Works with IE6+, Firefox2+, Safari 3.2+, Chrome 3+, Opera 10+

Require.js Ground Up!

Asynchronous Module Dependency Injector

Dr. SYED AWASE KHIRNI

Encapsulation

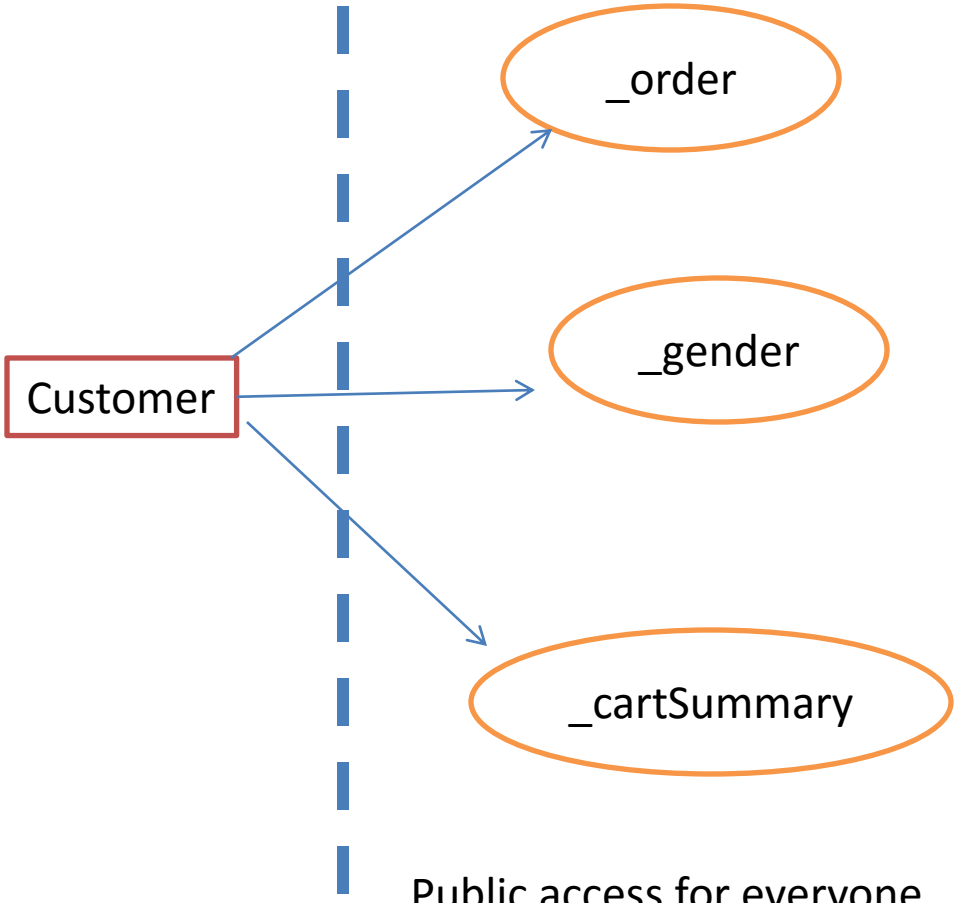
A language mechanism for restricting access to some of the Object's components

Protect variable from **unwanted changes**

JavaScript is prototype-based

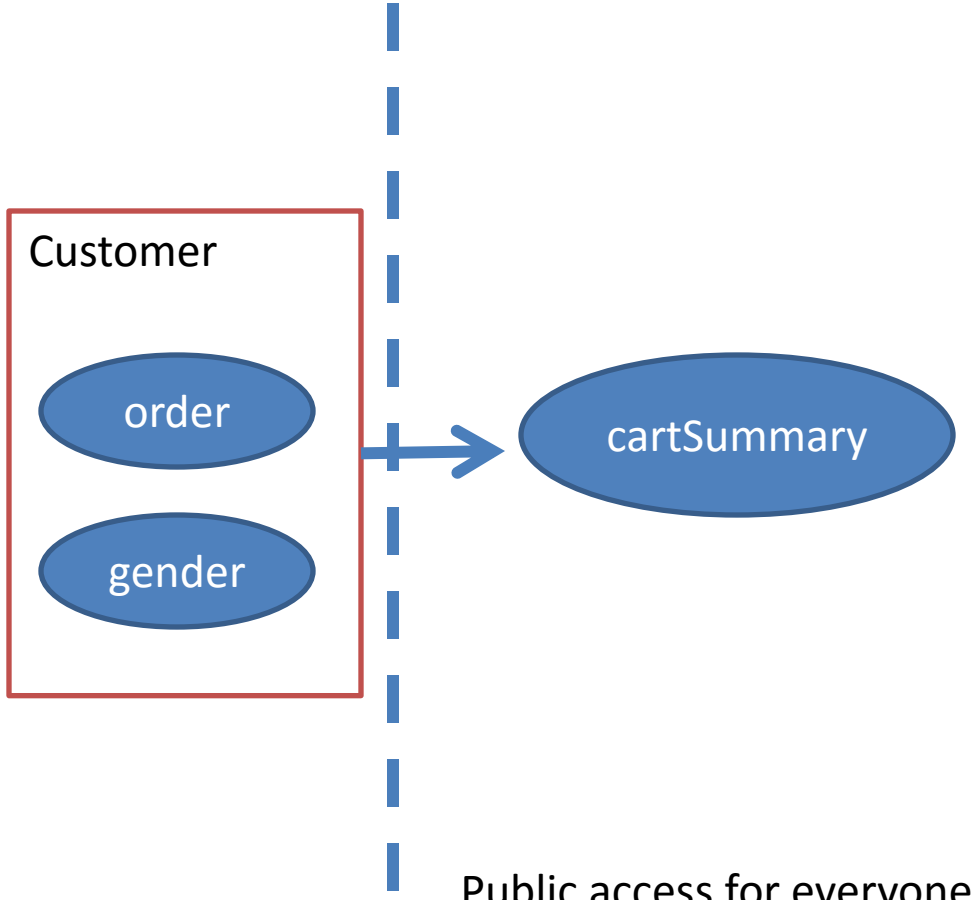
Convention in JavaScript

```
function Customer(gender){
// Private
this._order=1;
this._gender = gender || 'Male';
//public
this.cartSummary= function(){
    this._order++;
    };
}
```



Privileged Method

```
function Customer(gender){
// Private
var order =1,
    gender = gender||'Male';
//public
this.cartSummary= function(){
    order++;
    };
}
```



RequireJS

- An Asynchronous JavaScript module and file loader, optimized for web browser usage.
- Adhering to SOLID Principles – Single responsibility, the code is separated into modules, dependencies need to be configured when loading files.
- RequireJS creates code on demand and adds it to the page header as an script tag (using head.appendChild()) taking care of all its dependencies (loading them as well in the proper order).

Problem Statement

- Increase in code complexity for enterprise class applications.
- Web applications serving multi-modal devices
- Assembly gets harder
- Compliance with SOLID Principles
- Developers wants discrete JS files/modules
- Increase web application performance with fewer HTTP calls for resources using bundling and minification
- Encapsulation of data

Traditional Approach

- Slow
  - Many HTTP requests to the server for fetching resources
  - Blocking render
- Manual Dependencies
- Lack Encapsulation

Solution

- Some sort of #include/import/require for dependency injection
- Ability to load nested dependencies asynchronously
- Ease of use for developer backed by an optimization tool that helps deployment.
- Transform Websites into Web Apps
  - More
  - More Complexity
  - Ease of Development
- Adhering to Design Principles
  - Separate into reusable components
  - Avoid global namespace pollution

Module Pattern

```
var Module = (function () {
  var privateMethod = function () {};
  return {
    publicMethodOne: function () {
      // I can call `privateMethod()` you know...
    },
    publicMethodTwo: function () {
    },
    publicMethodThree: function () {
    }
  };
})();
```

Locally Scoped Object Literal

```
var Module = (function () {
  // locally scoped Object
  var myObject = {};
  // declared with `var`, must be "private"
  var privateMethod = function () {};
  myObject.someMethod = function () {
    // take it away Mr. Public Method
  };
  return myObject;
})();
```

Where do modules live?

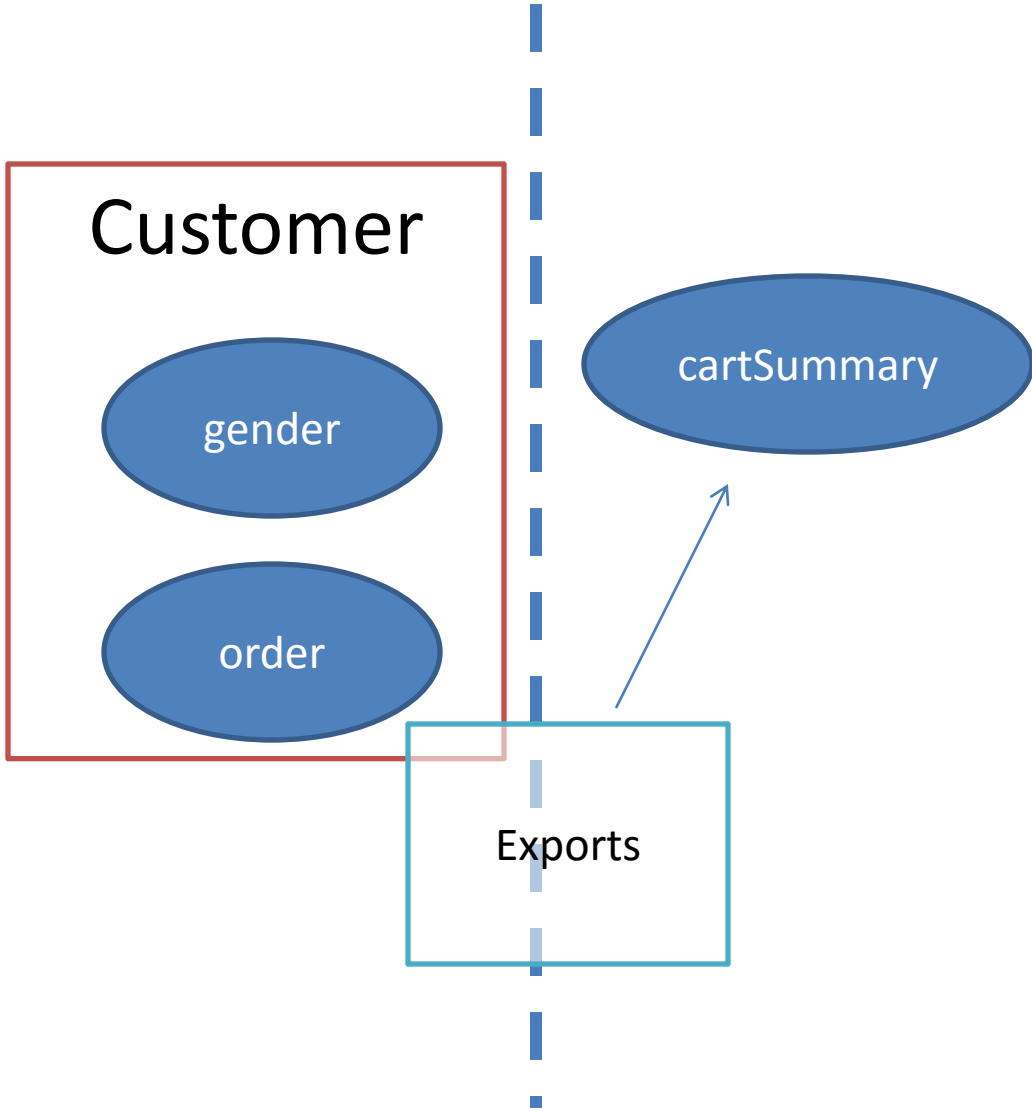
- Global? Or Local? Are they reusable?

Module is a resource

Modules may require some dependencies

Module Pattern

```
function Customer(gender){
// Private
var order =1,
    gender = gender||'Male';
//public
return {
    cartSummary: function(){
    order++;
    }
    };
}
```



Stacked locally scoped object literal

```
var Module = (function () {
  var privateMethod = function () {};
  var myObject = {
    someMethod:  function () {
    },
    anotherMethod:  function () {
    }
  };
  return myObject;
})();
```

Revealing Module Pattern

- public pointers to methods inside the Module's scope are revealed.

```
var Module = (function () {
  var privateMethod = function () {
    // private
  };
  var someMethod = function () {
    // public
  };
  var anotherMethod = function () {
    // public
  };
  return {
    someMethod: someMethod,
    anotherMethod: anotherMethod
  };
})();
```

Module Pattern

Advantage

- Easy to implement
- Keep private data safe

Disadvantage

- Hard to inherit
- Lots of modules
- Complex dependency injection
- One file per module
- Lots of files
  - Performance issue
  - Asynchronous loading required

Augmentation

- One of the limitations of the module pattern so far is that the entire module must be in **one file.**
- **In large code-base, modules are extensible components.**
- **Augment modules, provides the functionality to import the module, and add or extend the functionality and then export it.**
- **Tight Augmentation implies a set loading order, but allows overrides.**
- **Complex dependency**
  - load modules by order

```
var Module = (function () {
  var privateMethod = function () {
    // private
  };
  var someMethod = function () {
    // public
  };
  var anotherMethod = function () {
    // public
  };
  return {
    someMethod: someMethod,
    anotherMethod: anotherMethod
  };
})();
```

Tight Augmentation

Extending an existing module

```
var ModuleTwo = (function (Module) {
    Module.extension = function () {
        // another method!
    };
    return Module;
})(Module);
```

Loose Augmentation

- In the above Augmentation scenario, initial module creation to be first and then the augmnetation has to happen second in a specified order.
- Alternatively, it doesn't always happen in the same order, to improve performance, javascript allows to load scripts asynchronously.
- We can create flexibile **multi-part modules that can load themselves in any order with loose augmentation.**

```
var Module = (function () {
  var privateMethod = function () {
    // private
  };
  var someMethod = function () {
    // public
  };
  var anotherMethod = function () {
    // public
  };
  return {
    someMethod: someMethod,
    anotherMethod: anotherMethod
  };
})();
```

Extending an existing module

```
var ModuleTwo = (function (Module) {
    Module.extension = function () {
        // another method!
    };
    return Module;
})(Module || {});
```

3 RequireJS API

- Define() : define and register dependencies
  - Define () must return a **Object which is cached in a registry**
- Require() :callback function invoked when all defines() have completed
  - Acts as **intialization or root of the dependency tree**
  - **Starts the cascade of dependency checks and script loading**
  - **Starts the cascade of define() triggers**
- Config() :configure source paths, shims and aliases
  - Requirejs.config() – configuration or aliases and shims

Sub-modules

- It is to create a sub-set within an existing module.

```
Module.submoduleOne = (function () {
  var privateMethod = function () {
    // private
  };
  var someMethod = function () {
    // public
  };
  var anotherMethod = function () {
    // public
  };
  return {
    someMethod: someMethod,
    anotherMethod: anotherMethod
  };
})();
```

CommonJS

- A module ecosystem for Javascript:
  - Require() method
  - Exports object
- Created for javascript on the server
- Doesn't work well in the browser natively

Asynchronous Module Definition (AMD)

- A javascript specification that defines an API for defining code modules and their dependencies and loading them asynchronously if desired.
- Created to work directly in browsers
- It specifies a mechanism for defining modules where the module and its dependencies can be asynchronously loaded.
- RequireJS is an implementation of this specification.

AMD

- Asynchronous
  - Don't block browser
  - Parallel download and interpretation
  - Callbacks everywhere
- Loaded via script tag injection (not XHR +eval)
  - Debugging
  - X-Domain
- Each module says which modules it needs.
- The module's **"factory"** is called after all of those modules are loaded.

- Efficient Loading
  - Flexible development life cycle
  - Useful for a few or external resources
- Production/Deployment
  - Easy packaging
  - Single file or multiple packages
  - Command line tools to resolve dependencies at build time.
  - Inline text resources like template files

AMD

- Predecessors
  - DOJO
  - LABJS
  - COMMONJS
- Current
  - Jquery 1.7
  - Dojo 1.7
  - MooTools 2.0
  - EmbedJS
  - Backbone/underscore will
- AMD Drawback
  - No Standards
  - Lots of up-front work
  - No semantic versioning
  - Heavyweight tools (RequireJS)
- Alternative to AMD
  - Browserify
    - Simple syntax: require
    - Great if you are into Node+ npm
    - Intended for bundling, not so much for async module loading.

Using AMD

```
define(
module_name /*Optional*/,
[dependencies] /*Optional*/,
definition function /*function for instantiating the module or object*/
);
```

```
define('myModuleName', ['jQuery','Backbone'], function($, Backbone){
    var myModuleName= {
      //Define code here
    };
    //if anyone requires this module, they get this object
    return myModuleName;
});
```

RequireJS

- AMD Implementation by James Burke
- Asynchronous resource loader
- Call anytime
- Trace nested
- Avoid polluting the global namespace
- Optimization tool for JS/CSS

```
<!-- data-main attribute tells require.js to load
        scripts/main.js after require.js loads. -->
<script data-main="scripts/main" src="scripts/require.js"></script>
```

RequireJS (Loading Scripts)

- Creates script tags on demand and adds them to the head.

```
var head = document.getElementsByTagName('head')[0],
script=document.createElement('script');
script.src =url;
head.appendChild(script);
```

- Non-blocking page rendering approach for better web experience
- Works **after** page load
- Not constrained by same origin policy
- Need to know the dependencies upfront

RequireJS Module Pattern

- Allows the dependencies to be loaded as fast as possible, even out of order, but evaluated in the correct dependency order, and since global variables are not created. It makes it possible to load multiple versions of a module in a page.
- Dependency management
  - Dependencies can be nested
  - Load once, use in multiple locations
  - Plugins for text dependencies, load order, etc
  - Compatible with JavaScript protypal inheritance

- Extension of widely used JS module pattern
- Minimal boilerplate code for developers to adopt
- Defines a well-scoped object that avoids polluting the global namespace.
- Explicitly lists its dependencies
- Receives dependencies as arguments to the function that defines the module
- Does not need globals to refer to other modules

RequireJS Optimizer

- It includes an optimization tool to help deployment
  - Combine modules into fewer files for deployment
  - Concatenate and minify JS files
  - Shallow file exclusion for tweaking or debugging individual JS files
- Concatenate CSS@import files
- Minify CSS files
- File systems vs URLs

Using require() vs define()

Require()

- Used to run immediate functionalities
- Require calls use the requireJS baseURL(prepending it) in all cases but
  - When query string parameters are present
  - When absolute path are present
  - A protocol like http or https is contained in the url
  - When the path ends with .js (js extensions)

Define()

- Used to define modules for use in multiple locations.
- Reusable modules are usually defined
- Used for module definitions. A module can be basically any kind of object and is not available in the global namespace.
- Defining modules using requireJS is that no global objects are used.
- Modules are loaded (potentially) in different order as they are requested, but they are evaluated in the same order, so it is possible to load multiple version of the same module in the same page.

Managing the Order of Dependent Files

- RequireJS uses AMD for loading files. Each dependent module will start loading through asynchronous requests in the given order.
- Even though the file order is considered, it is not guaranteed that the first file is loaded before the second file due to the asynchronous nature.
- requireJS uses **shim** config to define the sequence of files which need to be loaded in correct order.

```
requirejs.config({
  shim: {
    'source1': ['dependency1','dependency2'],
    'source2': ['source1']
  }
});
```

Specifying Configuration

- RequireJS comes with a lot of configuration options to easily create different configurations for different situations
  - Development
  - Production/deployment
  - testing
- Yeoman generators generally set this up for us.

Configuration Options for requireJS

Option	Description
baseUrl	Base path appended to all modules loaded via requireJS, default path of the HTML page that loads requireJS, in case the “data-main” attribute is used, the default is the path of this attribute.
Paths	Path mapping for modules, relative to the baseUrl
Shim	Mechanism to support the configuration and usage of non-AMD libraries with requireJS. It defines their dependencies, exports them using their global variables or calls their no conflict functions explicitly.
Map	Different paths for some modules, useful in case the same application needs to use different versions of the same module for different purposes and needs to share their ids in order to use the same code for different situations.
Config	A way of passing configuration to the modules. This is done by using the special module ‘ module’ and its function module.config()
urlArgs	Query parameters that are added to all resources loaded via require JS. Very useful for cache busting during development, to be used with care on production
waitSeconds	Number of seconds before throwing a timeout, default 7 seconds, 0 means no time out will be thrown.

Configuration file

```
requirejs.config({
    baseUrl: "/web/app",
    paths: {
        "moduleA": "js/moduleA"
    },
    urlArgs: "version=" +  (new Date()).getTime(),
    waitSeconds: 15
});
```

```
requirejs.config({
    baseUrl:'js',
    paths: {
        angular: ['https://code.angularjs.org/1.5.6/angular.min.js','angular.min.js'],
        extCore:'ext-core.js',
        jquery: ['jquery-3.0.0'],
        mootools:'mootools.min.js',
        swfObj:'swfobject'
    }


});
```

Requirejs.config

```
requirejs.config({

    paths: {
        jquery: 'https://cdnjs.cloudflare.com/ajax/libs/jquery/2.2.3/jquery.min'
    }
});
```

```
define([],
    function () {
        var companyModule = function () {
            var _name = 'SycliQ';
            this.getName = function () {
                return _name;
            }
        }; return companyModule;
    });
```

```
define(['module/company'], function (companyref) {
    var company = new companyref();
    var _name = 'Syed Awase Khirni!';
    var employeeModule = function () {
        this.getName = function () {
            return company.getName() + ' ' + _name;
        }
    }; return employeeModule;
});
```

Requirejs.config

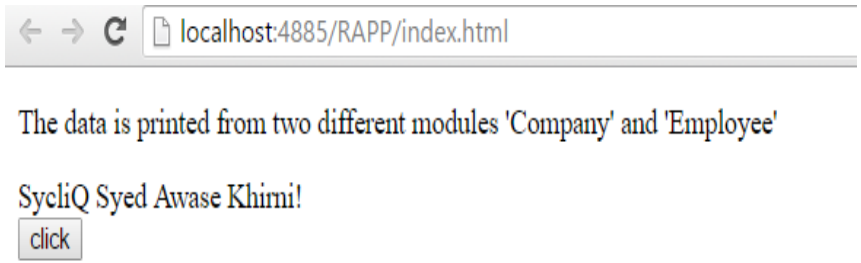
```
require( ['module/Company', 'module/Employee','jquery'],
    function(companyref, employeeref,$){
        $("#req_click").click(function(){
            alert("I'm clicked");
        });
        var company = new companyref(),
            employee = new employeeref();
        $("#result").html(employee.getName());
    });
```

View

```
<head>
    <meta charset="utf-8">
    <title>Company and Employee Application with RequireJS</title>
</head>
<body>
    <div id="container">
        <p>
           The data is printed from two different modules 'Company' and 'Employee'

        </p> <div id="result"></div>
    </div>

    <script data-main="main" src="../Scripts/require.js"></script> <input type="button"
      value="click" id="req_click" />
</body>
```



Example2

Counter.js

```
define([], function() {
    console.log("Executing counter -> Creating the counter AMD module");
      var counter = 0;

      return {
          increment: function() { counter++; },
          get: function() { return counter; }
      };
});
```

Example 2

Test1.js

```
define(['module/counter'], function(counter) {
    console.log("Executing test1 -> Incrementing the counter in module test1");
      counter.increment();
 });
```

Test2.js

```
define(['module/counter'], function(counter) {
    console.log("Executing test2-> Incrementing the counter in module test2");
      counter.increment();
  });
```

Example 2: Using RequireJs

Main.js

```
requirejs.config({

    paths: {
        jquery: 'https://cdnjs.cloudflare.com/ajax/libs/jquery/2.2.3/jquery.min'
    }
});


require(['module/test1', 'module/test2', 'module/counter'],
    function (test1, test2, counter) {
    console.log("Executing main");
    console.log(counter.get());
});
```

Example2: View

```
<!DOCTYPE html>
<html>
<head>
    <title>My Counter Demo</title>

    <script data-main="main2" src="../Scripts/require.js"></script>
</head>
<body>
    <h1>Counter Demo</h1>
</body>
```

Example 3:viewModel

```
define(['ko'], function (ko) {
    var ViewModel = function (first, last) {
        var self = this;
        self.firstName = ko.observable(first);
        self.lastName = ko.observable(last);
        self.fullName = ko.computed(function () {
            return self.firstName() + ' ' + self.lastName();
        });
    };
    return ViewModel;
});
```

Example3:requirejs.config

```
requirejs.config({
    paths: {
        jquery: 'https://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.4/jquery.min',
        ko: 'https://cdnjs.cloudflare.com/ajax/libs/knockout/3.3.0/knockout-min',
        bootstrap: 'https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.4/js/
          bootstrap.min'
    }
});

define(['jquery', 'ko', 'module/viewmodel'], function ($, ko, ViewModel) {
    $(document).ready(function () {
        ko.applyBindings(new ViewModel("Awase Khirni", "Syed"));
    });
});
```

Example3:View

```
<head>
    <title>RequireJS Example</title>

    <script src="../Scripts/require.js" data-main="main3"></script>
</head>
<body>
    <div class="panel panel-default">
        <div class="panel-heading">Test Form</div>
        <div class="panel-body">
            <input type="text" class="form-control" placeholder="First name" data-
              bind="value: firstName, valueUpdate: 'keyup'" />
            <input type="text" class="form-control" placeholder="Last name" data-
              bind="value: lastName, valueUpdate: 'keyup'" />
        </div>
        <h3><span title="Click to edit" class="label label-success">Hello, <span
          id="name" data-bind="text: fullName"></span>!</span></h3>
    </div>
</body>
```

Example 4:accountModule.js/CartModule.js

```
1  //module definition
2  define([],
3      function () {
4
5          var accountModule = function () {
6              var _id = 'UK1231241';
7              var _accountName = 'sak008';
8              this.getaccountName = function () {
9                  return _accountName;
10             }
11             this.getId = function () {
12                 return _id;
13             }
14         };
15         return accountModule;
16 });
```

```
1  define(['module/accountModule'],
2      function () {
3
4          var cartModule = function () {
5              var _productId = 'MS12321312',
6                  _productName = 'Red Tape',
7                  _price = 2900;
8              this.getproductId = function () {
9                  return _productId;
10             };
11             this.getproductName = function () {
12                 return _productName;
13             };
14             this.getprice = function () {
15                 return _price;
16             };
17         };
18         return cartModule;
19 });
```

Example4: requirejs.config

```
1  requirejs.config({
2
3      paths: {
4          jquery: 'https://cdnjs.cloudflare.com/ajax/libs/jquery/2.2.3/jquery.min'
5      }
6  });
7
8  require(['module/accountModule', 'module/cartModule', 'jquery'], function
     (accountModule, cartModule, $) {
9
10     var account = new accountModule(),
11         cart = new cartModule();
12     $("#result").html(account.getId() + " " + cart.getproductId() + " " +
         cart.getproductName() + " " + cart.getprice());
13 });
```

Example4:View

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
</head>
<body>
    <h1> Display of data from two module</h1>
    <div id="result"></div>
     <script  data-main="retailconfig"src="../Scripts/require.js"></script>
</body>
</html>
```

r.js- the RequireJS Optimizer

- Build and Deploy RequireJS Apps
  - Deploy uncompressed
  - Deploy concatenate and uglified
- Plugins available for
  - Grunt (grunt-requirejs)
  - Gulp(gulp-requirejs)
  - Broccoli (broccoli-requirejs)

Contact Us
sak@syclig.com
sak@territorialprescience.com

For code driven trainings for Technology Firms
Reach out to us +91-9035433124
We are hardcore
Technologists/Architects/Programmers
trainings are offered by Dr. Syed Awase

Thank You

We also provide Code Driven Open House Trainings

INDIA
HYDERABAD | BANGALORE | CHENNAI | PUNE

OVERSEAS
SINGAPORE | MALAYSIA | DUBAI | EUROPE | NORTH AMERICA

TERRITORIAL™ PRESCIENCE
provoke thought, invoke action

Current Offerings

- AngularJS 1.5.x
- Typescript /CoffeeScript /Dart/
- D3.JS/NVD3, HighCharts
- AngularJS 2 (with NodeJS)
- KnockOutJS (with NodeJS)
- BackBoneJS (with NodeJS)
- Ember JS / Ext JS (with NodeJS)
- Raspberry Pi
- Responsive Web Design with Bootstrap, Google Material Design and KendoUI, Zurb Foundation
- C# ASP.NET MVC , WEB API, WPF
- JAVA , SPRING, HIBERNATE
- Python , Django
- R Statistical Programming
- Android Programming
- Python/Django
- Ruby on Rails
- NoSQL (MongoDB – Essentials & Advanced)
- GIS Tools, SAFE FME, ArcGIS