

# Require JS

# Introduction

- RequireJS is a JavaScript file and module loader.
- It is optimized for in-browser use, but it can be used in other JavaScript environments as well such as Node
- Using a modular script loader like RequireJS will improve the speed and quality of your code.

# Directory Structure

- **project-directory/**

- project.html

- scripts/

- main.js

- require.js

- helper/

- util.js

# Including Require JS in HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Sample Project</title>
    <!-- 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>
  </head>
  <body>
    <h1>My Sample Project</h1>
  </body>
</html>
```

The data-main attribute is a special attribute that require.js will check to start script loading

# Usage

```
require.config({
  paths: {
    foo: 'libs/foo-1.1.3'
  }
});

<script src="scripts/require.js"></script>
<script>
  require(['scripts/config'], function() {
    // Configuration loaded now, safe to do other require calls
    // that depend on that config.
    require(['foo'], function(foo) {
      });
    });
</script>
```

# Define Blocks

- A module is different from a traditional script file in that it defines a well-scoped object that avoids polluting the global namespace.
- It can explicitly list its dependencies and get a handle on those dependencies
- If the module does not have any dependencies, and it is just a collection of name/value pairs

```
define({
  color: "black",
  size: "unisize"
});
```

# Dependency Management using Define

```
//my/main.js now has some dependencies, a cart and inventory
//module in the same directory as main.js
define(["./cart", "./inventory"], function(cart, inventory) {
    //return an object to define the "my/shirt" module.
    return {
        color: "blue",
        size: "large",
        addToCart: function() {
            inventory.decrement(this);
            cart.add(this);
        }
    }
});
```

# Define Module as a Function

- Modules do not have to return objects.
- Any valid return value from a function is allowed.

```
define(["my/cart", "my/inventory"],
  function(cart, inventory) {
    //return a function to define "foo/title".
    //It gets or sets the window title.
    return function(title) {
      return title ? (window.title = title) :
        inventory.storeName + ' ' + cart.name;
    }
});

```

## require() vs define()

- The require() function is used to run immediate functionalities, while define() is used to define modules for use in multiple locations.

# Installation

- Two options
  - npm install requirejs
  - Download: <https://requirejs.org/>