Basics of javascript

Instructors: Anumol | Varun

A COMPLETE WEBPAGE

A standard webpage is made up of
3 components

HTML

Provides the visual and functional content of the web pages

CSS

Provides visual enhancements of HTML elements in the page.

Script

Provides control over interactions within the page and assists in programming the behavior of web pages.

What is javascript ?

JavaScript is the programming language of HTML and the Web.

Javascript is a dynamic computer programming language.

It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

JavaScript is the default scripting language in HTML.

Where to include?

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

1. JavaScript in <head>
2. JavaScript in <body>
3. External JavaScript
4. External References

Example

```
<!DOCTYPE html>
<html>
  <head>
    <script>
        function myFunctionHead() {
          //Write javascript code here
        }
    </script>
    <script src="myScript.js"></script>
    <script src="https://www.extSite.com/test.js"></script>
  </head>
  <body>
    <div>
    <p>This is a paragraph.</p>
  </div>
  <script>
     function myFunctionBody() {
        //Write javascript code here
     }
   </script>
  </body>
</html>
```

document.write()

'document.write()' writes a string into our HTML document.

This function can be used to write text, HTML, or both.

Example 1 Example2

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

window.alert()

You can use an alert box to display data.

Example

console.log()

For debugging purposes, you can use the console.log() method to display data.

Example

JavaScript Statements

A javaScript statements consists of:
Values
Operators
Expressions
Keywords
Comments.

JavaScript Operators

JavaScript uses arithmetic operators (+ - * /) to compute values

It uses assignment operator (=) to assign values to variables

```
eg:  var x, y, z;
     5 + 6;
```

JavaScript Keywords

JavaScript keywords are used to identify actions to be performed.

```
eg:  var x = 10;
```

The var keyword tells the browser to create variables

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**.

Literals

Numbers are written with or without decimals

```
eg: 100, 10.56, -986
```

Strings are text, written within double or single quotes

```
eg: "Text Sting 1", 'Test String 2'
```

Variables

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.

```
eg:  var x;
     x = 6;
```

JavaScript Expressions

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

Expressions can also contain variable values

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

```
eg:  5 * 10;
     x * 10;
     "John" + " " + "Doe"
```

JavaScript Comments

Not all JavaScript statements are "executed".

Comments are ignored, and will not be executed

Line Comments

```
// This is a line comment
```

Block Comments

```
/*  This is a block comment
    This is a block comment
    This is a block comment */
```

Basic Rules and Nomenclature

JavaScript Identifiers

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

In JavaScript, the first character must be a letter, or an underscore (_), or a dollar sign ($). Numbers are not allowed as the first character.

All JavaScript identifiers are case sensitive.

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

JavaScript and Camel Case

JavaScript programmers tend to use camel case that starts with a lowercase letter:

eg: firstName, lastName, masterCard, interCity.

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

Semicolons and Whitespaces

Semicolons separate JavaScript statements.

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

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

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

JavaScript variable can hold a value of any data type.

Unlike many other languages, you don't have to tell JavaScript during variable declaration what type of value the variable will hold.
The value type of a variable can change during the execution of a program and JavaScript takes care of it automatically.

JavaScript Strings

A string (or a text string) is a series of characters like "John Doe".
Strings are written with single or double quotes
eg: var carName = "Volvo XC60";

JavaScript Numbers

JavaScript has only one type of numbers.
Numbers can be written with, or without decimals.
eg: var x1 = 34.00;

JavaScript Boolean

Booleans can only have two values: true or false.
Booleans are often used in conditional testing.
eg: var x = true;

JavaScript Arrays

JavaScript arrays are written with square brackets.
Array items are separated by commas.
eg: var cars = ["Audi", "Volvo", "BMW"];

JavaScript Data Types

JavaScript variables can hold any data types:

- Numbers
- String
- Objects
- Boolean

In JavaScript, the variable decalaration in done using var

Javascript Functions

A JavaScript function is a block of code designed to perform a particular task.

A JavaScript function is executed when "something" invokes it (calls it).

A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses ().

Syntax

```
function_name( parameter1, parameter2, parameter3 ) {
   //code to be executed
}
```

Function Invocation

The code inside the function will execute when "something" invokes (calls) the function:

- When an event occurs (when a user clicks a button)
- When it is invoked (called) from JavaScript code
- Automatically (self invoked)

Function Return

When JavaScript reaches a return statement, the function will stop executing.

If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.

Functions often compute a return value. The return value is "returned" back to the "caller"

```
var x = myFunction(4, 3);

function myFunction(a, b) {
   return a * b;
}
```

Javascript Events

HTML events are "things" that happen to HTML elements.

When JavaScript is used in HTML pages, JavaScript can "react" on these events.

HTML Events

An HTML event can be something the browser does, or something a user does, like:
An HTML web page has finished loading
An HTML input field was changed
An HTML button was clicked

JavaScript lets you execute code when events are detected.

Syntax

<element event='some JavaScript'>

Example

Example 1

Example 2

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

Commonly Used JavaScript Functions

String Length

Example

The length() property returns the length of a string

```
var len = txt.length;
```

String Search

Example

The search() method searches a string for a specified value and returns the position of the match

```
var pos = str.search("item");
```

String Slice

Example

slice() extracts a part of a string and returns the extracted part in a new string.

```
var res = str.slice(x, y);
```

String Case

Example

A string is converted to upper case with toUpperCase()

A string is converted to lower case with toLowerCase()

```
var text2 = text1.toUpperCase();
var text2 = text1.toLowerCase();
```

String Concat

Example

concat() joins two or more strings

```
var newString = string_1.concat(string_2);
```

Decimal Round-off

Example

toFixed() returns a string, with the number rounded-off with a specified number of decimals

```
num_Value.toFixed(2);
```

Number Conversion

Example

parseInt() parses a string and returns a whole number.

If the number cannot be converted, NaN (Not a Number) is returned.

```
parseInt("10");
```

Miscellaneous JavaScript Functions

Math Object

Example

The JavaScript Math object allows you to perform mathematical tasks on numbers.

JavaScript Dates

Example

The Date object lets you work with dates (years, months, days, hours, minutes, seconds, and milliseconds)

'get' Date Options

Method	Description
getDate()	Get the day as a number (1-31)
getDay()	Get the weekday as a number (0-6)
getFullYear()	Get the four digit year (yyyy)
getHours()	Get the hour (0-23)
getMilliseconds()	Get the milliseconds (0-999)
getMinutes()	Get the minutes (0-59)
getMonth()	Get the month (0-11)
getSeconds()	Get the seconds (0-59)
getTime()	Get the time (milliseconds since January 1, 1970)

Conditional Statements

if Statement

else Statement

else if Statement

Switch Case

Use the if statement to specify a block of JavaScript code to be executed if a condition is true.

Use the else statement to specify a block of JavaScript code to be executed if a condition is false.

Use the else if statement to specify a new condition if the first condition is false.

Use the switch statement to select one of many blocks of code to be executed.

When JavaScript reaches a break keyword, it breaks out of the switch block.

The default keyword specifies the code to run if there is no case match

Examples

if	else	else if	switch

Looping Statements

Loops can execute a block of code a number of times.

Loops are handy, if you want to run the same code over and over again, each time with a different value.

JavaScript supports different kinds of loops:

for - loops through a block of code a number of times
for/in - loops through the properties of an object
while - loops through a block of code while a specified condition is true
do/while - also loops through a block of code while a specified condition is true

for Loop

The for loop has the following syntax:

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

Statement 1 - executed before loop starts.
Statement 2 - defines the condition for running the loop.
Statement 3 - is executed each time after the loop has been executed.

for **for..in**

while Loop

The while loop syntax:

```
while (condition) {
  // code block to be executed
}
```

The while loop loops through a block of code as long as a specified condition is true.
The do/while loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

while **do/ while**

Thank You !