

---

# Object.create

---

For inheritance and for your sanity.

---

# Basic Inheritance

---

```
var mammal = {  
    speak: function() {console.log("My name is " + this.name);}  
};  
var dog = Object.create(mammal);  
mammal.speak(); // "My name is undefined"  
dog.speak(); // "My name is undefined"  
  
dog.name = "Sparky";  
dog.speak(); // "My name is Sparky"  
mammal.speak(); // "My name is undefined"
```

---

# Basic Inheritance & Initialization

---

```
function Dog(that, name) {  
    that.name = name;  
}
```

```
var dog = Object.create(mammal);  
dog.name // undefined  
Dog(dog, "Sparky");
```

```
dog.name // "Sparky"  
dog.speak(); // "My name is Sparky"  
mammal.speak(); // "My name is undefined"
```

---

# Basic Inheritance & Initialization Using Dynamic this

---

```
function Dog(name) {  
    this.name = name;  
}
```

```
var dog = Object.create(mammal);  
dog.name // undefined  
Dog.call(dog, "Sparky");
```

```
dog.name // "Sparky"  
dog.speak(); // "My name is Sparky"  
mammal.speak(); // "My name is undefined"
```

---

# Deep Inheritance

---

```
function Corgi(name) {  
    Dog.call(this, name);  
    this.breed = "Pembroke Welsh Corgi";  
}
```

```
var corgi = Object.create(dog);  
corgi.name // undefined  
corgi.breed // undefined  
Corgi.call(corgi, "Sparky");  
corgi.breed // "Pembroke Welsh Corgi"  
corgi.speak(); // "My name is Sparky"
```

---

# Two Types of Descriptors: Data & Accessor

---

Both have the following 2 shared properties:

- configurable
    - true if and only if the type of this property descriptor may be changed and if the property may be deleted from the corresponding object.
    - Defaults to false.
  - enumerable
    - true if and only if this property shows up during enumeration of the properties on the corresponding object.
    - Defaults to false.
-

# Data Descriptors

---

Must have the following 2 properties:

- value
    - The value associated with the property. Can be any valid JavaScript value (number, object, function, etc).
    - Defaults to undefined.
  - writable
    - true if and only if the value associated with the property may be changed with an assignment operator.
    - Defaults to false.
-

# Getters & Setters

---

```
var corgi = Object.create(dog, {  
  hasTail: {  
    value: true,  
    writable: true  
  }  
});  
Corgi.call(corgi, "Sparky");  
  
corgi.hasTail // true  
corgi.hasTail = false // false
```

---



# Accessor Descriptors

---

Meta Programming:

Metaprogramming is the writing of computer programs that write or manipulate other programs (or themselves) as their data, or that do part of the work at compile time that would otherwise be done at runtime.

---

# Accessor Descriptor

---

Optionaly have the following 2 properties:

- set
    - A function which serves as a setter for the property, or undefined if there is no setter. The function will receive as only argument the new value being assigned to the property.
    - Defaults to undefined.
  - get
    - A function which serves as a getter for the property, or undefined if there is no getter. The function return will be used as the value of property.
    - Defaults to undefined.
-

# Getters & Setters

---

```
var corgi = Object.create(dog, {  
  hasTail: {  
    get: function() { /* return a value or don't */},  
    set: function(value) { /* set value or don't */}  
  }  
});  
Corgi.call(corgi, "Sparky");  
  
corgi.hasTail // invokes getter  
corgi.hasTail = true // invokes setter
```

---