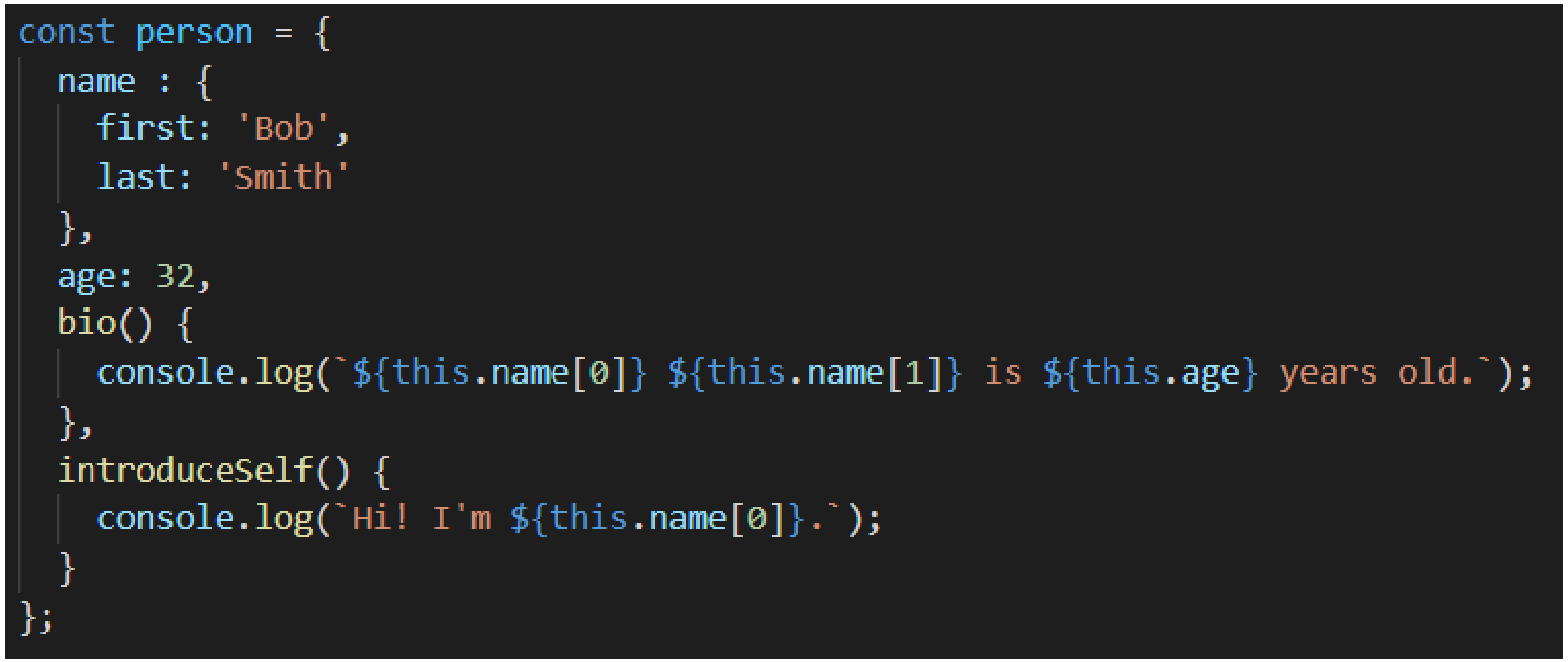
기초 웹

22 Winter CNU 기초 스터디

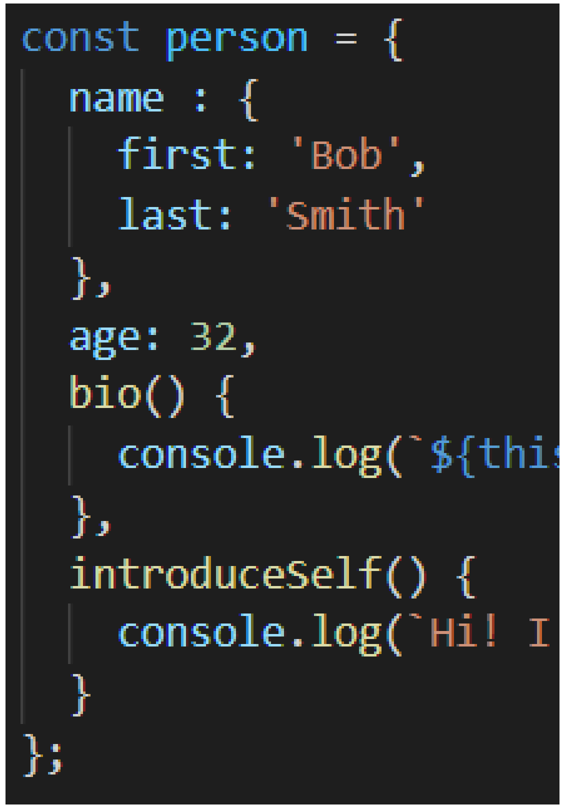
21 남정연
21 박준서

Javascript Object



```
const person = {
  name : {
    first: 'Bob',
    last: 'Smith'
  },
  age: 32,
  bio() {
    console.log(`${this.name[0]} ${this.name[1]} is ${this.age} years old.`);
  },
  introduceSelf() {
    console.log(`Hi! I'm ${this.name[0]}.`);
  }
};
```

Object Notations

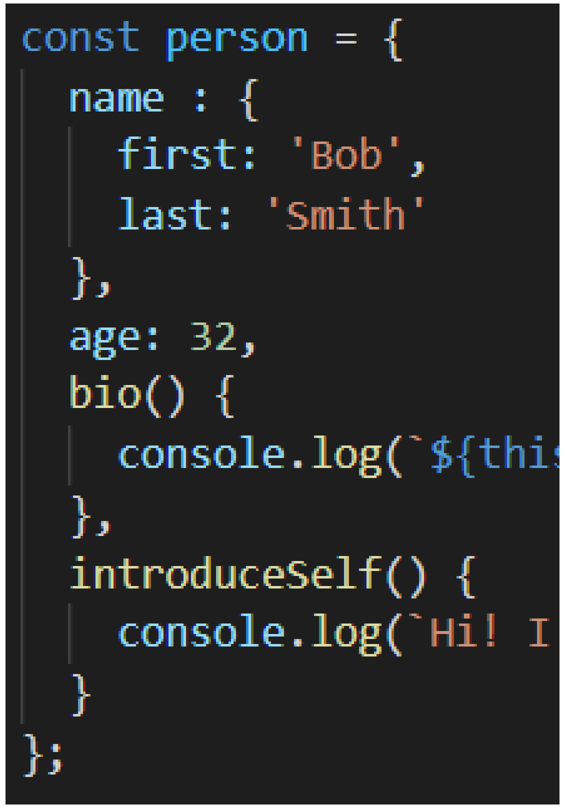


```
const person = {
  name : {
    first: 'Bob',
    last: 'Smith'
  },
  age: 32,
  bio() {
    console.log(`${thi
  },
  introduceSelf() {
    console.log(`Hi! I
  }
};
```

```
person.age
person.name.first
```

```
person['age']
person['name']['first']
```

Object Notations

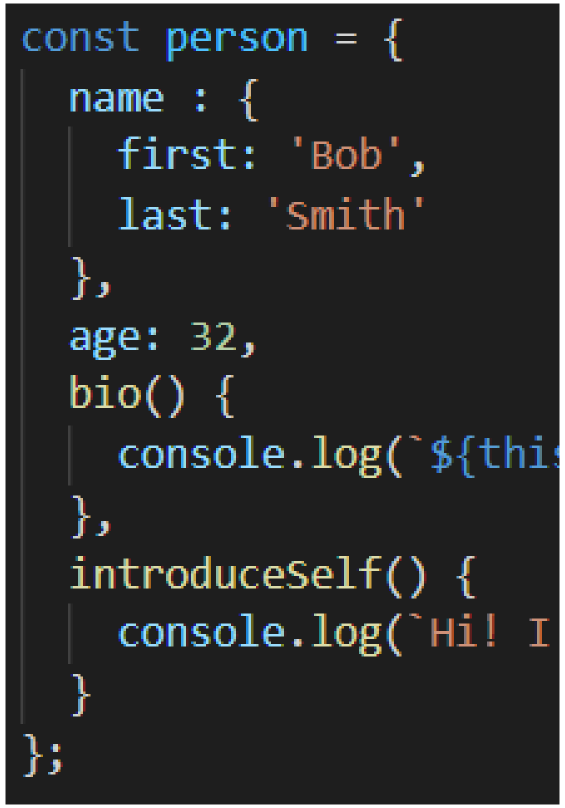


```
const person = {
  name : {
    first: 'Bob',
    last: 'Smith'
  },
  age: 32,
  bio() {
    console.log(`${thi
  },
  introduceSelf() {
    console.log(`Hi! I
  }
};
```

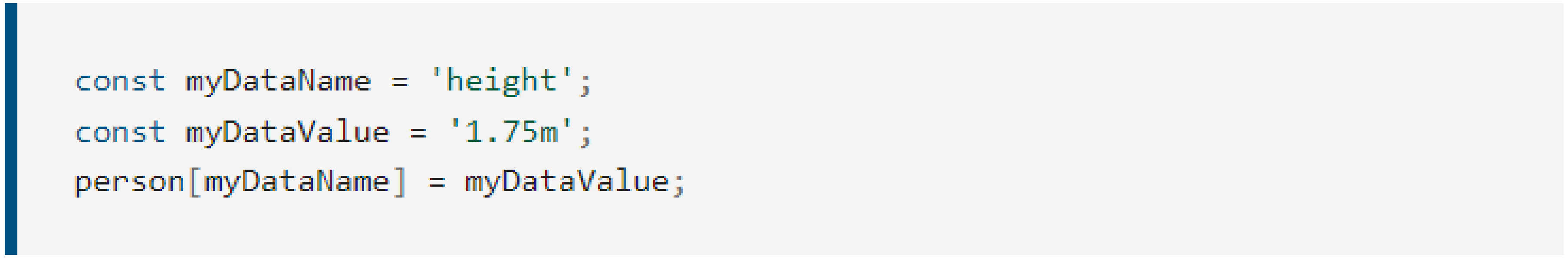
```
person.age = 45;
person['name']['last'] = 'Cratchit';
```

```
person['eyes'] = 'hazel';
person.farewell = function() {
  console.log('Bye everybody!');
}
```

Object Notations



```
const person = {
  name : {
    first: 'Bob',
    last: 'Smith'
  },
  age: 32,
  bio() {
    console.log(`${thi
  },
  introduceSelf() {
    console.log(`Hi! I
  }
};
```

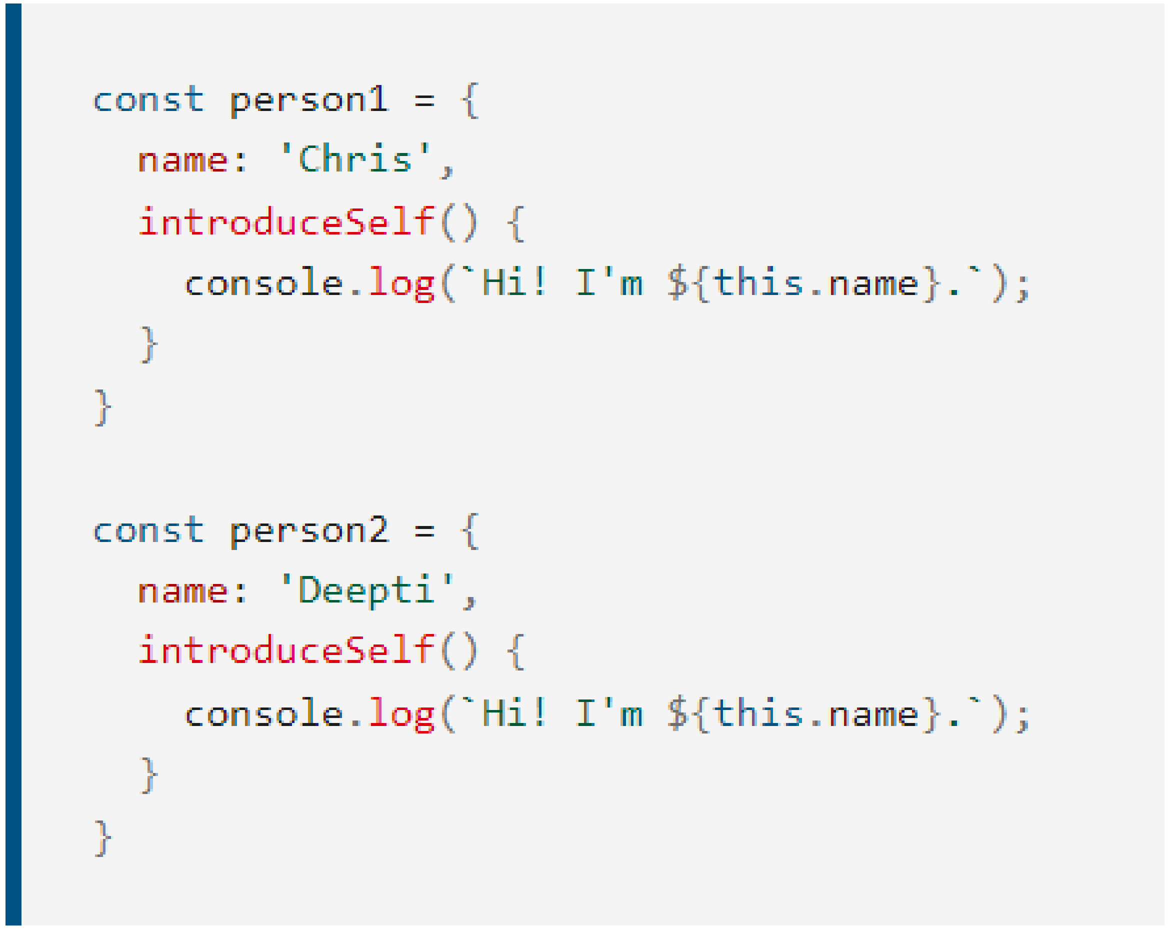


```
const myDataName = 'height';
const myDataValue = '1.75m';
person[myDataName] = myDataValue;
```

‘this’ keyword

```
introduceSelf() {
  console.log(`Hi! I'm ${this.name[0]}.`);
}
```

‘this’ keyword



```
const person1 = {
  name: 'Chris',
  introduceSelf() {
    console.log(`Hi! I'm ${this.name}.`);
  }
}

const person2 = {
  name: 'Deepti',
  introduceSelf() {
    console.log(`Hi! I'm ${this.name}.`);
  }
}
```

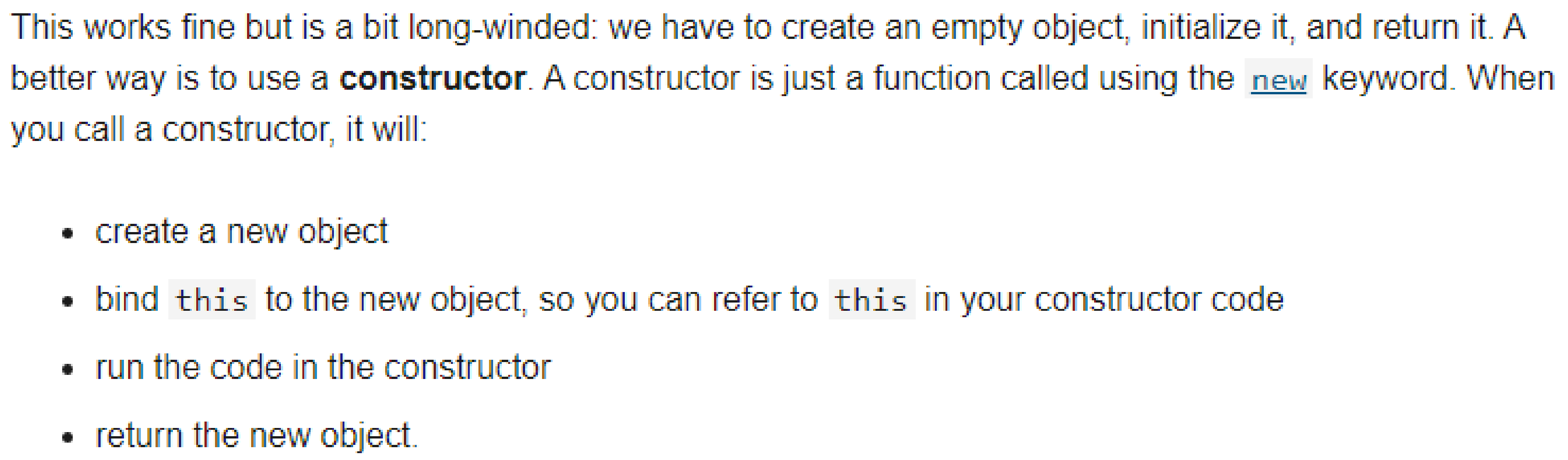
Javascript Constructor

```
function createPerson(name) {
  const obj = {};
  obj.name = name;
  obj.introduceSelf = function() {
    console.log(`Hi! I'm ${this.name}.`);
  }
  return obj;
}
```

```
const salva = createPerson('Salva');
salva.name;
salva.introduceSelf();

const frankie = createPerson('Frankie');
frankie.name;
frankie.introduceSelf();
```

Javascript Constructor



This works fine but is a bit long-winded: we have to create an empty object, initialize it, and return it. A better way is to use a **constructor**. A constructor is just a function called using the `new` keyword. When you call a constructor, it will:

- create a new object
- bind `this` to the new object, so you can refer to `this` in your constructor code
- run the code in the constructor
- return the new object.

Javascript Constructor

```
function Person(name) {
  this.name = name;
  this.introduceSelf = function() {
    console.log(`Hi! I'm ${this.name}.`);
  }
}
```

```
const salva = new Person('Salva');
salva.name;
salva.introduceSelf();

const frankie = new Person('Frankie');
frankie.name;
frankie.introduceSelf();
```

Prototype Chain

```javascript
const myObject = {
  city: 'Madrid',
  greet() {
    console.log(`Greetings from ${this.city}`);
  }
}

myObject.greet(); // Greetings from Madrid
```

```javascript
myObject.toString(); // "[object Object]"
```

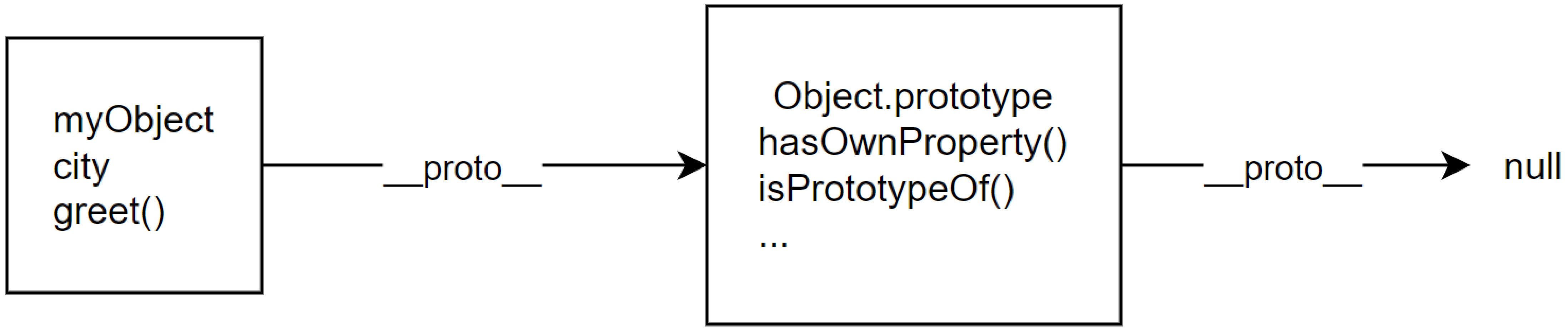
Prototype Chain

```
const myObject = {
  city: 'Madrid',
  greet() {
    console.log(`Greetings from ${this.city}`);
  }
}

myObject.greet(); // Greetings from Madrid
```

```
__defineGetter__
__defineSetter__
__lookupGetter__
__lookupSetter__
__proto__
city
constructor
greet
hasOwnProperty
isPrototypeOf
propertyIsEnumerable
toLocaleString
toString
toValueOf
```

Prototype Chain



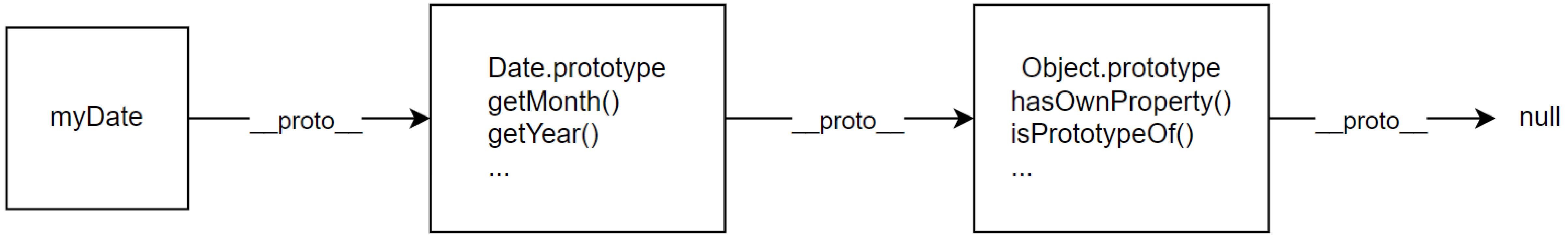
Prototype Chain

```
const myDate = new Date();
let object = myDate;

do {
  object = Object.getPrototypeOf(object);
  console.log(object);
} while (object);

// Date.prototype
// Object {...}
// null
```

Prototype Chain



Shadowing Property

```
const myDate = new Date(1995, 11, 17);

console.log(myDate.getYear()); // 95

myDate.getYear = function() {
  console.log('something else!')
};

console.log(myDate.getYear()); // 'something else!'
```

Setting a prototype

```
const personPrototype = {
  greet() {
    console.log('hello!');
  }
}

const carl = Object.create(personPrototype);
carl.greet();  // hello!
```

```
const personPrototype = {
  greet() {
    console.log(`hello, my name is ${this.name}!`);
  }
}

function Person(name) {
  this.name = name;
}

Person.prototype = personPrototype;
Person.prototype.constructor = Person;
```

Setting a prototype

```
const personPrototype = {
  greet() {
    console.log('hello!');
  }
}

const carl = Object.create(personPrototype);
carl.greet();  // hello!
```

```
const personPrototype = {
  greet() {
    console.log(`hello, my name is ${this.name}!`);
  }
}

function Person(name) {
  this.name = name;
}

Person.prototype = personPrototype;
Person.prototype.constructor = Person;
```

Setting a prototype

```
const irma = new Person('Irma');

console.log(Object.hasOwn(irma, 'name')); // true
console.log(Object.hasOwn(irma, 'greet')); // false
```

Setting a prototype

```
const irma = new Person('Irma');

console.log(Object.hasOwn(irma, 'name')); // true
console.log(Object.hasOwn(irma, 'greet')); // false
```