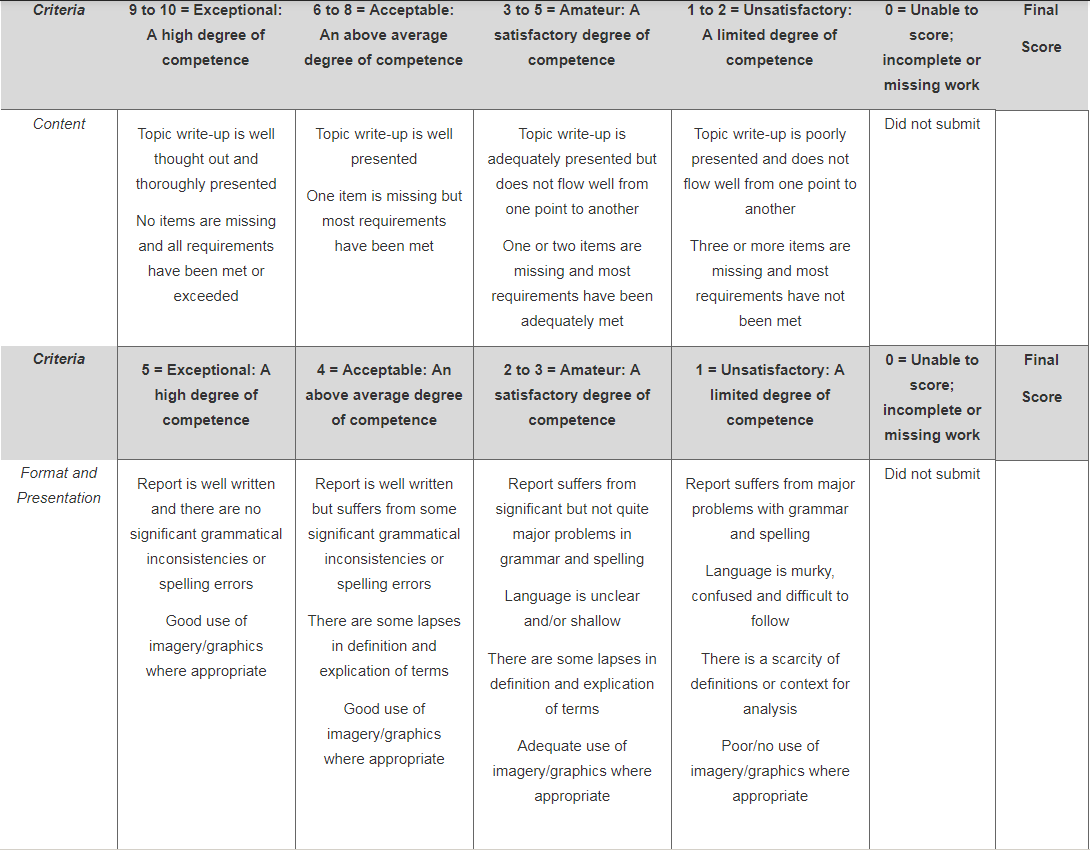
<https://addyosmani.com/resources/essentialjsdesignpatterns/book/> - js design patterns

OUR DESIGN PATTERN

I also started a github repo for this project so we can all contribute to the single webpage report we are going to hand in.

Design Patterns:

* Design patterns are reusable, generic solutions to regularly occurring situations in software design.
* Useful to create designs that are easily maintainable.

Anti-Patterns:

* A developer may introduce an ineffective design pattern to an application which may negatively influence the end quality of the application.
* Anti patterns in javascript seem to mostly be attributed with poor programming practices such as:
  + Polluting global space
  + Use of the eval() function
  + Use of document.write()

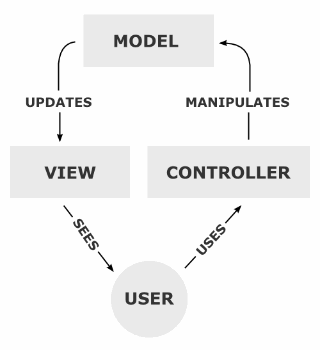
------ START OF REPORT ------

# Design Patterns

When working with any program language it can become difficult to maintain a system of organization and consistency. Design patterns provide solutions that may be descriptive toward the problem, while also being generic and easily reused for similar projects in the future. There are three varieties of design patterns which target different areas of applications. Creational patterns focus on controlling object creation in order to avoid uneccesary complexity in a project. Structural patterns establish relationships between different objects in order to restrict unnecessary change through the application. The last category is the behavioural pattern which focuses on communication between objects.

# MVC

MVC (Model-View-Control) is a design pattern developed from the idea of “Separated Presentation”. Modern frameworks like Backbone.js, Ember.js, React.js and more create a clear division between data, and presentation while also importantly creating structure for the app. This division allows for a more controlled and organized system, and can allow for portions of the model to be reused in other interfaces of the application.



The user does not get to directly interact with the model, but rather the interaction is controlled by the controller. When the model is then updated the model will ask the view to update what the user sees. Through this process an interactive application with a maintainable codebase is developed.

## Model

In a MVC environment the model is ignorant of what the user sees, but rather can be seen as the raw information the app is displaying. We may create a model in order to store information about anything, but the stored information tends to be quantitative and meaningful. The information stored in the model is generated either from values passed into the constructor which may have been sent from a controller, or initialized to a default value.

--- Vehicle model constructor with defaults, and mixin support snippet ---

This vehicle object represents a simple model. I can create a new vehicle object as follows:

--- default constructor snippet ---

This will create a vehicle object, and since there are no attributes passed in, it will set default values onto the variable defaultCar.

--- constructor with attrs snippet ---

This creates another vehicle object but the default values are overwritten with the values contained in the hash of key value pairs which is sent to the constructor. In order to change and ask for the values stored in these variables I may create functions set and get:

--- set get functions snippet ---

As an important side note we declare these functions as prototypes on the Vehicle object in order to not create get and set functions for every vehicle object created. The set function allows for changes in values to a vehicle object, such as if I wanted to update the mileage on defaultCar:

--- set usage snippet ---

The get function is mostly used internally to compare with, calculate or generate a view. The simple vehicle model is now complete.

## View

The view is effectively your application’s ambassador to the user — representing your application to the user and providing the basis on which the application is first judged.

The view provides different ways to present the data received from the model. They may be templates where that data is filled, some common templates that are used include Underscore, Mustache, Handlebars, jquery-tmpl. There may be several different views and the controller has to decide which one to use.

In order for an object to be displayed, the object will inherit the properties/methods from the prototype and add the function for rendering the object.

--- render function snippet ---

A target element is defined and will contain the elements created dynamically to hold the vehicle object attributes using the object's getter.

---create element snippet ---

Finally the object data is appended to the the main container in preparation for being displayed.

---append element snippet ---

## Controller

The Controller acts as a bridge between one or more of an application's model(s) and view(s). Controller objects will help inform the view(s) and the model(s) about changes. At the same time, the controller keeps the view(s) and the model(s) separate.

|  |
| --- |
| $('#defaultCar').on('click', function(e) {  var $tar = $('#defaultCarContainer');  defaultCar.render($tar);  });  $('#myCar').on('click', function(e) {  var $tar = $('#myCarContainer');  myCar.render($tar);  }); |