

Using JavaScript Constructor Functions



Jim Cooper

Software Engineer

@jimthecoop | jcoop.io



```
let Person = {  
  firstName: '',  
  lastName: '',  
  age: 18  
}
```

```
let Student = {  
  courses: [],  
  enroll(...) { ... }  
}
```

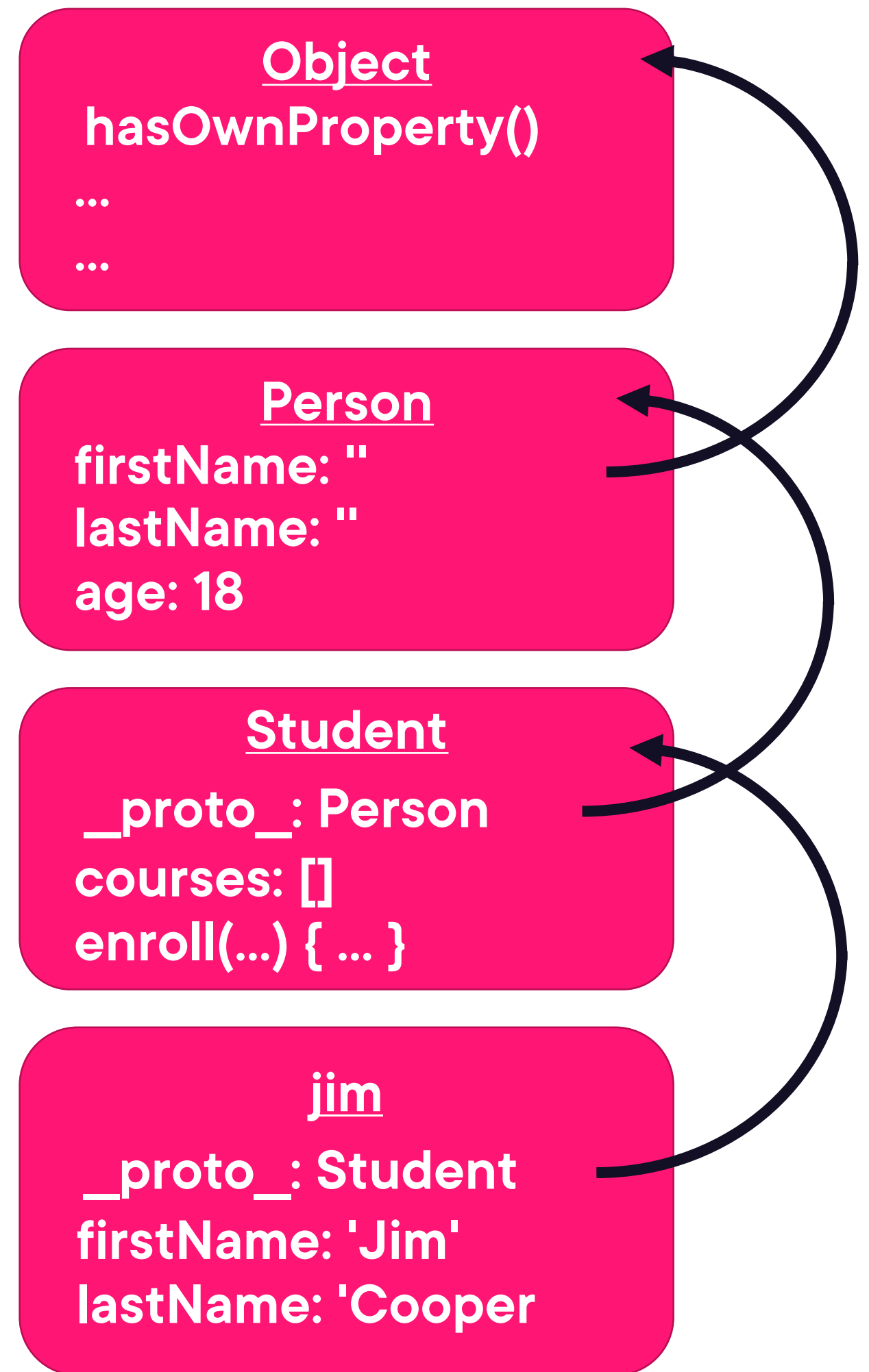
```
Object.setPrototypeOf(Student, Person);
```

```
let jim = {  
  firstName: 'Jim',  
  lastName: 'Cooper'  
}
```

```
Object.setPrototypeOf(jim, Student);
```

```
jim.age;
```

```
jim.hasOwnProperty('age'); //false
```



```
function Person(firstName, lastName, age){  
  this.firstName = firstName;  
  this.lastName = lastName;  
  this.age = age;  
  this.fullName = function() {  
    return ...;  
  }  
}
```

```
let jim = new Person('Jim', 'Cooper', 29);
```

1. Creates a new object
2. Binds `this` to the new object
3. Calls the function
4. Implicitly returns the new object

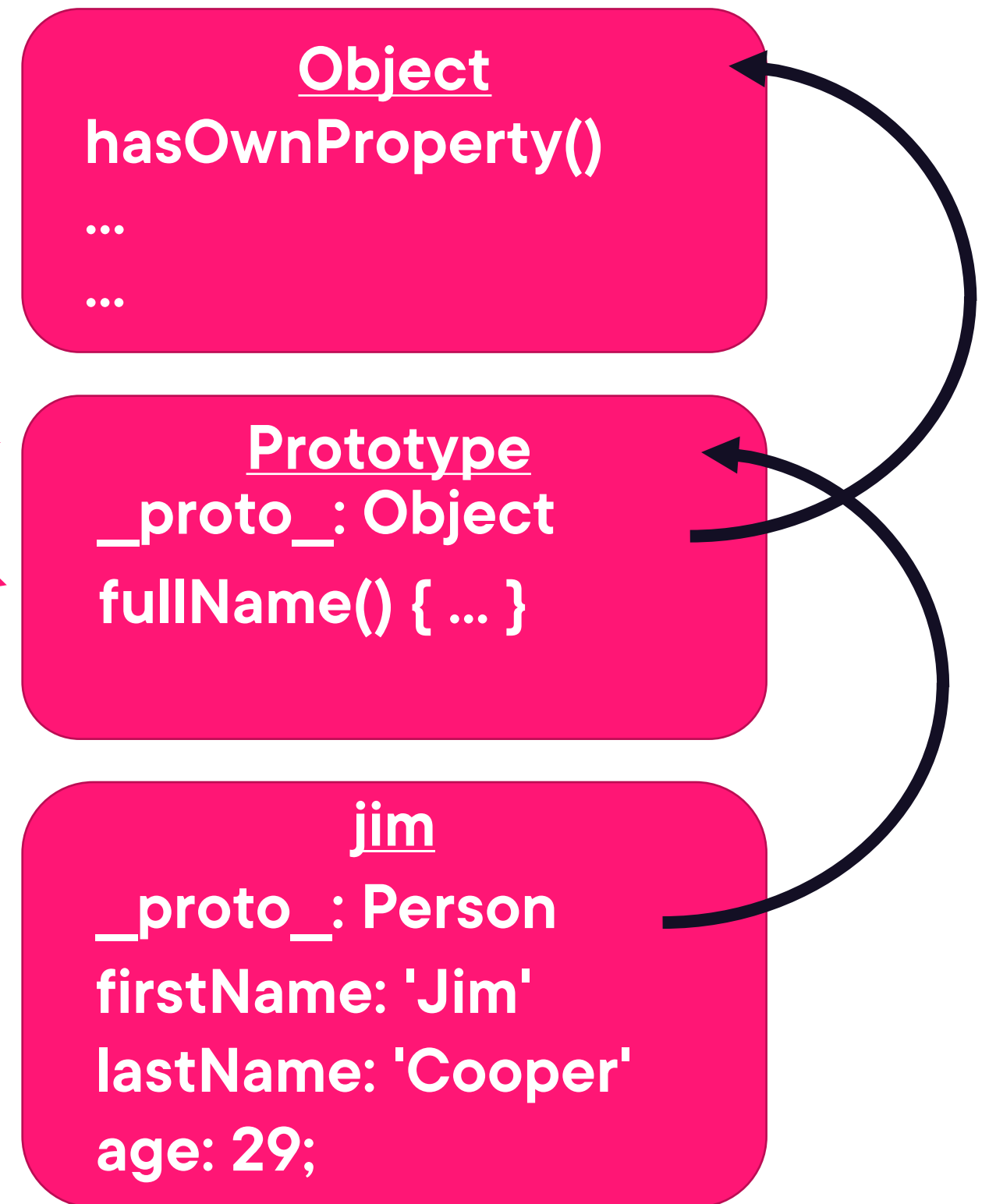
firstName: 'Jim'
lastName: 'Cooper'
age: 29
fullName: function()

```
function Person(firstName, lastName, age){  
  this.firstName = firstName;  
  this.lastName = lastName;  
  this.age = age;  
}
```

Person.prototype:

```
Person.prototype.fullName = function () {  
  ...  
};
```

```
let jim = new Person('Jim', 'Cooper', 29);
```



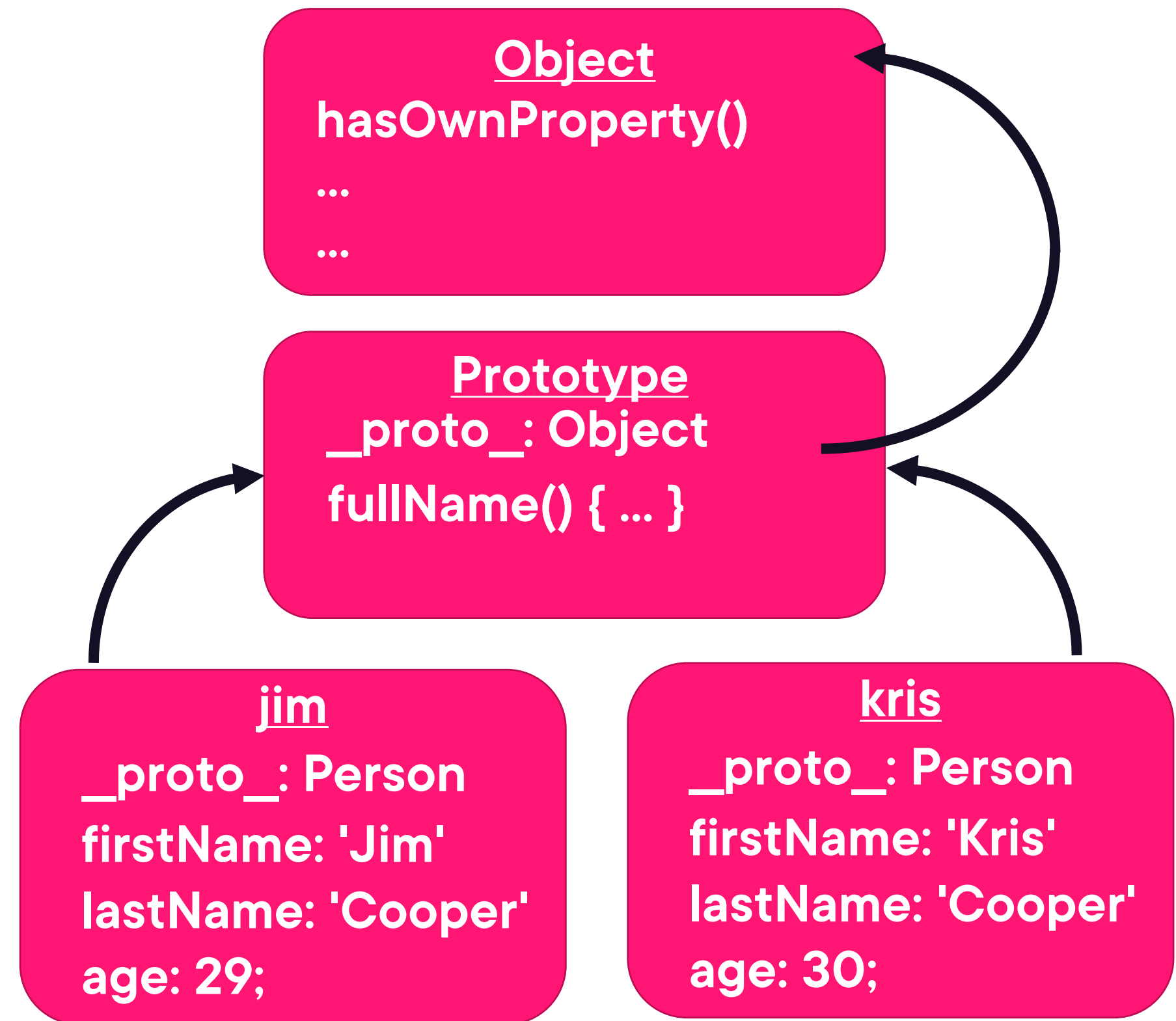
```
function Person(firstName, lastName, age){  
  this.firstName = firstName;  
  this.lastName = lastName;  
  this.age = age;  
}
```

Person.prototype:

```
Person.prototype.fullName = function () {  
  ...  
};
```

```
let jim = new Person('Jim', 'Cooper', 29);
```

```
let kris = new Person('Kris', 'Cooper', 30);
```



Inheritance with Constructor Functions

