JS Objects

Object literals: is an object that is created directly in the language by wrapping all its properties and methods in curly braces {}. They allow objects to be created quickly without the need for defining a class and provide a useful way of organizing your code without polluting the global namespace:

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

Creating Objects

```
const spiderman = {};
const spiderman = new Object(); //NOT RECOMMENDED
```

if a property key is the same as a variable name that the property value is assigned to:

```
const name = 'Iron Man'; const realName = 'Tony Stark';
// long way
const ironMan = { name: name, realName: realName }; // short ES6 way
const ironMan = { name, realName };
```

Accessing properties of classes

dot notation = `superman.name`

bracket notation = `superman['name']`

Calling Methods

dot notation = `superman.fly()`

bracket notation = `superman['fly']()`

Checking if properties exist in Object

Property = `'city' in superman;` or `superman.city !== undefined;` or `superman.hasOwnProperty('city');`

Retrieving all properties in Object

You create a key/or value variable(s) to temporarily store property key name/value(s).

for-in method: - **!! This method will retrieve inherited properties !!** You iterate over object.

```
for(const key in superman) {
console.log(key + ": " + superman[key]); }
```

!! To avoid retrieving inherited properties: !!

```
for(const key in superman) {
        if(superman.hasOwnProperty(key)){
            console.log(key + ": " + superman[key]); } }
```

for-of Object.keys() method – This returns an array of all keys in an object. You access keys by iterating over array.

```
for(const key of Object.keys(superman)) {
        console.log(key); }
```

for-of Object.values() method – This returns an array of all values in an object. You access values by iterating over array.

```
for(const value of Object.values (superman)) {
        console.log(key); }
```

for-of Object.entries() method – This returns an array of all key-value pairs in an object. You access key-values by iterating over array and destructuring.

```
for(const [key,value] of Object. entries (superman)) {
        console.log(`${key}: ${value}`); }
```

Adding Properties	**Changing Properties**	**Deleting Properties**
superman.city = 'Metropolis';	superman.city = 'Boston';	delete superman.city

Nested Objects

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

Accessing Nested Objects

dot notation = justiceLeague.wonderWoman.realName

bracket notation = justiceLeague ['flash']['realName']

mixed notation = justiceLeague.batman['realName'] and justiceLeague['superman'].realName

this

"*how to refer to the object within..*"

```
const dice = {
    sides: 6,
    roll() {return Math.floor(this.sides * Math.random()) + 1;} }
dice.roll();
```

Namespacing

Namespacing is separating your variables/functions into an object to avoid clashes with other similarly named variables functions that might exist in libraries, apis, or coding in teams.

Built-in Objects

Math Object

Date Object

RegExp Object

JSON