Web Programming

JavaScript Programming Basics #2

Instructor: Prof. SoYeop Yoo
School of Computing, Gachon University

CONTENTS

01 Looping

02 Functions

Looping

Looping Statements

Looping

- for loops, for/in loops
- while loops, do/while loops
- break statement
- continue statement

for Loops (1/2)

Syntax

- `for(`*`initial_expression`*`; `*`test_condition`*`; `*`change_expression`*`) {`
  `statements`
  `}`

- Iterates through statements for a number of times controlled by the test_condition

for Loops (2/2)

Example

```
<!DOCTYPE html>
<html>
    <head>
        <title>for Loop</title>
    </head>
    <body>
        <h2>Square numbers of 1 ~ 10</h2>
        <script>
            var counter;
            [redacted]
        </script>
    </body>
</html>
```

Square numbers of 1 ~ 10

1
4
9
16
25
36
49
64
81
100

for/in Loops (1/2)

Syntax

- `for(`*`counter`* `in` *`object`*`) {`
  `statements`
  `}`

- *Counter* and termination are determined by the length of the *object*
- *Statements* begin with *counter=0* and terminate when all the properties of the *object* have been exhausted
- For an array *object*, it terminates when no more elements are found

for/in Loops (2/2)

Example

```
<!DOCTYPE html>
<html>
    <head>
        <title>for/in Loop</title>
    </head>
    <body>
        <h2>for/in with Array</h2>
        <script>
            var books = ["Chinese", "English", "Korean"];
            [redacted]
        </script>
    </body>
</html>
```

for/in with Array

0: Chinese
1: English
2: Korean

while Loops (1/2)

Syntax

- `while(`*`condition`*`) {`
  `statements`
  `increment/decrement_statement`
  `}`

- Begins with checking a termination *condition* and keeps looping until the *condition* is not met

while Loops (2/2)

Example

```
<!DOCTYPE html>
<html>
    <head>
        <title>while Loop</title>
    </head>
    <body>
        <h2>While Loop</h2>
        <script>
            var counter = 100;
            [redacted]
        </script>
    </body>
</html>
```

While Loop

Number: 100
Number: 90
Number: 80
Number: 70
Number: 60
Number: 50
Number: 40
Number: 30
Number: 20
Number: 10
Number: 0

do/while Loops (1/2)

Syntax

- ```
  do {
     statements
     increment/decrement_statement
  } while(condition);
  ```

- Always executes *statements* in the first iteration of the loop
- The termination *condition* is placed at the end of the loop
  ```

do/while Loops (2/2)

Example

```
<!DOCTYPE html>
<html>
    <head>
        <title>do-while Loop</title>
    </head>
    <body>
        <h2>do/while Loop</h2>
        <script>
            var counter = 100;
            [redacted]
        </script>
    </body>
</html>
```

While Loop

Number: 100
Number: 90
Number: 80
Number: 70
Number: 60
Number: 50
Number: 40
Number: 30
Number: 20
Number: 10
Number: 0

Break and Continue(1/3)

break

- Halts execution in the middle of the loop
- Exits the loops

continue

- Halts execution in the middle of the loop and skips the remaining statements
- Continues with the loop

Break and Continue (2/3)

Example: break

```
<!DOCTYPE html>
<html>
    <head>
        <title>break/continue</title>
    </head>
    <body>
        <h2>break</h2>
        <script>
            for (var i = 0; i <= 10; i++) {
                if (i == 3)
                    break;
                document.write("The number is " + i + "<br>");
            }
        </script>
    </body>
</html>
```

break

The number is 0
The number is 1
The number is 2

Break and Continue (3/3)

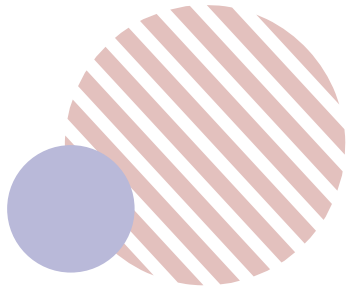
Example: continue

```
<!DOCTYPE html>
<html>
    <head>
        <title>break/continue</title>
    </head>
    <body>
        <h2>continue</h2>
        <script>
            for (var i = 0; i <= 10; i++) {
                [redacted]
                document.write("The number is " + i + "<br>");
            }
        </script>
    </body>
</html>
```

continue

The number is 0
The number is 1
The number is 2
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
The number is 10

Functions



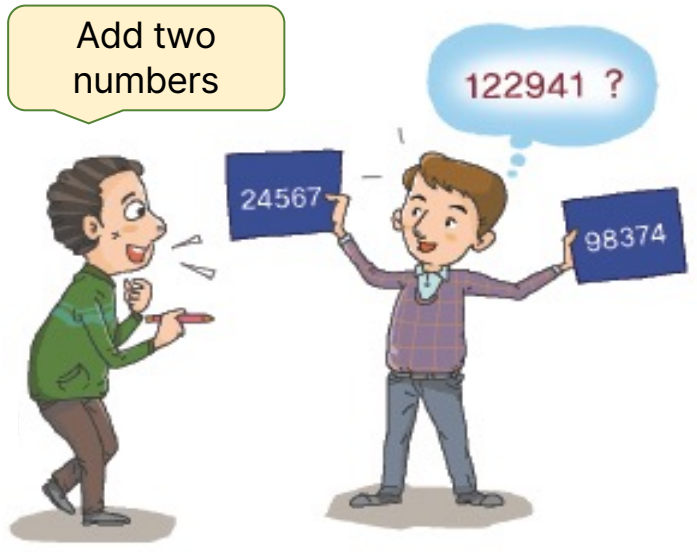
Function

What is a function?

- A block of code that receives data, processes it, and returns the results

Why do we need a function?

- Factor out code for repeated reuse
- (e.g.) a function that adds 2 numbers (24567 and 98374; 33398 and 77732; ...)



Functions

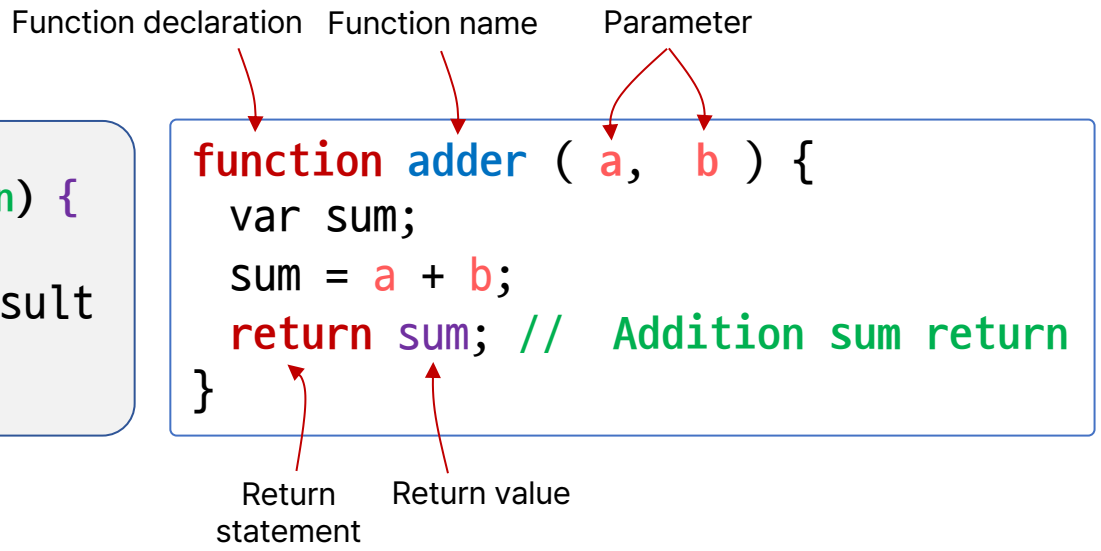
Functions

- Functions you have already used so far
  - `document.write("Hello, World Wide Web!");`
  - `var name = prompt("Input name:","");`
  - `alert(name);`
- A function has:
  - Function name
  - Parameters
  - Return value

Defining and Calling Functions (1/6)

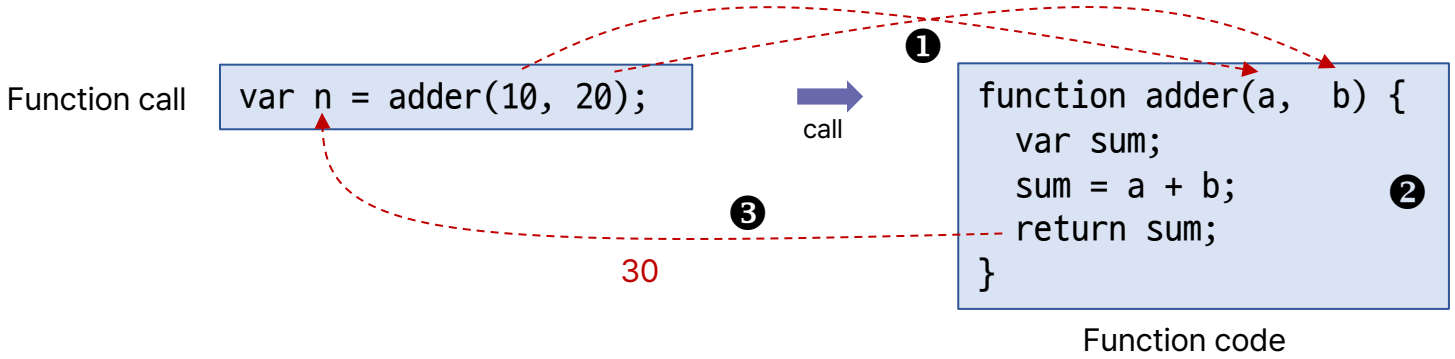
Defining a Function

```
function function_name(arg1, arg2,..., argn) {
    ...program code...
  Return statement that returns the result
}
```



Calling a Function

- The main code uses the name of the function to request the execution of the function



Defining and Calling Functions (2/6)

Function Code Placement

- Functions must be created (defined) before it is called
- Function definition
  - Placed in <head> section
  - <body> is OK, but let's just put it in <head> to make sure it comes before a function call
- Function call
  - Placed in <body> section

```
<html>
  <head>
    <script>
     // function definition comes here
     function myFunc(...) {
       ...
     }
    </script>
  </head>
  <body>
    <script>
     // function calls come here
     var returnValue = myFunc(...);
    </script>
  </body>
</html>
```

Defining and Calling Functions (3/6)

Defining a Function

- User-defined function
- Consists of three parts:
  - Reserved word **function** followed by the function name (identifier)
  - Zero or more parameters required by the function
  - Function statements, delimited by curly braces { }

```
<script type="text/javascript">
  function addStrings(str1, str2) {
    var result = str1 + str2;
    return result;
  }
</script>
```

Defining and Calling Functions (4/6)

Calling a Function

- Statement including the function name followed by a list of arguments in parentheses
- Parameters of function take on the values of arguments passed to the function through the function call
- Do not forget JavaScript is case-sensitive!

```
<script type="text/javascript">
    function addStrings(str1, str2) {
        ...
    }
    var string1 = "Hello ";
    var string2 = "World";
    var result = addStrings(string1,string2);
    document.write(result);
</script>
```

Defining and Calling Functions (5/6)

Function Returning a Value

- Functions may return no value
  - Just perform some tasks
- Functions may return a value
  - `var returnValue = function(parameters);`
  - A return statement must be included in the function definition

Defining and Calling Functions (6/6)

Example

```
<!DOCTYPE html>
<html>
    <head>
        <title>Function</title>
        <script>
            [redacted]
        </script>
    </head>
    <body>
        <h2>A function that returns the square of a number</h2>
        <script>
            var number = prompt("Enter a number: ");
            [redacted]
            document.write("The square of " + number + ": " + result);
        </script>
    </body>
</html>
```

A function that returns the square of a number

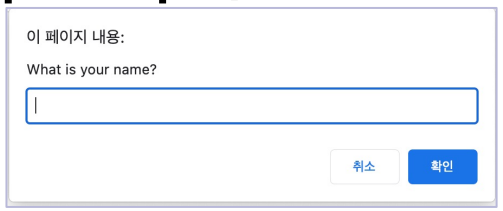
The square of 3: 9

Built-in Functions (1/8)

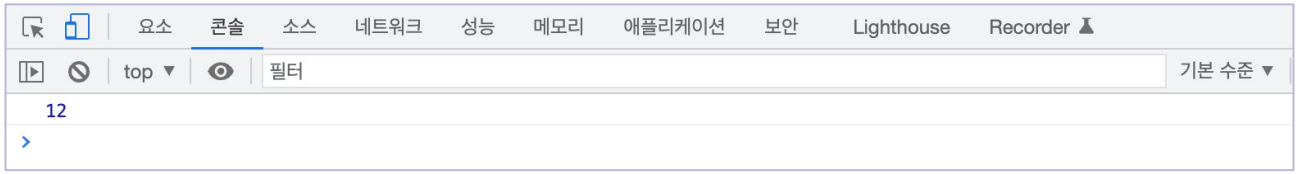
alert()

confirm()

prompt()



console.log()



Built-in Functions (2/8)

Type Casting Global Methods

isNaN(value)	Determines whether a value is an illegal number
parseInt(str)	Parses a string and returns an integer
parseFloat(str)	Parses a string and returns a floating point number
String(value)	Converts an object's value to a string
Number(value)	Converts an object’s value to a number

Built-in Functions (3/8)

isNaN

- isNaN(value)
- Determines whether a value is NaN (Not a Number)
- Returns true if the value is NaN, and false otherwise
- Example

```
<!DOCTYPE html>
<html>
	<head>
		<title>Function</title>
	</head>
	<body>
		<h2>Built-in Global Functions</h2>
		<script>
			document.write(isNaN(123) + "<br/>");
			document.write(isNaN(5 - 2) + "<br/>");
			document.write(isNaN("123") + "<br/>");
			document.write(isNaN("5-2") + "<br/>");
			document.write(isNaN("Hello") + "<br/>");
			document.write(isNaN("2005/12/12") + "<br/>");
		</script>
	</body>
</html>
```

Built-in Global Functions

false
false
false
true
true
true

Built-in Functions (4/8)

parseInt

- parseInt(string, radix)
- Converts a string into an integer
  - radix - specifies numeral system: 16 (hexadecimal), 2 (binary), 10 (decimal, default)
  - If the string begins with “0x”, the radix is 16 (hexadecimal)
  - If the string begins with any other value, the radix is 10 (decimal)
  - e.g., parseInt(“0x10”) = 16, parseInt(“2”)+3=5, parseInt(“11”, 2)=3
- Note:
  - Only the first number in the string is returned!
  - Leading and trailing spaces are allowed
  - If the first character cannot be converted to a number, parseInt() returns NaN

Built-in Functions (5/8)

parseInt

Example

```
<!DOCTYPE html>
<html>
	<head>
		<title>Function</title>
	</head>
	<body>
		<h2>Built-in Global Functions</h2>
		<script>
			document.write(parseInt("10") + "<br/>");
			document.write(parseInt("10.33") + "<br/>");
			document.write(parseInt("34 45 66") + "<br/>");
			document.write(parseInt(" 60 ") + "<br/>");
			document.write(parseInt("40 years") + "<br/>");
			document.write(parseInt("He was 40") + "<br/>");
			document.write(parseInt("10", 10) + "<br/>");
			document.write(parseInt("010") + "<br/>");
			document.write(parseInt("10", 8) + "<br/>");
			document.write(parseInt("0x10") + "<br/>");
			document.write(parseInt("10", 16) + "<br/>");
		</script>
	</body>
</html>
```

Built-in Global Functions

10
10
34
60
40
NaN
10
10
8
16
16

Built-in Functions (6/8)

parseFloat

- parseFloat(object)
- Finds a floating-point value at the beginning of a string

Number

- Number(object)
- Converts an object argument into a number that represents the object’s value

String

- String(object)
- Converts an object argument into a string that represents the object’s value

Built-in Functions (7/8)

Example

```
<!DOCTYPE html>
<html>
	<head>
		<title>Function</title>
	</head>
	<body>
		<h2>Built-in Global Functions</h2>
		<script>
			document.write(parseFloat("10.33") + "<br/>");
			document.write(Number("10.33") + "<br/>");
			document.write(Number("40 years") + "<br/>");
			document.write(String(10.33) + "<br/>");
			document.write(String(true) + "<br/>");
		</script>
	</body>
</html>
```

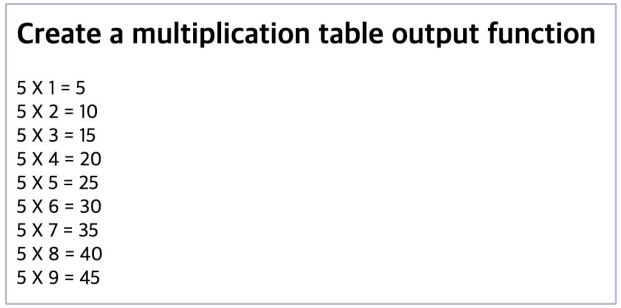
Built-in Global Functions

10.33
10.33
NaN
10.33
true

Built-in Functions (8/8)

Example

Multiplication table




```
<!DOCTYPE html>
<html>
	<head>
		<title>Function</title>
		<script>
			function multiplicationTable(n) {
				// converting string n to number
				[redacted]

				if (isNaN(num) || n < 1 || n > 9) {
					alert("You entered incorrectly.");
				} else {
					// show the multiplication table of num
					[redacted]
				}
			}
		</script>
	</head>
	<body>
		<h2>Create a multiplication table output function</h2>
		<script>
			var n = prompt("Enter the dan you want?");

			multiplicationTable(n);
		</script>
	</body>
</html>
```

Exercises

Exercise 1

Exercise 1: FizzBuzz

- Iterate through all numbers from 1 to 45
- Print the following:
  - For multiples of 3 print “Fizz”
  - For multiples of 5 print “Buzz”
  - For multiples of 3 and 5 print “FizzBuzz”

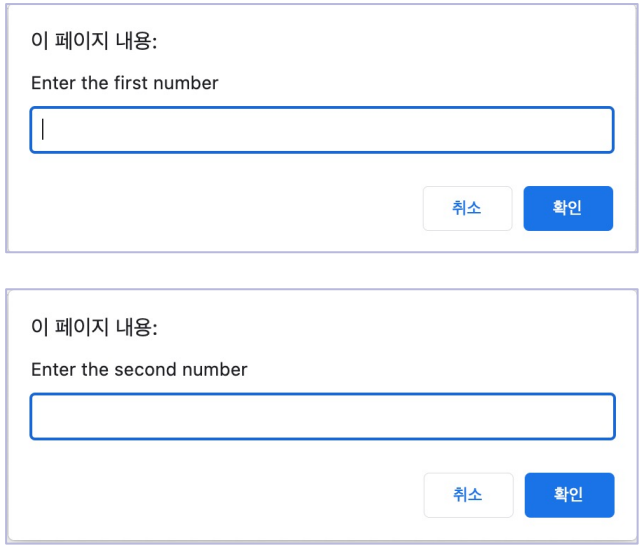
FizzBuzz

1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
17
Fizz
19
Buzz
Fizz
22
23
Fizz
Buzz

Exercise 2

Exercise 2

- Create two functions add() and sub()
  - add(num1, num2) returns num1+num2; sub(num1, num2) returns num1-num2
- Get two integers using prompt()
- Then print the results of two functions using document.write()

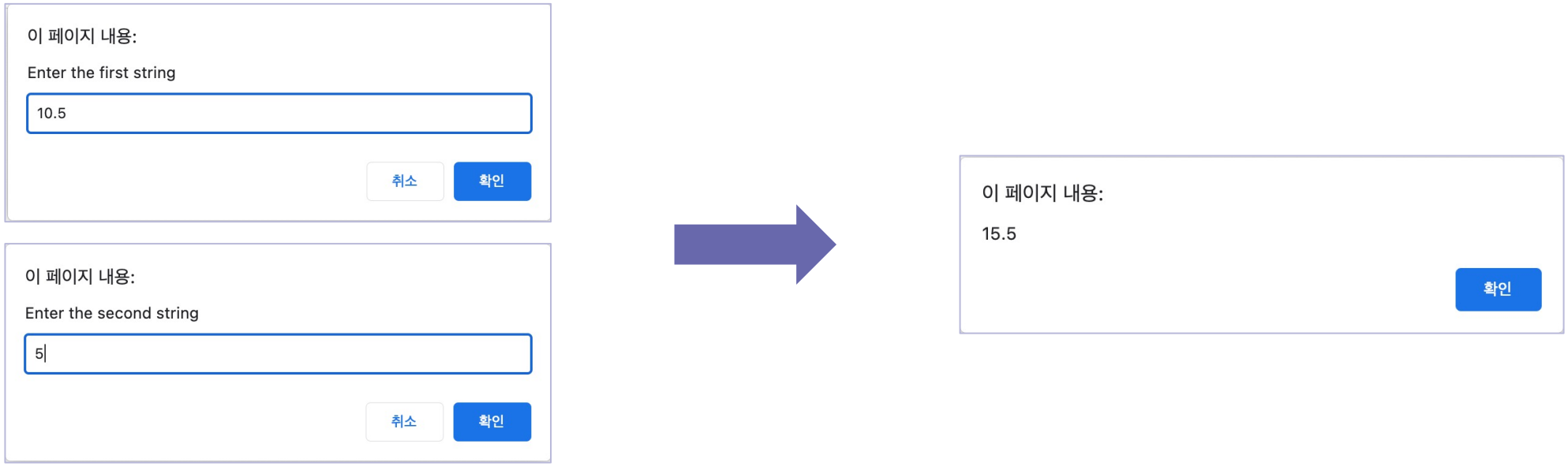


3 + 2 = 32
3 - 2 = 1

Exercise 3

Exercise 3

- Write a function
  - Given two string parameters, converts them into integer or floating-point values, adds them, converts the result into a string, and then returns the string
- Get two strings using prompt()
- Show the final result using alert()



End of Class