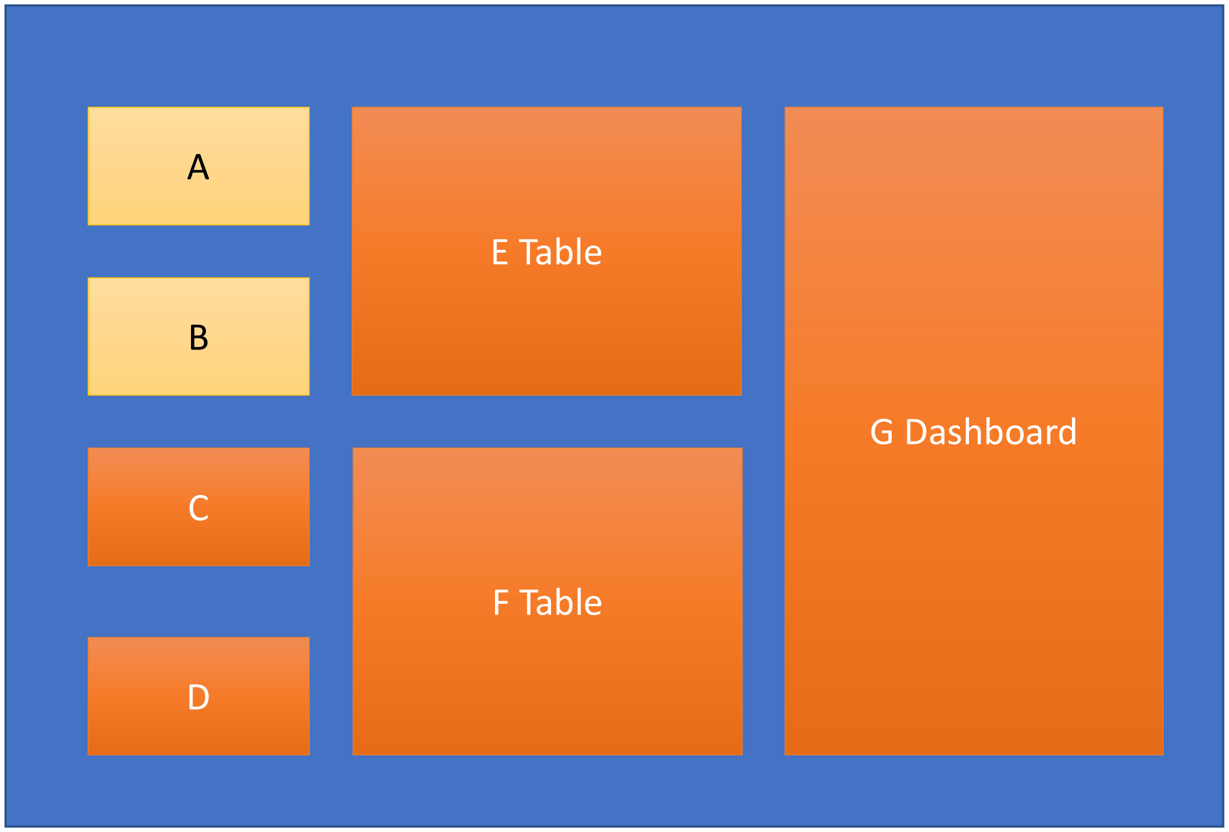
Knockout JS


A
B
C
D
E Table
F Table
G Dashboard

Features

- Declarative Bindings
  - Easily associate DOM elements with model data using a concise, readable syntax
- Automatic UI Refresh
  - When your data model's state changes, your UI updates automatically
- Dependency Tracking through Observables
  - Implicitly set up chains of relationships between model data, to transform and combine it
- Templating
  - Quickly generate sophisticated, nested UIs as a function of your model data

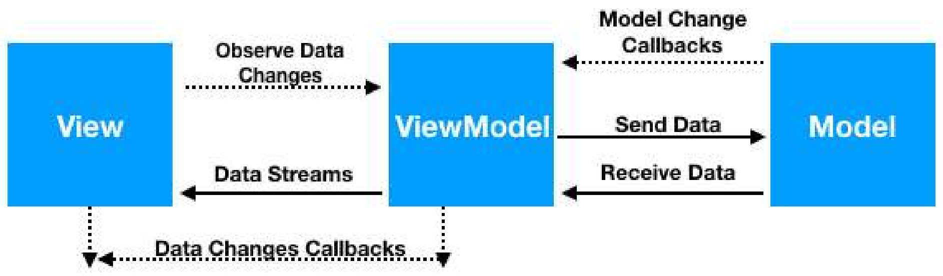
Additional Benefits

- Additional benefits:
  - **Pure JavaScript library** - works with any server or client-side technology
  - **Can be added on top of your existing web application** without requiring major architectural changes
  - **Compact** - around 13kb after gzipping
  - **Works on any mainstream browser** (IE 6+, Firefox 2+, Chrome, Safari, Edge, others)
  - **Comprehensive suite of specifications** (developed BDD-style) means its correct functioning can easily be verified on new browsers and platforms

M

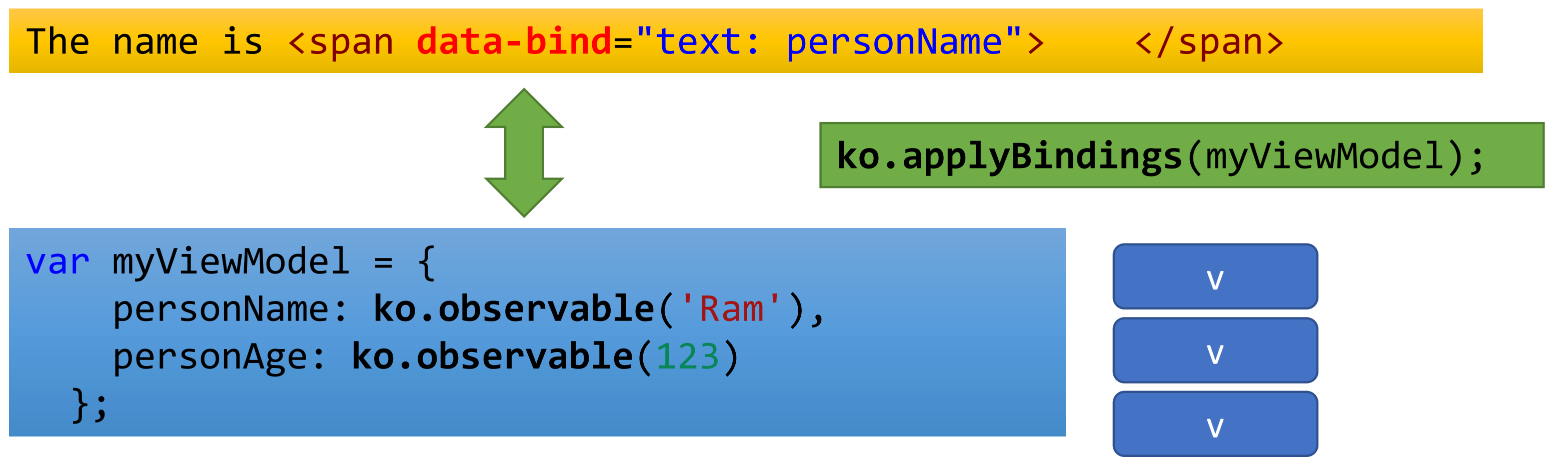
MVVM Architecture Pattern

- A **model**: your application's stored data.
- A **view model**: a pure-code representation of the data and operations on a UI.
- A **view**: a visible, interactive UI representing the state of the view model.



Observables

- Variables that can be observed
- So that rest of the code is notified of the changes in the observed variable
- Achieved through data binding



```
The name is <span data-bind="text: personName">    </span>
```

```
ko.applyBindings(myViewModel);
```

```
var myViewModel = {
    personName: ko.observable('Ram'),
    personAge: ko.observable(123)
  };
```

Accessing the Observables

```
// Reading and writing to an observable
this.x = this.lastName();
this.lastName(x)
```

Computed Observables

- These are functions that are dependent on one or more other observables, and will **automatically update** whenever any of these dependencies change.

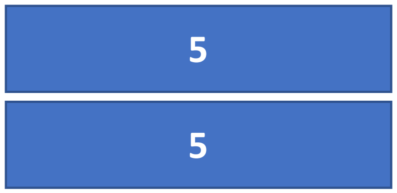
```
function AppViewModel() {
    var self = this;

    self.firstName = ko.observable('Ram');
    self.lastName = ko.observable('Kumar');
    self.fullName = ko.computed(function() {
        return self.firstName() + " " + self.lastName();
    }, this);
}
```

Observable Arrays

- If you want to detect and respond to changes on one object, you'd use **observables**. If you want to detect and respond to changes of a collection of things, use an **observableArray**.
- This is useful in many scenarios where you're displaying or editing multiple values and need repeated sections of UI to appear and disappear as items are added and removed.

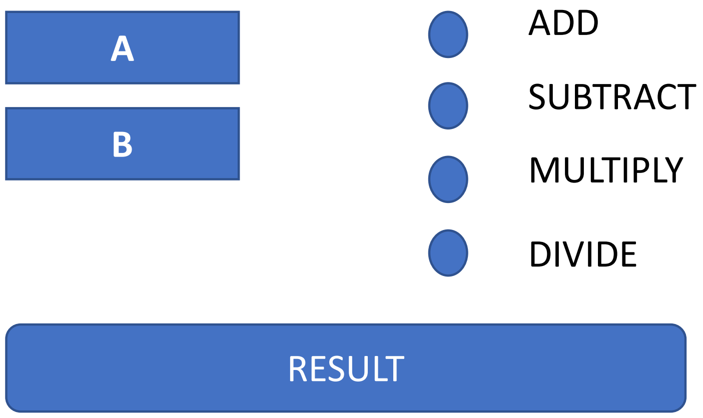
```
var arr = ko.observableArray([
    { name: "Bungle", type: "Bear" },
    { name: "George", type: "Hippo" },
    { name: "Zippy", type: "Unknown" }
]);
```

Knockout Lab

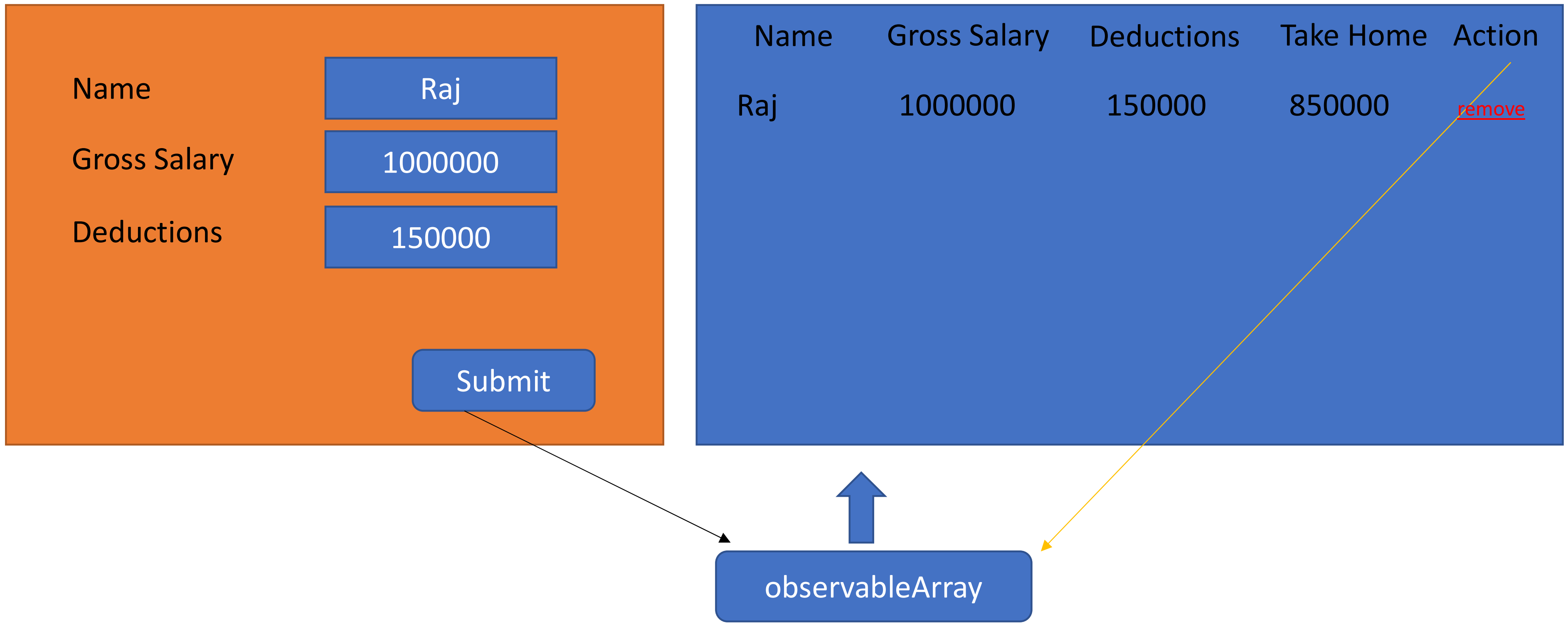


The sum of squares of two numbers is **50**

Creative Lab



Creative Lab


Name
Raj
Gross Salary
1000000
Deductions
150000
Submit
Name
Gross Salary
Deductions
Take Home
Action
Raj
1000000
150000
850000
remove
observableArray