Section 9 - JavaScript Objects

Learning Outcomes

JavaScript objects allow related attributes to be grouped together and methods simplify "spaghetti code". All that and they are easy to understand.

Content Links

- YouTube: OO Programming in 7 minutes - https://www.youtube.com/watch?v=pTB0EiLXUC8
- JavaScript Objects - https://www.w3schools.com/js/js_object_definition.asp
- Object Properties - https://www.w3schools.com/js/js_object_properties.asp
- Object Methods - https://www.w3schools.com/js/js_object_methods.asp

JavaScript Objects

See variables from Section 2.

If you understand objects, you understand JavaScript. In JavaScript, almost "everything" is an object. All JavaScript values, except primitives, are objects.

- Booleans can be objects (if defined with the new keyword)
- Numbers can be objects (if defined with the new keyword)
- Strings can be objects (if defined with the new keyword)
- Dates are always objects
- Maths are always objects
- Regular expressions are always objects
- Arrays are always objects
- Functions are always objects
- Objects are always objects

All JavaScript values, except primitives, are objects.

Primitives

A primitive value is a value that has no properties or methods. A primitive data type is data that has a primitive value. JavaScript defines 5 types of primitive data types:

- string
- number
- boolean
- null
- undefined

Primitive values are immutable (they are hardcoded and therefore cannot be changed).

> if `x = 3.14`, you can change the value of `x`. But you cannot change the value of `3.14`.

Value	Type	Comment
"Hello"	string	"Hello" is always "Hello"
3.14	number	3.14 is always 3.14

Value	Type	Comment
true	boolean	true is always true
false	boolean	false is always false
null	null (object)	null is always null
undefined	undefined	undefined is always undefined

Objects as Variables

JavaScript variables can contain single values:

```
let person = "John Doe";
```

JavaScript variables can also contain many values. Objects are variables too. But objects can contain many values.

Object values are written as `name : value` pairs (name and value separated by a colon). It is a common practice to declare objects with the `const` keyword.

```
const person = {
    firstName:"John",
    lastName:"Doe",
    age:50,
    eyeColor:"blue"
};
```

This example creates an empty JavaScript object, and then adds 4 properties:

```
const person = {};
person.firstName = "John";
person.lastName = "Doe";
person.age = 50;
person.eyeColor = "blue";
```

JavaScript Objects are Mutable

Objects are mutable: They are addressed by reference, not by value.

If `person` is an object, the following statement will not create a copy of `person` :

```
const x = person;  // Will not create a copy of person.
```

The object `x` is not a copy of person. It **is** `person` . Both `x` and `person` are the same object.

Any changes to `x` will also change `person` , because `x` and `person` are the same object.

```
const person = {
  firstName:"John",
  lastName:"Doe",
  age:50,
  eyeColor:"blue"
}
const x = person;
x.age = 10;      // Will change both x.age and person.age
```

Object Properties

- Properties are the most important part of any JavaScript object.
- Properties are the values associated with a JavaScript object.

- A JavaScript object is a collection of unordered properties.
- Properties can usually be changed, added, and deleted, but some are read only.

The syntax for accessing the property of an object is:

```
objectName.property       // person.age
```

or

```
objectName["property"]    // person["age"]
```

Object Methods

JavaScript methods are actions that can be performed on objects. A JavaScript method is a property containing a function definition.

```
const person = {
    firstName: "",
    lastName: "",
    fullName: function() {
        return this.lastName + ", " + this.firstName;
    }
}
person.firstName = "Dave";
person.lastName = "Jones";
console.log(person.fullName()); // writes 'Jones, Dave' to console
```

Nested Objects

Values in an object can be another object:

```
const person = {
  name:"John",
  age:30,
  cars: {
    car1:"Ford",
    car2:"BMW",
    car3:"Fiat"
  }
}
```

You can access nested objects using the dot notation or the bracket notation:

```
myObj.cars.car2;
```

Nested Arrays and Objects

Values in objects can be arrays, and values in arrays can be objects:

```
const myObj = {
  name: "John",
  age: 30,
  cars: [
    {name:"Ford", models:["Fiesta", "Focus", "Mustang"]},
    {name:"BMW", models:["320", "X3", "X5"]},
    {name:"Fiat", models:["500", "Panda"]}
  ]
}
```

To access arrays inside arrays, use a for-in loop for each array:

```
for (let i in myObj.cars) {
  x += "<h1>" + myObj.cars[i].name + "</h1>";
```

```
  for (let j in myObj.cars[i].models) {
    x += myObj.cars[i].models[j];
  }
}
```