JavaScript Objects

Topics Covered:

- JavaScript Objects
- Object Properties
- Object Methods
- Accessing object properties
- Accessing object methods
- this Keyword
- Constructors
- Object Maps

Topics in Detail:

JavaScript Objects

- **JavaScript** is an **Object-based** programming language.
- **JavaScript Objects** are a collection of key-value pairs.
- The **Key** of the property is a **string** and the **value** of the property can have **any value,** even a function.
- An **object** is a **reference data type**.
- **Objects** are the **building blocks** of JavaScript.

Example: Key name and **value are** separated by a **colon (:)**

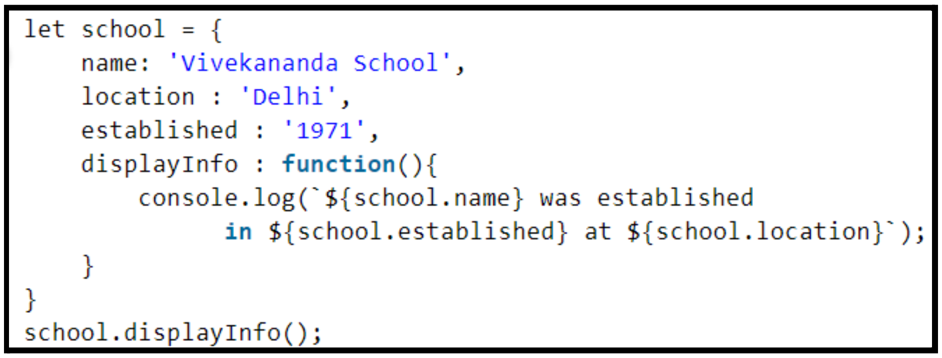
```
const person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

Object Properties

- In JavaScript, the **named variables** are called **properties**.
- Object **properties** are **variables** that are used **internally** in the **methods** of objects.
- These properties can also be **globally visible**.
- Considering the above example
    - **firstName, lastName, age, and eyeColor** are **keys.**
    - **John, Doe, 50, and blue** are **values**.
- Each of these **key-value** pairs is a **property** of the object.

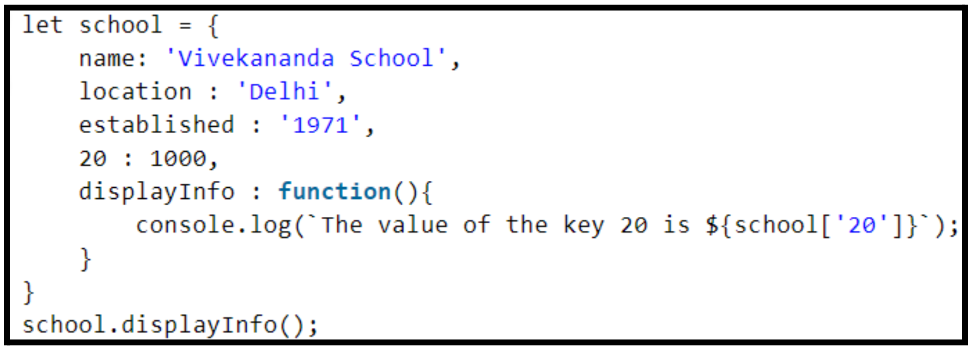
Object Methods

- The Object with the **function** as a member is known as **Object Methods**.
- Object **methods** are functions that allow **objects to do something**.
- **Methods** are always **attached** to an **object** and are **referenced** by **this** keyword.



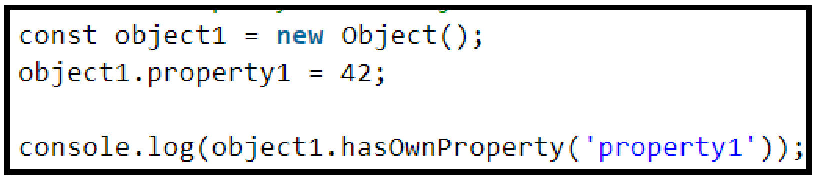
```
let school = {
    name: 'Vivekananda School',
    location : 'Delhi',
    established : '1971',
    displayInfo : function(){
        console.log(`${school.name} was established
              in ${school.established} at ${school.location}`);
    }
}
school.displayInfo();
```

- If property names are **numbers**, they can be accessed using the **bracket notation** as follows



```
let school = {
    name: 'Vivekananda School',
    location : 'Delhi',
    established : '1971',
    20 : 1000,
    displayInfo : function(){
        console.log(`The value of the key 20 is ${school['20']}`);
    }
}
school.displayInfo();
```

- Object **Properties** that are **inherited** from an **object's prototype** are known as **inherited properties** of that object. **hasOwnproperty** method can be used to check whether that property is the **object's own property**.



```
const object1 = new Object();
object1.property1 = 42;

console.log(object1.hasOwnProperty('property1'));
```

Accessing Object Properties

- Object's properties can be accessed in two ways
    - **The Dot Notation (.)**

Syntax

```
(objectName.memberName)
```

-
    - **The Array-Like Notation ([ ])**

Syntax

```
objectName["memberName"]
```

Accessing Object Methods

- Object's methods can be accessed as follows

Syntax

```
objectName.methodName()
```

- Object's method when invoked **with ( )** the **method** will be **executed**.
- Object's method, when accessed **without ( )** the **function definition,** will be **returned**.

'this' Keyword

- The **'this' keyword** refers to an object.
- The value of **this** cannot be changed.
- In the **function** definition, **this** refers to the **owner of the function**.

```
const person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```

this.firstName means the **firstName** property of a **person** object.

The ‘this’ keyword refers to different objects depending on how it is used:

Places used	Reference
object method	this refers to the object
this Keyword	this refers to the global object
function	this refers to the global object
function, in strict mode	this is undefined
Event	this refers to the element that received the event
Methods like call(), apply(), and bind()	this refers to any object

Object Constructors

- The **object constructor function** is used to create an **object type**.
- In the below example,

```
function Person(first, last, age, eye) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eye;
}
```

- **Function Person()** is an object constructor function.
- By calling the **object constructor function** with the **new keyword,** we can create the objects of the same type.

```
const myFather = new Person("John", "Doe", 50, "blue");
const myMother = new Person("Sally", "Rally", 48, "green");
```

- To add a **new property or method** to a constructor, first **add it** to the **constructor function** as follows

```
function Person(first, last, age, eye) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eye;
  this.nationality = "English";
  this.name = function() {
    return this.firstName + " " + this.lastName
  };
}
```

Built-in JavaScript Constructors

```
new String()
new Number()
new Boolean()
new Object()
new Array()
new RegExp()
new Function()
new Date()
```

Object Maps

- A Map has **key-value** pairs. The key can be of any datatype.
- Map has a property to represent the **size of the map**.

Methods of Object Map

Method	Description
new Map()	Creates a new Map object
set()	Sets the value for a key in a Map
get()	Gets the value for a key in a Map
clear()	Removes all the elements from a Map
delete()	Removes a Map element specified by a key
has()	Returns true if a key exists in a Map

Method	Description
forEach()	Invokes a callback for each key/value pair in a Map
entries()	Returns an iterator object with the [key, value] pairs in a Map
keys()	Returns an iterator object of the keys in a Map
values()	Returns an iterator object of the values in a Map

Properties of Object Map

Property	Description
size	Returns the number of Map elements