Breaking Javascript Objects

by Shaun Kriel

Object: "a set of **variables** and **functions** that are *grouped* together to create a model which can also take on *new names*."

- **Variables** become **properties**
- **Functions** become **methods**

Create an object

```
// Creating an object
const person = {
    name: 'Happy Gilmore',
    age: 35,
    PGA: true,
    address: {
        street: '123 Wild Drive',
        city: 'San Diego',
        state: 'CA',
    },
    hobbies: ['music', 'sports'],
};
```

Accessing an object

```
document.write(person.name);
document.write(person['age']);
document.write(person['PGA']);
document.write(person.address.street);
document.write(person.hobbies[1]);
```

BJS Objects
(.name)
Happy Gilmore

BJS Objects
(.age)
35

BJS Objects
(.PGA)
true

BJS Objects
(.address.street)
123 Wild Drive

BJS Objects
(.hobbies)
sports

you access the properties or methods of an object using dot notation or square brackets

Updating an object

```
// Updating properties
person.name = 'Johnny Cash';
person['PGA'] = false;
```

** so we have changed the property names of “name” and “PGA” to NEW property values of “Johnny Cash” and “false”

BJS Objects

Johnny Cash

BJS Objects

false

to update property values, use dot notation or square brackets.
To update property of an object but not methods, use square brackets syntax as in the example for ‘PGA”

New Property Objects

```
// Create new properties
person.hasSuperPowers = true;
console.log(person);
```

```
{name: 'Johnny Cash' , age: 35, PGA: false, address: {...}, hobbies:
Array(2), ...}
PGA: false
address: {street: '123 Wild Drive' , city: 'San Diego' , state: 'CA'}
age: 35
hasSuperPowers: true
hobbies: (2) ['music', 'sports']
name: 'Johnny Cash'
[[prototype]]: Object
```

"superpowers" has been added as a new property

property name

assign operator

person . name = 'value';

object

dot-notation

property value

take the object (person), use the dot notation(.) followed by the NEW property name then using the assign operator (=) enter the properties value

to delete a property

delete person . name;

to clear a value

person . name = ' ';

Adding a function

```
// Add functions
person.greet = function () {
    document.write(`Hello, my name is ${this.name}`)};

person.greet();
```

BJS Objects

Hello, my name is Johnny Cash

this = keyword that refers to an object.
It is not a variable

```
// Add functions
person.greet = function () {
    document.write(`Hello, my name is ${this.name}
        and I am ${this.age} years old.
        I live in ${this.address.state}`)};

person.greet();
```

BJS Objects

Hello, my name is Johnny Cash and I am 35 years old. I live in CA

As you can see, Template Literals - **`${}`** have been used in the function. Template literals are ***string literals*** that allow *embedded expressions* (**variables**) into your code. They are enclosed by **backticks (`)** instead of single (‘) or double (“) quotes.

Keys with multiple words

```
// Keys with multiple words
const person2 = {
    'first name': 'Bubba',
    'last name': 'Gum',
};
```

```
console.log(person2);
```

```
▼ {first name: 'Bubba', last name: 'Gum'} i
    first name: "Bubba"
    last name: "Gum"
  ▶ [[Prototype]]: Object
```

```
// Keys with multiple words
const person2 = {
    'first name': 'Bubba',
    'last name': 'Gum',
};
x = person2['first name'];
y = person2['last name'];

document.write(x, y);
```

BJS Objects

BubbaGum