🡪JavaScript is the programming language of HTML and the Web.

# Why Study JavaScript?

JavaScript is one of the 3 languages all web developers must learn:

1. HTML to define the content of web pages

2. CSS to specify the layout of web pages

3. JavaScript to program the behavior of web pages

This tutorial is about JavaScript, and how JavaScript works with HTML and CSS.

# 1)Program to display date and time.

<!DOCTYPE html>

<html>

<body>

<h1>My First JavaScript</h1>

<button type="button"

onclick="document.getElementById('demo').innerHTML = Date()">

Click me to display Date and Time.</button>

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

</body>

</html>

# 2)JavaScript Can Change HTML Content

# This example uses the method to "find" an HTML element (with id="demo") and changes the element content (innerHTML) to "Hello JavaScript".

<!DOCTYPE html>

<html>

<body>

<h1>What Can JavaScript Do?</h1>

<p id="demo">JavaScript can change HTML content.</p>

<button type="button" onclick="document.getElementById('demo').innerHTML = 'Hello JavaScript!'">Click Me!</button>

</body>

</html>

# 3.JavaScript Can Change HTML Attributes

This example changes an HTML image by changing the src (source) attribute of an <img> tag:

<!DOCTYPE html>

<html>

<body>

<h1>What Can JavaScript Do?</h1>

<p>JavaScript can change HTML attributes.</p>

<p>In this case JavaScript changes the src (source) attribute of an image.</p>

<button onclick="document.getElementById('myImage').src='pic\_bulbon.gif'">Turn on the light</button>

<img id="myImage" src="pic\_bulboff.gif" style="width:100px">

<button onclick="document.getElementById('myImage').src='pic\_bulboff.gif'">Turn off the light</button>

</body>

</html>

# 4.JavaScript Can Change HTML Styles (CSS)

Changing the style of an HTML element, is a variant of changing an HTML attribute:

<!DOCTYPE html>

<html>

<body>

<h1>What Can JavaScript Do?</h1>

<p id="demo">JavaScript can change the style of an HTML element.</p>

<button type="button" onclick="document.getElementById('demo').style.fontSize='30px'">Click Me!</button>

</body>

</html>

# 5.JavaScript Can Show HTML Elements

Showing hidden HTML elements can also be done by changing the display style:

Example:

<!DOCTYPE html>

<html>

<body>

<h1>What Can JavaScript Do?</h1>

<p>JavaScript can show hidden HTML elements.</p>

<p id="demo" style="display:none">Hello JavaScript!</p>

<button type="button" onclick="document.getElementById('demo').style.display='block'">Click Me!</button>

</body>

</html>

# JavaScript Where To

JavaScript can be placed in the <body> and the <head> sections of an HTML page.

# The <script> Tag

In HTML, JavaScript code must be inserted between <script> and </script> tags.

Example

<script>

document.getElementById("demo").innerHTML = "My First JavaScript";

</script>

# JavaScript Functions and Events

A JavaScript function is a block of JavaScript code, that can be executed when "asked" for.

For example, a function can be executed when an event occurs, like when the user clicks a button.

You will learn much more about functions and events in later chapters.

# JavaScript in <head> or <body>

You can place any number of scripts in an HTML document.

Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.

Keeping all code in one place, is always a good habit.

# JavaScript in <head>

In this example, a JavaScript function is placed in the <head> section of an HTML page.

The function is invoked (called) when a button is clicked:

Example

<!DOCTYPE html>

<html>

<head>

<script>

function myFunction() {

document.getElementById("demo").innerHTML = "Paragraph changed.";

}

</script>

</head>

<body>

<h1>My Web Page</h1>

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

<button type="button" onclick="myFunction()">Try it</button>

</body>

</html>

# External JavaScript

Scripts can also be placed in external files:

# myScript.js

function myFunction() {

document.getElementById("demo").innerHTML = "Paragraph changed.";

}

External scripts are practical when the same code is used in many different web pages.

JavaScript files have the file extension .js.

To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:

Example:

<!DOCTYPE html>

<html>

<body>

<h1>External JavaScript</h1>

<p id="demo">A Paragraph.</p>

<button type="button" onclick="myFunction()">Try it</button>

<p><strong>Note:</strong> myFunction is stored in an external file called "myScript.js".</p>

<script src="myScript.js"></script>

</body>

</html>

You can place an external script reference in <head> or <body> as you like.

The script will behave as if it was located exactly where the <script> tag is located.

External scripts cannot contain <script> tags.

# External JavaScript Advantages

Placing JavaScripts in external files has some advantages:

1.It separates HTML and code

2.It makes HTML and JavaScript easier to read and maintain

3.Cached JavaScript files can speed up page loads

# JavaScript Output

JavaScript does NOT have any built-in print or display functions.

# JavaScript Display Possibilities

JavaScript can "display" data in different ways:

1.Writing into an alert box, using window.alert().

2.Writing into the HTML output using document.write().

3.Writing into an HTML element, using innerHTML.

4.Writing into the browser console, using console.log().

# Using window.alert()

You can use an alert box to display data:

# Example:

<!DOCTYPE html>

<html>

<body>

<h1>My First Web Page</h1>

<p>My first paragraph.</p>

<script>

window.alert(5 + 75);

</script>

</body>

</html>

# Using document.write()

For testing purposes, it is convenient to use document.write():

# Example:

<!DOCTYPE html>

<html>

<body>

<h1>My First Web Page</h1>

<p>My first paragraph.</p>

<script>

document.write(5 + 6);

</script>

</body>

</html>

# Using document.write() after an HTML document is fully loaded, will delete all existing HTML:

# Example:

<!DOCTYPE html>

<html>

<body>

<h1>My First Web Page</h1>

<p>My first paragraph.</p>

<button type="button" onclick="document.write(5 + 6)">Try it</button>

</body>

</html>

The document.write() method should only be used for testing.

# Using innerHTML

To access an HTML element, JavaScript can use the document.getElementById(id) method.

The id attribute defines the HTML element. The innerHTML property defines the HTML content:

# Example:

<!DOCTYPE html>

<html>

<body>

<h1>My First Web Page</h1>

<p>My First Paragraph</p>

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

<script>

document.getElementById("demo").innerHTML = 5 + 6;

</script>

</body>

</html>

To "display data" in HTML, (in most cases) you will set the value of an innerHTML property.

Using console.log()

In your browser, you can use the console.log() method to display data.

Activate the browser console with F12, and select "Console" in the menu.

# Using console.log()

To access an HTML element, JavaScript can use the console.log()method.

# Example:

<!DOCTYPE html>

<html>

<body>

<h1>My First Web Page</h1>

<p>My first paragraph.</p>

<p>

Activate debugging in your browser (Chrome, IE, Firefox) with F12, and select "Console" in the debugger menu.

</p>

<script>

console.log(5 + 6);

</script>

</body>

</html>

# JavaScript Syntax

JavaScript syntax is the set of rules, how JavaScript programs are constructed.

# JavaScript Programs

A computer program is a list of "instructions" to be "executed" by the computer.

In a programming language, these program instructions are called statements.

JavaScript is a programming language.

JavaScript statements are separated by semicolons.

# Example

In HTML, JavaScript programs can be executed by the web browser.

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Statements</h1>

<p>Statements are separated by semicolons.</p>

<p>The variables x, y, and z are assigned the values 5, 6, and 11:</p>

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

<script>

var x = 5;

var y = 6;

var z = x + y;

document.getElementById("demo").innerHTML = z;

</script>

</body>

</html>

# JavaScript Statements

JavaScript statements are composed of:

Values, Operators, Expressions, Keywords, and Comments.

# JavaScript Values

The JavaScript syntax defines two types of values: Fixed values and variable values.

Fixed values are called literals. Variable values are called variables.

# JavaScript Literals

The most important rules for writing fixed values are:

Numbers are written with or without decimals:

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Numbers</h1>

<p>Number can be written with or without decimals.</p>

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

<script>

document.getElementById("demo").innerHTML = 10.50;

</script>

</body>

</html>

Strings are text, written within double or single quotes:

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Strings</h1>

<p>Strings can be written with double or single quotes.</p>

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

<script>

document.getElementById("demo").innerHTML = 'John Doe';

</script>

</body>

</html>

# JavaScript Variables

In a programming language, variables are used to store data values.

JavaScript uses the var keyword to declare variables.

An equal sign is used to assign values to variables.

In this example, x is defined as a variable. Then, x is assigned (given) the value 6:

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Variables</h1>

<p>In this example, x is defined as a variable.

Then, x is assigned the value of 6:</p>

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

<script>

var x;

x = 6;

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

# JavaScript Operators

JavaScript uses an assignment operator ( = ) to assign values to variables:

<!DOCTYPE html>

<html>

<body>

<h1>Assigning Values</h1>

<p>In JavaScript the = operator is used to assign values to variables.</p>

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

<script>

var x = 5;

var y = 6;

document.getElementById("demo").innerHTML = x + y;

</script>

</body>

</html>

# JavaScript Expressions

An expression is a combination of values, variables, and operators, which computes to a value.

The computation is called an evaluation.

For example, 5 \* 10 evaluates to 50:

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Expressions</h1>

<p>Expressions compute to values.</p>

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

<script>

document.getElementById("demo").innerHTML = 5 \* 10;

</script>

</body>

</html>

The values can be of various types, such as numbers and strings.

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Expressions</h1>

<p>Expressions compute to values.</p>

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

<script>

document.getElementById("demo").innerHTML = "John" + " " + "Doe";

</script>

</body>

</html>

# JavaScript Keywords

JavaScript keywords are used to identify actions to be performed.

The var keyword tells the browser to create a new variable:

<!DOCTYPE html>

<html>

<body>

<h1>The var Keyword Creates a Variable</h1>

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

<script>

var x = 5 + 6;

var y = x \* 10;

document.getElementById("demo").innerHTML = y;

</script>

</body>

</html>

# JavaScript Comments

Not all JavaScript statements are "executed".

Code after double slashes // or between /\* and \*/ is treated as a comment.

Comments are ignored, and will not be executed:

<!DOCTYPE html>

<html>

<body>

<h1>Comments are NOT Executed</h1>

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

<script>

var x = 5;

// var x = 6; I will not be executed

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

# JavaScript Identifiers

Identifiers are names.

In JavaScript, identifiers are used to name variables (and keywords, and functions, and labels).

The rules for legal names are much the same in most programming languages.

In JavaScript, the first character must be a letter, an underscore (\_), or a dollar sign ($).

Subsequent characters may be letters, digits, underscores, or dollar signs.

Numbers are not allowed as the first character.This way JavaScript can easily distinguish identifiers from numbers.

JavaScript is Case Sensitive

All JavaScript identifiers are case sensitive.

The variables lastName and lastname, are two different variables.

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript is Case Sensitive</h1>

<p>Try change lastName to lastname.</p>

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

<script>

var lastName = "Doe";

var lastname = "Peterson";

document.getElementById("demo").innerHTML = lastName;

</script>

</body>

</html>

JavaScript does not interpret VAR or Var as the keyword var.

# JavaScript and Camel Case

Historically, programmers have used three ways of joining multiple words into one variable name:

# Hyphens:

first-name, last-name, master-card, inter-city.

# Underscore:

first\_name, last\_name, master\_card, inter\_city.

# Camel Case:

FirstName, LastName, MasterCard, InterCity.

camelCase

In programming languages, especially in JavaScript, camel case often starts with a lowercase letter:

firstName, lastName, masterCard, interCity.

Hyphens are not allowed in JavaScript. It is reserved for subtractions.

# JavaScript Character Set

JavaScript uses the Unicode character set.

Unicode covers (almost) all the characters, punctuations, and symbols in the world.

# JavaScript Statements

In HTML, JavaScript statements are "instructions" to be "executed" by the web browser.

# JavaScript Statements

This statement tells the browser to write "Hello Dolly." inside an HTML element with id="demo":

# Example

<!DOCTYPE html>

<html>

<body>

<p>In HTML, JavaScript statements are "commands" to the browser.</p>

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

<script>

document.getElementById("demo").innerHTML = "Hello Dollccy.";

</script>

</body>

</html>

# JavaScript Programs

Most JavaScript programs contain many JavaScript statements.

The statements are executed, one by one, in the same order as they are written.

In this example x, y, and z are given values, and finally z is displayed:

# Example

<!DOCTYPE html>

<html>

<body>

<p>JavaScript code (or just JavaScript) is a list of JavaScript statements.</p>

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

<script>

var x = 5;

var y = 6;

var z = x + y;

document.getElementById("demo").innerHTML = z;

</script>

</body>

</html>

JavaScript programs (and JavaScript statements) are often called JavaScript code.

# Semicolons ;

Semicolons separate JavaScript statements.

Add a semicolon at the end of each executable statement:

<!DOCTYPE html>

<html>

<body>

<p>JavaScript statements are separated by semicolon.</p>

<p id="demo1"></p>

<script>

a = 11;

b = 2;

c = a + b;

document.getElementById("demo1").innerHTML = c;

</script>

</body>

</html>

When separated by semicolons, multiple statements on one line are allowed:

<!DOCTYPE html>

<html>

<body>

<p>Multiple statements on one line is allowed.</p>

<p id="demo1"></p>

<script>

a = 1; b = 2; c = a + b;

document.getElementById("demo1").innerHTML = c;

</script>

</body>

</html>

On the web, you might see examples without semicolons.

Ending statements with semicolon is not required, but highly recommended.

# JavaScript White Space

JavaScript ignores multiple spaces. You can add white space to your script to make it more readable.

The following lines are equivalent:

var person = "Hege";

var person="Hege";

A good practice is to put spaces around operators ( = + - \* / ):

var x = y + z;

# JavaScript Line Length and Line Breaks

For best readability, programmers often like to avoid code lines longer than 80 characters.

If a JavaScript statement does not fit on one line, the best place to break it, is after an operator:

# Example:

<!DOCTYPE html>

<html>

<body>

<h1>My Web Page</h1>

<p>

The best place to break a code line is after an operator or a comma.

</p>

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

<script>

document.getElementById("demo").innerHTML =

"Hello.";

</script>

</body>

</html>

Dolly

JavaScript statements can be grouped together in code blocks, inside curly brackets {...}.

The purpose of code blocks is to define statements to be executed together.

One place you will find statements grouped together in blocks, are in JavaScript functions:

# Example:

<!DOCTYPE html>

<html>

<body>

<h1>My Web Page</h1>

<p id="myPar">I am a paragraph.</p>

<p id="myDiv">I am a div.</p>

<button type="button" onclick="myFunction()">Try it</button>

<script>

function myFunction() {

document.getElementById("myPar").innerHTML = "Hello Dolly.";

document.getElementById("myDiv").innerHTML = "How are you?";

}

</script>

<p>When you click on "Try it", the two elements will change.</p>

</body>

</html>

In this tutorial we use 4 spaces of indentation for code blocks.

You will learn more about functions later in this tutorial.

# JavaScript Keywords

JavaScript statements often start with a keyword to identify the JavaScript action to be performed.

Here is a list of some of the keywords you will learn about in this tutorial:

# Keyword Description

break Terminates a switch or a loop

continue Jumps out of a loop and starts at the top

debugger Stops the execution of JavaScript, and calls (if available) the debugging function

do ... while Executes a block of statements, and repeats the block, while a condition is true

for Marks a block of statements to be executed, as long as a condition is true

function Declares a function

if ... else Marks a block of statements to be executed, depending on a condition

return Exits a function

switch Marks a block of statements to be executed, depending on different cases

try ... catch Implements error handling to a block of statements

var Declares a variable

JavaScript keywords are reserved words. Reserved words cannot be used as names for variables.

# JavaScript Comments

JavaScript comments can be used to explain JavaScript code, and to make it more readable.

JavaScript comments can also be used to prevent execution, when testing alternative code.

# Single Line Comments

Single line comments start with //.

Any text between // and the end of the line will be ignored by JavaScript (will not be executed).

This example uses a single-line comment before each code line:

# Example

// Change heading:

<!DOCTYPE html>

<html>

<body>

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

<script>

var x = 5; // Declare x, give it the value of 5

var y = x + 2; // Declare y, give it the value of x + 2

document.getElementById("demo").innerHTML = y; // Write y to demo

</script>

<p><strong>Note:</strong> The comments are not executed.</p>

</body>

</html>

This example uses a single line comment at the end of each line to explain the code:

# Multi-line Comments

Multi-line comments start with /\* and end with \*/.

Any text between /\* and \*/ will be ignored by JavaScript.

This example uses a multi-line comment (a comment block) to explain the code:

# Example

<!DOCTYPE html>

<html>

<body>

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

<script>

var x = 5; // Declare x, give it the value of 5

var y = x + 2; // Declare y, give it the value of x + 2

document.getElementById("demo").innerHTML = y; // Write y to demo

</script>

<p><strong>Note:</strong> The comments are not executed.</p>

</body>

</html>

It is most common to use single line comments.

Block comments are often used for formal documentation.

# Using Comments to Prevent Execution

Using comments to prevent execution of code is suitable for code testing.

Adding // in front of a code line changes the code lines from an executable line to a comment.

This example uses // to prevent execution of one of the code lines:

# Example

This example uses a comment block to prevent execution of multiple lines:

<!DOCTYPE html>

<html>

<body>

<h1 id="myH"></h1>

<p id="myP"></p>

<script>

/\*

The code below will change

the heading with id = "myH"

and the paragraph with id = "myp"

in my web page:

\*/

document.getElementById("myH").innerHTML = "My First Page";

document.getElementById("myP").innerHTML = "My first paragraph.";

</script>

<p><strong>Note:</strong> The comment block is not executed.</p>

<body>

</html>

# Example

<!DOCTYPE html>

<html>

<body>

<h1 id="myH"></h1>

<p id="myP"></p>

<script>

/\*

document.getElementById("myH").innerHTML = "Welcome to my Homepage";

document.getElementById("myP").innerHTML = "This is my first paragraph.";

\*/

</script>

<p><strong>Note:</strong> The comment-block is not executed.</p>

</body>

</html>

# JavaScript Variables

JavaScript variables are containers for storing data values.

In this example, x, y, and z, are variables:

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Variables</h1>

<p>In this example, x, y, and z are variables</p>

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

<script>

var x = 5;

var y = 6;

var z = x + y;

document.getElementById("demo").innerHTML = z;

</script>

</body>

</html>

# Much Like Algebra

In this example, price1, price2, and total, are variables:

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Variables</h1>

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

<script>

var price1 = 5;

var price2 = 6;

var total = price1 + price2;

document.getElementById("demo").innerHTML =

"The total is: " + total;

</script>

</body>

</html>

# JavaScript Identifiers

All JavaScript variables must be identified with unique names.

These unique names are called identifiers.

Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

The general rules for constructing names for variables (unique identifiers) are:

Names can contain letters, digits, underscores, and dollar signs.

Names must begin with a letter

Names can also begin with $ and \_ (but we will not use it in this tutorial)

Names are case sensitive (y and Y are different variables)

Reserved words (like JavaScript keywords) cannot be used as names

JavaScript identifiers are case-sensitive.

# The Assignment Operator

In JavaScript, the equal sign (=) is an "assignment" operator, not an "equal to" operator.

This is different from algebra. The following does not make sense in algebra:

x = x + 5

In JavaScript, however, it makes perfect sense: it assigns the value of x + 5 to x.

(It calculates the value of x + 5 and puts the result into x. The value of x is incremented by 5.)

The "equal to" operator is written like == in JavaScript.

# JavaScript Data Types

JavaScript variables can hold numbers like 100 and text values like "John Doe".

In programming, text values are called text strings.

JavaScript can handle many types of data, but for now, just think of numbers and strings.

Strings are written inside double or single quotes. Numbers are written without quotes.

If you put a number in quotes, it will be treated as a text string.

# Example

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Variables</h1>

<p>Strings are written with quotes.</p>

<p>Numbers are written without quotes.</p>

<p>Try to experiment by removing the // comments.</p>

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

<script>

var pi = 3.14;

var person = "John Doe";

var answer = 'Yes I am!';

document.getElementById("demo").innerHTML = pi;

//document.getElementById("demo").innerHTML = person;

//document.getElementById("demo").innerHTML = answer;

</script>

</body>

</html>

# Declaring (Creating) JavaScript Variables

Creating a variable in JavaScript is called "declaring" a variable.

You declare a JavaScript variable with the var keyword:

var carName;

After the declaration, the variable has no value. (Technically it has the value of undefined)

To assign a value to the variable, use the equal sign:

carName = "Volvo";

You can also assign a value to the variable when you declare it:

var carName = "Volvo";

In the example below, we create a variable called carName and assign the value "Volvo" to it.

Then we "output" the value inside an HTML paragraph with id="demo":

# Example

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Variables</h1>

<p>Create a variable, assign a value to it, and display it:</p>

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

<script>

var carName = "Volvo";

document.getElementById("demo").innerHTML = carName;

</script>

</body>

</html>

# JavaScript Operators

### Example

Assign values to variables and add them together:

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Operators</h1>

<p>x = 5, y = 2, calculate z = x + y, and display z:</p>

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

<script>

var x = 5;

var y = 2;

var z = x + y;

document.getElementById("demo").innerHTML = z;

</script>

</body>

</html>

# JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic on numbers (literals or variables).

# Operator Description

+ Addition

- Subtraction

\* Multiplication

/ Division

% Modulus

++ Increment

-- Decrement

The addition operator (+) adds numbers:

# Adding

<!DOCTYPE html>

<html>

<body>

<h1>The + Operator</h1>

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

<script>

var x = 5;

var y = 2;

var z = x + y;

document.getElementById("demo").innerHTML = z;

</script>

</body>

</html>

The multiplication operator (\*) multiplies numbers.

# Multiplying

<!DOCTYPE html>

<html>

<body>

<h1>The \* Operator</h1>

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

<script>

var x = 5;

var y = 2;

var z = x \* y;

document.getElementById("demo").innerHTML = z;

</script>

</body>

</html>

# JavaScript Assignment Operators

Assignment operators assign values to JavaScript variables.

# Operator Example Same As

= x = y x = y

+= x += y x = x + y

-= x -= y x = x - y

\*= x \*= y x = x \* y

/= x /= y x = x / y

%= x %= y x = x % y

The assignment operator (=) assigns a value to a variable.

# Assignment

<!DOCTYPE html>

<html>

<body>

<h1>The = Operator</h1>

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

<script>

var x = 10;

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

The addition assignment operator (+=) adds a value to a variable.

# Assignment

<!DOCTYPE html>

<html>

<body>

<h1>The += Operator</h1>

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

<script>

var x = 10;

x += 5;

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

# JavaScript String Operators

The + operator can also be used to add (concatenate) strings.

When used on strings, the + operator is called the concatenation operator.

# Example

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Operators</h1>

<p>The + operator concatenates (adds) strings.</p>

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

<script>

var txt1 = "John";

var txt2 = "Doe";

document.getElementById("demo").innerHTML = txt1 + " " + txt2;

</script>

</body>

</html>

The result of txt3 will be:

John Doe

The += assignment operator can also be used to add (concatenate) strings:

# Example

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Operators</h1>

<p>The assignment operator += can concatenate strings.</p>

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

<script>

txt1 = "What a very ";

txt1 += "nice day";

document.getElementById("demo").innerHTML = txt1;

</script>

</body>

</html>

The result of txt1 will be:

What a very nice day

# Adding Strings and Numbers

Adding two numbers, will return the sum, but adding a number and a string will return a string:

# Example

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Operators</h1>

<p>Adding a number and a string, returns a string.</p>

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

<script>

var x = 5 + 5;

var y = "5" + 5;

var z = "Hello" + 5;

document.getElementById("demo").innerHTML =

x + "<br>" + y + "<br>" + z;

</script>

</body>

</html>

The result of x, y, and z will be:

10

55

Hello5

The rule is: If you add a number and a string, the result will be a string!

# JavaScript Comparison and Logical Operators

# Operator Description

== equal to

=== equal value and equal type

!= not equal

!== not equal value or not equal type

> greater than

< less than

>= greater than or equal to

<= less than or equal to

? ternary operator

Comparison and logical operators are described in the JS Comparisons chapter.

# JavaScript Type Operators

# Operator Description

typeof Returns the type of a variable

instanceof Returns true if an object is an instance of an object type

# JavaScript Arithmetic

A typical thing to do with numbers is arithmetic.

# JavaScript Arithmetic Operators

Arithmetic operators perform arithmetic on numbers (literals or variables).

# Operator Description

+ Addition

- Subtraction

\* Multiplication

/ Division

% Modulus

++ Increment

-- Decrement

# Arithmetic Operations

A typical arithmetic operation operates on two numbers.

The two numbers can be literals:

# Example

<!DOCTYPE html>

<html>

<body>

<p>A typical arithmetic operation takes two numbers and produces a new number.</p>

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

<script>

var x = 100 + 50;

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

or variables:

# Example

<!DOCTYPE html>

<html>

<body>

<p>A typical arithmetic operation takes two numbers (or variables) and produces a new number.</p>

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

<script>

var a = 100;

var b = 50;

var x = a + b;

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

or expressions:

# Example

<!DOCTYPE html>

<html>

<body>

<p>A typical arithmetic operation takes two numbers (or expressions) and produces a new number.</p>

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

<script>

var a = 3;

var x = (100 + 50) \* a;

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

# Operators and Operands

The numbers (in an arithmetic operation) are called operands.

The operation (to be performed between the two operands) is defined by an operator.

# Operand Operator Operand

100 + 50

The addition operator (+) adds numbers:

# Adding

<!DOCTYPE html>

<html>

<body>

<h1>The + Operator</h1>

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

<script>

var x = 5;

var y = 2;

var z = x + y;

document.getElementById("demo").innerHTML = z;

</script>

</body>

</html>

The subtraction operator (-) subtracts numbers.

# Subtracting

<!DOCTYPE html>

<html>

<body>

<h1>The - Operator</h1>

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

<script>

var x = 5;

var y = 2;

var z = x - y;

document.getElementById("demo").innerHTML = z

</script>

</body>

</html>

The multiplication operator (\*) multiplies numbers.

# Multiplying

<!DOCTYPE html>

<html>

<body>

<h1>The \* Operator</h1>

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

<script>

var x = 5;

var y = 2;

var z = x \* y;

document.getElementById("demo").innerHTML = z;

</script>

</body>

</html>

The division operator (/) divides numbers.

# Dividing

<!DOCTYPE html>

<html>

<body>

<h1>The / Operator</h1>

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

<script>

var x = 5;

var y = 2;

var z = x / y;

document.getElementById("demo").innerHTML = z;

</script>

</body>

</html>

The modular operator (%) returns the division remainder.

# Modulus

<!DOCTYPE html>

<html>

<body>

<h1>The % Operator</h1>

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

<script>

var x = 5;

var y = 2;

var z = x % y;

document.getElementById("demo").innerHTML = z;

</script>

</body>

</html>

The increment operator (++) increments numbers.

# Incrementing

<!DOCTYPE html>

<html>

<body>

<h1>The ++ Operator</h1>

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

<script>

var x = 5;

x++;

var z = x;

document.getElementById("demo").innerHTML = z;

</script>

</body>

</html>

The decrement operator (--) decrements numbers.

# Decrementing

<!DOCTYPE html>

<html>

<body>

<h1>The -- Operator</h1>

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

<script>

var x = 5;

x--;

var z = x;

document.getElementById("demo").innerHTML = z;

</script>

</body>

</html>

# Operator Precedence

Operator precedence describes the order in which operations are performed in an arithmetic expression.

Example

<!DOCTYPE html>

<html>

<body>

<p>Multiplication has precedence over addition.</p>

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

<script>

document.getElementById("demo").innerHTML = 100 + 50 \* 3;

</script>

</body>

</html>

Is the result of example above the same as 150 \* 3, or is it the same as 100 + 150?

Is the addition or the multiplication done first?

As in traditional school mathematics, the multiplication is done first.

Multiplication (\*) and division (/) have higher precedence than addition (+) and subtraction (-).

And (as in school mathematics) the precedence can be changed by using parentheses:

# Example

<!DOCTYPE html>

<html>

<body>

<p>Multiplication has precedence over addition.</p>

<p>But parenthesis has precedence over multiplication.</p>

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

<script>

document.getElementById("demo").innerHTML = (100 + 50) \* 3;

</script>

</body>

</html>

When using parentheses, the operations inside the parentheses are computed first.

When many operations have the same precedence (like addition and subtraction), they are computed from left to right:

# Example

<!DOCTYPE html>

<html>

<body>

<p>When many operations has the same precedence, they are computed from left to right.</p>

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

<script>

document.getElementById("demo").innerHTML = 100 + 50 - 3;

</script>

</body>

</html>

JavaScript Operator Precedence Values

# Value Operator Description Example

19 ( ) Expression grouping (3 + 4)

18 . Member person.name

18 [] Member person["name"]

17 () Function call myFunction()

17 new Create new Date()

16 ++ Postfix Increment i++

16 -- Postfix Decrement i--

15 ++ Prefix Increment ++i

15 -- Prefix Decrement --i

15 ! Logical not !(x==y)

15 typeof Type typeof x

14 \* Multiplication 10 \* 5

14 / Division 10 / 5

14 % Modulo division % 5

14 \*\* Exponentiation 10 \*\* 2

13 + Addition 10 + 5

13 - Subtraction 10 - 5

12 << Shift left x << 2

12 >> Shift right x >> 2

11 < Less than x < y

11 <= Less than or equal x <= y

11 > Greater than x > y

11 >= Greater than or equal x >= y

10 == Equal x == y

10 === Strict equal x === y

10 != Unequal x != y

10 !== Strict unequal x !== y

6 && And x && y

5 || Or x || y

3 = Assignment x = y

3 += Assignment x += y

3 -= Assignment x -= y

3 \*= Assignment x \*= y

3 /= Assignment x /= y

# JavaScript Assignment

# JavaScript Assignment Operators

Assignment operators assign values to JavaScript variables.

JavaScript Assignment Operators

Assignment operators assign values to JavaScript variables.

# Operator Example Same As

= x = y x = y

+= x += y x = x + y

-= x -= y x = x - y

\*= x \*= y x = x \* y

/= x /= y x = x / y

%= x %= y x = x % y

The = assignment operator assigns a value to a variable.

# Assignment

<!DOCTYPE html>

<html>

<body>

<h1>The = Operator</h1>

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

<script>

var x = 10;

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

The += assignment operator adds a value to a variable.

# Assignment

<!DOCTYPE html>

<html>

<body>

<h1>The += Operator</h1>

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

<script>

var x = 10;

x += 5;

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

The -= assignment operator subtracts a value from a variable.

# Assignment

<!DOCTYPE html>

<html>

<body>

<h1>The -= Operator</h1>

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

<script>

var x = 10;

x -= 5;

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

The \*= assignment operator multiplies a variable.

# Assignment

<!DOCTYPE html>

<html>

<body>

<h1>The \*= Operator</h1>

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

<script>

var x = 10;

x \*= 5;

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

The /= assignment divides a variable.

# Assignment

<!DOCTYPE html>

<html>

<body>

<h1>The /= Operator</h1>

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

<script>

var x = 10;

x /= 5;

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

The %= assignment operator assigns a remainder to a variable.

# Assignment

<!DOCTYPE html>

<html>

<body>

<h1>The %= Operator</h1>

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

<script>

var x = 10;

x %= 5;

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

# JavaScript Data Types

String, Number, Boolean, Array, Object.

# JavaScript Data Types

JavaScript variables can hold many data types: numbers, strings, arrays, objects and more:

var length = 16; // Number

var lastName = "Johnson"; // String

var cars = ["Saab", "Volvo", "BMW"]; // Array

var x = {firstName:"John", lastName:"Doe"}; // Object

# The Concept of Data Types

In programming, data types is an important concept.

To be able to operate on variables, it is important to know something about the type.

Without data types, a computer cannot safely solve this:

var x = 16 + "Volvo";

Does it make any sense to add "Volvo" to sixteen? Will it produce an error or will it produce a result?

JavaScript will treat the example above as:

var x = "16" + "Volvo";

When adding a number and a string, JavaScript will treat the number as a string

# Example:

<!DOCTYPE html>

<html>

<body>

<p>When adding a number and a string, JavaScript will treat the number as a string.</p>

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

<script>

var x = 16 + "Volvo";

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

JavaScript evaluates expressions from left to right. Different sequences can produce different results:

# JavaScript:

<!DOCTYPE html>

<html>

<body>

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

<script>

var x = 16 + 4 + "Volvo";

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

In the first example, JavaScript treats 16 and 4 as numbers, until it reaches "Volvo".

In the second example, since the first operand is a string, all operands are treated as strings.

# JavaScript Has Dynamic Types

JavaScript has dynamic types. This means that the same variable can be used as different types:

# Example:

var x; // Now x is undefined

var x = 5; // Now x is a Number

var x = "John"; // Now x is a String

# JavaScript Strings

A string (or a text string) is a series of characters like "John Doe".

Strings are written with quotes. You can use single or double quotes:

# Example

var carName = "Volvo XC60"; // Using double quotes

var carName = 'Volvo XC60'; // Using single quotes

You can use quotes inside a string, as long as they don't match the quotes surrounding the string:

# Example

<!DOCTYPE html>

<html>

<body>

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

<script>

var carName1 = "Volvo XC60";

var carName2 = 'Volvo XC60';

var answer1 = "It's alright";

var answer2 = "He is called 'Johnny'";

var answer3 = 'He is called "Johnny"';

document.getElementById("demo").innerHTML =

carName1 + "<br>" +

carName2 + "<br>" +

answer1 + "<br>" +

answer2 + "<br>" +

answer3;

</script>

</body>

</html>

# JavaScript Numbers

JavaScript has only one type of numbers.

Numbers can be written with, or without decimals:

# Example

var x1 = 34.00; // Written with decimals

var x2 = 34; // Written without decimals

Extra large or extra small numbers can be written with scientific (exponential) notation:

# Example

<!DOCTYPE html>

<html>

<body>

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

<script>

var x1 = 34.00;

var x2 = 34;

var y = 123e5;

var z = 123e-5;

document.getElementById("demo").innerHTML = x1 + "<br>" + x2 + "<br>" + y + "<br>" + z

</script>

</body>

</html>

# JavaScript Booleans

Booleans can only have two values: true or false.

# Example

var x = true;

var y = false;

Booleans are often used in conditional testing.

You will learn more about conditional testing later in this tutorial.

# JavaScript Arrays

JavaScript arrays are written with square brackets.

Array items are separated by commas.

The following code declares (creates) an array called cars, containing three items (car names):

# Example

<!DOCTYPE html>

<html>

<body>

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

<script>

var cars = ["Saab","Volvo","BMW"];

document.getElementById("demo").innerHTML = cars[0];

</script>

</body>

</html>