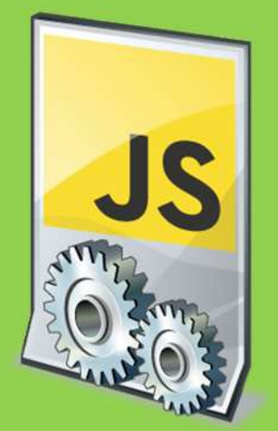
JavaScript Fundamentals With Closures and Prototype



Closures

- A function inside another function is called a closure.

```
function add(number1, number2){
  function doAdd(){
    return number1 + number2;
  }
  return doAdd();
}
add(1,1) // 2
```

- A closure has access to outer variables.
- Often callbacks are declared as closures.

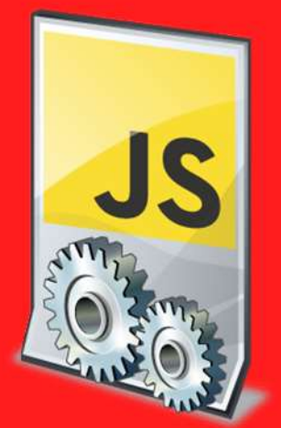
If outer variables change, your closure variables will change too.

```
function trickyAdd(number1, number2){
  function doAdd(){
    return number1 + number2;
  }
  number1 += 1;
  number2 += 2;
  return doAdd();
}
trickyAdd(1,1) // 5
```

Returning Functions

Functions can be used to create other functions.

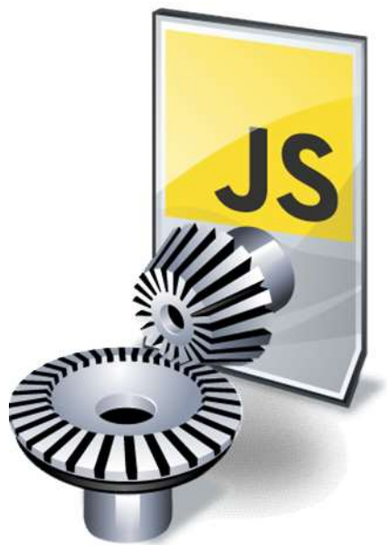
```
function createIncrementByNumber(number){
  return function(x) { return number+x; }
}

var inc = createIncrementByNumber(2);
inc(3) // 5
inc(10) //12

var inc2 = createIncrementByNumber(10);
inc2(3) // 13
inc2(10) // 20
```



Prototypes

JavaScript Prototypes

- All objects in JavaScript have a prototype property.
- You can assign an object to a prototype.
- You can chain prototypes.



Prototype Functions

You can add functions to an object by using:

```
function A(){
  this.getName = function(){ return "A"; }
};
```

With prototypes, use the following syntax:

```
function A(){};
A.prototype.getName=function(){return "A";};
```

Prototype Dynamic Object Modification

Adding properties and methods to the prototype will affect all instances.

```
function A(){};
var inst1 = new A();
var inst2 = new A();
A.prototype.getName=function(){return "A";};
console.log(inst1.getName()); // A
console.log(inst2.getName()); // A
```

Prototype Chain

```
function A(){};
A.prototype.getName = function(){
  return "proto A"
};
function B(){};
B.prototype = new A();
var b = new B();
console.log(b.getName()); // proto A
```

Where Can I Learn More?

Resource	Website
ECMAScript Language Specification 5.1 Edition	http://www.ecma-international.org/ecma-262/5.1/
Douglas Crockford's JavaScript	http://javascript.crockford.com/
Mozilla Developer Network – JavaScript reference	https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference