JavaScript: Novice to Ninja Chapter 5 — Objects

https://www.sitepoint.com/premium/books/javascript-novice-to-ninja-2nd-edition/read/5

Object Literals

object: a self-contained set of related values and functions.

property: a key-value pair associated with an object.

method: a property that is also a function associated with the object.

Example:

```
> const superman = {
... name: 'Superman',
... 'real name': 'Clark Kent',
... height: 75,
... weight: 235,
... hero: true,
... villian: false,
... allies: ['Batman','Supergirl','Superboy'],
... fly() {
..... return 'Up, up and away!';
..... }
... };
undefined
>
```

'real name' could just as easily be realName. Just showing that you can use a space in the key, and how to if the need arises.

Create an object:

easy: enter a pair of curly braces

not so easy: constructor function

```
      >const superman = new Object();
```

Accessing Properties

dot notation

```
> superman.name;
'Superman'
>
```

bracket notation

```
> superman['name'];
'Superman'
>> // Also can be used with non-standard property names
undefined
> superman["real" + " " + "name"]
'Clark Kent'
> // this is why you should use standard names
```

A value can be an expression. Here is a cool example from the book:

```
> let bewitched = false;
undefined
> const donkeyDoug = { name: 'Donkey Doug', hero: bewitched ? false :
true };
undefined
> donkeyDoug.hero;
true
```

But changing the variable doesn't just change the property value.

```
> bewitched = true;
true
> donkeyDoug.hero
true
>
```

But it's easy enough to change it:

```
> donkeyDoug.hero = false;
false
> donkeyDoug.hero
false
>
```

Calling methods

dot or bracket notation: dot is easier and easier to remember

Does a Certain Method or Property exist?

```
> 'city' in superman;
false
// or
> superman.hasOwnProperty('city');
false
> superman.hasOwnProperty('name');
true
```

Finding all properties of an Object

```
> for(const key in superman) {
... console.log(key + ": " + superman[key]);
... };
name: Superman
real name: Clark Kent
height: 75
weight: 235
hero: true
villian: false
allies: Batman,Supergirl,Superboy
fly: fly() { return 'Up, up and away!' }
```

Add a property

```
> superman.city = 'Metropolis';
'Metropolis'
> for(const key in superman) {
... console.log(key + ": " + superman[key]);
... }
name: Superman
real name: Clark Kent
height: 75
weight: 235
hero: true
villian: false
allies: Batman,Supergirl,Superboy
fly: fly() { return 'Up, up and away!' }
city: Metropolis
```

Change property

```
> superman['real name'] = 'Kal El';
'Kal El'
> for(const key in superman) {
... console.log(key + ": " + superman[key]);
... }
name: Superman
real name: Kal El
height: 75
weight: 235
hero: true
villian: false
allies: Batman,Supergirl,Superboy
fly: fly() { return 'Up, up and away!' }
```

```
city: Metropolis
```

Nester Objects OK

```
const jla = {
    superman: { realName: 'Clark Kent' },
    batman: { realName: 'Bruce Wayne' },
    wonderWoman: { realName: 'Diana Prince" },
    flash: { realName: 'Barry Allen' },
    aquaman: { realName: 'Arthur Curry' },
}
jla.wonderWoman.realName
<< "Diana Prince"
```

Objects Are Copied By Reference

So if you copy an object and change the copy, you'll also change the original object

Objects can be parameters to functions

```
function greet({greeting,name,age}) {
    return `${greeting}! My name is ${name} and I am ${age} years
old.`;
}
```

More about 'this'

In the "dice" object below, 'this' refers to the "dice" object:

```
> const dice = {
... sides: 6,
... roll() {
..... return Math.floor(this.sides * Math.random()) + 1;
..... }
... };
undefined
> dice.roll
[Function: roll]
> dice.roll();
4
> dice.roll();
3
> dice.roll();
4
> dice.roll();
```

```
4
> dice.roll();
5
```

'this' refers to the object that it is within. I can be used inside methods, like above, to access the method's properties.

JSON

JSON is a string representation of the object literal notation. Differences:

1. Property names must be double-quoted
2. Permitted values are double-quoted strings, numbers, true, false, null, arrays and objects
3. Functions are not permitted values

Since JSON is a string there is a parse() method that returns a JS object:

```
> const batman = '{"name": "Batman","real name": "Bruce
Wayne","height": 74, "weight": 210, "hero": true, "villain": false,
"allies": ["Robin","Batgirl","Superman"]}'
undefined
> JSON.parse(batman);
{
  name: 'Batman',
  'real name': 'Bruce Wayne',
  height: 74,
  weight: 210,
  hero: true,
  villain: false,
  allies: [ 'Robin', 'Batgirl', 'Superman' ]
}
```

and the stringify() method does the reverse:

```
> const wonderWoman = {
...     name: 'Wonder Woman',
...     'real name': 'Diana Prince',
...     height: 72,
...     weight: 165,
...     hero: true,
...     villain: false,
...     allies: ['Wonder Girl','Donna Troy','Superman'],
...     lasso: function(){
.....         console.log('You will tell the truth!');
.....     }
... };
undefined
> JSON.stringify(wonderWoman);
```

```
'{"name":"Wonder Woman","real name":"Diana
Prince","height":72,"weight":165,"hero":true,"villain":false,"allies":[
"Wonder Girl","Donna Troy","Superman"]}'
>
```

'lasso()' is a function and not part of the JSON spec. Bye bye!

You'll really need these when working with web servers using Ajax requests, or using localStorage. More on these later.

Math object

You're good

Date object

```
> const today = new Date();
undefined
> today
2021-10-01T00:36:58.014Z
> const tomorrow = new Date(2021, 9, 1);
undefined
> tomorrow
2021-10-01T05:00:00.000Z
>
```

```
new Date(year,month,day,hour,minutes,seconds,milliseconds)
```

Remember that computer programs start counting at zero, so January is 0, February is 1, and so on up to December, which is 11.

Working with dates and time zones can be tricky. The moment.js library gives you a large number of methods that make it easier to work with dates, as well as support for multiple locales.

RegExp

There is an exercise devoted to this one.