Front-end Essentials

JavaScript

Table of Contents

Lesson Objectives

- Understand JavaScript and its syntax
- Practice well with JavaScript

Section 1

Overview

Overview – What is JavaScript?

- Programming language that can be included on web pages to make them more interactive.
  - ✓ You can use it to **check** or **modify** the **contents of forms**, **change images**, **open new windows** and **write dynamic page content**.
- Inside a host environment (for example, a web browser), JavaScript can be connected to the objects of its environment to provide programmatic control over them.

Overview – What is JavaScript?

- **Core** JavaScript can be extended for a variety of purposes by supplementing it with additional objects:
  - ✓ Client-side JavaScript extends the core language by supplying objects to control a browser and its Document Object Model (DOM).
  - ✓ Server-side JavaScript extends the core language by supplying objects relevant to running JavaScript on a server.

Overview – Why JavaScript?

- **To add** dynamic functionality to your web page.
- JavaScript does things that HTML can't—like logic.
  - ✓ You can change HTML on the fly.
- To shoulder some of the form-processing burden.
  - ✓ JavaScript runs in the browser, not on the Web server.

Overview – Why JavaScript?

- Make web app **more smooth**.
- **To Validate** the data that users enter into the form, before it is sent to your Web application.
- **To add** animation

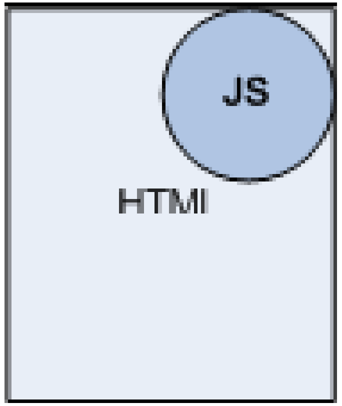
Overview – When not to use JavaScript?

- We cannot treat JavaScript as a full-fledged programming language.
- It lacks the following important features:
  - ✓ When you need to access other resources:
    - Files
    - Programs
    - Databases
  - ✓ When you are using sensitive or copyrighted data or algorithms.
  - ✓ Your JavaScript code is open to the public.

Overview – How to add JavaScript?

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

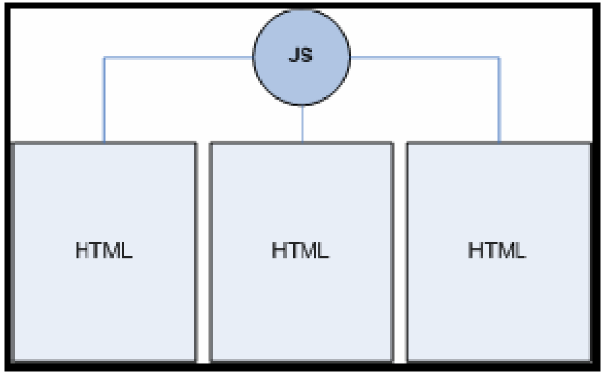
```
<html>
<head>
<script language="JavaScript">
    // JavaScript code
</script>
</head>
```



Overview – How to add JavaScript?

- **As a file, linked from the HTML page:**

```
<head>
<script language="JavaScript" src="script.js">
</script>
</head>
```



Overview – Functions

- A JavaScript function is a block of code designed to perform a particular task.
- A JavaScript function is executed when "something" invokes it (calls it).
- **Syntax:**

```
<script language="javascript">
    function myFunction(parameters) {
        // some logical grouping of code
    }
</script>
```

Overview – Events

- **HTML events** are “**things**” that happen to HTML elements.
- When JavaScript is used in HTML pages, Javascript can “**react**” on these events.
- An HTML event can be something the **browser** does, or something a **user** does.

Overview – Events

- JavaScript defines various **events**:
  - ✓ **onClick** – link or image is clicked
  - ✓ **onSubmit** – a form is submitted
  - ✓ **onMouseOver** – the mouse cursor moves over it
  - ✓ **onChange** – a form control is changed
  - ✓ **onLoad** – something gets loaded in the browser

  etc.
- JavaScript lets you **execute** code when **events** are **detected**

Overview – Events Example

```
<html>
<head>
    <script language=“javascript”>
    function funct() {
         // code
    }
    </script>
</head>
<body>
    <img src=“pic.gif” onClick=“funct();”>
</body>
</html>
```

Overview – Variables

- **JavaScript** has **untyped** variables.
- Variables are declared with the **var** keyword:

```
var num = 1;                 // Now num is a Number
var name = "Mel";   // Now name is a String
var phone ;         // Now phone is undefined
// JavaScript Booleans:
//     can only have two values: true or false.
var x = true;
var y = false;
```

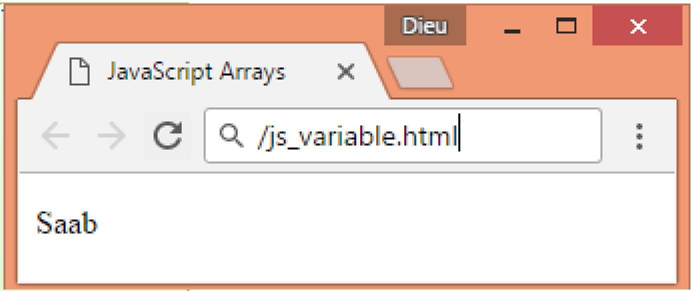
Overview – Arrays

- JavaScript arrays are written with square brackets [].
- Items are separated by **commas**.
- **Example:**

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

<script>
var cars = ["Saab","Volvo","BMW"];

document.getElementById("demo").innerHTML = cars[0];
</script>
</body>
```



Saab

Overview – Objects

- Objects are written with brackets {}
- Objects are collections of key/value pairs
- **Examples:**

```
<script>
var student = {
  name: ‘Fresher’,
  age: 20,
  class: ‘Front-end’
};
</script>
```

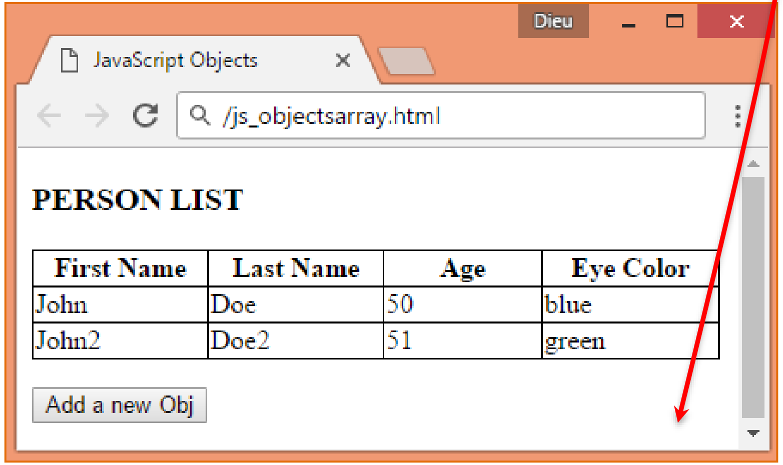
Overview – Objects Example

```
function addRow() {
var Persons = [ {
        firstName : "John",
        lastName : "Doe",
        age : 50,
        eyeColor : "blue"
}, {
        firstName : "John2",
        lastName : "Doe2",
        age : 51,
        eyeColor : "green"
} ];

var table = window.document.getElementById("dTable");
var row;
var cell1, cell2, cell3, cell4;
for (var i = 0; i < Persons.length; i++) {
        row = table.insertRow(i+1);
        cell1 = row.insertCell(0);
        cell2 = row.insertCell(1);
        cell3 = row.insertCell(2);
        cell4 = row.insertCell(3);

        cell1.innerHTML = Persons[i].firstName;
        cell2.innerHTML = Persons[i].lastName;
        cell3.innerHTML = Persons[i].age;
        cell4.innerHTML = Persons[i].eyeColor;
}
}
```

```
 <H3>PERSON LIST</H3>
<table id="dTable">
       <tr>
              <th>First Name</th>
              <th>Last Name</th>
              <th>Age</th>
              <th>Eye Color</th>
       </tr>
</table>
<p/>
<input type="button" value="Add a new Obj“
                          onclick="addRow();">
```



PERSON LIST

First Name	Last Name	Age	Eye Color
John	Doe	50	blue
John2	Doe2	51	green

Overview – Summary

- JavaScript is a dynamic computer programming language
- JavaScript interact with html elements in order to make interactive web user interface.
- JavaScript has **untyped** variables
- Support various data structure such as Arrays and Objects

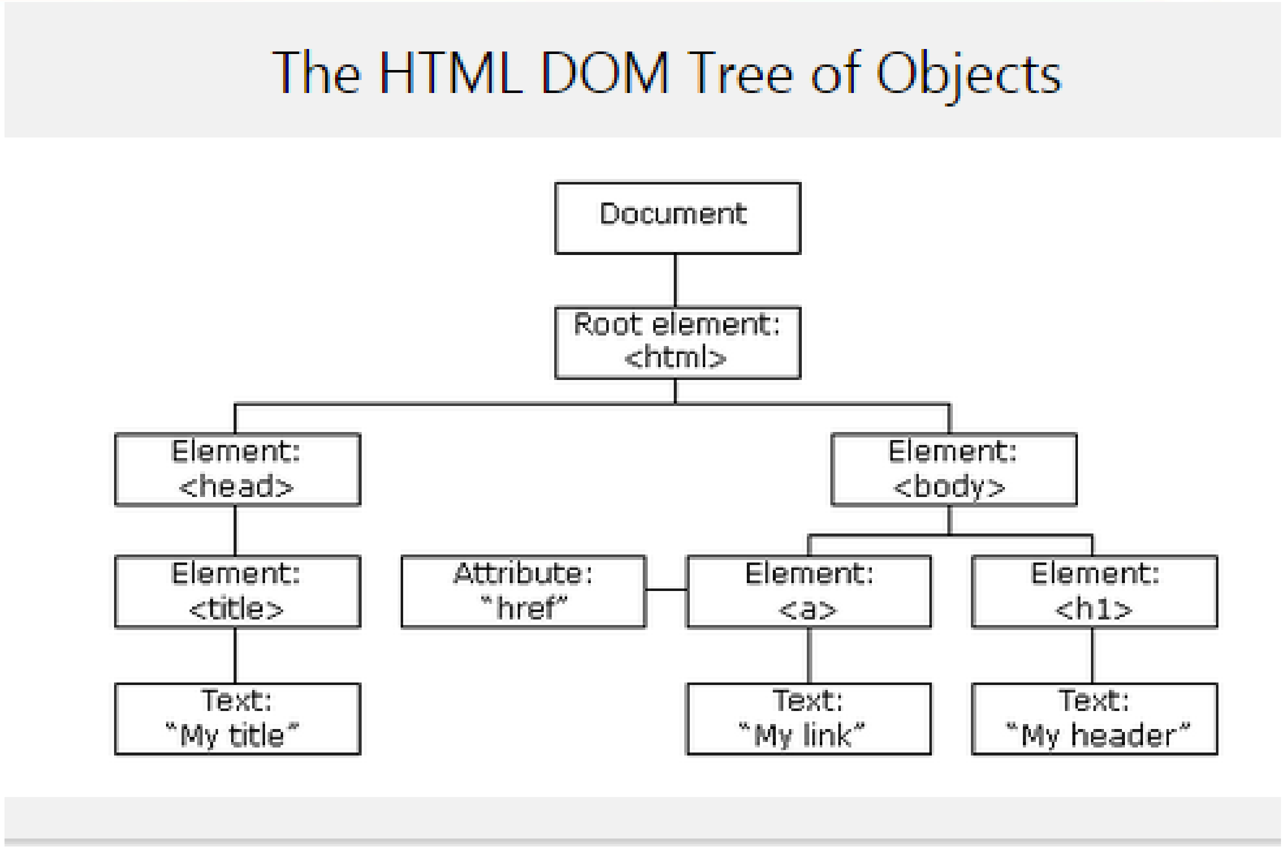
Section 2

Document Object Model

DOM – The HTML DOM

- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
- With the HTML DOM, JavaScript can access and change all the elements of an HTML document.

DOM – The HTML DOM Example



DOM – Part of the DOM

- **window** (browser window)
- **location** (URL)
- **document** (HTML page)
- **anchors** <a>P: The Anchor object represents an HTML <a> element.
- **body** <body>
- **images** <img>
- **forms** <form>
- **elements** <input>, <textarea>, <select>
- **frames** <frame>
- **tables** <table>
- **rows** <tr>
- **cells** <th>, <td>
- **title** <title>

DOM –Referencing the DOM

- Levels of the DOM are **dot-separated**.
- **By keyword and array number (0+)**

  window.document.images[0]

  window.document.forms[1].elements[4]
- **By names (the name attribute in HTML)**

  window.document.mygif (<img src=“file.gif” name=“mygif”>)

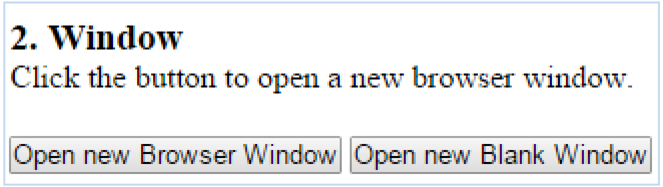
  window.document.catform.fname

      (<form name=“catform” . . .> <input name=“fname” . . .>)

DOM – window

- **Example:**

```
function openWindow1() {
    window.open("https://www.google.com.vn");
}
```

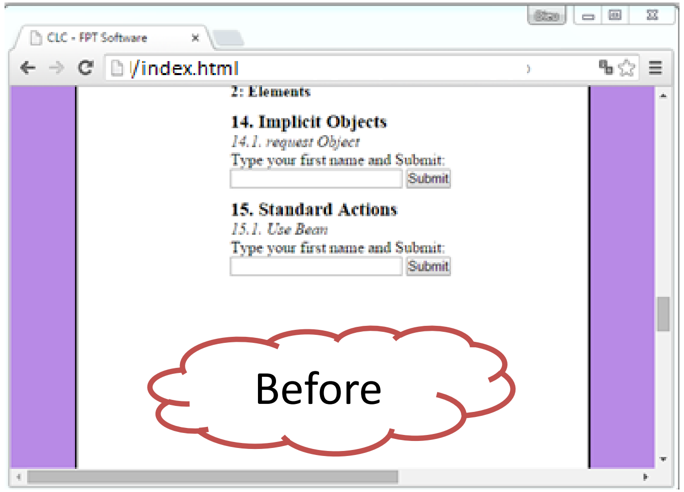

2. Window

Click the button to open a new browser window.

DOM – document and body

- **Example:**

```
function changeBody() {
      document.getElementsByTagName("BODY")[0].style.
                                    backgroundColor = “blue";
}
```



2: Elements

14. Implicit Objects

14.1. request Object

Type your first name and Submit:

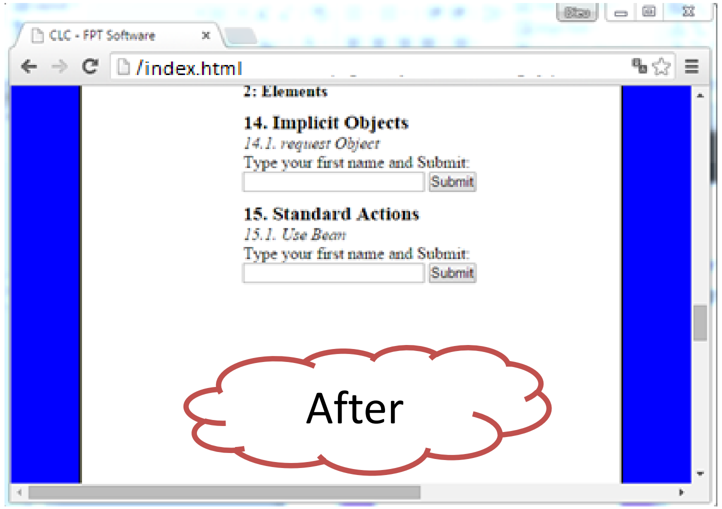
Submit

15. Standard Actions

15.1. Use Bean

Type your first name and Submit:

Submit



2: Elements

14. Implicit Objects

14.1. request Object

Type your first name and Submit:

Submit

15. Standard Actions

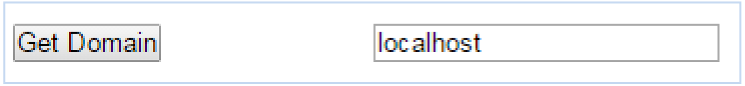
15.1. Use Bean

Type your first name and Submit:

Submit

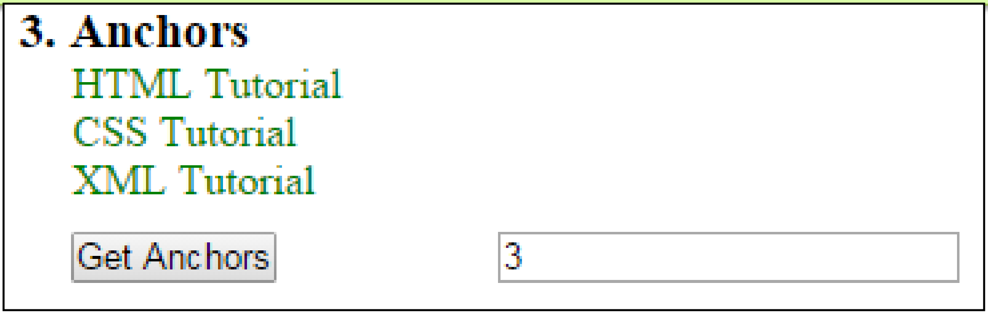
DOM – location

```
function getDomain() {
    document.getElementById("myText").value = document.domain;
    // or
    document.getElementById("myText").value =
         document.lastModified;
    var theText = document.getElementById("myText");
    theText.value = document.lastModified;
}
```



DOM – anchors

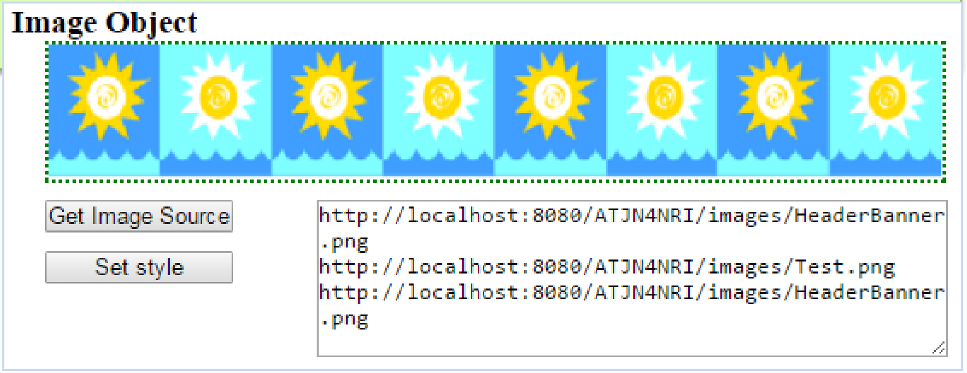
```
<div style="margin-top: 70px">
    <a name="html">HTML Tutorial</a><br>
    <a name="css">CSS Tutorial</a><br>
    <a name="xml">XML Tutorial</a><br>
    <button onclick="getAnchors()">Get Anchors</button>
    <input type="text" id="anchorText" value="Anchors">
</div>
function getAnchors() {
        document.getElementById("anchorText").value =
                                          document.anchors.length;
}
```


3. Anchors

HTML Tutorial
CSS Tutorial
XML Tutorial

DOM – images

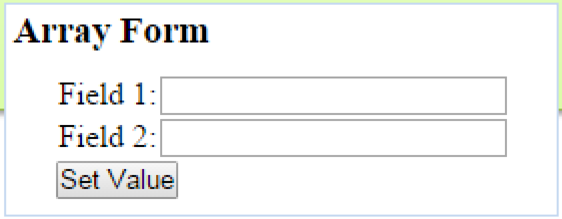
```
function getAllImages() {
    var srcImages = "";
    var arrImages = document.images;
    for (var i = 0; i < arrImages.length; i++) {
        srcImages = srcImages + arrImages[i].src + "\n";
    }
    document.getElementById("imgText").value= srcImages;
}
function setStyleImage() {
    document.images[0].style.border="2px dotted green";
}
```



```
http://localhost:8080/ATJN4NRI/images/HeaderBanner
.png
http://localhost:8080/ATJN4NRI/images/Test.png
http://localhost:8080/ATJN4NRI/images/HeaderBanner
.png
```

DOM – forms

```
function setValue(){
    document.forms[0].elements[0].value = "Field 1";
    document.forms[0].elements[1].value = "Field 2";
}
```


Array Form

Field 1:

Field 2:

DOM - Alerts

- A JavaScript alert is a little window that contains some message:

  alert(“This is an alert!”);

- Are generally used for warnings.
- Can get annoying—use sparingly

DOM – Alerts Sample

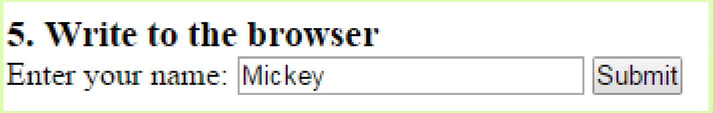
```
<html>
<head>
<script language="javascript">
function showAlert(text) {
    alert(text);
}
</script>
</head>
<body onload="showAlert
            ('This alert displays when the page is loaded!');">
. . .
//OR
<body onload="alert('This alert…');">
```

DOM – Write to Browser

- **JavaScript** can dynamically generate a **new HTML page**. Use **document.writeln**(“*text*”);
  - ✓ **Cannot** add to the current page.
- When you’re done, use **document.close()**;
- This flushes the buffer, and the generated document is then loaded into the browser.
- If the HTML code you’re generating contains quotation marks, you must escape them with a backslash.

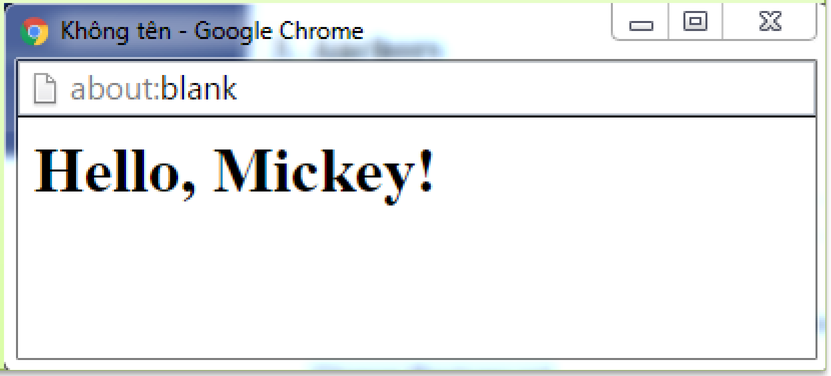
DOM – Write to Browser Example

```
<script language="javascript">
     function dynamicName() {
          var who = window.document.myform.name.value;
          var myWindow = window.open("", "myWindow", "width=600, height=800");
          myWindow.document.writeln("<html><body>");
          myWindow.document.writeln("<h1>Hello, " + who + "!</h1>");
          myWindow.document.writeln("</body></html>");
          myWindow.document.close();
     }
</script>
</head>
<body>
<form name="myform" onSubmit="dynamicName();">
      Enter your name: <input type="text" name="name">
      <input type="submit" value="Submit">
</form>
</body>
```



5. Write to the browser

Enter your name: Mickey



Không tên - Google Chrome

about:blank

Hello, Mickey!

DOM – Page Navigation

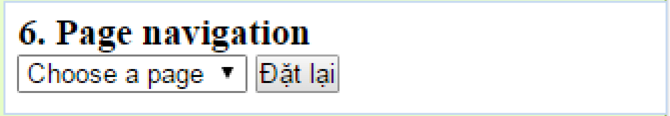
- Use the location API to change the HTML file that is loaded in the window.
- Just set location to another value:

  location = “*page.html*”;

DOM – Page Navigation Sample

```
<script language="javascript">
    function goPage() {
    var pg = document.theForm.aPage.value;
    location = "page" + pg + ".html";
}
</script>

<form name="theForm">
    <select name="aPage" onChange="goPage();">
    <option selected>Choose a page</option>
    <option value="1">Page 1</option>
    <option value="2">Page 2</option>
    <option value="3">Page 3</option>
    <option value="4">Page 4</option></select>
    <input type="reset">
</form>
```


6. Page navigation

DOM – Images

- The image swap is really a sleight-of-hand trick.
- There are two images, each slightly different than the other one.
- Use the src API in JavaScript to replace one image with the other.

```
<script language="javascript">
function swap(file) {
           document.globe.src=file;
}
</script>
. . .
<img name="globe" src="globe.jpg"     onMouseOver="swap('globe2.jpg');"
                                   onMouseOut="swap('globe.jpg');">
```

DOM – Practical Time (1)

- Here is a sample html file with a submit button. Now modify the style of the paragraph text through Javascript code.

```
<!DOCTYPE html>
<html><br><head>
<meta charset=utf-8 />
<title>JS DOM paragraph style</title>
</head>
<body>
     <p id ='text'>JavaScript Exercises - w3resource</p>
     <div>
          <button id="jsstyle"onclick="js_style()">Style
          </button>
     </div>
</body>
</html>
```

DOM – Practical Time (2)

- Write a JavaScript function to get/set the values of First and Last name of the following form.
- Write a JavaScript function to change image, link.

DOM - Summary

- Browser creates a **DOM** of the page based on HTML layout
- JavaScript can access and change all the elements of an HTML document with **DOM**
- **DOM** is composed of many part such as: window, location, document...
- Each part represents a part of the web page

Section 3

JavaScript Regular Expressions

RegExp – What is Regular Expression?

- A regular expression is a sequence of characters that forms a **search pattern**.
- The search pattern can be used for **text search** and **text replace** operations.
- **Syntax:**

  /pattern/modifiers;

RegExp – What is Regular Expression?

- **Using String Methods:**

Method	Description
search()	The search() method uses an expression to search for a match, and returns the position of the match.
replace()	The replace() method returns a modified string where the pattern is replaced.

RegExp – search() and replace()

- **search**() method:

```
var str = "Visit MySchools";
var n = str.search(/myschools/i);
// The result in n will be: 6
```

- **replace**() method:

```
var str = "Visit Microsoft!";
var res = str.replace(/microsoft/i, "MySchools");
// The result in res will be: Visit MySchools!
```

RegExp – Modifiers

- **Regular Expression Modifiers:**

Modifier	Description
i	Perform case-insensitive matching
g	Perform a global match (find all matches rather than stopping after the first match)
m	Perform multiline matching

RegExp – Syntax (1)

- **Brackets** are used to find a range of characters:

Expression	Description
[abc]	Find any of the characters between the brackets
[0-9]	Find any of the digits between the brackets
(x|y)	Find any of the alternatives separated with |

RegExp – Syntax (2)

- **Metacharacters** are characters with a special meaning:

Metacharacter	Description
\d	Find a digit
\s	Find a whitespace character
\b	Find a match at the beginning or at the end of a word
\uxxxx	Find the Unicode character specified by the hexadecimal number xxxx

RegExp – Syntax (3)

- **Quantifiers** define quantities:

Quantifier	Description
n+	Matches any string that contains at least one n
n*	Matches any string that contains zero or more occurrences of n
n?	Matches any string that contains zero or one occurrences of n

RegExp – Using RegExp Object (1)

- Using **test**() method:
  - ✓ The test() method is a RegExp expression method.
  - ✓ It searches a string for a pattern, and returns true or false, depending on the result.
- **Example 2:**

```
var patt = /in/;
patt.test("The best things in life are free!");
// the output of the code above will be: true
```

- **Example 2:**

```
 // allow letters, numbers, and underscores
var illegalChars = /\W/; // Equivalent to [^A-Za-z0-9_].
illegalChars.test("dieunt1");
 // the output of the code above will be: true
```

RegExp – Using RegExp Object (2)

- The exec() method is a RegExp expression method.
  - ✓ It searches a string for a specified pattern, and returns the found text.
  - ✓ If no match is found, it returns *null.*
- **Example:**

```
var patt = /in/;
patt.exec("The best things in life are free!");
// the output of the code above will be: in
```

RegExp – Summary

- Regular Expression is a powerful tool for **text search** and **text replace**
- Using **RegExp**, you can search a string in another string or if a string matches a pattern

Section 4

Form Validation

Form Validation – Example

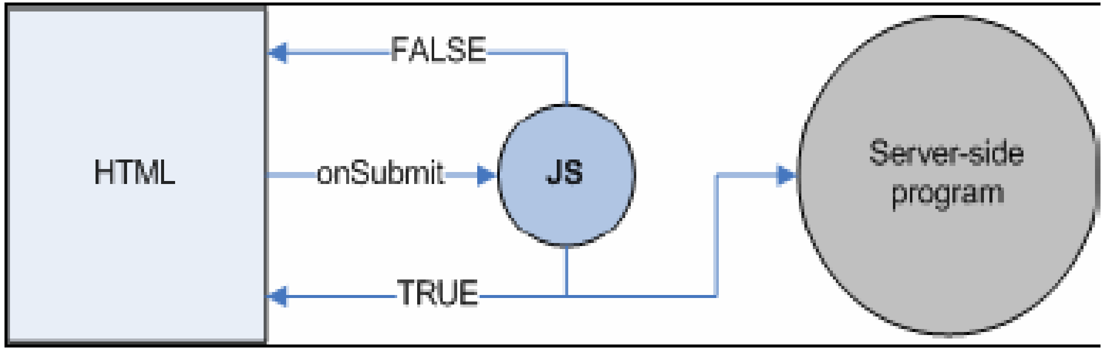
```
<script language="javascript">
function checkAll() {
    for (i = 0; i < document.forms.elements.length; i++) {
     var f = document.fields.elements[i];
     if (f.value == "") {
          alert("Please enter a value for Field " + (i + 1));
          f.style.borderColor="#FF0000";
          f.focus();
          return false;
     }
    }
    return true;
}
</script>
```

Form Validation – Overview (1)

- Have **JavaScript** validate data for the server-side program—more efficient.
  - ✓ **Processing done** on the client.
  - ✓ **Data sent** to server only once.
  - ✓ JavaScript can **update** the original HTML if errors occur
  - ✓ **Server-side** program would have to regenerate the HTML page.
  - ✓ **Server-side** program gets the data in the format it needs.

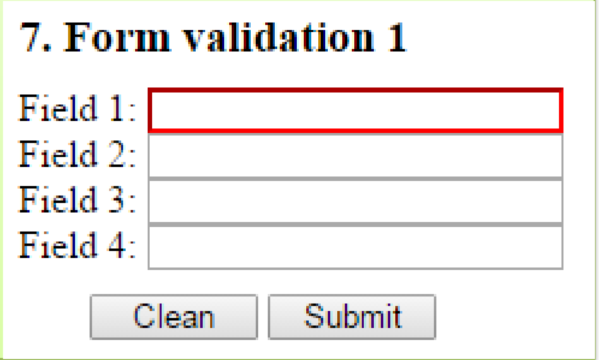
Form Validation – Overview (2)

- **Step 1:** Add an **onSubmit** event for the form.
- **Step 2:** Use the **return** keyword to get an answer back from JavaScript about whether the data is valid or not.
    - ***return false***: server-side program is not called, and the user must fix the field(s).
    - ***return true***: the valid data is sent to the server-side program.



Form Validation – Example

```
<form   method="post" name="fields" action="/cgi-bin/pgm"
            onsubmit="javascript: return checkAll();">
    <p>Field 1: <input type="text" name="f1">
    <br>Field 2: <input type="text" name="f2">
    <br>Field 3: <input type="text" name="f3">
    <br>Field 4: <input type="text" name="f4"></p>
    <input type="reset">
    <input type="submit" value="Submit">
</form>
```

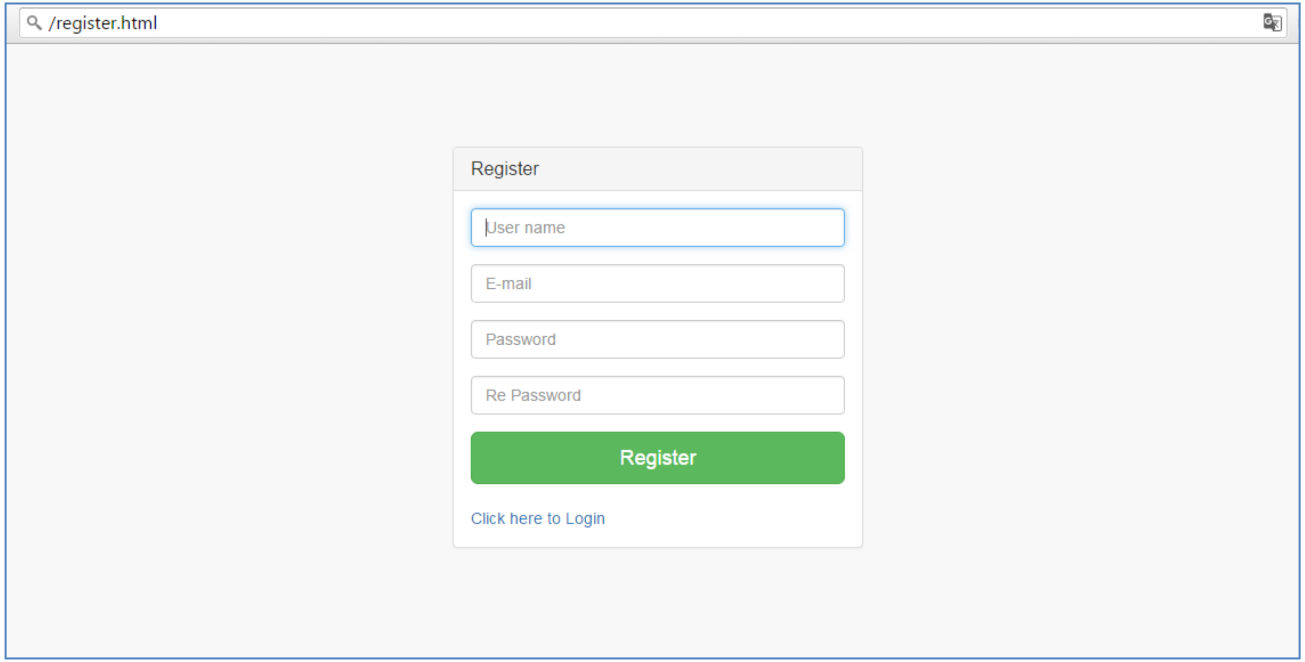


7. Form validation 1

Field 1:
Field 2:
Field 3:
Field 4:

Form Validation – Practical Time

- In this practice, we will validate data in the list item definition:



Form Validation – Summary

- **Form Validation** play an important role in every web app
- It make our app more efficient by:
  - ✓ **Processing** on the client
  - ✓ **Sent data** to server only once
  - ✓ **Feedback** errors to users

Section 5

Cookies

Cookies – Overview (1)

- Cookies let you **store user information** in web pages.
- **Cookies are data**, stored in **small text files**, on **your computer**.
- When a web server has sent a web page to a browser, the connection is shut down, and the server forgets everything about the user.

Cookies – Overview (2)

- Cookies were invented to solve the problem "**how to remember information about the user**":
  - ✓ When a user **visits** a web page, **his name** can be **stored in a cookie**.
  - ✓ **Next time** the user visits the page, the cookie "**remembers**" his name.
- Cookies are saved in name-value pairs like:

username=John Doe

Cookies – Overview (3)

- **By default**, cookies are destroyed when the browser window is closed, unless you explicitly set the expires attribute.
  - ✓ **To persist** a cookie, set the expires attribute to a future date.
  - ✓ **To delete** a cookie, set the expires attribute to a past date.

Cookies – Overview (4)

- **By default**, cookies can only be read by the web page that wrote them unless you specify one or more of these attributes:
  - ✓ **path** – allows more than one page on your site to read a cookie.
  - ✓ **domain** – allows multiple servers to read a cookie.

Cookies – Create and Read (1)

- JavaScript can **create**, **read**, and **delete** cookies with the **document.cookie** property.
- **To Create** a Cookie with JavaScript:
  - ✓ With JavaScript, a cookie can be **created** like this:

```
document.cookie="username=John Doe";
```

  - ✓ You can also add an **expiry date** (in UTC time). By default, the cookie is deleted when the browser is closed:

```
document.cookie="username=John Doe;
expires=Thu, 18 Dec 2013 12:00:00 UTC";
```

Cookies – Create and Read (2)

- **To Create** a Cookie with JavaScript:
  - ✓ With a **path parameter**, you can tell the browser what path the cookie belongs to. By default, the cookie belongs to the current page.

```
document.cookie="username=John Doe;
expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/";
```

- **Read** a Cookie with JavaScript

```
var x = document.cookie;
```

Cookies – Example (1)

- In the example to follow, we will create a cookie that stores the name of a visitor.
  - ✓ The **first time** a visitor arrives to the web page, he will be asked to fill in his name.
  - ✓ The **next time** the visitor arrives at the same page, he will get a welcome message.

Cookies – Example (2)

- For the example we will create 3 JavaScript functions:
  - ✓ A function to set a cookie value
  - ✓ A function to get a cookie value
  - ✓ A function to check a cookie value

Cookies – Example (3)

- **A Function to Set a Cookie:**

```
function setCookie(cname, cvalue, exdays) {
    var d = new Date();
    d.setTime(d.getTime() + (exdays*24*60*60*1000));
    var expires = "expires="+d.toUTCString();
    document.cookie = cname + "=" + cvalue + "; " + expires;
}
```

Cookies – Example (4)

- **A Function to Get a Cookie:**

```
function getCookie(cname) {
    var name = cname + "=";
    var ca = document.cookie.split(';');
    for(var i=0; i<ca.length; i++) {
        var c = ca[i];
        while (c.charAt(0)==' ') c = c.substring(1);
        if (c.indexOf(name)== 0)
            return c.substring(name.length,c.length);
    }
    return "";
}
```

Cookies – Example (5)

- **A Function to Check a Cookie:**

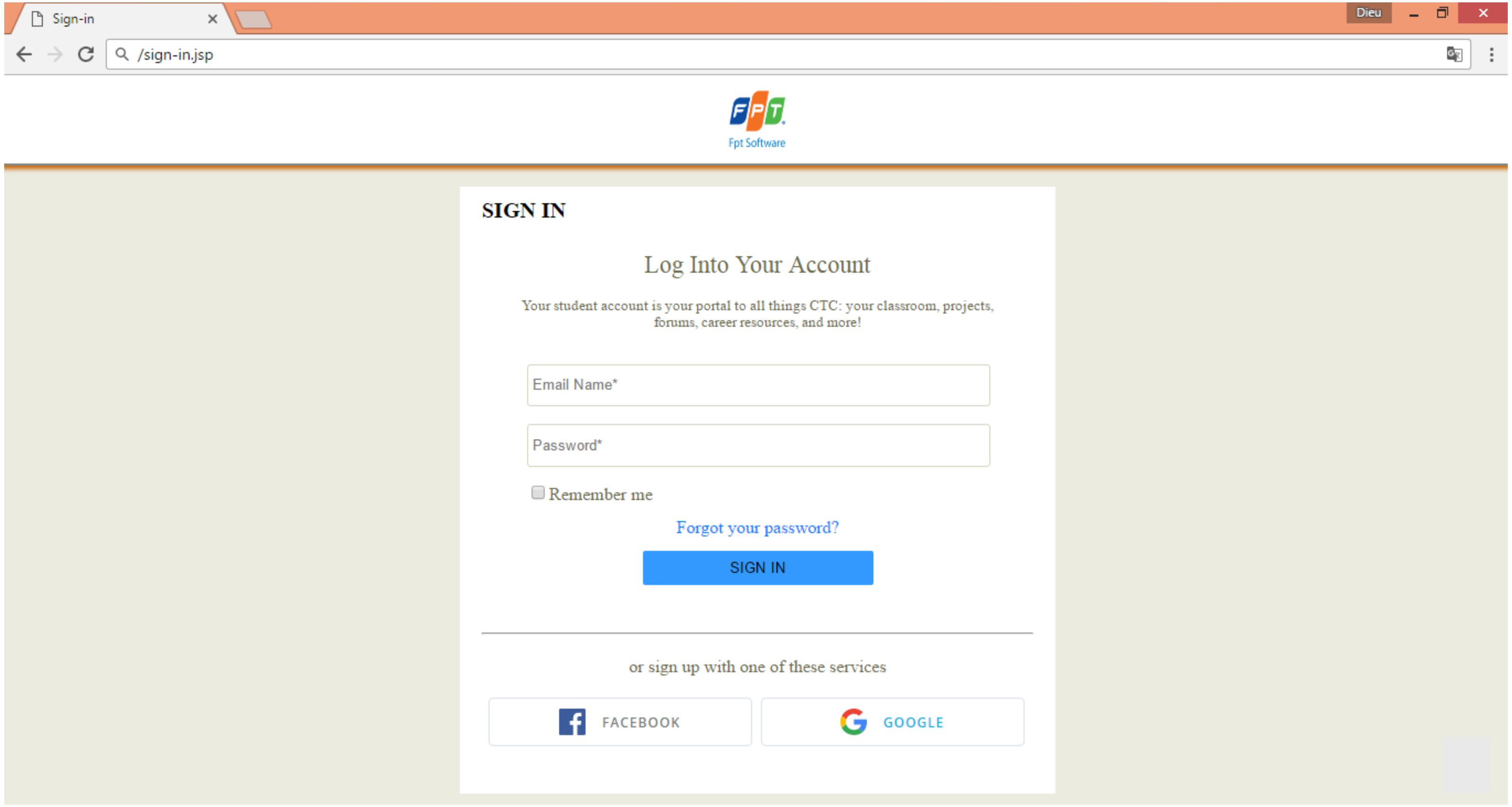
```
function checkCookie() {
    var username=getCookie("username");
    if (username!="") {
        alert("Welcome again " + username);
    }else{
        username = prompt("Please enter your name:", "");
        if (username != "" && username != null) {
            setCookie("username", username, 365);
        }
    }
}
```

Cookies – Example (6)

- **Create a form**

```
<body onload=“checkCookie();”>
<form name="cookieForm" onsubmit=“javascript: return setCookie();“
                        action=“/cgi-bin/login" method="post">
    User ID:    <input type="text" name="username"><br>
    Password:   <input type="password" name="pwd"><br>
   <input type="checkbox" name="persist"> Remember user ID <br>
   <input type="submit" value="Submit">
</form>
```

Cookies – Practical Time



Sign-in

/sign-in.jsp

SIGN IN

Log Into Your Account

Your student account is your portal to all things CTC: your classroom, projects, forums, career resources, and more!

Email Name*

Password*

☐ Remember me

Forgot your password?

SIGN IN

or sign up with one of these services

FACEBOOK GOOGLE

Cookies – Summary

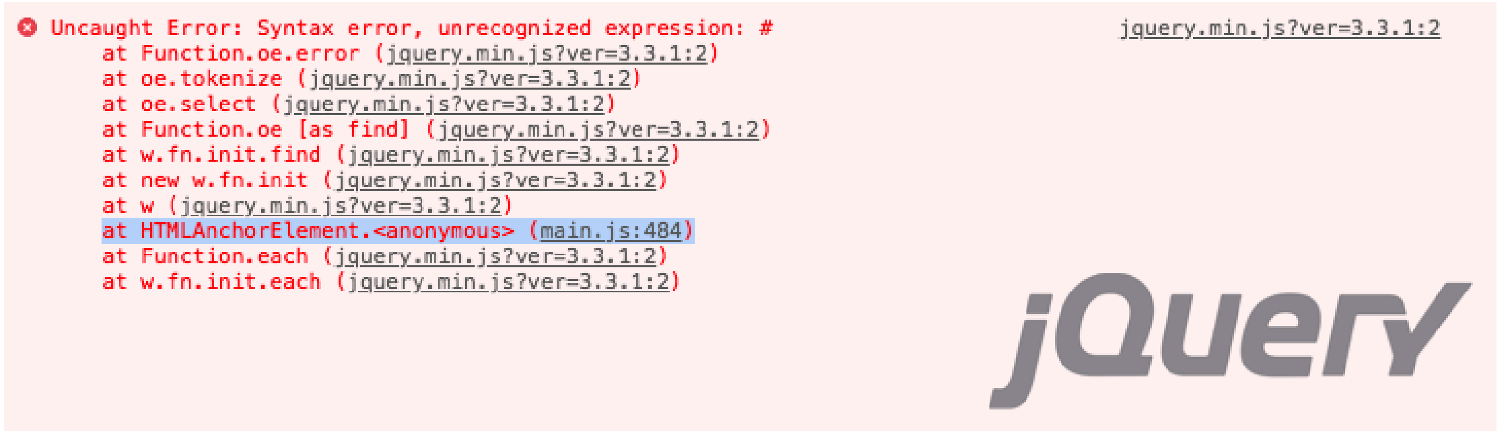
- Cookies let you **store user information** in web pages.
- **Cookies are data**, stored in **small text files**, on **your computer**.
- Cookies were invented to solve the problem "**how to remember information about the user**":
- Cookies are saved in name-value pairs like
- JavaScript can **create**, **read**, and **delete** cookies with the **document.cookie** property.

Section 6

Debugging JavaScript

Debugging JavaScript – Problem

- Difficult because the language is interpreted.
  - ✓ No compiler errors/warnings.
  - ✓ Browser will try to run the script, errors and all



```
Uncaught Error: Syntax error, unrecognized expression: #                    jquery.min.js?ver=3.3.1:2
    at Function.oe.error (jquery.min.js?ver=3.3.1:2)
    at oe.tokenize (jquery.min.js?ver=3.3.1:2)
    at oe.select (jquery.min.js?ver=3.3.1:2)
    at Function.oe [as find] (jquery.min.js?ver=3.3.1:2)
    at w.fn.init.find (jquery.min.js?ver=3.3.1:2)
    at new w.fn.init (jquery.min.js?ver=3.3.1:2)
    at w (jquery.min.js?ver=3.3.1:2)
    at HTMLAnchorElement.<anonymous> (main.js:484)
    at Function.each (jquery.min.js?ver=3.3.1:2)
    at w.fn.init.each (jquery.min.js?ver=3.3.1:2)
```

Debugging JavaScript – Tips

- Make use of Console Developer Tools
- Make each line as **granular** as possible (use variables).
- Use `console/alert` to get values of variables and see which lines are not getting processed.
- When testing form validation, set the action attribute to a dummy HTML page—not the server-side form. If you get the page, the script works.

Debugging JavaScript – Summary

- Difficult because the language is interpreted.
  - ✓ No compiler errors/warnings.
  - ✓ Browser will try to run the script, errors and all.
- Make each line as **granular** as possible (use variables).
- Use `console/alerts` to get values of variables and see which lines are not getting processed.

Thank you