HTML and JavaScript

Prof. Hyuk-Yoon Kwon
https://sites.google.com/view/seoultech-bigdata

Most parts are based on slides used in
(http://ce.sharif.edu/~zarrabi/courses/2013/ce419/notes/)

HTML Forms

Forms

- HTML forms are used to get user input
- Form elements include:
  - Text Fields
  - Buttons
  - Menus
  - Checkboxes
  - Radio Buttons

Form Definition

- Forms are defined using <form> tag

```
<form>
  First name:
  <input type="text" name="fname"><br>
  Last name:
  <input type="text" name="lname"><br>
</form>
```

First name:

Last name:

Form Action

- Submit button is used to send data to server
- The form tag attributes:
  - action: a URL to which the information is sent
  - method: HTTP method for sending data (get or post)

```
<form action="get-form.py" method="get">
  First name:
  <input type="text" name="fname"><br>
  Last name:
  <input type="text" name="lname"><br>
  <input type="submit" value="Submit">
</form>
```

Form Inputs

- The <input> tag is multipurpose
- Input type is specified using type attribute
  - text, password, checkbox, radio, button, submit, …
- They should all have name attribute
- Their initial state can be set by value attribute
- They can be disabled by disabled attribute

Checkboxes

- <input type="checkbox" ...>
- The name attribute names the checkbox
- The value attribute specifies the value bound to name if checkbox is submitted
- The checked attribute indicates a pre-checked checkbox

```
<form action="" method="get">
  <input type="checkbox" name="device"
    value="iPhone">iPhone<br>
  <input type="checkbox" name="device"
    value="iPad">iPad<br>
  <input type="submit" value="Submit">
</form>
```

Radio Buttons

- <input type="radio" ...>
- Used to select *one of many* options

```
<form action="" method="get">
  <input type="radio" name="device"
    value="iPhone">iPhone<br>
  <input type="radio" name="device"
    value="iPad">iPad<br>
  <input type="submit" value="Submit">
</form>
```

Text Boxes

- <input type="text" ...>
- The size attribute specifies the width in characters
- The maxlength attribute specifies the maximum number of characters

```
Your Full Name:
<input type="text" name="fullname" size="30" maxlength="50">
```

Passwords

- <input type="password" ...>
- Identical to a text box, but text typed into the box is not readable on browser
- Useful for submitting sensitive information, like passwords, but not secure at all

```
Password: <input type="password" name="pass">
```

Hidden Objects

- <input type="hidden" ...>
- Represents a hidden input, invisible to the user
- Useful for sending hidden data to server, or keeping track of data as user traverses a collection of pages

```
<input type="hidden" name="id" value="a84re">
```

Buttons

- <input type="submit" ...>
  - A button that submits the form to the server
- <input type="reset" ...>
  - A button that resets all form fields to their default state
- <input type="button" ...>
  - A button that does nothing!
  - Print alert messages when click

```
<input type="button" onclick="alert('print something')" value="click">
```

Button Tag

- The <button> tag can be alternatively used to create buttons
- The type attribute specifies the type of button
  - can be button, submit, reset
- Inside button element you can put text or image
  - this is the main difference with input buttons

```
<button type="button">Click Here!</button>
```

Text Areas

- The <textarea> is used for multiline text input
- The rows and cols attributes specify the number of rows and columns

```
<textarea rows="30" cols="50" name="text">
This is the text that you will see
and can edit in the area.
</textarea>
```

Menus

- The <select> tag is used to create menus
- Each option is enclosed in an option tag
- The size attribute determines how many options to be displayed at once

```
<select name="device">
  <option value="iPhone">iPhone</option>
  <option value="iPad">iPad</option>
  <option value="iMac">iMac</option>
</select>
```

Labels

- The <label> tag defines a label for an input element
- The for attribute of the label must be equal to the id attribute of the input element

```
<input type="checkbox" name="iPhone" id="iPhone">
<label for="iPhone">I like iPhone</label><br>
<input type="checkbox" name="iPad" id="iPad">
<label for="iPad">I like iPad</label>
```

Practice

1. Make a simple page and put two text inputs and a Submit button into it. Ask for the user's name and address...

Name:
Address:
제출

2. Create two radio buttons and place them inside the form element, under the text "Favorite color".
   - Add the name attribute with a value of "color" to both radio buttons.
   - Also add a value attribute and specify "blue" for the first and "red" for the second.
   - Lastly, specify some text that corresponds with the values.

3. Create a button (using input types) that says "Click Me", which alerts "Hello World" when you click on it.

4. Add a drop down list with name="cars" to the form. Include the following options: "volvo", "ford", "fiat", and "audi".

HTML5 New Elements

HTML5 Main Features

- 2D graphics with <canvas> and <svg>
- New media elements
- Drag and drop support
- New form controls, like calendar and data list

Canvas

- **The HTML <canvas> element is used to draw graphics, on the fly, via JavaScript.**

```
<canvas id="myCanvas"></canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(0,0);
ctx.lineTo(200,100);
ctx.stroke();
</script>
```

SVG

- SVG has several methods for drawing paths, boxes, circles, text, and graphic images

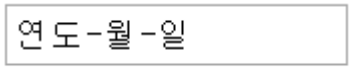
```
<!DOCTYPE html>
<html>
<body>

<svg height="100" width="100">
  <circle cx="50" cy="50" r="40" stroke="black"
stroke-width="3" fill="red" />
  Sorry, your browser does not support inline
SVG.
</svg>

</body>
</html>
```

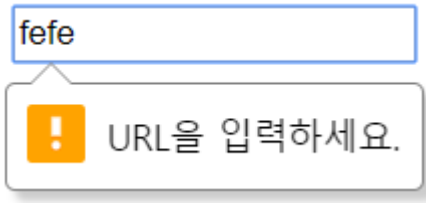
Drag and Drop

- **Example**
  - https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_draganddrop2

New Input Elements

- date
- number
- range
- color
- email
- url
- …





New Media Elements

<audio controls>	Defines sound content
<video controls>	Defines a video or movie
<source>	Defines multiple media resources for <video> and <audio>
<embed src="">>	Defines a container for an external application or intera ctive content (a plug-in)

Practice

1. **To play an audio file, use the <audio controls> element**
   - Use the following source: "https://www.w3resource.com/html-css-exercise/basic/solution/beach.mp3"

2. **To play a video file, use the <video controls> element**
   - Use the following source: "http://www.w3resource.com/html-css-exercise/basic/solution/php-demo1.mp4"

3. **Embed a video file, use the <embed src> element**
   - Use the following source: "https://www.youtube.com/v/tgbNymZ7vqY"

JavaScript – Part1

Outline

- What is JavaScript?
- Using JavaScript
  - Adding JavaScript to HTML
  - Sample Usages
- JavaScript Basics
- JavaScript Functions

Introduction

- JavaScript is a programming language for use in HTML pages
- JavaScript is a full-featured programming language (has nothing to do with Java!)
- JavaScript programs are run by an interpreter built into the client's web browser (not on the server)

JavaScript Uses

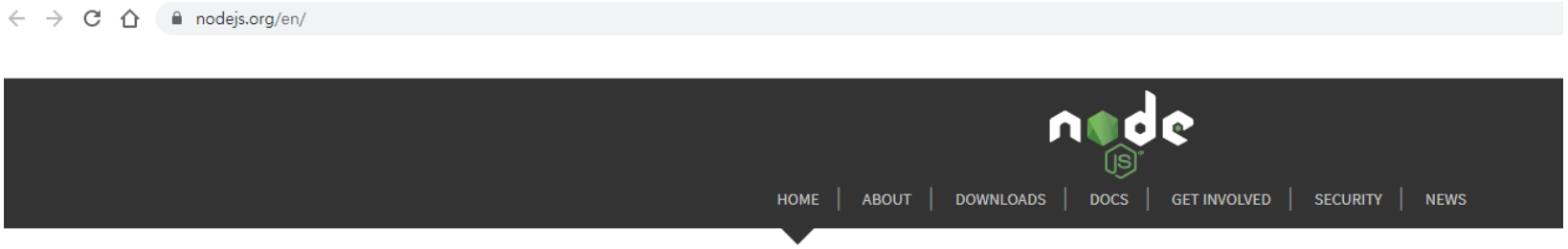
- dynamically modifying an HTML page
- responding to user action
- validating user input
- communicate with server
- storing/restoring data
- animation and drawing
- playing audio and video, …

Outside Web Pages

- Gadgets
  - Desktop apps, Browser extensions
- Tools
  - Adobe Acrobat, Open-Office, Flash
- Game development
  - DX-Studio, Re-Animator
- Server-side scripting
  - Node.js

Practice: Node.JS

Install Node.JS in your Web server



Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

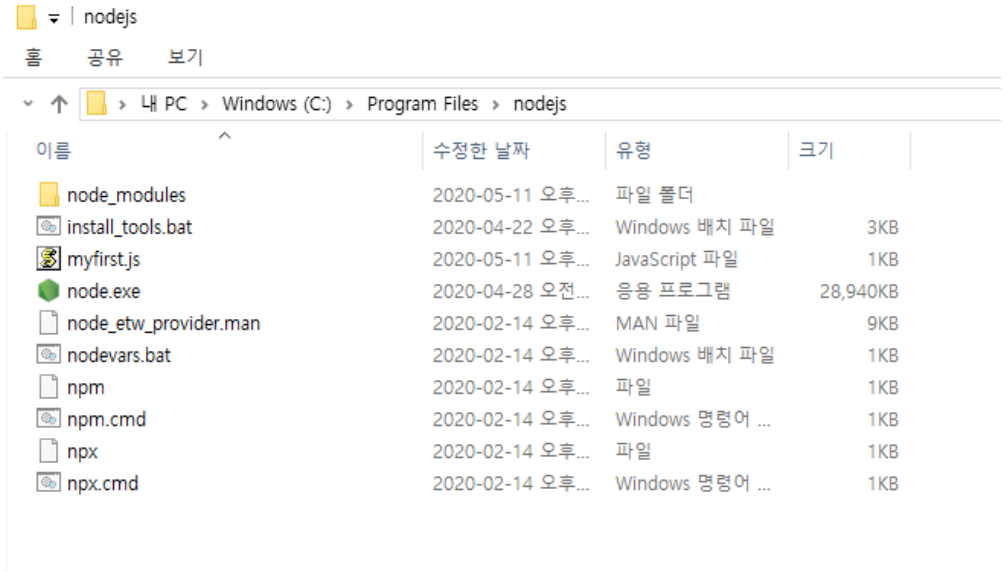
Download for Windows (x64)

12.16.3 LTS
Recommended For Most Users

14.2.0 Current
Latest Features

Other Downloads | Changelog | API Docs Other Downloads | Changelog | API Docs

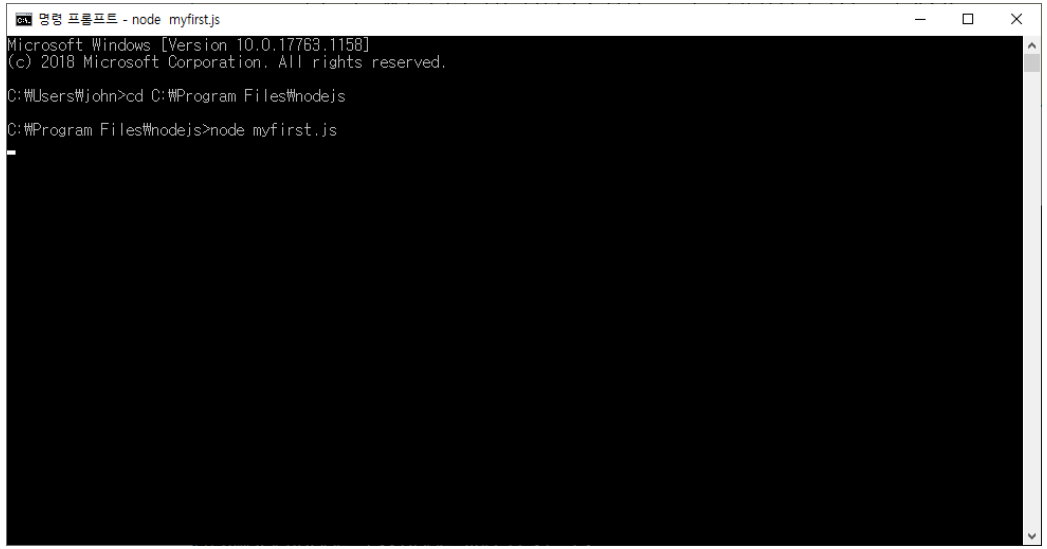
Or have a look at the Long Term Support (LTS) schedule.



Make a following JavaScript, named myfirst.js

```
var http = require('http');

http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.end('Hello World!');
}).listen(8080);
```

Execute myfirst.js



Access http://localhost:8080

Using JavaScript

Inline Script

- JavaScript can be inserted into HTML pages by using the <script> tag

```
<!DOCTYPE html>
<html>
<body>
.
.
<script>
   document.write("Hello World!");
</script>
.
.
</body>
</html>
```

Where to Put Inline Scripts

- You can have any number of scripts
- Scripts can be placed in the <body> or in the <head>
  - In the <head>, scripts are run before the page is displayed
  - In the <body>, scripts are run as the page is displayed
- In the <head> is the right place to define functions and variables that are used by scripts within the <body>

External Scripts

- Scripts can also be loaded from an external file
- Useful for complicated scripts or set of functions that are used in different pages

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

Sample Usages

- Responding to Events

```
<button type="button" onclick="alert('Hi!')">
Click Here!</button>
```

JavaScript Basics

Statements

- Each statement is optionally ended with a semicolon
  - It is good practice to add semicolons
- Comments are delimited with // and /* */ as in Java and C++

Variables

- JavaScript has variables that you can declare with the optional var keyword
- Variables declared within a function are local to that function
- Variables declared outside of any function are global variables
  - Note: variables declared without var are made global

```
var str = "Hello";
```

Basic Data Types

- string
- number
- Boolean
- null
- undefined

Operators

- JavaScript has Java/C-like operators
  - Arithmetic (+, - , *, /, %)
  - Assignment (=, +=, -=, *= /=, %=, ++, --)
  - Comparison (<, >, <=, >=, ==)
  - Logical (&&, ||, !)
- Notes:
  - + also does string concatenation
  - === checks both value and type

Practice – Developer Tools

- **Open Developer Tools in Browser (short key: F12 in Chrome or Internet Explorer)**

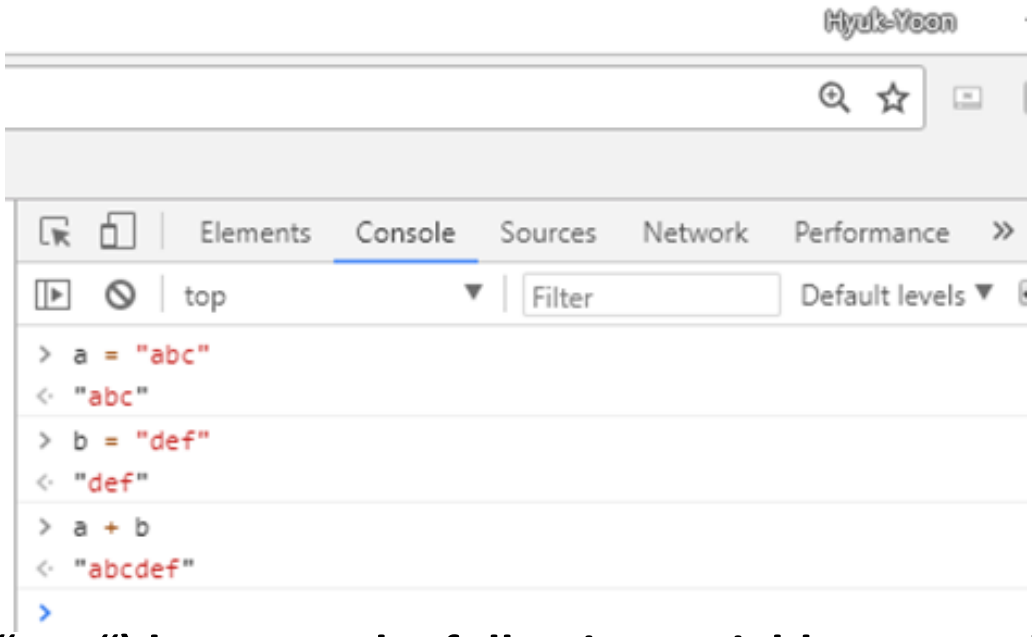
- **In console tab, make some expressions using the following operators and check the result**
  - Arithmetic (+, - , *, /, %)

    e.g, a = 1; b = 2; a + b
  - Assignment (=, +=, -=, *= /=, %=, ++, --)

    e.g., a += 1
  - Comparison (<, >, <=, >=, ==)

    e.g., 1 < 2
  - Logical (&&, ||, !)

    e.g., (1>2) && (1<2)



- **Compare the result of equality operators ("==" or "===") between the following variables num, obj, and str**
  - num = 0
  - obj = new String("0")
  - str = "0"

Type Conversion

- In expressions involving a string, the + operator and a number, numbers are converted to strings

```
res = 'The total is ' + 12;
res = 12 + 'is the total';
```

- With other operators, strings are converted to numbers

```
a = 20 - '32';    // -12
b = 1 * '5.4';    // 5.4
c = 3 * 'a2';     // NaN
```

Control Structures

- JavaScript has C-like structures
- Conditionals
  - if, else
  - switch, case
- Loops
  - for, while
  - (can break and continue)

Arrays

- Arrays are indexed from 0
- Special version of for works with arrays

```
var colors = ['red', 'geen', 'yellow'];
for (var i in colors) {
   document.write('<div style="background-color:'
                  + colors[i] + ';">'
                  + colors[i] + '</div>\n');
}
```

Practice - Loop

1. Write a JavaScript for loop that will iterate from 0 to 15. For each iteration, it will check if the current number is odd or even, and display a message to the screen

2. For given numbers, write a JavaScript program insert dashes (-) between each two even numbers. For example if you accept 025468 the output should be 0-254-6-8
   - Hint: convert numbers to string and check characters of string by accessing array

JavaScript Functions

Functions

- You can define functions using the function keyword
- Functions can return a value using the return keyword (or return undefined by default)

```
function test(n) {
   var a = 2 / n;
   return a;
}
```

Function Arguments

- JavaScript is very flexible with function arguments
- A function can be called with more or less arguments than the number of declared parameters
- Too few arguments: leaves parameters undefined

```
function show(x, y) {
  document.write(x + '\n');
  document.write(y);
}
show('hello'); // prints hello undefined
```

Function Arguments

- All the arguments to a function can be accessed through the arguments pseudo-array

```
function show() {
  for (var i = 0; i < arguments.length; i++) {
    document.write(arguments[i]);
  }
}

show('a', 'b', 'c', 'd', 42);
```

eval

- The eval() function evaluates or executes an argument.

```
var str = '2 * 4 + 5';
var x = eval(str);

document.write(x);
```

Simple User Interaction

- JavaScript has some built-in functions providing simple user interaction
  - alert(msg): alerts the user that something has happened
  - confirm(msg): asks the user to confirm (or cancel) something
  - prompt(msg, default): asks the user to enter some text

```
alert('The email is not correct!');

confirm('Are you sure you want to do that?');

prompt('Enter you name');
```

Practice – User Interaction

- **Receive an input from the user until given input is ‘john’. Then, print a message including input**
  - Hint: Use prompt() to receive an input

Objects

- JavaScript objects are collections of name/value pairs
- Objects in JavaScript are similar to
  - Dictionaries in Python
  - Hashes in Perl and Ruby
  - Associative arrays in PHP
  - HashMaps in Java

Object Structure

- The name part is a simple string, while the value can be any JavaScript value, including other objects

```
var person = {
  name: 'Hamid',
  family: 'Fereydoon',
  id: 12
}
```

Object Properties

- The values in an object are usually called properties
- Property names can also be numbers

```
var person = {
   1: 'Hamid',
   2: 'Fereydoon',
   others: 0
 }
```

Creating Objects

- Two ways to create objects

```
var obj = {};

var obj = new Object();
```

- These two are equivalent

Accessing Properties

- Object properties can be accessed through the . or [] operators

```
var person = {
  name: 'Hamid',
  11: '9121122090'
}

person.name
person['name']
person[11]
person[name]     // Error: name is not defined
```

Nested Objects

```
var obj = {
    name: 'T-Shirt',
    'for': 'ACM',
    details: {
        color: 'Black',
        size: 10
    }
};

obj.details.color;          // Black
obj['details']['size'];      // 10
```

Practice - Object

1. Write a JavaScript program to list the properties of a JavaScript object.
   - *Sample object*:

```
var student = {
   name : "David Rayy",
   sclass : "VI",
   rollno : 12
};
```

   - *Sample Output*: name,sclass,rollno

2. Write a JavaScript function to get a copy of the object where the keys have become the values and the values the keys

IN Operator

- **How to access the Name**

```
Color = {red: "#FF0000", green: "#00FF00", white: "#FFFFFF"};
for (var element in Color) {
        alert(element);                    // key
}

Color = ["red", "green", "white"];
for (var element in Color) {
        alert(element);                    // index
}
```

Array

- **How to access the Value in Array**

```
Color = ["red", "green", "white"];
for (var element of Color) {
        alert(element);                    // value
};

Color = ["red", "green", "white"];
Color.forEach (function(element) {
        alert(element);                    // value
});
```

How to access both Name and Value in Array

```
Color = ["red", "green", "white"];
Color.forEach (function(value, index) {
        alert(index + " - " + value);
});
```

C.f.,) In Python

```
Color = [‘red’, ‘green’, ‘white’];
for element in Color:
        print element;        // value

Color = {'red': "#FF0000", 'green': "#00FF00", 'white': "#FFFFFF"};
for element in Color:
        print element;        // name
```

Object Operators

Update

- We can add/update properties on the fly, using access operators

```
var obj = {
  name : 'Ali'
};

obj.name = 'Hamid';
obj['family'] = 'Fereydoon';
```

Delete

- We can remove properties using delete operator
- Can be used to remove objects

```
var obj = {
  name: 'Ali'
};

delete obj.name
delete obj
```

Property Existence

- The in operator determines whether an object has a certain property

```
if ('property ' in obj) { ... }

if (typeof obj.property !== 'undefined') { ... }

if (obj.hasOwnProperty('property')) { ... }
```

Typeof

- The typeof operator returns a string representing the type of the argument
- Can be one of
  - "function"
  - "string"
  - "number"
  - "boolean"
  - "object"
  - "undefined"