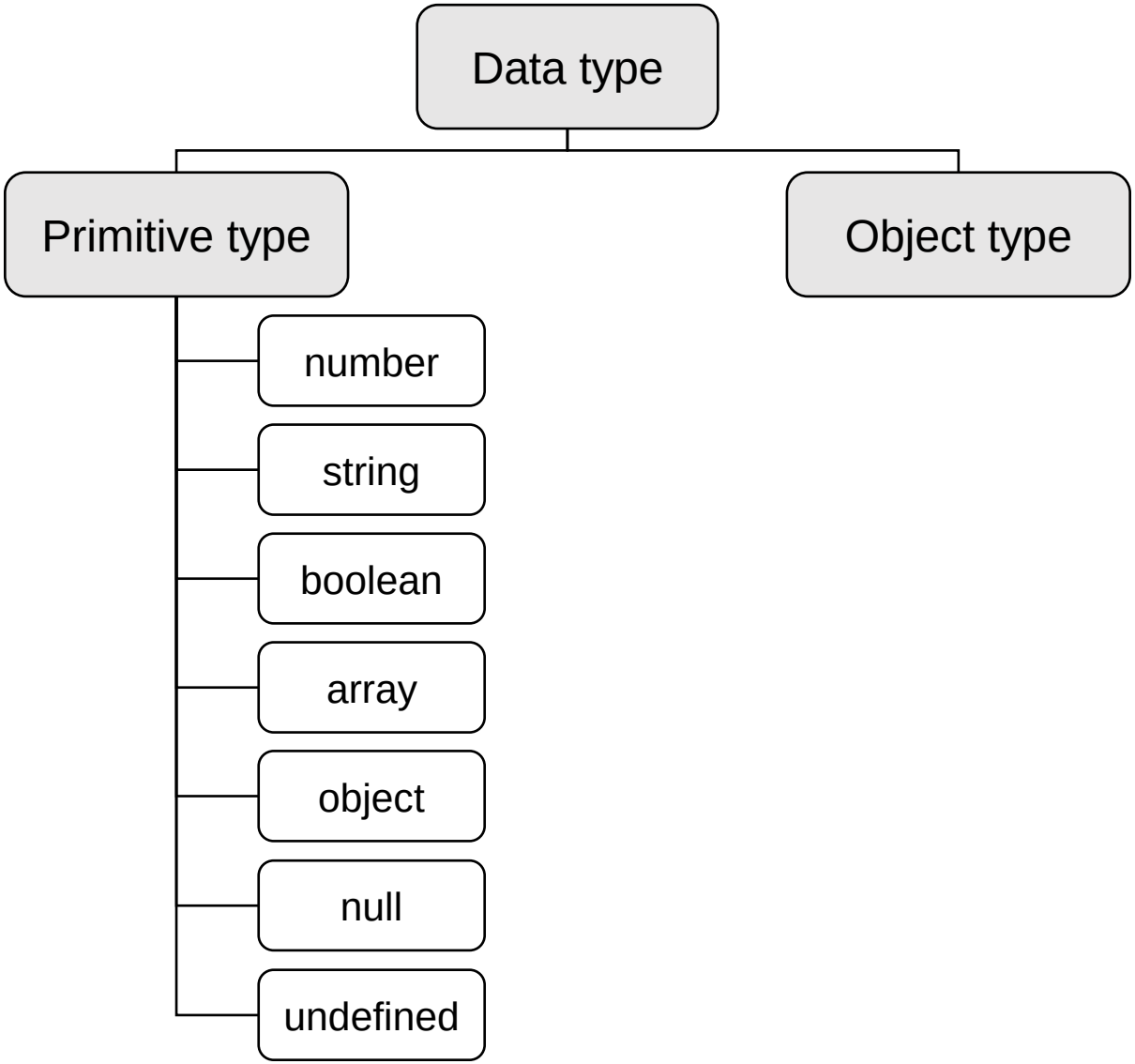
Lab 08. JavaScript 02

인터넷과웹기초

Data Type in JS



Data Type in JS (cont'd)

- Checking data type using the typeof operator

number
number
string
boolean
object
object
undefined
object

```
<html>
    <head></head>
    <body>
        <script>
            var num;
            var obj=null;
            document.write(typeof 100 + "<br>");
            document.write(typeof 10.5 + "<br>");
            document.write(typeof "name" + "<br>");
            document.write(typeof true + "<br>");
            document.write(typeof [1,2,3] + "<br>");
            document.write(typeof {name:'name'} + "<br>");
            document.write(typeof num + "<br>");
            document.write(typeof obj + "<br>");
        </script>
    </body>
</html>
```

Control statement

Control statement

- Control statement in JavaScript

type	description	structure
conditional statements	Selectively executes the following statements according to conditions	• if • if~else • multiple if~else • switch~case
loop statements	Processing the same instruction multiple times or processing a specific operation repeatedly	• for • while • do~while
break or continue statements	If a conditional statement is encountered, skip or terminate the iteration.	• continue • break

`if~else` statement

- `if` statement

```
if(condition) {
     statement;
}
```

```
if(condition_a) {
     statement;
     if(condition_b) {
          statement;
     }
}
```

- `if~else` statement

```
if (condition) {
     statement;
}
else {
     statement;
}
```

- example: if-else.js

multiple `if~else`

- multiple `if~else` statement

```
if (condition A) {
     statement;
}
else if (condition B) {
     statement;
}
else if (condition C) {
     statement;
}
else {
     statement;
}
```

- example: multiple_if-else.js

switch~case statement

- switch~case statement

```
switch (integer) {
     case n:
          statement;
          break;
     case n:
          statement;
          break;
     default:
          statement;
}
```

- example: switch-case.js

for statement

- for statement

```
for (initialization; condition; final-expression) {
     statement;
}
```

  - initialization : 반복 변수값 초기화함
  - condition : 블록 내 문장을 얼마나 반복할지 결정함
  - final-expression : 초기화한 변수의 값을 증가 혹은 감소시킴

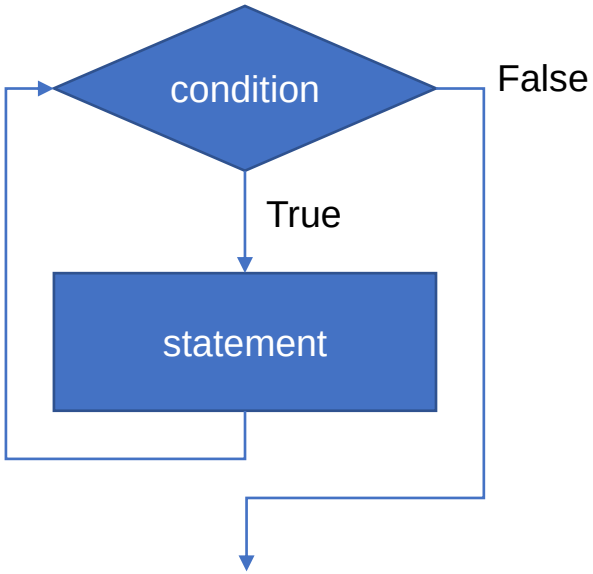
- example: for.js

`while` statement

- `while` statement

```
while (condition) {
     statement;
}
```

  - condition → True or False



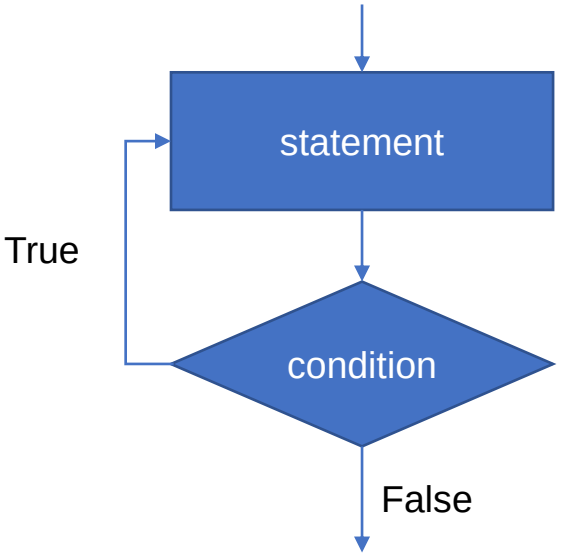
  - example: while.js

do~while statement

- do~while statement

```
do {
     statement;
} while(condition);
```

  - condition → True or False



  - example: do-while.js

`break` and `continue`

- `break`
  - get out of loop (for, while, do~while)
  - example: break.js

- continue
  - return to the start of loop
  - example: continue.js

Object

Object

- Properties
  - object.property
  - object[property]
  - property → name:value pair
- Methods
  - object.method()

Object	Properties	Methods
	car.name = Fiat	car.start()
	car.model = 500	car.drive()
	car.weight = 850kg	car.brake()
	car.color = white	car.stop()

Object: creation

- 1. Create an object and assign it to an object variable

```
var car = {
     name: ‘Booroong’,
     speed: 50,
     color: ‘silver’,
     door: 4,
     start: function () {
          return this.speed+10;
     }
};
```

- 2. Create an object using a constructor function

```
var car = new Object();
     car.name=‘Booroong’;
     car.speed=50;
     car.color=‘silver’;
     car.speedup = function() {
          return this.speed+10;
     };
```

Object: creation

- 3. Defining a constructor function

```
function Car(name, color, speed) {
     this.name=name;
     this.color=color;
     this.speed=speed;
     this.speedup=function() {
          return this.speed+10;
     };
     this.speeddown=function() {
          return this.speed-10;
     };
}
```

```
var 14바2087 = new Car(‘sedan’, ‘white’, 50);
var 50도9931 = new Car(‘suv’, ‘black’, 30);
var 135사1125 = new Car(‘hatchback’, ‘white’, 60);
```

Object: example

- control_car.html

```
var car = {
name: 'suv',
speed: 100,
color: 'white',
speedup: function(param) {
    var sp = this.speed+param;
    if(sp>=300) {
        sp=300
        return sp
        console.log("speed limit: 300")
    }
    else {
        return sp;
    }
},
speeddown: function(param) {
    var sp = this.speed-param;
    if(sp<0) {
        sp = 0;
        return sp;
    }
    else {
        return sp;
    }
}
};
```

Array

Array

- Array

index	0	1	2	3	4	5	6	7	8	9
element	a	b	c	d	e	f	g	h	i	j

  - size of array : 10
  - index : 0 ~ 9
  - data of index 5 : f

Array

- create an array with array literal

```
var array = [element-1, element-2, element-3, … ];
```

- create an array and assign elements

```
var array = [];
array[0] = ‘A’;
array[1] = ‘B’;
array[2] = ‘C’;
```

- assign elements with different data types

```
var x = 5;
var array = [100, ‘string’, true, x];
```

- examples
  - array01.js
  - array02.js (NaN?)

Array: methods

- splice
  - add or remove elements from an array
  - splice(arg1, arg2, arg3, …);
    - arg1 : index to start
    - arg2 : number to remove
    - arg3, … : data to add
  - example: splice.js

```
var data=['a', 'b', 'c', 'd', 'e’]

var str1=data.splice(1, 2)

var str2=data.splice(1, 1, 'f', 'g’)

var str3=data.splice(2, Number.MAX_VALUE)
```

Array: methods

- slice
  - creates an array by selecting only the elements within a specific range of the array.
  - If the argument is negative, the last index is considered -1
  - example: slice.js

```
var data=['a','b','c','d','e','f','g’]

var str1=data.slice(0, 4)

var str2=data.slice(2, -1)

var str3=data.slice(-4, -2)
```

Array: methods

- join
  - converts all elements stored in an array to a string, then concatenates and prints it
  - example: join.js

```
var arr=['Apple', 'Lemon', 'Melon']
var join1 = arr.join()
var join2 = arr.join('-')
var join3 = arr.join(' and ')
console.log(join1)
console.log(join2)
console.log(join3)
```

Array: methods

- concat
  - Combining each data in an array or combining different array objects
  - example: concat.js

```
var arr=['Apple', 'Lemon', 'Melon']
var join1 = arr.join()
var join2 = arr.join('-')
var join3 = arr.join(' and ')
console.log(join1)
console.log(join2)
console.log(join3)
```

- reverse
  - reverse the order of elements in an array
  - example: reverse.js

```
var data=[1, 2, 3, 4, 5, 6, 7, 8, 9]
console.log(data)
var rData=data.reverse()
console.log(rData)
```

Array: methods

- filter
  - creates a new array by returning only the data for which the condition is true among the data of elements in the array.
  - example: filter.js

```
var data=[22, 40, 35, 17, 29, 10, 33, 28, 16]

function filterArr(value) {
    return value>=20
}

var fData1 = data.filter(filterArr)

var fData2 = data.filter(value => value>=20)
```

Association array

- Association array
  - 연관 배열
  - set index of array to string

```
var array = {key1:value1, key2:value2, …, keyN:valueN};
```

  - example: association_array.js

```
var data={'first':100, 'second':200, 'third':300}

data['fourth']=400
data.fifth=500

console.log(data.first)
console.log(data.second)
console.log(data['third'])
console.log(data['fourth'])
console.log(data['fifth'])
```