W03 - Objects

Object literal - created directly in the language by wrapping all properties and methods in curly brackets.

description of Man of Steel as an object literal

```
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!';
  }
};
```

Properties are key-value pairs separated by comma

fly() property is a method - additional methods would follow comma-separated

- objects are mutable, can be changed while program is running.

```
const name = 'Iron Man';
const real_name = 'Tony Stark';
// Long way
const ironMan = { name: name, realname: real_name };
// short ES6 way
const ironMan = { name, real-name };
```

Accessing Properties - dot notation

Superman.name ← returns 'Superman'

- bracket notation

Superman['name'] ← advantages: can access non-standard property and method names, can evaluate an expression and use as key

W03 - objects

Computed Properties

```
Const bewitched = true;                    Values can be an expression
const captainBritain = { name: 'Captain Britain',
    hero: bewitched ? false : true };
captainBritain ← { name: 'Captain Britain', hero: false }
```

Symbol data type can be used as a computed property key

```
const name = Symbol('name');
Const supergirl = {[name]: 'Supergirl'} ← access using bracket notation

const realName = Symbol('real name');
Supergirl[realName] = 'Kara Danvers';
```

} new property added to object using Symbol

Symbols can be reused by any other object

Check if a property or method exists — in operator

```
'city' in Superman; ← returns false
superman.city !== undefined; ← returns false
Superman.hasOwnProperty('city'); ← returns false
superman.hasOwnProperty('name'); ← returns true
```

Loop through using for in loop to log all properties of an object

Adding Properties to an object - dot notation

```
superman.city = 'Metropolis'; ← adds key-value pair to object
```

* objects are not ordered lists

Changing Properties - use assigment operator

REMOVING PROPERTIES

```
delete superman.city; ← returns true
```

W03 - Objects

NESTED OBJECTS - access using dot or bracket notation. can mix

* Objects are copied by reference

Objects as parameters ← useful when large numbers of parameters need to be passed. Arguments can be provided by name in any order.

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

greet({greeting: 'What's up dude', age: 10, name: 'Bart'});
```

this ← key word: refers to object

Namespacing ← Naming collisions occur when same variable or function name is re-used for a different purpose.

use an object literal for groups of related functions

Built-in Objects

JSON
- parse() ← converts a string of JSON data to a JavaScript obje
- stringify() ← converts a JavaScript Object to JSON data

Math ← built in Object with several mathematical constants and methods

Date ← date and time .getFullYear();

RegExp ← Regular Expression Object