# Introduction to JavaScript: Part 1

## Introduction to Internet and Web

부산대학교 정보·의생명공학대학
정보컴퓨터공학부

# Contents

- ❖ JavaScript Introduction
- ❖ JavaScript Syntax
- ❖ JavaScript Functions
- ❖ JavaScript HTML DOM navigation

# JAVASCRIPT INTRODUCTION

# Why Study JavaScript?

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

❖ 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

❖ Web pages are not the only place where JavaScript is used. Many desktop and server programs use JavaScript.

- Node.js, MongoDB, CouchDB …

# JavaScript Where To

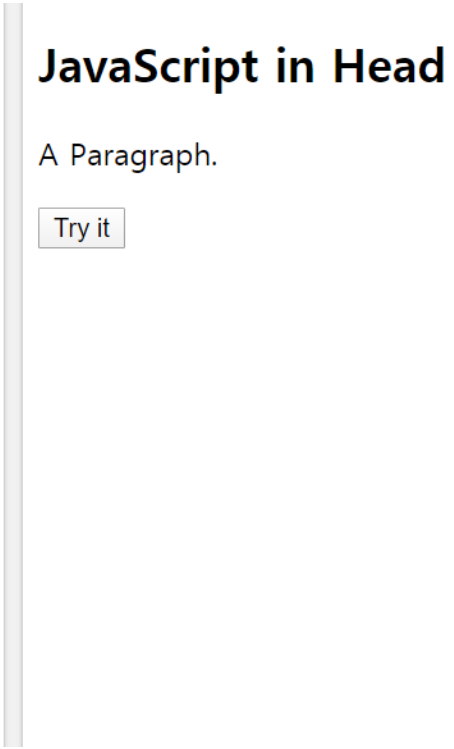
❖ In HTML, JavaScript code is inserted between <script> and </script> tags

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

<h2>JavaScript in Body</h2>

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

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

</body>
</html>
```

# JavaScript in <head>

- Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both
- In this example, a JavaScript function is placed in the <head> section of an HTML page.
- The function is invoked when a button is clicked:

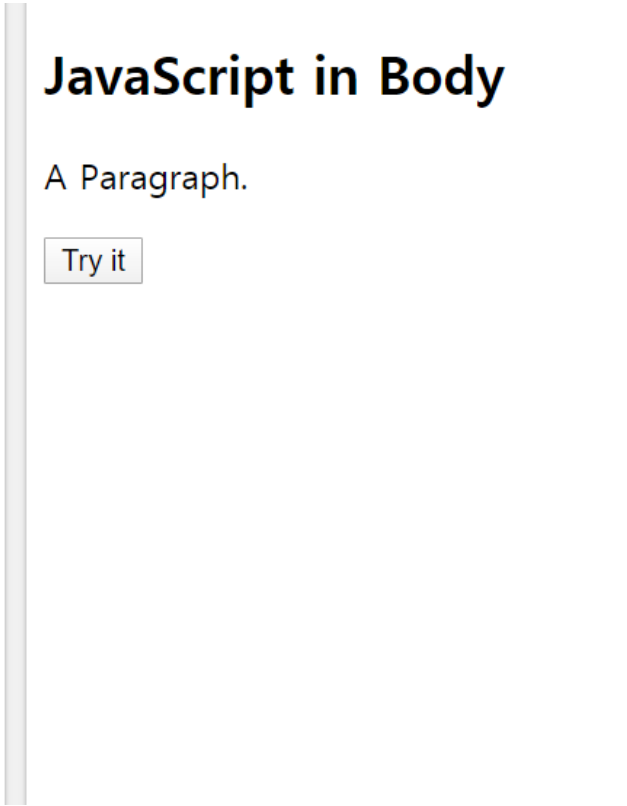
```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
  document.getElementById("demo").innerHTML =
"Paragraph changed.";
}
</script>
</head>
<body>

<h2>JavaScript in Head</h2>

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

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

</body>
</html>
```



# JavaScript in <body>

- In this example, a JavaScript function in placed in the <body> section of an HTML page.
- The function is invoked when a button is clicked:

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

<h2>JavaScript in Body</h2>

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

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

<script>
function myFunction() {
  document.getElementById("demo").innerHTML =
"Paragraph changed.";
}
</script>

</body>
</html>
```



# External JavaScript

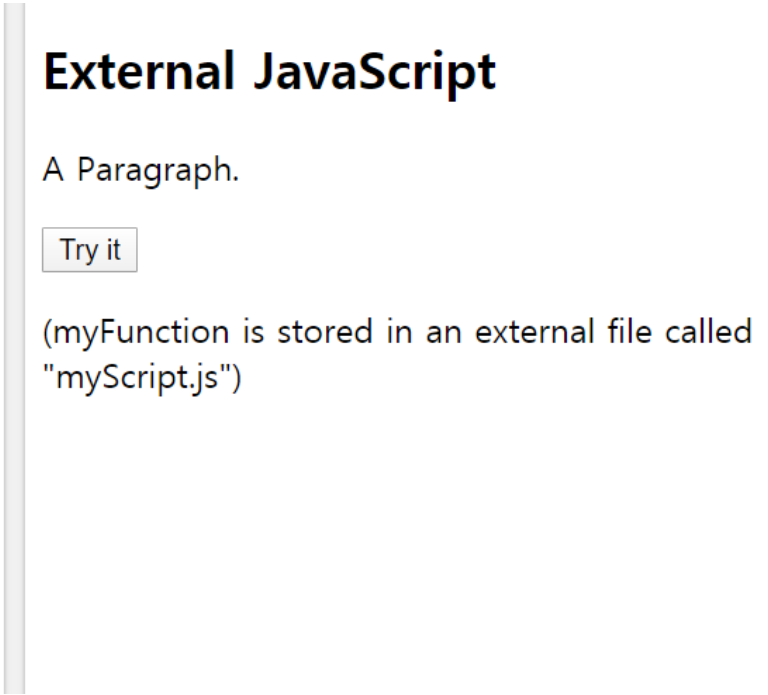
- Script can also be placed in external files:
- External scripts are practical when the same code is used in many different web pages
- JavaScript files have the file extension .js
- To use an external script, put the name of the script file in the src attribute of a <script> tag:
- Placing scripts in external files has some advantages:
  - It separates HTML and code
  - It makes HTML and JavaScript easier to read and maintain
  - Cached JavaScript files can speed up page loads

# External JavaScript

## External file: myScript.js

```
function myFunction() {
 document.getElementById("demo").innerHTML = "Paragraph changed.";
}
```

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

<h2>External JavaScript</h2>

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

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

<p>(myFunction is stored in an external file called
"myScript.js")</p>

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

</body>
</html>
```



# External References

- External scripts can be referenced with a full URL or with a path relative to the current web page.
- This example uses a full URL to link to a script:

```
<script src="https://www.w3schools.com/js/myScript1.js"></script>
```

- This example uses a script located in a specified folder on the current web sites:

```
<script src="/js/myScript1.js"></script>
```

# JAVASCRIPT SYNTAX

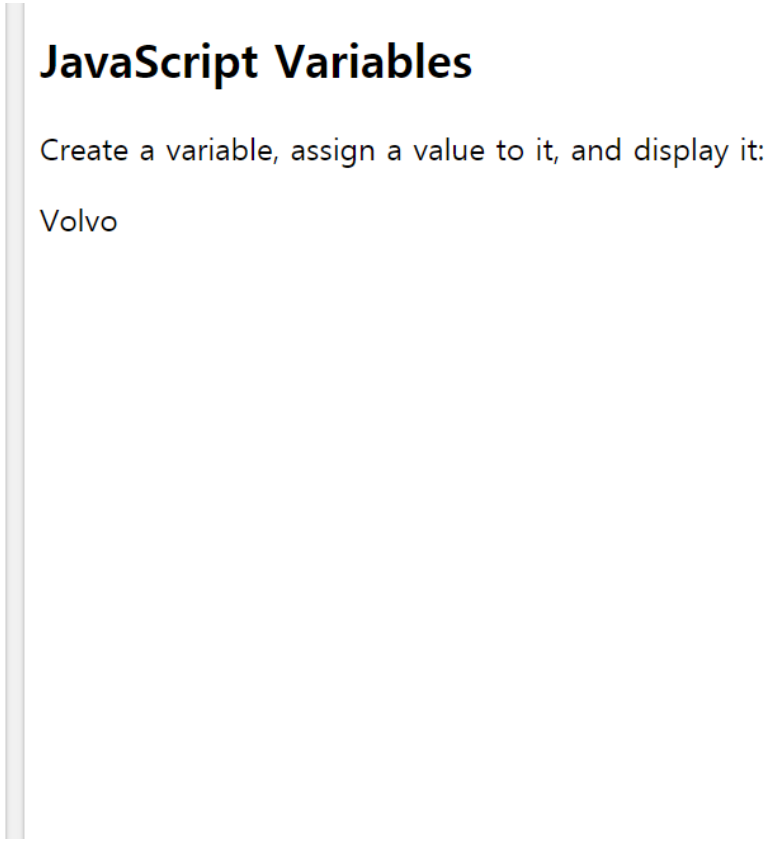
# JavaScript Values

- **The JavaScript syntax defines two types of values: Fixed values and variable values.**
- **Fixed values are called literals.**
  - **Numbers** are written with or without decimals: 10.5, 1001
  - **Strings** are text, written within double or single quotes: “John Doe”, ‘John Deo’
- **Variables values are called variables.**
  - JavaScript uses the **var** keyword to declare variables.
  - An equal sign is used to assign values to variables

```
var x;
x = 6;
```

# Declaring (Creating) JavaScript Variables

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

<h2>JavaScript Variables</h2>

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

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

<script>
var carName1;

var carNmae2;
carName2 = "Volvo"

var carName3 = "Volvo";

document.getElementById("demo").innerHTML = carName;
</script>

</body>
</html>
```




```
var person = "John Doe", carName = "Volvo", price = 200;
```

# JavaScript Identifier

- **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 Keywords

❖ JavaScript keywords are used to identify actions to be performed

| 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 Operators

❖ **Arithmetic operators are used to perform arithmetic on numbers**

❖ **The + operator can also be used to concatenate strings**

- Adding a number and a string will return a string

| Operator | Description |
|---|---|
| + | Addition |
| - | Subtraction |
| * | Multiplication |
| ** | Exponentiation (ES2016) |
| / | Division |
| % | Modulus (Division Remainder) |
| ++ | Increment |
| -- | Decrement |

Example

```
var x = 5 + 5;
var y = "5" + 5;
var z = "Hello" + 5;
```

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

```
10
55
Hello5
```

```
var txt1 = "John";
var txt2 = "Doe";
var txt3 = txt1 + " " + txt2;
```

The result of txt3 will be:

```
John Doe
```

# JavaScript Operators

- Assignment operators assign values to JavaScript variables.
- Bit operators work on 32 bits numbers.

| 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 |
| **= | x **= y | x = x ** y |

| Operator | Description |
|---|---|
| & | AND |
| \| | OR |
| ~ | NOT |
| ^ | XOR |
| << | Zero fill left shift |
| >> | Signed right shift |
| >>> | Zero fill right shift |

# 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:
- **Expressions can also contain variable values: x * 10**
- **The values can be of various types, such as numbers and strings.**
  - For example, "John" + " " + "Doe", evaluates to "John Doe":

# JavaScript Comments

- Not all JavaScript statements are “executed”.
- Code after double slashes // or written /* and */ is treated as a comment
- Comments are ignored, and will not be executed.

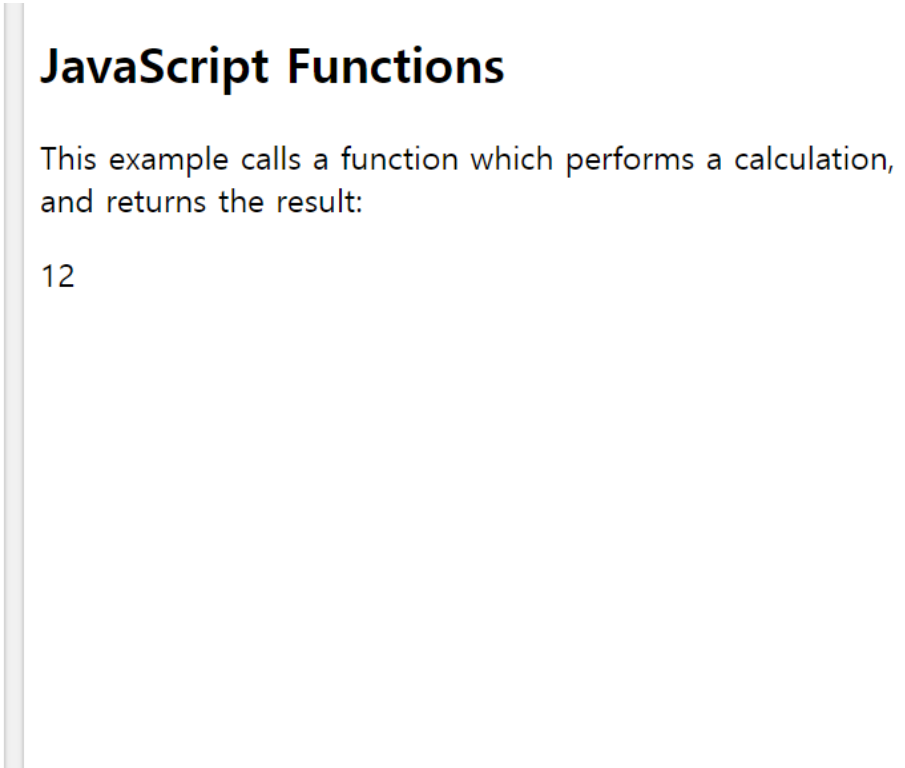
```
var x = 5;   // I will be executed

// var x = 6;   I will NOT be executed
```

# JAVASCRIPT FUNCTIONS

# 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

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

<h2>JavaScript Functions</h2>

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

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

<script>
function myFunction(p1, p2) {
  return p1 * p2;
}
document.getElementById("demo").innerHTML =
myFunction(4, 3);
</script>

</body>
</html>
```



# JavaScript Function Syntax

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

- Function name can contain letters, digits, underscores, and dollar signs. (same rules as variables)
- The parentheses may include parameter names separated by commas: (parameter1, parameter2, …)

❖ **The code to be executed, by the function, is placed inside curly brackets: {}**

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

# Function Invocation / Return

- The code inside the function will execute when "something" invokes (calls) the function:
  - e.g., Events, JavaScript code
- 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"

```
<script>
function myFunction(p1, p2) {
  return p1 * p2;
}
document.getElementById("demo").innerHTML =
myFunction(4, 3);
</script>
```

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

<script>
function displayDate() {
  document.getElementById("demo").innerHTML = Date();
}
</script>
```

# Function Invocation / Return

- Function parameters are listed inside the parentheses () in the fuction definition
- Function arguments are the values receives by the function when it is invoked

Calculate the product of two numbers, and return the result:

```
var x = myFunction(4, 3);   // Function is called, return value will end up in x

function myFunction(a, b) {
  return a * b;             // Function returns the product of a and b
}
```

# Why Functions?

- ❖ You can reuse code: Define the code once, and use it many times
- ❖ You can use the same code many times with different arguments, to produce different results.

Convert Fahrenheit to Celsius:

```
function toCelsius(fahrenheit) {
  return (5/9) * (fahrenheit-32);
}
document.getElementById("demo").innerHTML = toCelsius(77);
```

# Local variables

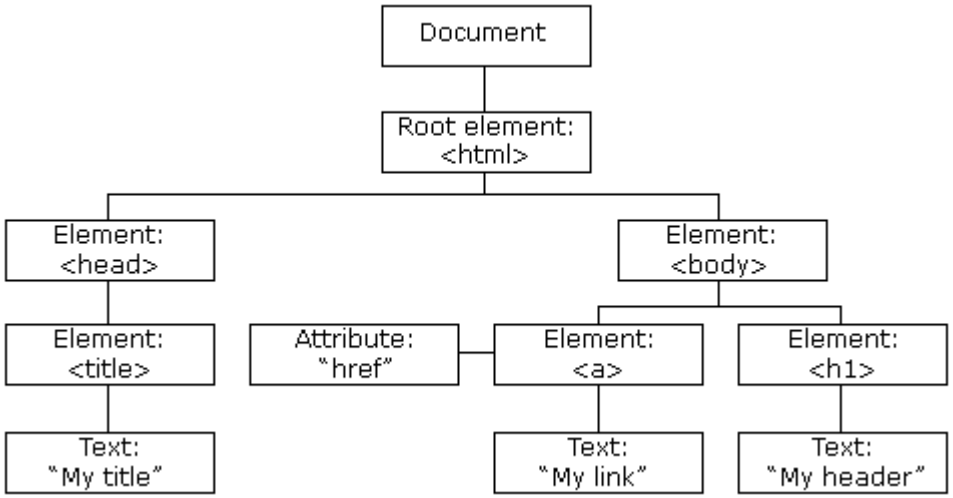
- Variables declared within a JavaScript function, become LOCAL to the function.
- Local variables can only be accessed from within the function
- Since local variables are only recognized inside their functions, variables with the same name can be used in different functions
- Local variables are created when a function starts, and deleted when the function is completed.
  - The arguments behave as local variables.

```
// code here can NOT use carName

function myFunction() {
  var carName = "Volvo";
  // code here CAN use carName
}

// code here can NOT use carName
```

# JAVASCRIPT HTML DOM NAVIGATION

# HTML DOM

- With the HTML DOM, all nodes in the node tree can be accessed by JavaScript.
- New nodes can be created, and all nodes can be modified or deleted.



# Navigating Between Nodes

❖ **The following node properties to navigate between nodes with JavaScript:**

- parentNode
- childNodes[nodenumber]
- firstChild
- lastChild
- nextSibling
- previousSibling

❖ **innerHTML property to retrieve the content of an HTML element.**

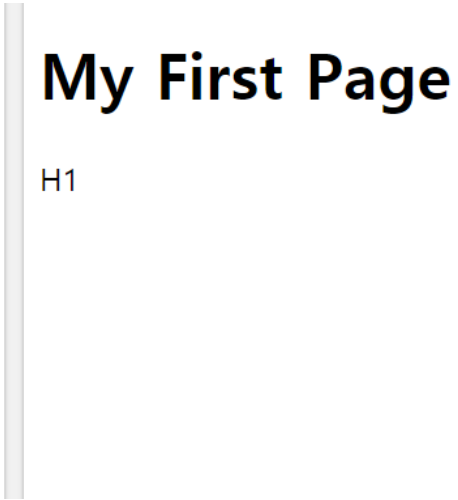
```
<title id="demo">DOM Tutorial</title>
var myTitle = document.getElementById("demo").innerHTML;
var myTitle = document.getElementById("demo").firstChild.nodeValue;
var myTitle = document.getElementById("demo").childNodes[0].nodeValue;
```

# The nodeName Property

## ❖ The nodeName property specifies the name of a node

- nodeName is read-only
- nodeName of an element node is the same as the tag name
- nodeName of an attribute node is the attribute name
- nodeName of a text node is always #text
- nodeName of the document node is always #document

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

<h1 id="id01">My First Page</h1>
<p id="id02"></p>

<script>
document.getElementById("id02").innerHTML =
document.getElementById("id01").nodeName;
</script>

</body>
</html>
```



# The nodeValue / nodeType Properties

## ❖ The nodeValue property specifies the value of a node

- nodeValue for element nodes is null
- nodeValue for text nodes is the text itself
- nodeValue for attribute nodes is the attribute value

## ❖ The nodeType property is read-only. It returns the type of a node

| Node | Type | Example |
|---|---|---|
| ELEMENT_NODE | 1 | <h1 class="heading">W3Schools</h1> |
| ATTRIBUTE_NODE | 2 | class = "heading" (deprecated) |
| TEXT_NODE | 3 | W3Schools |
| COMMENT_NODE | 8 | <!-- This is a comment --> |
| DOCUMENT_NODE | 9 | The HTML document itself (the parent of <html>) |
| DOCUMENT_TYPE_NODE | 10 | <!Doctype html> |

# 요 약

## ➢ JavaScript Introduction

- How to import JavaScript into HTML document

## ➢ JavaScript Syntax

- Values, Variables, Keywords, Identifier

## ➢ JavaScript Functions

- How to declare function
- How to invoke function

## ➢ JavaScript HTML DOM navigation

- How to access DOM node in the HTML document