

7. JS Tutorial

웹프로그래밍

2016년 1학기

충남대학교 컴퓨터공학과

목차

- ❏ JS Introduction
- ❏ JS Where To, Output, Syntax
- ❏ JS Statements and Comments
- ❏ JS Variables & Operators
- ❏ JS Data Types: Numbers, Strings, Arrays, and more
- ❏ JS Functions
- ❏ JS Objects
- ❏ JS Scope
- ❏ JS Events
- ❏ JS Control Statements

JavaScript Introduction

- ❏ Some examples of what JavaScript can do.
- ❏ JavaScript can change HTML content. *Try it!*
- ❏ JavaScript can change HTML attributes. *Try it!*
- ❏ JavaScript can change HTML Styles(CSS). *Try it!*
- ❏ JavaScript can validate data. *Try it!*

JavaScript Introduction (cont'd)

❏ Did you know?

- ❏ JavaScript and Java are completely different languages, both in concept and design.
- ❏ JavaScript was invented by Brendan Eich in 1995, and became an ECMA standard in 1997.
- ❏ ECMA-262 is the official name. ECMAScript 6 (June 2015) is the current standard.

JavaScript Where To

- JavaScript can be placed in the <body> and the <head> sections of an HTML page.

- The <script> Tag

- JavaScript code must be inserted between <script> and </script> tags.

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

- JavaScript Functions and Events

- JavaScript **function** is a block of JavaScript code, that can be executed when "asked" for.

For example, a function can be executed when an **event** occurs, like when the user clicks a button.

JavaScript Where To (cont'd)

- ❏ JavaScript in <head> *Try it!*
- ❏ JavaScript in <body> *Try it!*
- ❏ External JavaScript *Try it!*
 - ❏ File extension .js
 - ❏ To use an external script, put the name of the script file in the src (source) attribute of the <script> tag.
 - ❏ External scripts cannot contain <script> tags.
- ❏ External JavaScript Advantages
 - ❏ It separates HTML and code.
 - ❏ It makes HTML and JavaScript easier to read and maintain.
 - ❏ Cached JavaScript files can speed up page load.

JavaScript Output

JavaScript Display Possibilities

- Writing into an alert box, using `window.alert()` *Try it!*
- Writing into the HTML output using `document.write()` *Try it!*
- Writing into an HTML element, using `innerHTML` *Try it!*
- Writing into the browser console, using `console.log()` *Try it!*

JavaScript Syntax

JavaScript Programs *Try it!*

- JavaScript is a programming language.
- JavaScript statements are separated by semicolon.
- In HTML, JavaScript programs can be executed by the web browser.

JavaScript Statements

- Composed of : Values, Operators, Expressions, Keywords, and Comments

JavaScript Values

- Two types of values : Fixed values and variable values
 - Fixed values are called literals.
 - Variable values are called variables.

JavaScript Syntax (cont'd)

JavaScript Literals

- Numbers are written with or without decimals. *Try it!*
- Strings are text, written within double or single quotes. *Try it*
- Expression can also represent fixed values. *Try it!*

JavaScript Variables *Try it!*

- Used to store data values.
- JavaScript uses the **var** keyword to define variables.
- Equal sign** is used to **assign values** to variables.

JavaScript Syntax (cont'd)

JavaScript Operators

- Assignment operator (=) *Try it!*
- Arithmetic operators (+ - * /) to compute values. *Try it!*

JavaScript Keywords

- Used to identify actions to be performed.
 - The **var** keyword tells the browser to create a new variable.

JavaScript Comments

- Single line comment : // *Try it!*
- Multi-line Comment : /* and end with */

JavaScript Syntax (cont'd)

JavaScript is Case Sensitive

- The variables **lastName** and **lastname**, are two different variables.

Try it!

- *JavaScript does not interpret **VAR** or **Var** as the keyword **var**.*



JavaScript and Camel Case

- In JavaScript, camel case often starts with a lowercase letter:
 - firstName, lastName, masterCard, interCity
- Hypes are not allowed in JavaScript. It is reserved for subtractions.

JavaScript Statements

- ❏ In HTML, JavaScript statements are “instructions” to be “executed” by the web browser.
- ❏ JavaScript Statements *Try it!*
- ❏ JavaScript Programs
 - Contain many JavaScript statements.
 - The statements are executed, one by one, in the same order as they are written. *Try it!*

JavaScript Statements (cont'd)

❏ Semicolons ;

- ❏ Separate JavaScript statements.
- ❏ Add a semicolon at the end of each executable statement. *Try it!*
- ❏ Multiple statements on one line are allowed. *Try it!*

❏ JavaScript Code Blocks *Try it!*

- ❏ JavaScript statements can be grouped together in code blocks, inside curly brackets {...}.
- ❏ To define statements to be executed together.

JavaScript Statements (cont'd)

JavaScript Keywords

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 Comments

Single Line Comments

- Single line comments start with `//`. *Try it*
- Any text between `//` and the end of line. *Try it*

Multi-line Comments

- Multi-line comments start with `/*` and end with `*/`. *Try it*
- Any text between `/*` and `*/` will be ignored by JavaScript.

JavaScript Variables

JavaScript Identifiers

- All JavaScript variables must be identified with unique names.
 - Unique names are called identifiers
- 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 _.
 - Names are case sensitive (y and Y are different variables).
 - Reserved words (like JavaScript keywords) cannot be used as names.

JavaScript Variables (cont'd)

❖ Declaring (Creating) JavaScript Variables *Try it!*

- Declare JavaScript variables with the **var** keyword.

```
var carName;
```

- After the declaration, the variable is empty. (it has no value)
- To assign a value to the variable, use the equal sign.

```
carname = "Volvo";
```

- You can also assign a value to the variable when you declare it.

```
var carname = "Volvo";
```

JavaScript Variables (cont'd)

- ❖ One Statement, Many Variables *Try it!* *Try it!*
- ❖ Value = undefined *Try it!*
 - ⦿ Variable declared without a value will have the value **undefined**.
- ❖ Re-Declaring JavaScript Variables
 - ⦿ If you re-declare a JavaScript variable, it will not lose its value. *Try it!*
- ❖ *Test Yourself with Exercises!*

JavaScript Operators

JavaScript Arithmetic Operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement

Try it!

Try it!

Try it!

Try it!

Try it!

Try it!

Try it!

JavaScript Operators (cont'd)

JavaScript Assignment Operators

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

Try it!

Try it!

Try it!

Try it!

Try it!

Try it!

JavaScript Operators (cont'd)

JavaScript String Operators

- + operator can also be used to concatenate (add) strings.

Try it! *Try it!* *Try it!* *Try it!*

Adding Strings and Numbers

- If you add a number and a string, the result will be a string! *Try it!*

JavaScript Data Types

JavaScript Data Types

- JavaScript variables can hold many data types : numbers, strings, arrays, objects and more.

```
var length = 16;           // Number
var lastName = "Johnson"; // String
var cars = ["Saab", "Volvo", "BMW"]; // Array
var x = {firstName:"John", lastName:"Doe"}; // Object
```

JavaScript Has Dynamic Types

- The same variable can be used as different types.

```
var x;           // Now x is undefined
var x = 5;       // Now x is a Number
var x = "John";  // Now x is a String
```

JavaScript Data Types (cont'd)

JavaScript Strings *Try it!*

- Can use single or double quotes.

JavaScript Numbers *Try it!*

- Can be written with, or without decimals.
- Extra large or extra small numbers can be written with scientific (exponential) notation.

JavaScript Booleans

- Only have two values : true or false

```
var x = true;  
var y = false;
```

JavaScript Data Types (cont'd)

JavaScript Arrays *Try it!*

- Are written with square brackets.
- Array items are separated by commas.
- Array indexes are zero-based, which means that first item is [0], second is [1], and so on.

JavaScript Objects *Try it!*

- Are written with curly braces.
- Are written as name:value pairs, separated by commas.

The typeof Operator *Try it!*

- To find the type of a JavaScript variable.

JavaScript Data Types (cont'd)

❏ Undefined *Try it!*

- A variable without a value

❏ Empty Values *Try it!*

<< *Here is the end of "Introduction to JavaScript" in Codecademy.com. >>*

JavaScript Functions

JavaScript Function is Try it!

- A block of code designed to perform a particular task.
- Executed when “something” invokes it(calls it).

JavaScript Function Syntax

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

Function Invocation

- The code inside the function will execute when “something” invokes the function.
 - When an event occurs (when a user clicks a button)
 - When it is invoked (called) from JavaScript code
 - Automatically (self invoked)

JavaScript Functions (cont'd)

❏ Function Return *Try it!*


❏ Why Functions? *Try it!*

- You can reuse code: Define the code once, and use it many times.

❏ *Test Yourself with Exercises!*

JavaScript Objects

Real Life Objects, Properties and methods

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

JavaScript Objects (cont'd)

JavaScript Objects

- Objects are variables. Objects can contain many values. *Try it!*
 - The values are written as name:value pairs*

Object Properties

- name:values pairs are called **properties**.

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

Property	Property Value
firstName	John
lastName	Doe
age	50
eyeColor	blue

JavaScript Objects (cont'd)

Accessing Object Properties

`objectName.propertyName`

Try it!

or

`objectName[propertyName]`

Try it!

Accessing Object Methods *Try it!* *Try it!*

`objectName.methodName()`

Test Yourself with Exercises!

JavaScript Scope

JavaScript Scope

- In JavaScript, scope is the set of variables, objects, and functions you have access to.

Local JavaScript Variables *Try it!*

- Variables declared within a JavaScript function, become **LOCAL** to the function.
- Local variables have local scope : They can only be accessed within the function.

Global JavaScript Variables *Try it!*

- Variable declared outside a function, becomes **GLOBAL**.
- Global variable has global scope : All scripts and functions on a web page can access it.

JavaScript Scope (cont'd)

❏ Automatically Global Try it!

- If you assign a value to a variable that has not been declared, it will automatically become a **GLOBAL** variable.

<< *Here is the end of "Functions" in Codecademy.com.* >>

JavaScript Events

HTML Events

- An HTML event can be something the browser does, or something a user does.
 - An HTML web page has finished loading.
 - An HTML input field was changed.
 - An HTML button was clicked.
- HTML allows event handler attributes, with JavaScript code, to be added to HTML elements. *Try it!* *Try it!* *Try it!*
 - With single quotes

```
<some-HTML-element some-event='some JavaScript'>
```
 - With double quotes

```
<some-HTML-element some-event="some JavaScript">
```

JavaScript Events (cont'd)

Common HTML Events

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

 *Test Yourself with Exercises!*

JavaScript Strings

JavaScript Strings

- Stores a series of characters like "John Doe".
- Can be any text inside (single or double) quotes. Try it! Try it!

String Length

- Built in property **length** Try it!

Special Characters Try it!

- Special characters : \', \", \\, \n, \r, \t, \b, \f*

JavaScript Strings (cont'd)

❏ Strings Can be Objects. *Try it!*

❏ String Properties and Methods

○ String Properties

Property	Description
constructor	Returns the function that created the String object's prototype
length	Returns the length of a string
prototype	Allows you to add properties and methods to an object

○ String Methods (next slide)

❏ *Test Yourself with Exercises!*

JavaScript Strings Methods

❖ Finding a String in a String

- `indexOf ()` *Try it!*
- `lastIndexOf ()` *Try it!*
 - JavaScript counts position from zero. 0 is the first position in a string, 1 is the second, 2 is the third...
 - Both the `indexOf ()` and the `lastIndexOf ()` methods return -1 if the text is not found.

❖ Extracting String Parts

- `slice (start, end)` *Try it!* *Try it!*
- `substring(start, end)` *Try it!*
- `substr (start, length)` *Try it!*

JavaScript Strings Methods (cont'd)

- ❏ Replacing String Content *Try it!*
- ❏ Converting to Upper and Lower Case
 - ❏ toUpperCase() *Try it!*
 - ❏ toLowerCase() *Try it!*
- ❏ The concat() Method *Try it!*
- ❏ Converting a String to an Array
 - ❏ split() *Try it!*
- ❏ *Complete JavaScript String Reference*
- ❏ *Test Yourself with Exercises!*

JavaScript Numbers

JavaScript Numbers are Always 64-bit Floating Point.

Value (aka Fraction/Mantissa)	Exponent	Sign
52 bits (0 - 51)	11 bits (52 - 62)	1 bit (63)

Precision

- Integers are considered accurate up to 15 digits. *Try it!*
- The maximum number of decimals is 17. Floating point arithmetic is not always 100% accurate. *Try it!*

Infinity

- Infinity (or –Infinity) is the value. JavaScript will return if you calculate a number outside the largest possible number. *Try it!* *Try it!* *Try it!*

JavaScript Numbers (cont'd)

❖ NaN – Not a number *Try it! Try it! Try it!*

- JavaScript reserved word indicating that the result of a numeric operation was not a number.

❖ Numbers Can be Objects *Try it!*

❖ Number Properties and Methods

- Number Properties

Property	Description
MAX_VALUE	Returns the largest number possible in JavaScript
MIN_VALUE	Returns the smallest number possible in JavaScript
NEGATIVE_INFINITY	Represents negative infinity (returned on overflow)
NaN	Represents a "Not-a-Number" value
POSITIVE_INFINITY	Represents infinity (returned on overflow)

- Number Methods (next slide)

❖ *Test Yourself with Exercises!*

JavaScript Numbers Methods

📘 Global Methods

- Can be used on all JavaScript data types

Method	Description
Number()	Returns a number, converted from its argument.
parseFloat()	Parses its argument and returns a floating point number
parseInt()	Parses its argument and returns an integer

Try it!

Try it!

Try it!

JavaScript Numbers Methods (cont'd)

Number Methods

-  Can be used on numbers.

Method	Description
toString()	Returns a number as a string
toExponential()	Returns a string, with a number rounded and written using exponential notation.
toFixed()	Returns a string, with a number rounded and written with a specified number of decimals.
toPrecision()	Returns a string, with a number written with a specified length
valueOf()	Returns a number as a number

Try it!

Try it!

Try it!

Try it!

Try it!

Complete JavaScript Number Reference

JavaScript Math Object

❏ The Math Object

- ❏ Allows you to perform mathematical tasks.
- ❏ Includes several mathematical methods.

❏ **Math.min() and Math.max()** *Try it!* *Try it!*

❏ **Math.random()** *Try it!*

❏ **Math.round()** *Try it!*

❏ **Math.ceil()** *Try it!*

JavaScript Math Object (cont'd)

- ❏ `Math.floor()` *Try it!*
- ❏ `Math.Constants()` *Try it!*
- ❏ *Complete Math Object Reference*
- ❏ *Test Yourself with Exercises!*

JavaScript Dates

JavaScript Date Formats

- Can be written as a string :

Sun Mar 29 2015 08:03:13 GMT +0900 (대한민국 표준시)

- Or as a number : **1427583793458**

- Dates written as numbers, specifies the number of milliseconds since January 1, 1970, 00:00:00.

Displaying Dates *Try it!*

JavaScript Dates (cont'd)

❏ Creating Date Objects

- 4 ways of initiating a date

```
new Date() Try it!  
new Date(milliseconds) Try it!  
new Date(dateString) Try it!  
new Date(year, month, day, hours, minutes, seconds, milliseconds) Try it!
```

❏ Displaying Dates

- toString() Try it!
- toUTCString() Try it!
- toDateString() Try it!

❏ Test Yourself with Exercises!

JavaScript Date Methods

Date Get Methods

Method	Description
<code>getDate()</code>	Get the day as a number (1-31)
<code>getDay()</code>	Get the weekday as a number (0-6)
<code>getFullYear()</code>	Get the four digit year (yyyy)
<code>getHours()</code>	Get the hour (0-23)
<code>getMilliseconds()</code>	Get the milliseconds (0-999)
<code>getMinutes()</code>	Get the minutes (0-59)
<code>getMonth()</code>	Get the month (0-11)
<code>getSeconds()</code>	Get the seconds (0-59)
<code>getTime()</code>	Get the time (milliseconds since January 1, 1970)

[Try it!](#) [Try it!](#)

[Try it!](#)

[Try it!](#)

JavaScript Date Methods (cont'd)

Date Set Methods

Method	Description
setDate()	Set the day as a number (1-31)
setFullYear()	Set the year (optionally month and day)
setHours()	Set the hour (0-23)
setMilliseconds()	Set the milliseconds (0-999)
setMinutes()	Set the minutes (0-59)
setMonth()	Set the month (0-11)
setSeconds()	Set the seconds (0-59)
setTime()	Set the time (milliseconds since January 1, 1970)

Try it!

Try it!

 *Complete JavaScript Date Reference*

JavaScript Arrays

❏ Creating an Array

- Syntax

```
var array-name = [item1, item2, ...];
```

- Example

```
var car= ["Saab", "Volvo", "BMW"];
```

❏ Access the Elements of an Array

- Refer to an array element by referring to **the index number**.

```
var name = cars[0];
```

- Modifies the first element in cars.

```
cars[0] = "Opel";
```

JavaScript Arrays (cont'd)

❖ You Can Have Different Objects in One Array.

```
myArray[0] = Date.now;  
mymyArray[1] = myFunction;  
myArray[2] = myCars;
```

❖ Arrays are Objects.

- Arrays use numbers to access its “elements”. *Try it!*
- Objects use names to access its “members”. *Try it!*

❖ Array Properties and Methods

```
var x = cars.length;           // The length property returns the number of elements in cars  
var y = cars.sort();           // The sort() method sort cars in alphabetical order
```

JavaScript Arrays (cont'd)

- ❏ The length Property *Try it!*
- ❏ Adding Array Elements *Try it!* *Try it!*
- ❏ Looping Array Elements *Try it!*
- ❏ *Test Yourself with Exercise!*

JavaScript Array Methods

❏ Converting Arrays to Strings

- `valueOf()` *Try it!*
- `toString()` *Try it!*
- `join()` *Try it!*

❏ Popping and Pushing

- `pop()` *Try it!*
- `Push()` *Try it!*

❏ Shifting Elements

- `shift()` *Try it!*
- `unshift()` *Try it!*

JavaScript Array Methods (cont'd)

❏ Changing Elements *Try it!* *Try it!*

❏ Deleting Elements *Try it!*

❏ Splicing an Array

○ splice () *Try it!* *Try it!*

❏ Sorting an Array

○ sort () *Try it!*

❏ Reversing an Array

○ reverse () *Try it!*

JavaScript Array Methods (cont'd)

❖ The Compare Function

- To define an alternative sort order

```
function(a, b) {return a-b}
```

❖ Find the Highest (or Lowest) Try it! Try it!

❖ Joining Arrays

- `concat()` Try it!

❖ Slicing an Array

- `slice()` Try it!

❖ Complete JavaScript Array Reference

❖ Test Yourself with exercise!

JavaScript Booleans

- Two values : "true" or "false"
- The Boolean() Function *Try it!*
- Comparisons and Conditions

Operator	Description	Example
==	equal to	if (day == "Monday")
>	greater than	if (salary > 9000)
<	less than	if (age < 18)

JavaScript Booleans (cont'd)

- ❏ Everything With a Real Value is True. *Try it!*
- ❏ Everything Without a Real Value is False.
 - ⦿ *Try it!* *Try it!* *Try it!*
- ❏ *Complete Boolean Reference*

JavaScript Comparison and Logical Operators

Comparison Operators

- Given that x=5

Operator	Description	Comparing	Returns
==	equal to	x==8	false
		x==5	true
===	exactly equal to (equal value and equal type)	x==="5"	false
		x===5	true
!=	not equal	x!=8	true
!==	not equal (different value or different type)	x!== "5"	true
		x!==5	false
>	greater than	x>8	false
<	less than	x<8	true
>=	greater than or equal to	x>=8	false
<=	less than or equal to	x<=8	true

JavaScript Comparison and Logical Operators (cont'd)

Logical Operators

- Given that $x=6$ and $y=3$

Operator	Description	Example
&&	and	$(x < 10 \ \&\& \ y > 1)$ is true
	or	$(x==5 \ \ y==5)$ is false
!	not	$!(x==y)$ is true

Conditional Operator *Try it!*

- Syntax

variablename = (condition) ? value1 : value2

Test Yourself with Exercise!

JavaScript if...Else Statements

❏ The if statement

- ❏ To specify a block of JavaScript code to be executed if a condition is true.

Try it!

❏ The else Statement

- ❏ To specify a block of code to be executed if the condition is false. *Try it!*

❏ The else if statement

- ❏ *To specify a new condition if the first condition is false.* *Try it!*

JavaScript Switch Statement

❏ The JavaScript Switch Statement *Try it!*

❏ The Break Keyword

- When the JavaScript code interpreter 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 *Try it!*

JavaScript For Loop

JavaScript Loops *Try it!*

Different Kinds of Loops

- for *Try it!*
 - Loops through a block of code a number of times
- for/in *Try it!*
 - Loops through the properties of an object
- while *Try it!*
 - Loops through a block of code while a specified condition is true
- do/while *Try it!*
 - Also loops through a block of code while a specified condition is true

JavaScript Break and Continue

- ❏ The Break Statement *Try it!*
- ❏ The Continue Statement *Try it!*
- ❏ JavaScript Labels *Try it!*
- ❏ *Test Yourself with Exercise!*

JavaScript Errors

JavaScript try and catch

- try statement

- Allows you to define a block of code to be tested for errors while it is being executed.

- catch statement

- Allows you to define a block of code to be executed, if an error occurs in the try block.

```
try {  
    Block of code to try  
}  
catch(err) {  
    Block of code to handle errors  
}
```

JavaScript Errors (cont'd)

❏ The throw Statement

- ⦿ Allows you to create a custom error.

```
throw "Too big";    // throw a text
throw 500;          // throw a number
```

❏ Input Validation Example Try it!

❏ The finally Statement Try it!

```
try {
    Block of code to try
}
catch(err) {
    Block of code to handle errors
}
finally {
    Block of code to be executed regardless of the try / catch result
}
```

<< Here is the end of *"For' Loops in JavaScript"* in Codecademy.com. >>

<< Here is the end of *"While' Loops in JavaScript"* in Codecademy.com. >>

<< Here is the end of *"Control Flow"* in Codecademy.com. >>