HINDUSTAN
INSTITUTE OF TECHNOLOGY & SCIENCE
(DEEMED TO BE UNIVERSITY)

CSB4301 - WEB TECHNOLOGY

B.Tech – V Semester

UNIT II

Dr. Muthukumaran M
Associate Professor
School of Computing Sciences,
Department of Computer Science and Engineering

JavaScript Objects

Real Life Objects, Properties, and Methods

In real life, a car is an object.

A car has properties like weight and color, and methods like start and stop:

	Properties	Methods
	car.name = Fiat	car.start()
	car.model = 500	car.drive()
	car.weight = 850kg	car.brake()
	car.color = white	car.stop()

All cars have the same properties, but the property values differ from car to car.

All cars have the same methods, but the methods are performed at different times.

JavaScript Objects

You have already learned that JavaScript variables are containers for data values.

This code assigns a simple value (Fiat) to a variable named car:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Variables</h2>

<p id="demo"></p>

<script>
// Create and display a variable:
let car = "Fiat";
document.getElementById("demo").innerHTML = car;
</script>

</body>
</html>
```

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

This code assigns many values (Fiat, 500, white) to a variable named car:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Objects</h2>
<p id="demo"></p>

<script>
// Create an object:
const car = {type:"Fiat", model:"500", color:"white"};

// Display some data from the object:
document.getElementById("demo").innerHTML = "The car type
is " + car.type;
</script>

</body>
</html>
```

The 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.

Object Definition

You define (and create) a JavaScript object with an object literal:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Objects</h2>
<p id="demo"></p>
<script>
// Create an object:
const person = {firstName:"John", lastName:"Doe", age:50,
eyeColor:"blue"};

// Display some data from the object:
document.getElementById("demo").innerHTML =
person.firstName + " is " + person.age + " years old.";
</script>
</body>
</html>
```

Spaces and line breaks are not important. An object definition can span multiple lines:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Objects</h2>
<p id="demo"></p>
<script>
// Create an object:
const person = {
  firstName: "John",
  lastName: "Doe",
  age: 50,
  eyeColor: "blue"
};
// Display some data from the object:
document.getElementById("demo").innerHTML =
person.firstName + " is " + person.age + " years old.";
</script>

</body>
</html>
```

Object Properties

The name:values pairs in JavaScript objects are called properties:

Property	Property Value
firstName	John
lastName	Doe
age	50
eyeColor	blue

Accessing Object Properties

You can access object properties in two ways:

```
objectName.propertyName
```

or

```
objectName["propertyName"]
```

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Objects</h2>
<p>There are two different ways to access an object property.</p>
<p>You can use person.property or person["property"].</p>
<p id="demo"></p>
<script>
// Create an object:
const person = {
  firstName: "John",
  lastName : "Doe",
  id     :  5566
};
// Display some data from the object:
document.getElementById("demo").innerHTML =
person.firstName + " " + person.lastName;
</script>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Objects</h2>

<p>There are two different ways to access an object property.</p>
<p>You can use person.property or person["property"].</p>
<p id="demo"></p>
<script>
// Create an object:
const person = {
  firstName: "John",
  lastName : "Doe",
  id     :  5566
};
// Display some data from the object:
document.getElementById("demo").innerHTML =
person["firstName"] + " " + person["lastName"];
</script>
</body>
</html>
```

Object Methods

Objects can also have methods.

Methods are actions that can be performed on objects.

Methods are stored in properties as function definitions.

Property	Property Value
firstName	John
lastName	Doe
age	50
eyeColor	blue
fullName	function() {return this.firstName + " " + this.lastName;}

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

The this Keyword

In a function definition, this refers to the "owner" of the function.

In the example above, this is the person object that "owns" the fullName function.

In other words, this.firstName means the firstName property of this object.

Accessing Object Methods

You access an object method with the following syntax:

```
objectName.methodName()
```

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Objects</h2>
<p>An object method is a function definition, stored as a property value.</p>
<p id="demo"></p>
<script>
// Create an object:
const person = {
  firstName: "John",
  lastName: "Doe",
  id: 5566,
  fullName: function() {
    return this.firstName + " " + this.lastName;
  }
};
// Display data from the object:
document.getElementById("demo").innerHTML = person.fullName();
</script>

</body>
</html>
```

If you access a method without the () parentheses, it will return the function definition:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Objects</h2>

<p>If you access an object method without (), it will return the function definition:</p>
<p id="demo"></p>
<script>
// Create an object:
const person = {
  firstName: "John",
  lastName : "Doe",
  id     : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
// Display data from the object:
document.getElementById("demo").innerHTML = person.fullName;
</script>
</body>
</html>
```

Do Not Declare Strings, Numbers, and Booleans as Objects!

When a JavaScript variable is declared with the keyword "new", the variable is created as an object:

```
x = new String();        // Declares x as a String object
y = new Number();        // Declares y as a Number object
z = new Boolean();       // Declares z as a Boolean object
```

Avoid String, Number, and Boolean objects. They complicate your code and slow down execution speed.