JAVASCRIPT PATTERNS AND BEST PRACTICES

EIRIK REKSTEN (@EIREKSTEN)

TOOLING

GRUNT.JS

- Build Tool for Javascript Projects
- Configured in JS code
- Bursting with premade plugins!

EXERCISE 1

https://github.com/eireksten/jspatterns

STATIC CODE ANALYSIS

Avoiding some of the gotchas mentioned in part 1 of this series.

JSHINT

JSHint is a program that flags suspicious usage in programs written in JavaScript. It helps developers detect potential bugs in their JavaScript code and enforce their development team's JavaScript coding conventions.

(**http://www.jshint.com/docs/**)

JSHINT RULES

- Enforce `===`
- Remember `var` and `;`
- Unused variables
- `"use strict";` statement.
- And many more...

EXERCISE 2

https://github.com/eireksten/jspatterns

DOM MANIPULATION LIBRARIES

ZEPTO / JQUERY

WHAT IS JQUERY?

jQuery is a javascript library that adds the `jQuery` (alias `$`) function object. It is intended to simplify cross browser DOM manipulation, server communication and event handling, as well as providing a couple of generic utility methods.

DOM MANIPULATION

```
var $element = $('<div>');                  // create an element
$element.addClass('mydiv');                 // add css class
$element.removeClass('mydiv');              // remove css class

var $subelement = $('<p>');
$element.append($subelement);               // add child

var $tables = $('table#myid.myclass'); // find elements
var $cells = $tables.find('td');            // find subelements

$element.click(function () {
   // Click event handler for the element.
});
$element.remove();                          // remove an element from the DOM
                                            // and destroy event handlers
```

OTHER UTILITIES

```
jQuery(function () {
    // This code is run when the document is loaded
    var myapp = new Application();
    myapp.initialize($('div#myelement'));
});

$.ajax({ // options for the ajax call to the server
    url: 'http://my.url',
    type: 'POST'
}).done(function (response) {
    // This code is run when the http request returns successfully.
}).fail(function () {
    // This code is run when the http request returns an error.
}).always(function () {
    // This code is run when the http request completes.
});
```

PITFALLS

- Spaghetti is not a good pattern!
- Avoid duplicating DOM searches!
- Remember that jQuery is only a library!

EXERCISE 3

https://github.com/eireksten/jspatterns

PATTERN - MODULE

A module is an independent part of your program.

PROBLEM

- In the browser, all javascript files share the same namespace
- Most functions and variables are only interesting for a little part of your program.
- We want to avoid name collisions between different parts of the program.

IMPLEMENTING MODULE PATTERN

```
// module.js

(function () {
    // The module code goes here!
    // Variables declared here will only be visible inside the module.
}());
```

SINGLE GLOBAL VARIABLE

When exposing functionality from modules, a single global object can be used to avoid polluting the namespace.

```
// module.js
var mynamespace = mynamespace || {};
(function () {
    // Exporting a function
    var hello = "Hello Steria!";

    mynamespace.beAwesome = function () {
        console.log(hello);
    };

}());
```

EXERCISE 4

https://github.com/eireksten/jspatterns

REQUIRE.JS

Lets you implement client side modules with dependencies.

```
// mymodule.js
// defines a module with dependencies to note and notelist (in same folder)
define(["./note", "./notelist"], function(note, notelist) {
    // return an object to define the "mymodule" module.
    return {
        addnote: function (content) {
            var mynote = note.create(content);
            notelist.add(mynote);
        }
    }
});
```

http://requirejs.org/

PATTERN - CONTROLLER OBJECT

The Controller Object is an object responsible for a single component on the page. It should have a clearly defined element to work on, and not know anything about other elements on the page.

IMPLEMENTING THE CONTROLLER OBJECT

```
var highlightController = {
    bindEvents: function () {
        this.$element.on('mouseover', this.handleMouseOver.bind(this));
    },
    handleMouseOver: function () {
        this.$element.addClass('highlighted');
    }
};

exports.createHighlighter = function ($element) {
    var controller = Object.create(highlightController);

    controller.$element = $element;
    controller.bindEvents();

    return controller;
}
```

EXERCISE 5

https://github.com/eireksten/jspatterns

CONTROLLER OBJECTS IN FRAMEWORKS

Controller Objects exist in one form or another in most MV* Frameworks in Javascript, including

- Backbone
- Spine
- AngularJS
- ember.js
- React
- Javascript MVC
- And so on...

CLIENT SIDE TEMPLATING

Client Side Templating is the generation of HTML (from model data) in the browser.

HANDLEBARS.JS

```
<h4>Notelist ({{notecount}}</h4>
<ul>
    {{#notes}}
        <li data-noteid="{{id}}">{{name}}</li>
    {{/notes}}
</ul>
```

COMPILING AND RENDERING

Handlebars templates can be compiled into javascript functions using a command line tool or grunt plugin.

It can then be used as follows:

```
var html = templateobject.templatename({
    notecount: notelist.length,
    notes: notelist
});
```

EXERCISE 6

https://github.com/eireksten/jspatterns

INCREASING COHESION

Cohesion refers to the degree to which the elements of a module belong together.

THE CALLBACK PATTERN

The callback pattern consists of passing a callback function as argument to another function. This function is called when set conditions are met. Example:

```
myfunction(function (data) {
    // do stuff with data
});

var myfunction = function (callback) {
    var data = obtainData();
    callback(data);
};
```

PUBLISH/SUBSCRIBE

In the publish/subscribe pattern, a publisher will emit 'events' when certain conditions are met or an something has happened. Others can 'subscribe' to these events by registering callbacks to be called when they occur.

PUBLISH/SUBSCRIBE USAGE

This is the exact same pattern as is used on DOM elements, which emit events such as 'click', 'keyup' or 'mouseover'.

```
var mypublisher = new Publisher();

mypublisher.on('change', function (name) {
    console.log('Hello ' + name + '!');
});

mypublisher.emit('change', 'Steria');
```

EVENT EMITTERS

- Often defined as a mixin to be used in any object to make it a publisher
- Contains at least the functions
  - `emit = function (type, args...)`
  - `on = function (type, callback, context)`
  - `off = function (type, callback)`

DATA MODELS

To loosen our coupling between the note and notelist controllers, we can introduce a note data model.

- A data model is a representation of the actual data object
- It should be oblivious to the existence of a UI
- It should contain all logic directly related to the data it represents, including manipulation, storage etc.

EXERCISE 7 AND 8

https://github.com/eireksten/jspatterns

INHERITANCE VS. MIXINS

Inheritance traditionally represents the "is-a" relationship, while a mixin is a small set of related functionality to be "mixed into" an object.

- Both mixins and inheritance are means of reusing code across objects.
- Mixins allow for a kind of "multiple inheritance".
- In statically typed languages, inheritance allow code reuse through polymorphism. This does not apply to javascript.

SUMMARY

- Javascript libraries does not allow you to put design principles aside!
- A web component should not know about anything other than the parts it contains.
- Callbacks are a way of achieving higher cohesion between modules and avoiding cyclic dependencies.

REFERENCES

- **http://jshint.com/docs/**
- **http://api.jquery.com/**
- **http://zeptojs.com/**
- **http://handlebarsjs.com/**
- [Book] Javascript Patterns - *Stoyan Stefanov*
- [Book] Maintainable JavaScript - *Nicholas C. Zakas*