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:

```
var car = "Fiat";
```

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

```
var car = {type:"Fiat", model:"500", color:"white"};
```

The values are written as **name:value** pairs (name and value separated by a colon).

Object Properties

The name:values pairs (in JavaScript objects) are called **properties**.

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

Property	Property Value
firstName	John
lastName	Doe
age	50
eyeColor	blue

Object Definition

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

<p>Creating a JavaScript Object.</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>
```

Local JavaScript Variables

Variables declared within a JavaScript function, become **LOCAL** to the function.
Local variables have **local scope**: They can only be accessed within the function.

```
<!DOCTYPE html>
<html>
<body>
<p>The local variable carName cannot be accessed from code outside the function:</p>
<p id="demo"></p>
```

```
<script>
myFunction();
document.getElementById("demo").innerHTML =
"The type of carName is " + typeof carName;
function myFunction() {
  var carName = "Volvo";
}
</script>

</body>
</html>
```

Local JavaScript Variables

Variables declared within a JavaScript function, become **LOCAL** to the function.
Local variables have **local scope**: They can only be accessed within the function.

Example

```
// code here can not use carName

function myFunction() {
  var carName = "Volvo";

  // code here can use carName

}
```

Global JavaScript Variables

A variable declared outside a function, becomes **GLOBAL**.
A global variable has **global scope**: All scripts and functions on a web page can access it.

Example

```
var carName = " Volvo";

// code here can use carName

function myFunction() {

  // code here can use carName

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

```
<script>
var carName = "Volvo";
myFunction();

function myFunction() {
  document.getElementById("demo").innerHTML =
  "I can display " + carName;
}
```

HTML Events

An HTML event can be something the browser does, or something a user does.
Here are some examples of HTML events:

- An HTML web page has finished loading
- An HTML input field was changed
- An HTML button was clicked

Often, when events happen, you may want to do something.
JavaScript lets you execute code when events are detected.
HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.
With single quotes:
<element event='some JavaScript'>
With double quotes:
<element event="some JavaScript">
In the following example, an onclick attribute (with code), is added to a button element:

Example

```
<button onclick="document.getElementById('demo').innerHTML = Date()">The time is?</button>
```

Example

```
<button onclick="this.innerHTML = Date()">The time is?</button>
```

Example

```
<button onclick="displayDate()">The time is?</button>
```

Common HTML Events

Here is a list of some common HTML events:

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

JavaScript Regular Expressions

A regular expression is a sequence of characters that forms a search pattern.
The search pattern can be used for text search and text replace operations.

What Is a Regular Expression?

A regular expression is a sequence of characters that forms a **search pattern**. When you search for data in a text, you can use this search pattern to describe what you are searching for. A regular expression can be a single character, or a more complicated pattern. Regular expressions can be used to perform all types of **text search** and **text replace** operations.

Syntax

/pattern/modifiers;

Example

```
var patt = /w3schools/i;
```

Using String Methods

In JavaScript, regular expressions are often used with the two **string methods**: search() and replace().

The search() method uses an expression to search for a match, and returns the position of the match.

The replace() method returns a modified string where the pattern is replaced.

Using String search() With a Regular Expression

Example

Use a regular expression to do a case-insensitive search for "w3schools" in a string:

```
var str = "Visit W3Schools";
var n = str.search(/w3schools/i);
<button onclick="myFunction()">Try it</button>

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

<script>
function myFunction() {
   var str = "Visit W3Schools!";
   var n = str.search(/w3Schools/i);
   document.getElementById("demo").innerHTML = n;
}
</script>
```

Using String search() With String

The search method will also accept a string as search argument. The string argument will be converted to a regular expression:

Example

Use a string to do a search for "W3schools" in a string:

```
var str = "Visit W3Schools!";
var n = str.search("W3Schools");
```

Use String replace() With a Regular Expression

Example

Use a case insensitive regular expression to replace Microsoft with W3Schools in a string:

```
var str = "Visit Microsoft!";
var res = str.replace(/microsoft/i, "W3Schools");
<button onclick="myFunction()">Try it</button>
```

```
<p id="demo">Please visit Microsoft and Microsoft!</p>

<script>
function myFunction() {
   var str = document.getElementById("demo").innerHTML;
   var txt = str.replace(/microsoft/i,"W3Schools");
   document.getElementById("demo").innerHTML = txt;
}
</script>
```

Regular Expression Modifiers

Modifiers can be used to perform case-insensitive more global searches:

Modifier	Description
i	Perform case-insensitive matching
g	Perform a global match (find all matches rather than stopping after the first match)
m	Perform multiline matching

Regular Expression Patterns

Brackets are used to find a range of characters:

Expression	Description
[abc]	Find any of the characters between the brackets
[0-9]	Find any of the digits between the brackets
(x|y)	Find any of the alternatives separated with |

Meta characters are characters with a special meaning:

Meta character	Description
\d	Find a digit
\s	Find a whitespace character
\b	Find a match at the beginning or at the end of a word
\uxxxx	Find the Unicode character specified by the hexadecimal number xxxx

Quantifiers define quantities:

Quantifier	Description
n+	Matches any string that contains at least one *n*
n*	Matches any string that contains zero or more occurrences of *n*
n?	Matches any string that contains zero or one occurrences of *n*

Using the RegExp Object

In JavaScript, the RegExp object is a regular expression object with predefined properties and methods.

Using test()

The test() method is a RegExp expression method.
It searches a string for a pattern, and returns true or false, depending on the result.
The following example searches a string for the character "e":

Example

```
var patt = /e/;
patt.test("The best things in life are free!");
```

Since there is an "e" in the string, the output of the code above will be:
true
You don't have to put the regular expression in a variable first. The two lines above can be shortened to one:

```
/e/.test("The best things in life are free!");
```

Using exec()

The exec() method is a RegExp expression method.
It searches a string for a specified pattern, and returns the found text.
If no match is found, it returns *null.*
The following example searches a string for the character "e":

Example 1

```
/e/.exec("The best things in life are free!");
```

Since there is an "e" in the string, the output of the code above will be:
e

JavaScript Form Validation

```
<!DOCTYPE html>
<html>
<head>
<script>
function validateForm() {
  var x = document.forms["myForm"]["fname"].value;
  if (x == "") {
    alert("Name must be filled out");
    return false;
  }
}
</script>
</head>
<body>
```

```
<form name="myForm" action="/action_page_post.php"
onsubmit="return validateForm()" method="post">
Name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form>

</body>
</html>

<script type="text/javascript">
    function validateForm()
    {
        var a=document.forms["Form"]["answer_a"].value;
        var b=document.forms["Form"]["answer_b"].value;
        var c=document.forms["Form"]["answer_c"].value;
        var d=document.forms["Form"]["answer_d"].value;
        if (a==null || a=="",b==null || b=="",c==null || c=="",d==null ||
d=="")
        {
            alert("Please Fill All Required Field");
            return false;
        }
    }
</script>

<form method="post" name="Form" onsubmit="return validateForm()" action="">
     <textarea cols="30" rows="2" name="answer_a" id="a"></textarea>
     <textarea cols="30" rows="2" name="answer_b" id="b"></textarea>
     <textarea cols="30" rows="2" name="answer_c" id="c"></textarea>
     <textarea cols="30" rows="2" name="answer_d" id="d"></textarea></form>
```