**Knockout**

Knockout is a JavaScript library that helps us to create rich, responsive display and editor user interfaces with a clean underlying data model  
**MVVM and View Models:  
Model-View-View-Model(MVVM)** is a design pattern for building user interface. It describes how User Interface splitting into 3 parts  
**A Model:**Our application’s stored data. This data represents objects and operations in our business domain. When using ko, we will usually make a AJAX calls to some server side code to read and write this stored model data  
**A View Model:**It is a pure-code representation of the dataand operations on a UI  
Ex:  
create a view model with ko, just declare any JavaScript object  
var MyViewModel = {  
 FirstName: ‘Afsha’,  
 LastName: ‘Mohammed’  
};  
**A View:**a visible, interactive UI representing the state of the view model  
When using ko, it is a HTLM document with declarative binding to link it to the View Model  
EX:  
create a simple view of the above view model using declarative binding  
<input type=”text” name=”FirstName” data-bind=”text: FirstName”>

**Activate knockout:**To activate knockout, we have to add the below in the <script> block  
ko.applyBindings( MyViewModel);

**Knockout JS build three core features:**Observables and Dependency Tracking,  
Declarative Bindings and   
Templating **Observables:**In knockout, declaring a member as observable means when its value changes any other object watching the member gets notified it has been changed. This simple concept allows the Two-Way-Data-Binding to be implemented  
Ex:  
rewrite the above ViewModel objects  
var MyViewModel = function(fName, lName){

this.FirstName = ko.observables(fName);  
this.LastName= ko.observables(lName);  
};  
**Reading and writing observables**  
**Read:** To read value just call observable property without the parameters  
EX: MyViewModel.FisrtName();  
**Write:** To write/update value in observable property, just pass the desired value in parameters   
EX: MyViewModel.FirstName(‘Afsha’);  
**Write multiple**:  
Multiple ViewModels can be updated in a single row with the help of chaining syntax   
MyViewModel.FirstName(‘Afsha’).LastName(‘Mohammed’).MyAge(26);  
**Computed Observables**  
It is a special type of function that is dependent on one or more observables to work  
computed called to make a Full name we have to combine firstName and LastName  
**Ex:**var MyViewModel = function(fName, lName){

this.FirstName = ko.observables(fName);  
this.LastName= ko.observables(lName);  
this.FullName = ko.computed(function(){  
 return this.FirstName() + ‘ ‘+ this.LastName();  
},this)  
};