Requires JavaScript dependency injection

2. Defining and Requiring modules

* **Using External module as dependency:**

**Config section**

require.config({

paths:{

jquery:"jquery-2.1.1.min"

}

});

**Require function**

Require function is used to load the initial set of modules.

Each module is identified as string

require({dependency array}, call back function);

require(["jquery"], function($){);

* **Asynchronous Module dependency(AMD)**

**Asynchronous module definition** (**AMD**) is a [JavaScript](https://en.wikipedia.org/wiki/JavaScript) specification that defines an [API](https://en.wikipedia.org/wiki/Application_programming_interface) for defining code modules and their dependencies, and loading them asynchronously if desired. Implementations of AMD provide the following benefits:

* Website performance improvements. AMD implementations load smaller JavaScript files, and only load them when they are needed.
* Fewer page errors. AMD implementations allow developers to define dependencies that must load before a module is executed, so the module does not try to use outside code that is not yet available.

AMD modules can be loaded by any AMD module loader

* Require.js
* Curl.js
* **Defining AMD module**

**Syntax**: We can use either **self executing function** or **require’s define function**

* + 1. **Self executing function**

**var myModule = (function(){..})();**

* + 1. **require’s define function**

**2.1 When define a module inline with other code**

**define({modulenameString}, {dependencyArray }, callback function );**

**define('taskData', [] , function() {} );**

**2.2 Simplified commonJS Wrapper**

**define({moduleNameString}, function(require, export, module) {**

**var {module} = require({moduleName});**

**export.{publicMember} = {publicValue};**

**})**

**Design patterns for AMD**: We can use either **Module Pattern** or **Revealing Module Pattern**

1. **Module Pattern**: define only private members at top and define only public embers in return statement.

**var myModule = (function(){**

**var iamPrivate =”private”;**

**return{**

**iAmPublic : “public”**

**}**

**})();**

1. **Revealing Module Pattern**: define all members at top and return only public methods.

**var myModule = (function(){**

**var iamPrivate =”private”;**

**var iamPublic =”public”;**

**return{**

**iAmPublic : iamPublic**

**}**

**})();**

Generally we follow the revealing Design pattern.