ES6 MODULES

Modules

- Modules are self contained code blocks with file scope that can control the visibility to other modules
- Modules help control the namespace avoiding namespace collisions
- In ES6 introduced modules where code can be exported and imported to make sure we don't overwrite other code

Exporting and Importing

- Support for exporting/importing values from/to modules without global namespace pollution

```
// lib/math.js
export function sum (x, y) { return x + y }
export var pi = 3.141593
// someApp.js
import * as math from "lib/math"
console.log("2π = " + math.sum(math.pi, math.pi))
// otherApp.js
import { sum, pi } from "lib/math"
console.log("2π = " + sum(pi, pi))
```

Default & Wildcard

- Marking a value as the default exported value and mass-mixin of values

```
// lib/mathplusplus.js
export * from "lib/math"
export var e = 2.71828182846
export default (x) => Math.exp(x)
// someApp.js
import exp, { pi, e } from "lib/mathplusplus"
console.log("e^{π} = " + exp(pi))
```

CLASSES

Class Definition

- Intuitive, OOP-style and boilerplate-free classes

```
class Shape {
        constructor (id, x, y) {
                this.id = id
                this.move(x, y)}
        move (x, y) {
                this.x = x
                this.y = y
        }
}
```

Class Inheritance

```
class Rectangle extends Shape {
 constructor (id, x, y, width, height) {
   super(id, x, y)  // must be first line in constructor
   this.width  = width
   this.height = height
 }}
class Circle extends Shape {
 constructor (id, x, y, radius) {
   super(id, x, y)
   this.radius = radius
 }}
```

Getter/Setter

```
class Rectangle {
 constructor (width, height) {
   this._width  = width
   this._height = height}
 set width  (width)    { this._width = width             }
 get width  ()         { return this._width               }
 set height (height)   { this._height = height            }
 get height ()         { return this._height              }
 get area   ()         { return this._width * this._height }}
var r = new Rectangle(50, 20)       r.area === 1000
```

Static Members

```
class Rectangle extends Shape {
 // ...
 static defaultRectangle () {
   return new Rectangle("default", 0, 0, 100, 100)
 }
}
var defRectangle = Rectangle.defaultRectangle()
```

Base Class Access

```
class Shape {
 toString () {
   return `Shape(${this.id})` }}
class Rectangle extends Shape {
 constructor (id, x, y, width, height) {
   super(id, x, y) }
 toString () {
   return "Rectangle > " + super.toString()}}
```