Javascript Development

A module is a reusable piece of code that encapsulates implementation details and exposes a public API so it can be easily loaded and used by other code.

- **abstract code**: to delegate functionality to specialised libraries so that we don't have to understand the complexity of their actual implementation
- **encapsulate code**: to hide code inside the module if we don't want the code to be changed
- **reuse code**: to avoid writing the same code over and over again

anonymous function that is invoked when it is declared.

- **Asynchronous Module Definition (AMD)** - format is used in browsers and uses a define function to define modules.
- **CommonJS** - format is used in Node.js and uses require and module.exports to define dependencies and modules.
- **Universal Module Definition (UMD)** - format can be used both in the browser and in Node.js.
- **System.register** - was designed to support the ES6 module syntax in ES5

Definition (AMD) in RequireJS library

Exporting

```
// Calling define with a dependency array and a factory function
define(['dep1', 'dep2'], function (dep1, dep2) {
    //Define the module value by returning a value.
    return function () {};
});
```

Importing (RequireJS)

module or Node.js module

Exporting

```
var dep1 = require('./dep1');
var dep2 = require('./dep2');

module.exports = function() {
  // ...
}
```

Importing (RequireJS)

Definition (UMD)

```
(function (root, factory) {
  if (typeof define === 'function' && define.amd) {
    // AMD. Register as an anonymous module.
      define(['b'], factory);
  } else if (typeof module === 'object' && module.exports) {
    // Node. Does not work with strict CommonJS, but
    // only CommonJS-like environments that support module.exports,
    // like Node.
    module.exports = factory(require('b'));
  } else {
    // Browser globals (root is window)
    root.returnExports = factory(root.b);
  }
}(this, function (b) {
  //use b in some fashion.

  // Just return a value to define the module export.
  // This example returns an object, but the module
```

SystemJS

The module format (not supported in ES5)

Exporting

```
var s = 'local';

export function func() {
  return q;
}

export class C {
}
```

Importing