JAVASCRIPT PROPERTIES

CIT 260:03 – MOBILE APPLICATION DEVELOPMENT

JAVASCRIPT PROPERTIES

They 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.

ACCESSING JAVASCRIPT PROPERTIES

The syntax for accessing the property of an object is:

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

or

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

or

```
objectName[expression]       // x = "age"; person[x]
```

ACCESSING JAVASCRIPT PROPERTIES

Examples:

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

<p>There are two different ways to access an object property:</p>
<p>You can use .property or ["property"].</p>

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

<script>
var person = {
    firstname:"John",
    lastname:"Doe",
    age:50,
    eyecolor:"blue"
};
document.getElementById("demo").innerHTML =
person.firstname + " is " + person.age + " years old.";
</script>

</body>
</html>
```

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

<p>There are two different ways to access an object property:</p>
<p>You can use .property or ["property"].</p>

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

<script>
var person = {
    firstname:"John",
    lastname:"Doe",
    age:50,
    eyecolor:"blue"
};
document.getElementById("demo").innerHTML =
person["firstname"] + " is " + person["age"] + " years old.";
</script>

</body>
</html>
```

JAVASCRIPT FOR... IN LOOP

The JavaScript for... in statement loops through the properties of an object.

Syntax:

```
for (variable in object) {
    code to be executed
}
```

JAVASCRIPT FOR... IN LOOP

The block of code inside of the for... in loop will be executed once for each property.

Looping through the properties of an object.

JAVASCRIPT FOR... IN LOOP

Example:

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

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

<script>
var txt = "";
var person = {fname:"John", lname:"Doe", age:25};
var x;
for (x in person) {
    txt += person[x] + " ";
}
document.getElementById("demo").innerHTML = txt;
</script>

</body>
</html>
```

ADDING NEW PROPERTIES

We can add new properties to an existing object by simply giving it a value.

Assume that the person object already exists - we can then give it new properties.

```
person.nationality = "English";
```

ADDING NEW PROPERTIES

Example:

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

<p>You can add new properties to existing objects.</p>

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

<script>
var person = {
    firstname:"John",
    lastname:"Doe",
    age:50,
    eyecolor:"blue"
};
person.nationality = "English";
document.getElementById("demo").innerHTML =
person.firstname + " is " + person.nationality + ".";
</script>

</body>
</html>
```

DELETING PROPERTIES

The **delete** keyword deletes a property from an object.

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
delete person.age;   // or delete person["age"];
```

DELETING PROPERTIES

Example:

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

<p>You can delete object properties.</p>

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

<script>
var person = {
    firstname:"John",
    lastname:"Doe",
    age:50,
    eyecolor:"blue"
};
delete person.age;
document.getElementById("demo").innerHTML =
person.firstname + " is " + person.age + " years old.";
</script>

</body>
</html>
```

DELETING PROPERTIES

The **delete** keyword deletes both the value of the property and the property itself.

After deletion, the property cannot be used before it is added back again.

DELETING PROPERTIES

The delete operator is designed to be used on object properties. It has no effect on variables or functions.

The delete operator should not be used on predefined JavaScript object properties. It can crash your application.

PROPERTY ATTRIBUTES

All properties have a name. In addition they also have a value.

The value is one of the property's attributes.

Other attributes are: enumerable, configurable, and writable.

PROPERTY ATTRIBUTES

These attributes define how the property can be accessed (is it readable?, is it writable?)

In JavaScript, all attributes can be read, but only the value attribute can be changed (and only if the property is writable).

PROTOTYPE PROPERTIES

JavaScript objects inherit the properties of their prototype.

The **delete** keyword does not delete inherited properties, but if you delete a prototype property, it will affect all objects inherited from the prototype.

JAVASCRIPT PROPERTIES

SOURCE:

HTTP://WWW.W3SCHOOLS.COM/JS/JS_PROPERTIES.ASP