) { //function constructor

This.id = id;

This.description = description;

This.value = value;

});

})();

Var UIController = (function(){

Var DOMStrings = { //object to store strings of class names  
 inputType: ‘.add\_\_type’,

inputDescription: ‘.add\_\_description’,

inputValue: ‘.add\_\_value’,

inputBtn: ‘.add\_\_btn’

});

Return { //return as we want these objects public

getInput: function() {

return {

type: document.querySelector(DOMStrings.inputType).value,

description: document.querySelector

(DOMStrings.inputDescription).value,

Value: document.querySelector(DOMStrings.inputValue).value

};

},

getDOMStrings: function() { //returned so controller module can access

return DOMStrings;

}

})();

Var controller = (function(budgetCtrl, UICtrl){

Var setupEventListeners = function() { //event listeners

Document.querySelector(DOM.inputBtn).addEventListener(‘click’, ctrlAddItem);

Document.addEventListener(‘keypress’, function(e) {

If(e.keyCode === 13) {

ctrlAddItem();

}

});

};

Var DOM = UICtrl.getDOMStrings(); //getDOMStrings() from UIController accessed

Var ctrlAddItem = function () { //function fired above which gets input and shows on console

*//1. Get the field input data*

Var input = UICtrl.getInput();

Console.log(input);

*//2. Add the item to the budget controller*

*//3. Add the item to the UI*

*//4. Calculate the budget*

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

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)