Introduction to JavaScript: Part 2

Introduction to Internet and Web

부산대학교 정보·의생명공학대학
정보컴퓨터공학부

부산대학교
PUSAN NATIONAL UNIVERSITY

Contents

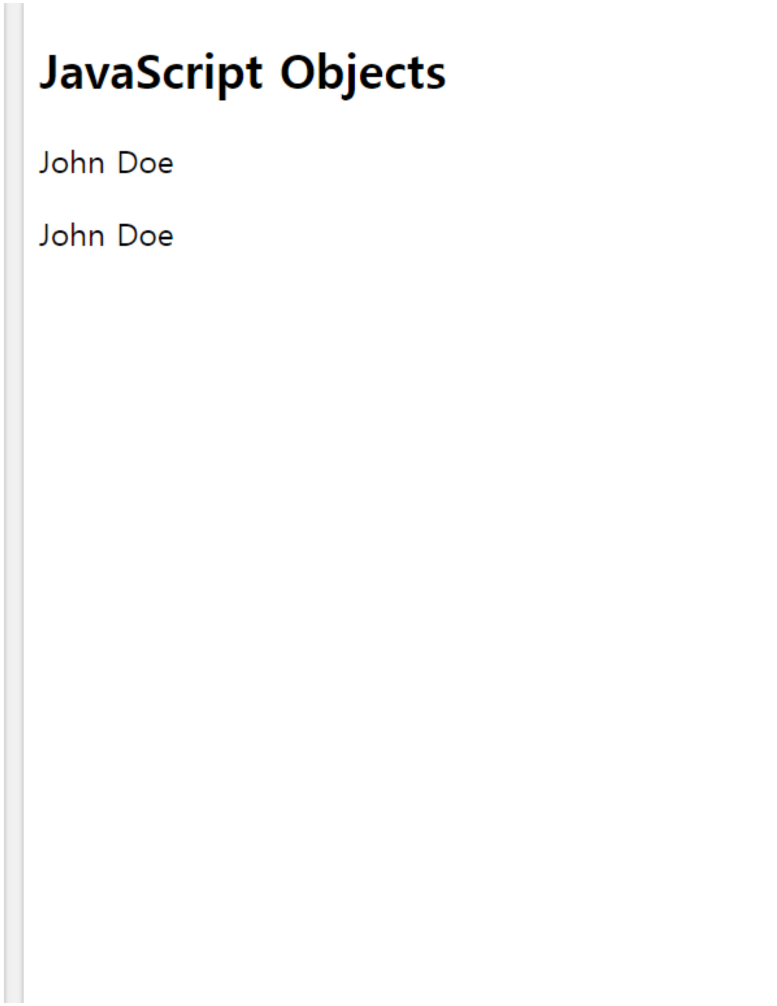
JAVASCRIPT OBJECT

JavaScript Object

- ❖ **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”;
- ❖ **Objects are variables too. But objects can obtain many values.**
  - var car = {type: “Fiat”, model: “500”, color:”white”};
  - The values are written as name:value pairs
  - The name:values pairs in JavaScript objects are called properties:

JavaScript Object Properties

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

<h2>JavaScript Objects</h2>

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

<p id="demo2"></p>

<script>
// Create an object:
var person = {
  firstName: "John",
  lastName : "Doe",
  id      :  5566
};
// Display some data from the object:
document.getElementById("demo").innerHTML =
person["firstName"] + " " + person["lastName"];

// Display some data from the object:
document.getElementById("demo2").innerHTML =
person.firstName + " " + person.lastName;
</script>

</body>
</html>
```



JavaScript Objects

John Doe

John Doe

JavaScript Object Methods

❖ Objects can also have methods.

❖ Methods are actions that can be performed on objects.

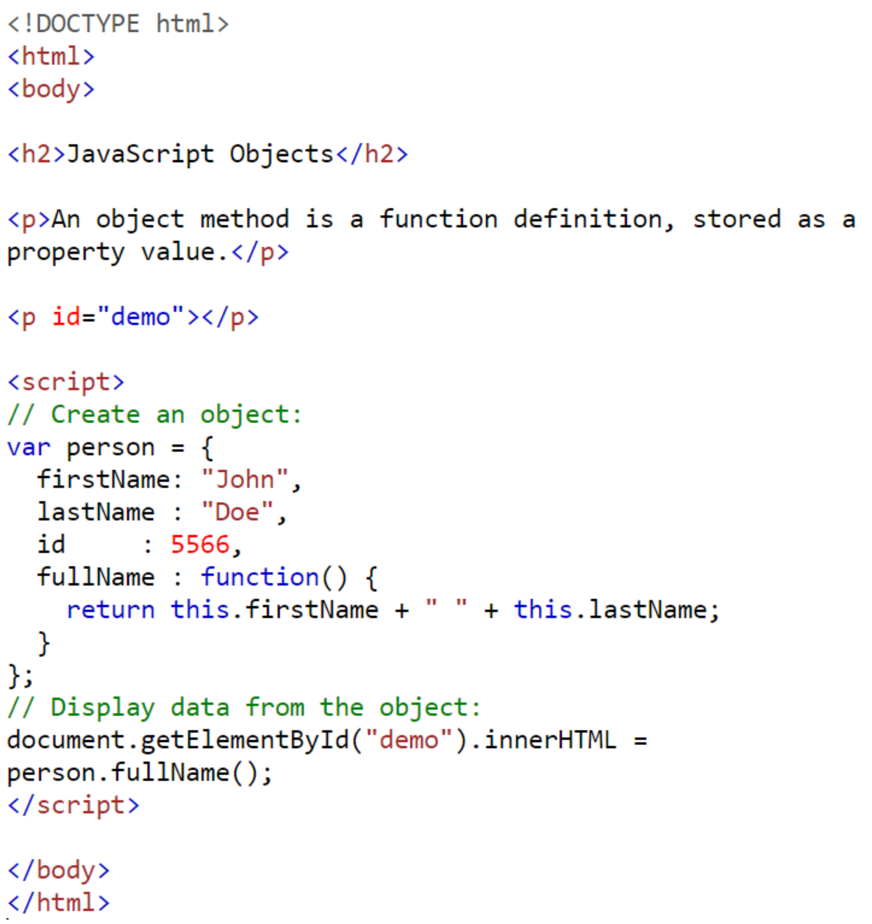
❖ Methods are stored in properties as function definitions.

❖ In a function definition, this refers to the “owner” of the function

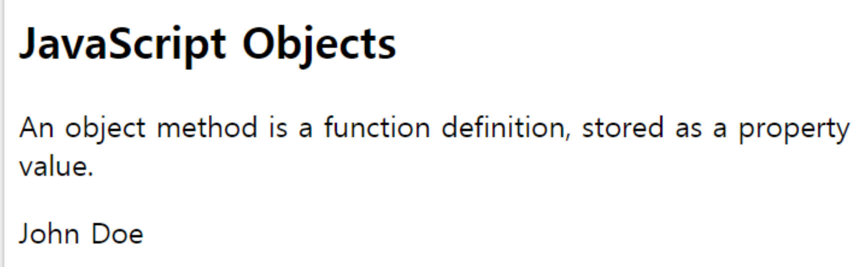
❖ In the example, this is the person object that “owns” the fullname function.

```
var person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```

JavaScript Object Methods



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

<h2>JavaScript Objects</h2>

<p>An object method is a function definition, stored as a
property value.</p>

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

<script>
// Create an object:
var person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
// Display data from the object:
document.getElementById("demo").innerHTML =
person.fullName();
</script>

</body>
</html>
```



JavaScript Objects

An object method is a function definition, stored as a property value.

John Doe

JAVASCRIPT STRING/NUMBER/ARRAY

JavaScript Strings

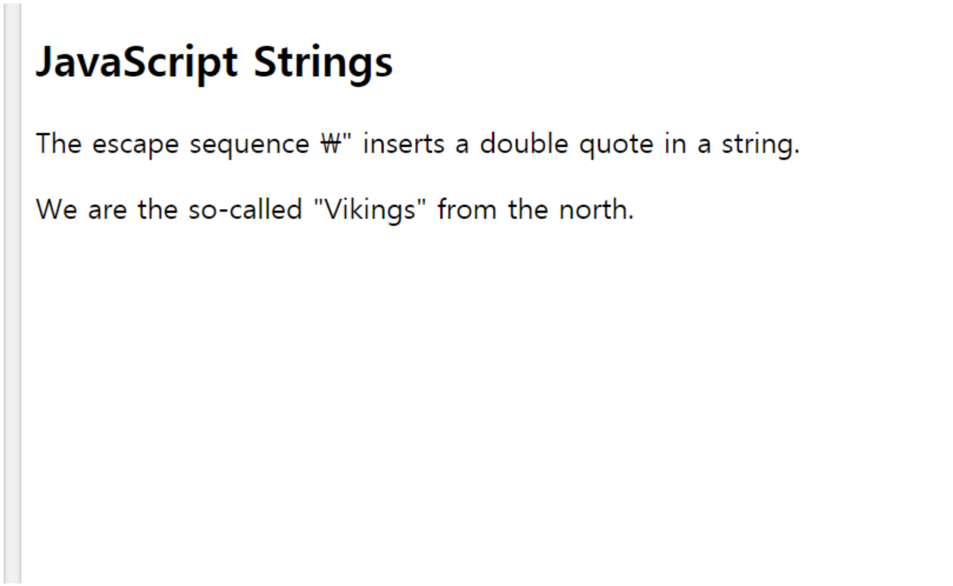
❖ Length Property

```
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
var sln = txt.length;
```

❖ Escape Character

- Single quote, double quote, backslash

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

<h2>JavaScript Strings</h2>

<p>The escape sequence \" inserts a double quote in a
string.</p>

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

<script>
var x = "We are the so-called \"Vikings\" from the north.";
document.getElementById("demo").innerHTML = x;
</script>

</body>
</html>
```



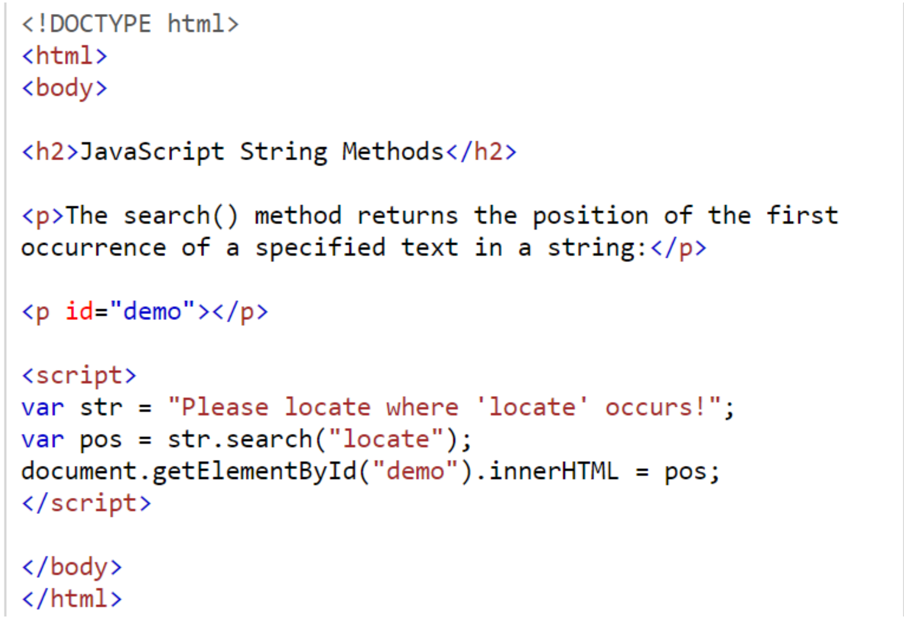
JavaScript Strings

The escape sequence ₩" inserts a double quote in a string.

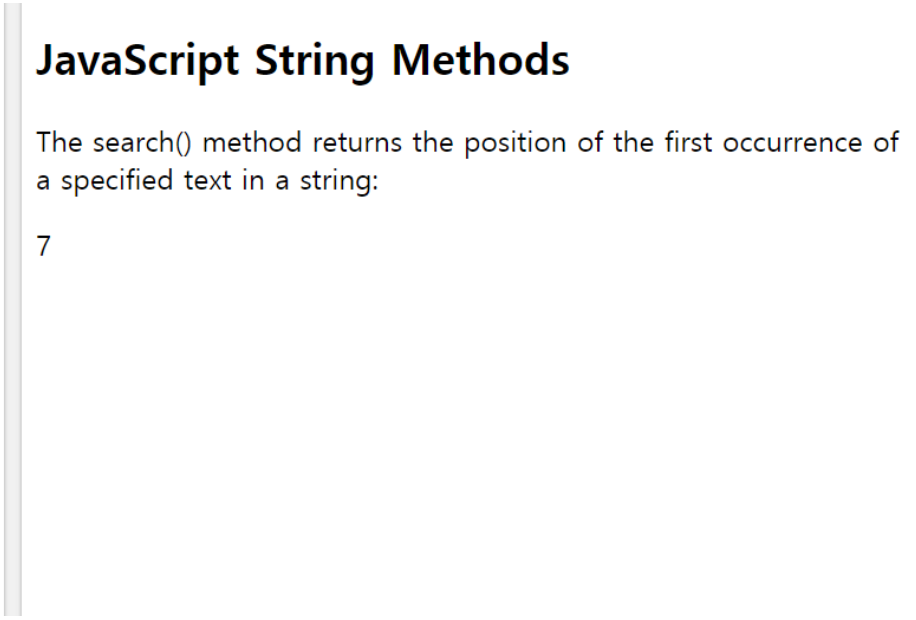
We are the so-called "Vikings" from the north.

JavaScript String Methods

- ❖ A number of methods are provided for String
  - indexOf(), lastIndexOf(), replace() …
- ❖ The search() method searches a string for a specified value and returns the position of the match



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

<h2>JavaScript String Methods</h2>

<p>The search() method returns the position of the first
occurrence of a specified text in a string:</p>

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

<script>
var str = "Please locate where 'locate' occurs!";
var pos = str.search("locate");
document.getElementById("demo").innerHTML = pos;
</script>

</body>
</html>
```



JavaScript String Methods

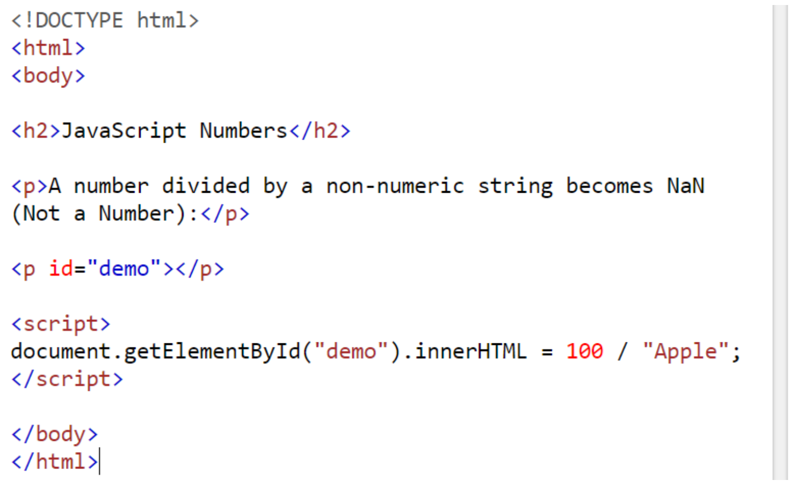
The search() method returns the position of the first occurrence of a specified text in a string:

7

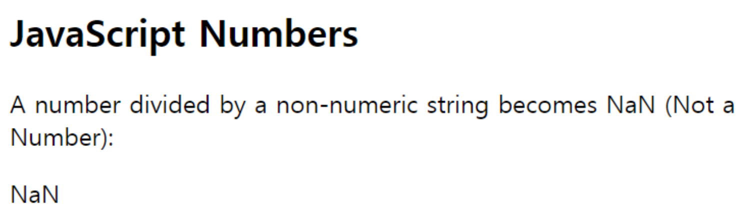
JavaScript Numbers

❖ NaN is a JavaScript reserved word indicating that a number is not a legal number

❖ Trying to do arithmetic with a non-numeric string will result in NaN



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

<h2>JavaScript Numbers</h2>

<p>A number divided by a non-numeric string becomes NaN
(Not a Number):</p>

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

<script>
document.getElementById("demo").innerHTML = 100 / "Apple";
</script>

</body>
</html>
```



```
JavaScript Numbers

A number divided by a non-numeric string becomes NaN (Not a
Number):

NaN
```

❖ JavaScript interprets numeric constants as hexadecimal if they are preceded by 0x.

```
var x = 0xFF;          // x will be 255
```

JavaScript Number Methods

- ❖ **A number of methods are provided for String**
- ❖ **toString()**
  - Returns a number as a string

```
var x = 123;
x.toString();            // returns 123 from variable x
(123).toString();        // returns 123 from literal 123
(100 + 23).toString();   // returns 123 from expression 100 + 23
```

- ❖ **parseInt()**
  - Parses its argument and returns an integer
  - Spaces are allowed. Only the first number is returned:

```
parseInt("10");         // returns 10
parseInt("10.33");      // returns 10
parseInt("10 20 30");   // returns 10
parseInt("10 years");   // returns 10
parseInt("years 10");   // returns NaN
```

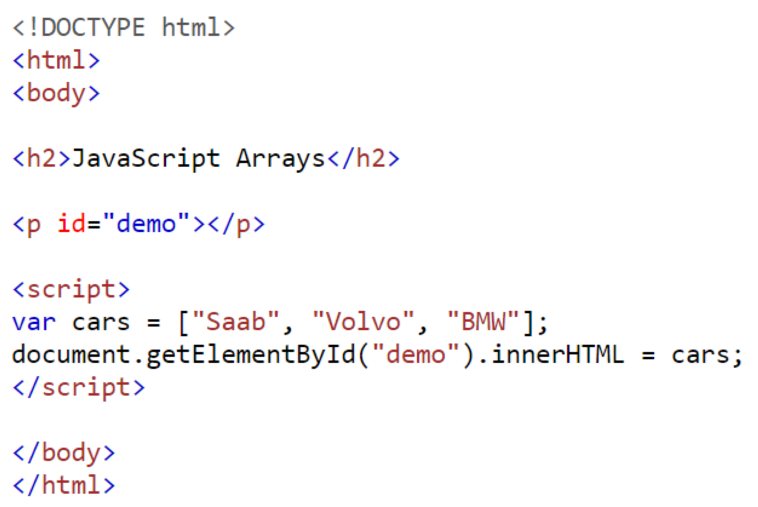
JavaScript Arrays

❖ **JavaScript arrays are used to store multiple values in a single variable.**

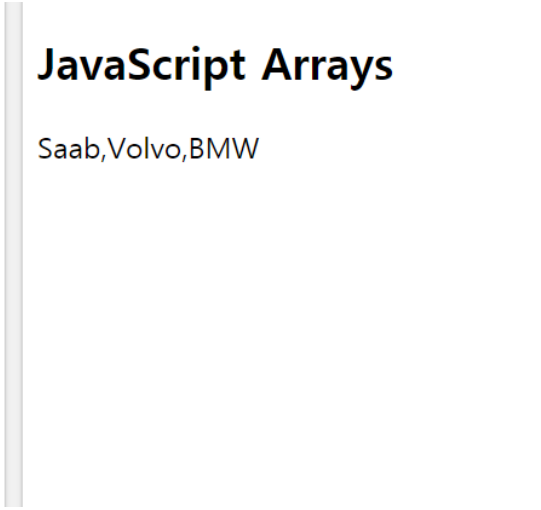
- var *array_name* = [*item1*, *item2*, ...];

❖ **An array can hold many values under a single name, and you can access the values by referring to an index number.**

- var name = cars[0];
- cars[0] = "Opel";



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

<h2>JavaScript Arrays</h2>

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

<script>
var cars = ["Saab", "Volvo", "BMW"];
document.getElementById("demo").innerHTML = cars;
</script>

</body>
</html>
```



JavaScript Arrays

```
Saab,Volvo,BMW
```

JavaScript Array Methods

❖ The pop() method removes the last element from an array

❖ The push() method adds a new element to an array (at the end)

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
var x = fruits.pop();      // the value of x is "Mango"
```

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
var x = fruits.push("Kiwi");   //  the value of x is 5
```

❖ The shift() method removes the first array element and “shift” all other elements to a lower index

❖ The unshift() method adds a new element to an array (at the beginning), and “unshift” older elements:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
var x = fruits.shift();    // the value of x is "Banana"
```

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.unshift("Lemon");    // Adds a new element "Lemon" to fruits
```

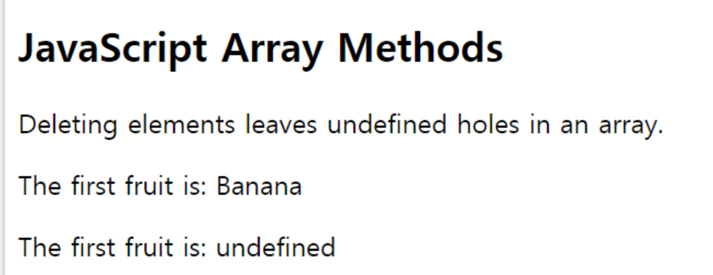
JavaScript Array Methods

❖ Changing Elements

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits[0] = "Kiwi";        // Changes the first element of fruits to
```

❖ Deleting Elements

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

<h2>JavaScript Array Methods</h2>

<p>Deleting elements leaves undefined holes in an array.
</p>

<p id="demo1"></p>
<p id="demo2"></p>

<script>
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML =
"The first fruit is: " + fruits[0];
delete fruits[0];
document.getElementById("demo2").innerHTML =
"The first fruit is: " + fruits[0];
</script>

</body>
</html>
```



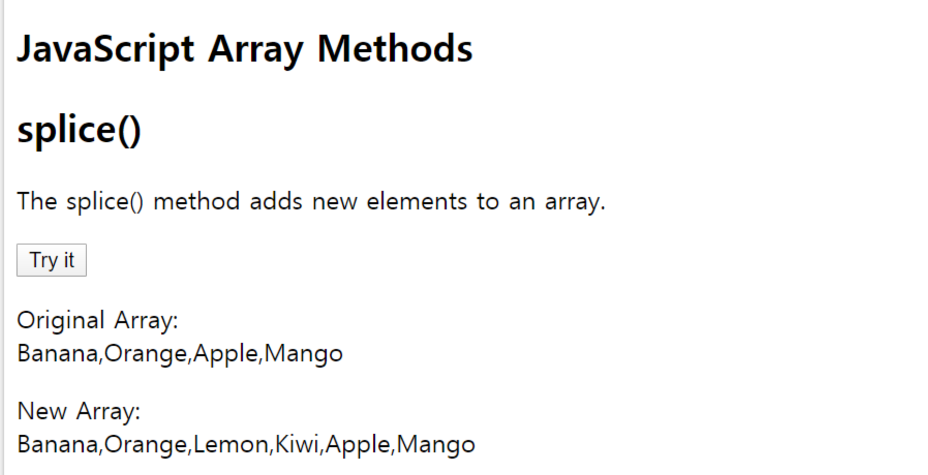
```
JavaScript Array Methods

Deleting elements leaves undefined holes in an array.

The first fruit is: Banana

The first fruit is: undefined
```

JavaScript Array Methods

❖ Splice() method

- The first parameter defines the position where new elements should be added (spliced in).
- The second parameter defines how many elements should be removed.
- The rest of the parameters define the new elements to be added.

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

<h2>JavaScript Array Methods</h2>

<h2>splice()</h2>

<p>The splice() method adds new elements to an array.</p>

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

<p id="demo1"></p>
<p id="demo2"></p>

<script>
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = "Original
Array:<br>" + fruits;
function myFunction() {
  fruits.splice(2, 0, "Lemon", "Kiwi");
  document.getElementById("demo2").innerHTML = "New Array:
<br>" + fruits;
}
</script>

</body>
</html>
```



JavaScript Array Methods

splice()

The splice() method adds new elements to an array.

Try it

Original Array:
Banana,Orange,Apple,Mango

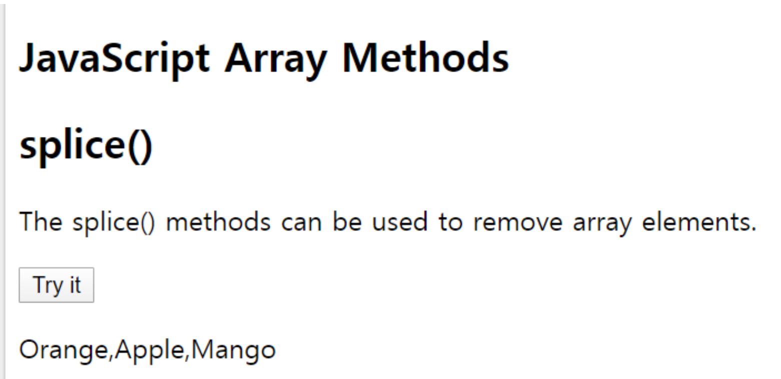
New Array:
Banana,Orange,Lemon,Kiwi,Apple,Mango

PUSAN NATIONAL UNIVERSITY 부산대학교

JavaScript Elements

❖ Using splice() to Remove Elements

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

<h2>JavaScript Array Methods</h2>

<h2>splice()</h2>

<p>The splice() methods can be used to remove array
elements.</p>

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

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

<script>
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits;
function myFunction() {
  fruits.splice(0, 1);
  document.getElementById("demo").innerHTML = fruits;
}
</script>

</body>
</html>
```



JavaScript Array Methods

splice()

The splice() methods can be used to remove array elements.

```
Orange,Apple,Mango
```

JAVASCRIPT CONDITIONS/SWITCH

JavaScript Conditions

❖ **Conditional statements are used to perform different actions based on different conditions.**

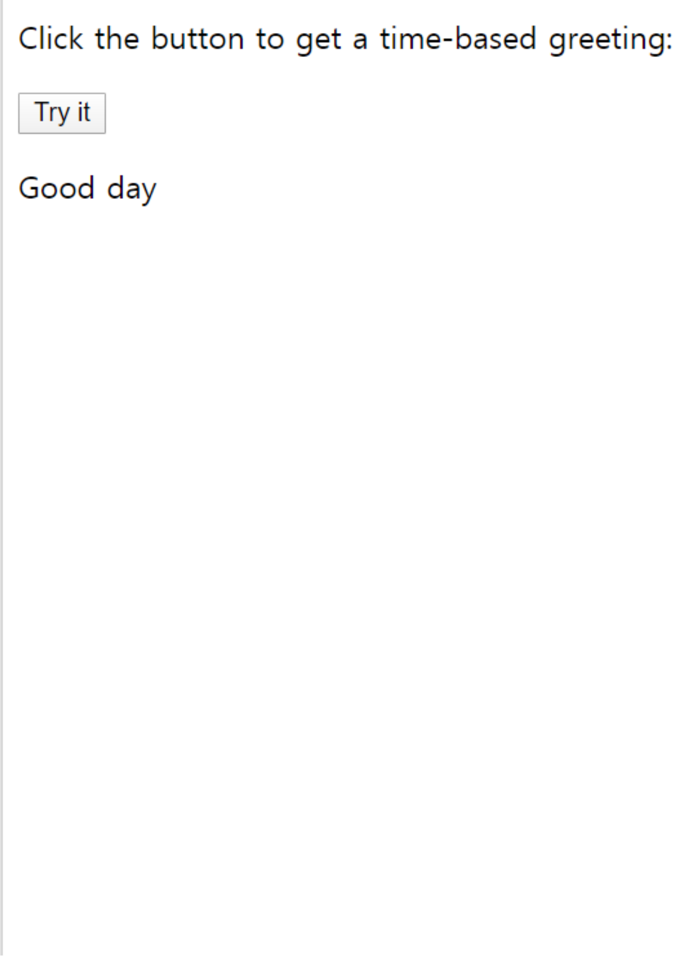
- Use **if** to specify a block of code to be executed, if a specified condition is true
- Use **else** to specify a block of code to be executed, if the same condition is false
- Use **else if** to specify a new condition to test, if the first condition is false
- Use **switch** to specify many alternative blocks of code to be executed

```
if (condition1) {
  //  block of code to be executed if condition1 is true
} else if (condition2) {
  //  block of code to be executed if the condition1 is false and condition2
is true
} else {
  //  block of code to be executed if the condition1 is false and condition2
is false
}
```

부산대학교 PUSAN NATIONAL UNIVERSITY

JavaScript Conditions

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

<p>Click the button to get a time-based greeting:</p>

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

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

<script>
function myFunction() {
  var greeting;
  var time = new Date().getHours();
  if (time < 10) {
    greeting = "Good morning";
  } else if (time < 20) {
    greeting = "Good day";
  } else {
    greeting = "Good evening";
  }
  document.getElementById("demo").innerHTML = greeting;
}
</script>

</body>
</html>
```



Click the button to get a time-based greeting:

Good day

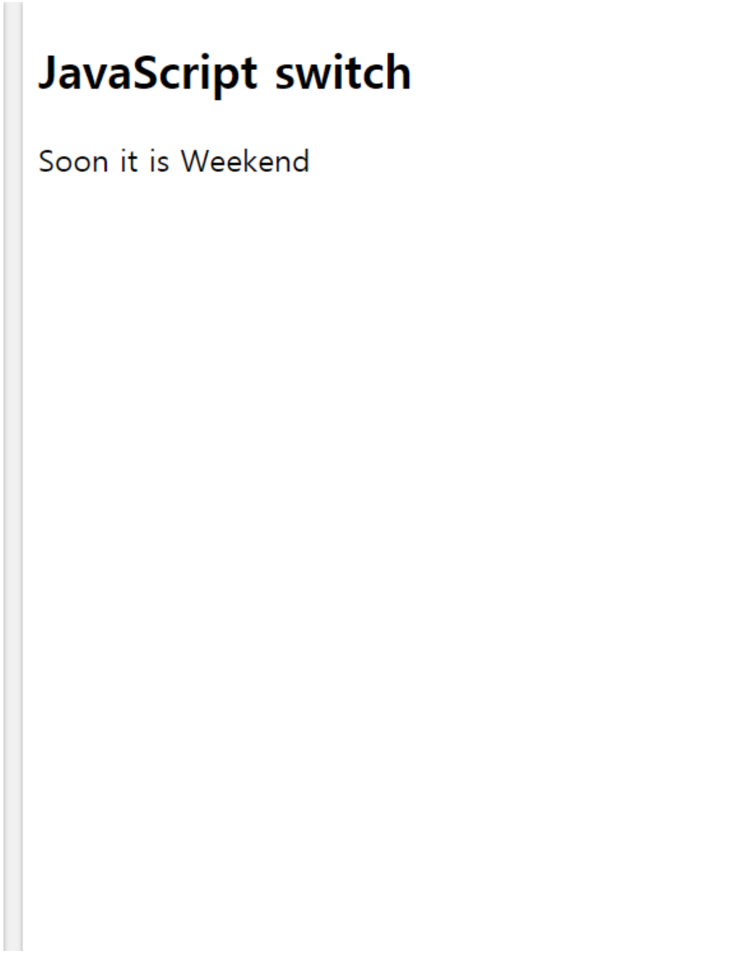
JavaScript Switch

- ❖ **Use the switch statement to select one of many code blocks 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**
- ❖ **Switching details**
  - If multiple case matches a case value, the first case is selected.
  - If no matching cases are found, the program continues to the default label.
  - If no default label is found, the program continues to the statement(s) after the swtich

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

JavaScript Switch

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

<h2>JavaScript switch</h2>

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

<script>
var text;
switch (new Date().getDay()) {
  case 4:
  case 5:
    text = "Soon it is Weekend";
    break;
  case 0:
  case 6:
    text = "It is Weekend";
    break;
  default:
    text = "Looking forward to the Weekend";
}
document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```



JavaScript switch

Soon it is Weekend

JAVASCRIPT LOOP FOR/WHILE

JavaScript Loops

❖ JavaScript Loops

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

```
text += cars[0] + "<br>";
text += cars[1] + "<br>";
text += cars[2] + "<br>";
text += cars[3] + "<br>";
text += cars[4] + "<br>";
text += cars[5] + "<br>";
```

```
var i;
for (i = 0; i < cars.length; i++) {
  text += cars[i] + "<br>";
}
```

❖ Different Kinds of Loops

- For – loops through a block of code a number of times
- For/in – loops through the properties of an object
- For/of – loops through the values of an iterable 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

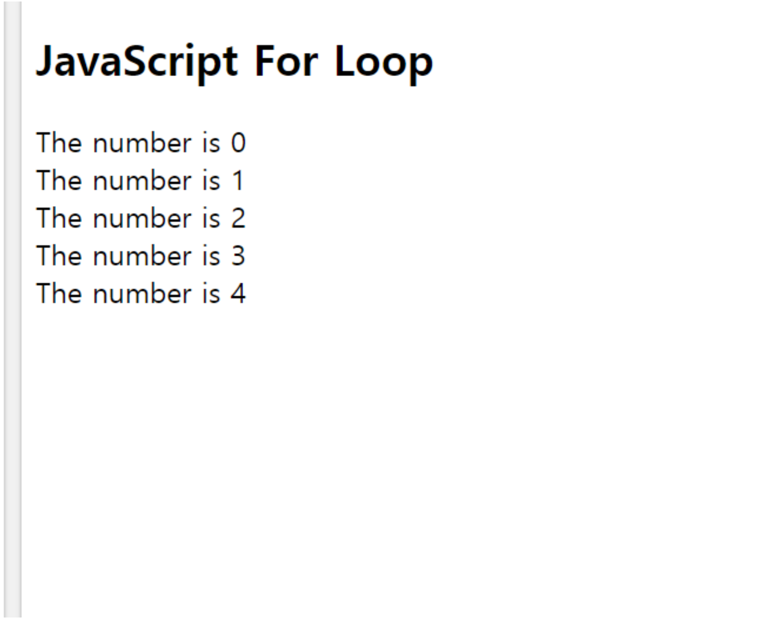
JavaScript For Loop

❖ Syntax

- Statement1 is executed (one time) before the execution of the code block
- Statement2 defines the condition for executing the code block
- Statement3 is executed (every time) after the code block has been executed

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

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

<h2>JavaScript For Loop</h2>

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

<script>
var text = "";
var i;
for (i = 0; i < 5; i++) {
  text += "The number is " + i + "<br>";
}
document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```



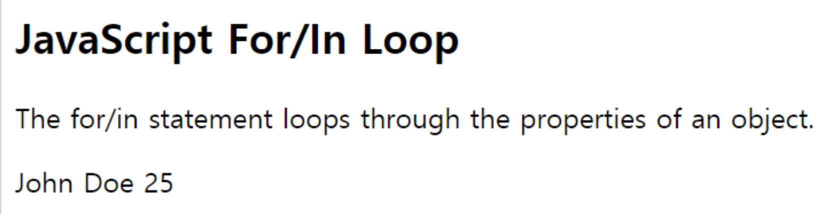
JavaScript For Loop

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
```

JavaScript For/in Loop

- The JavaScript for/in statement loops through the properties of an object

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

<h2>JavaScript For/In Loop</h2>

<p>The for/in statement loops through the properties of an
object.</p>

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

<script>
var txt = "";
var person = {fname:"John", lname:"Doe", age:25};
var x;
for (x in person) {
  txt += person[x] + " ";
}
document.getElementById("demo").innerHTML = txt;
</script>

</body>
</html>
```



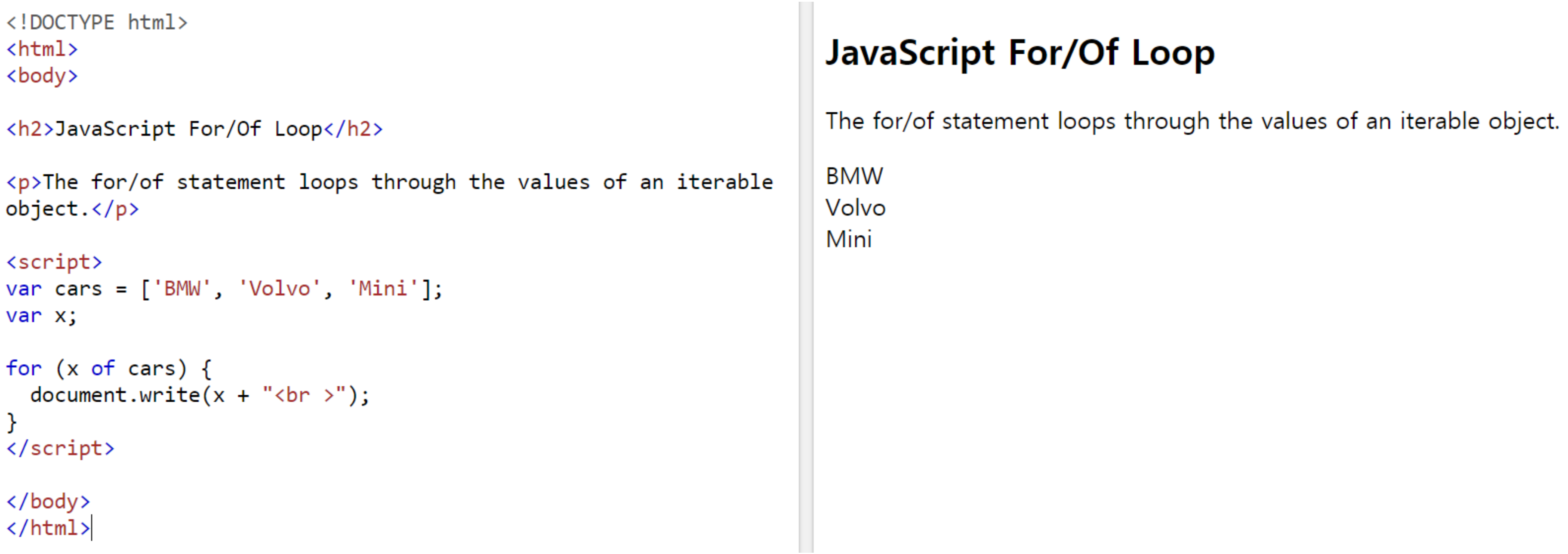
JavaScript For/In Loop

The for/in statement loops through the properties of an object.

John Doe 25

JavaScript For/Of Loop

❖ The JavaScript for/of statement loops through values of an iterable objects



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

<h2>JavaScript For/Of Loop</h2>

<p>The for/of statement loops through the values of an iterable
object.</p>

<script>
var cars = ['BMW', 'Volvo', 'Mini'];
var x;

for (x of cars) {
  document.write(x + "<br >");
}
</script>

</body>
</html>
```

```
JavaScript For/Of Loop

The for/of statement loops through the values of an iterable object.

BMW
Volvo
Mini
```

부산대학교 PUSAN NATIONAL UNIVERSITY

JavaScript While Loop

❖ While

- loops through a block of code as long as a specified condition is true.

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

```
while (i < 10) {
  text += "The number is " + i;
  i++;
}
```

❖ Do/while

- This 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.

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

```
var i = 11;

do {
  text += "<br>The number is " + i;
  i++;
}
while (i < 10);
```

요 약

- JavaScript Objects
- JavaScript String/Number/Array
- JavaScript Conditions/Switch
- JavaScript Loop For/While