Pemrograman Web

4. Javascript

M. Udin Harun Al Rasyid, S.Kom, Ph.D
http://lecturer.eepis-its.edu/~udinharun
udinharun@eepis-its.edu

Table of Contents

Introduction

- JavaScript is the most popular scripting language in the world. It is the standard language used in web pages.

- Also widely used by desktop apps, mobile phone apps, and internet servers.

- JavaScript is used in millions of Web pages to improve the design, validate forms, detect browsers, create cookies, and much more.

JavaScript™

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language (a scripting language is a lightweight programming language)
- A JavaScript consists of lines of executable computer code
- A JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)

Java vs JavaScript

- Java - Programming Language (PL)
  - Interactive Web Graphics
  - Creating web browser applications
  - Writing stand-alone applications
  - Developed by Sun Microsystems, a powerful and much more complex programming language - in the same category as C and C++.

- JavaScript - Scripting Language
  - Runs within the context of the Web browser
  - Customizing pages based on browser version
  - Visual Feedback to user actions
  - Validating data entered on HTML Forms

What can JavaScript do?

- **JavaScript can manipulate HTML**
  JavaScript can read and change the content of HTML elements.

- **JavaScript can manipulate CSS**
  JavaScript can read and change the style of HTML elements.

- **JavaScript can validate data**
  JavaScript can be used to validate data, like validating forms input.

- **JavaScript can react to events**
  JavaScript can be set to execute when something happens, like when a user clicks on an HTML element.

The <script> Tag

- A JavaScript is surrounded by a <script> and </script> tag.
- The lines between the <script> and </script> contain the JavaScript:.
- Example:

```
<script>
alert("My First JavaScript");
</script>
```

Manipulating HTML Elements

- To access an HTML element from JavaScript, you can use the document.getElementById(*id*) method.
- Use the "id" attribute to identify the HTML element:

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

<h1>My First Web Page</h1>

<p id="demo">My First Paragraph.</p>

<script>
document.getElementById("demo").innerHTML="My First JavaScript ";
</script>

</body>
</html>
```

My First Web Page

My First JavaScript

Writing to The Document Output

- Use document.write() only to write directly into the document output.

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

<h1>My First Web Page</h1>

<script>
document.write("<p>My First JavaScript</p>");
</script>

</body>
</html>
```

My First Web Page

My First JavaScript

- If you execute document.write after the document has finished loading, the entire HTML page will be overwritten:

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

<h1>My First Web Page</h1>

<p>My First Paragraph.</p>

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

<script>
function myFunction()
{
document.write("Oops! The document disappeared!");
}
</script>

</body>
</html>
```

My First Web Page

My First Paragraph.

Try it

Oops! The document disappeared!

A JavaScript Function in <head>

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction()
{
document.getElementById("demo").innerHTML="My First JavaScript Function";
}
</script>
</head>

<body>

<h1>My Web Page</h1>

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

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

</body>
</html>
```

My Web Page

A Paragraph.

Try it

My Web Page

My First JavaScript Function

Try it

A JavaScript Function in <body>

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

<h1>My First Web Page</h1>

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

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

<script>
function myFunction()
{
document.getElementById("demo").innerHTML="My First JavaScript Function";
}
</script>

</body>
</html>
```

Using an External JavaScript

- Scripts can also be placed in external files. External files often contain code to be used by several different web pages.
- External JavaScript files have the file extension .js.
- To use an external script, point to the .js file in the "src" attribute of the <script> tag:

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

<h1>My Web Page</h1>

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

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

<p><strong>Note:</strong> The actual script is in an external script file called
"myScript.js".</p>

<script type="text/javascript" src="myScript.js"></script>

</body>
</html>
```

JavaScript Statements

- JavaScript statements are "commands" to the browser. The purpose of the statements is to tell the browser what to do.

- Example:
  document.getElementById("demo").innerHTML="Hello Dolly";

- Semicolon separates JavaScript statements.
- Normally you add a semicolon at the end of each executable statement.

JavaScript Code

- JavaScript code (or just JavaScript) is a sequence of JavaScript statements.

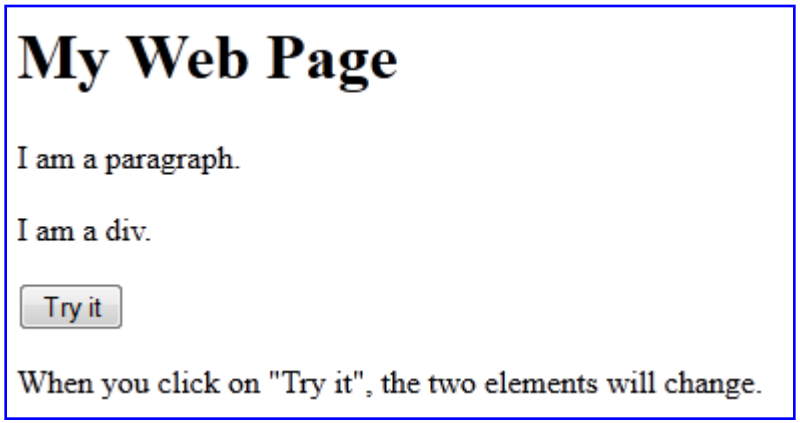
- Each statement is executed by the browser in the sequence they are written.

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

<h1>My Web Page</h1>

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

<div id="myDIV">A DIV.</div>

<script>
document.getElementById("demo").innerHTML="Hello Don";
document.getElementById("myDIV").innerHTML="How are you?";
</script>

</body>
</html>
```

JavaScript Code Blocks

- JavaScript statements can be grouped together in blocks start with a left curly bracket, and end with a right curly bracket.
- The purpose of a block is to make the sequence of statements execute together as JavaScript functions.

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

<h1>My Web Page</h1>

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

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

<script>
function myFunction()
{
document.getElementById("myPar").innerHTML="Hello Don";
document.getElementById("myDiv").innerHTML="How are you?";
}
</script>

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

</body>
</html>
```



JavaScript is Case Sensitive

- JavaScript is case sensitive.
- Watch your capitalization closely when you write JavaScript statements:
- A function getElementById is not the same as getElementbyID.
- A variable named myVariable is not the same as MyVariable.

JavaScript Comments

- Comments will not be executed by JavaScript.
- Comments can be added to explain the JavaScript, or to make the code more readable.
- Single line comments start with //.
- Multi line comments start with /* and end with */.

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

<h1 id="myH1"></h1>

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

<script>
// Write to a heading:
document.getElementById("myH1").innerHTML="Welcome to my Homepage";
// Write to a paragraph:
document.getElementById("myP").innerHTML="This is my first paragraph.";
</script>

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

</body>
</html>
```

Welcome to my Homepage

This is my first paragraph.

Note: The comments are not executed.

JavaScript Variables

- Variables are "containers" for storing information:

```
<!DOCTYPE html>
<html>
<body>
<script>
var answer1="He is called 'Johnny'";
var answer2='He is called "Johnny"';
var pi=3.14;
var x=123;
var y=123e5;
var z=123e-5;
var cars=["Saab","Volvo","BMW"];
var person={firstname:"John", lastname:"Doe", id:5566};
var carname
document.write(answer1 + "<br>")
document.write(answer2 + "<br>")
document.write(pi + "<br>")
document.write(x + "<br>")
document.write(y + "<br>")
document.write(z + "<br>")
document.write(cars[2] + "<br>")
document.write(person.firstname + " " + person["lastname"] + "<br>")
document.write(carname + "<br>")
</script>

</body>
</html>
```

He is called 'Johnny'
He is called "Johnny"
3.14
123
12300000
0.00123
BMW
John Doe
undefined

- Variable can have a short names, like x and y, or more descriptive names, like age, sum, or, totalvolume.
- Rules for JavaScript variable names:
  - Variable names are case sensitive (y and Y are two different variables)
  - Variable names must begin with a letter, the $ character, or the underscore character

- Example declare JavaScript variables with the **var** keyword:
  - var carname;
  - carname="Volvo";
  - var carname="Volvo";

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

<p>Click the button to create a variable, and display the result.</p>

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

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

<script>
function myFunction()
{
var carname="Volvo";
document.getElementById("demo").innerHTML=carname;
}
</script>

</body>
</html>
```

Click the button to create a variable, and display the result.

Try it

JavaScript Data Types

- There are many types of JavaScript variables, but for now, just think of two types: text and numbers.
- When you assign a text value to a variable, put double or single quotes around the value.
- When you assign a numeric value to a variable, do not put quotes around the value. If you put quotes around a numeric value, it will be treated as text.

One Statement, Many Variables

- You can declare many variables in one statement. Just start the statement with **var** and separate the variables by comma:

```
var name="Doe", age=30, job="carpenter";
```

- Your declaration can also span multiple lines:

```
var name="Doe",
age=30,
job="carpenter";
```

JavaScript Arithmetic

- As with algebra, you can do arithmetic with JavaScript variables, using operators like = and +:

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

<p>Given that y=5, calculate x=y+2, and display the result.</p>
<button onclick="myFunction()">Try it</button>

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

<script>
function myFunction()
{
var y=5;
var x=y+2;
var demoP=document.getElementById("demo")
demoP.innerHTML="x=" + x;
}
</script>

</body>
</html>
```

Given that y=5, calculate x=y+2, and display the result.

Try it

JavaScript Functions

- A function is a block of code that executes only when you tell it to execute.
- It can be when an event occurs, like when a user clicks a button, or from a call within your script, or from a call within another function.
- Functions can be placed both in the <head> and in the <body> section of a document, just make sure that the function exists, when the call is made.
- Syntax:

  function *functionname*()
  {
  some code
  }

```html
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction()
{
alert("Hello World!");
}
</script>
</head>

<body>

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

<p>By clicking the button above, a function will be called. The function will alert a message.</p>

</body>
</html>
```

Try it

By clicking the button above, a function will be called. The function will alert a message.

Calling a Function with Arguments

- When you call a function, you can pass along some values to it, these values are called *arguments* or *parameters*.
- These arguments can be used inside the function.
- You can send as many arguments as you like, separated by commas (,)
- Syntax: myFunction(*argument1,argument2*)
- Declare the argument, as variables, when you declare the function:

```
function myFunction(var1,var2)
    {
    some code
    }
```

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

<p>Click the button to call a function with arguments</p>

<button onclick="myFunction('Harry Potter','Wizard')">Try it</button>

<script>
function myFunction(name,job)
{
alert("Welcome " + name + ", the " + job);
}
</script>

</body>
</html>
```

Click the button to call a function with arguments

Try it

Welcome Harry Potter, the Wizard

OK

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

<p>Click one of the buttons to call a function with arguments</p>

<button onclick="myFunction('Harry Potter','Wizard')">Click for Harry Potter</button>
<button onclick="myFunction('Bob','Builder')">Click for Bob</button>

<script>
function myFunction(name,job)
{
alert("Welcome " + name + ", the " + job);
}
</script>

</body>
</html>
```

Click one of the buttons to call a function with arguments

Click for Harry Potter | Click for Bob

Functions With a Return Value

- Sometimes you want your function to return a value back to where the call was made.
- This is possible by using the *return* statement.
- When using the *return* statement, the function will stop executing, and return the specified value.

- Syntax:

```
function myFunction()
{
var x=5;
return x;
}
```

- The function-call will be replaced with the returnvalue:
  var myVar=myFunction();

- You can also use the returnvalue without storing it as a variable:
  document.getElementById("demo").innerHTML=myFunction();

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

<p>This example calls a function which perfoms a calculation, and returns the result:</p>

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

<script>
function myFunction(a,b)
{
return a*b;
}

document.getElementById("demo").innerHTML=myFunction(4,3);
</script>

</body>
</html>
```

This example calls a function which perfoms a calculation, and returns the result:

12

- **Local JavaScript Variables**
  - A variable declared (using var) within a JavaScript function becomes **LOCAL** and can only be accessed from within that function. (the variable has local scope).
  - You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.
  - Local variables are deleted as soon as the function is completed.

- **Global JavaScript Variables**
  - Variables declared outside a function, become **GLOBAL**, and all scripts and functions on the web page can access it.

- **The Lifetime of JavaScript Variables**
  - The lifetime JavaScript variables starts when they are declared.
  - Local variables are deleted when the function is completed.
  - Global variables are deleted when you close the page.

- **Assigning Values to Undeclared JavaScript Variables**
  - If you assign a value to variable that has not yet been declared, the variable will automatically be declared as a GLOBAL variable.
  - Statement: carname="Volvo";

  will declare the variable *carname* as a global variable , even if it is executed inside a function.

JavaScript Data Types

String, Number, Boolean, Array, Object, Null, Undefined.

JavaScript Strings

- A string is a variable which stores a series of characters which can be any text inside quotes. You can use simple or double quotes.

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

<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"';
var answer4="He is called \"Johnny\"";
document.write(carname1 + "<br>")
document.write(carname2 + "<br>")
document.write(answer1 + "<br>")
document.write(answer2 + "<br>")
document.write(answer3 + "<br>")
document.write(answer4 + "<br>")
document.write(carname1[7] + "<br>")
</script>

</body>
</html>
```

Volvo XC60
Volvo XC60
It's alright
He is called 'Johnny'
He is called "Johnny"
He is called "Johnny"
C

JavaScript Numbers

- JavaScript has only one type of numbers. Numbers can be written with, or without decimals:

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

<script>

var x1=34.00;
var x2=34;
var y=123e5;
var z=123e-5;

document.write(x1 + "<br>")
document.write(x2 + "<br>")
document.write(y + "<br>")
document.write(z + "<br>")

</script>

</body>
</html>
```

34
34
12300000
0.00123

JavaScript Booleans

- Booleans can only have two values: true or false.
  - var x=true
    var y=false

JavaScript Arrays

- The following code creates an Array called cars.
  - var cars=new Array();
    cars[0]="Saab";
    cars[1]="Volvo";
    cars[2]="BMW";
  - var cars=new Array("Saab","Volvo","BMW");
  - var cars=["Saab","Volvo","BMW"];

JavaScript Objects

- An object is delimited by curly braces. Inside the braces the object's properties are defined as name and value pairs (name : value). The properties are separated by commas:
  - var person={firstname:"John", lastname:"Doe", id:5566};
  - var person={
    firstname : "John",
    lastname : "Doe",
    id : 5566
    };

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

<script>
var person={
firstname : "John",
lastname  : "Doe",
id        :  5566
};
document.write(person.lastname + "<br>");
document.write(person["lastname"] + "<br>");
</script>

</body>
</html>
```

Doe
Doe

Null or Undefined

- Non-existing is the value of a variable with no value.
- Variables can be emptied by setting the value to null;
- Example:

```
cars=null;
person=null;
```

JavaScript Operators

- The assignment operator **=** is used to assign values to JavaScript variables.
- The arithmetic operator + is used to add values together.

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

<p>Click the button to calculate x.</p>
<button onclick="myFunction()">Try it</button>

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

<script>
function myFunction()
{
y=5;
z=2;
x=y+z;
document.getElementById("demo").innerHTML=x;
}
</script>

</body>
</html>
```

Click the button to calculate x.

Try it

JavaScript Arithmetic Operators

Operator	Description	Example	Result of x	Result of y
+	Addition	x=y+2	7	5
-	Subtraction	x=y-2	3	5
*	Multiplication	x=y*2	10	5
/	Division	x=y/2	2.5	5
%	Modulus (division remainder)	x=y%2	1	5
++	Increment	x=++y	6	6
		x=y++	5	6
--	Decrement	x=--y	4	4
		x=y--	5	4

JavaScript Assignment Operators

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

The + Operator Used on Strings

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

<p>Click the button to create and add string variables.</p>
<button onclick="myFunction()">Try it</button>

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

<script>
function myFunction()
{
txt1="What a very";
txt2="nice day";
txt3=txt1+txt2;
document.getElementById("demo").innerHTML=txt3;
}
</script>

</body>
</html>
```

Click the button to create and add string variables.

Try it

Adding Strings and Numbers

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

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

<p>Click the button to add numbers and strings.</p>
<button onclick="myFunction()">Try it</button>

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

<script>
function myFunction()
{
var x=5+5;
var y="5"+5;
var z="Hello"+5;

var demoP=document.getElementById("demo");
demoP.innerHTML=x + "<br>" + y + "<br>" + z;
}
</script>

</body>
</html>
```

Click the button to add numbers and strings.

Try it

Finish