JAVASCRIPT

Instructor:

Learning Goals

Understand JavaScript and Syntax

Practicce well with JavaScript

Table of contents

- Overview of JavaScript
- Document
- Object Model
- Form Validation
- Cookies
- Examples of JavaScript

Section 1

OVERVIEW OF JAVASCRIPT

What is JavaScript?

- **JavaScript** is a 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.
- **Core** JavaScript can be extended for a variety of purposes[mục đích] by supplementing it with additional objects; for example:
  - ✓ *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.

Why use JavaScript?

- To **add dynamic function** to your HTML.
- **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.
- **Better performance.**
- JavaScript can **validate** the data that users enter into the form, before it is sent to your Web application.

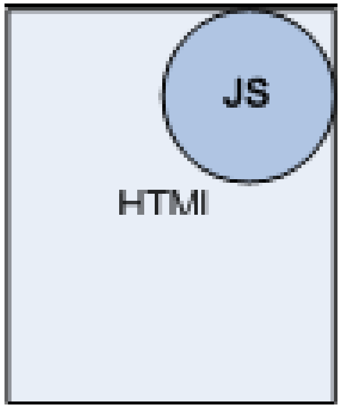
When not to use JavaScript?

- 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.**

Add JavaScript to HTML

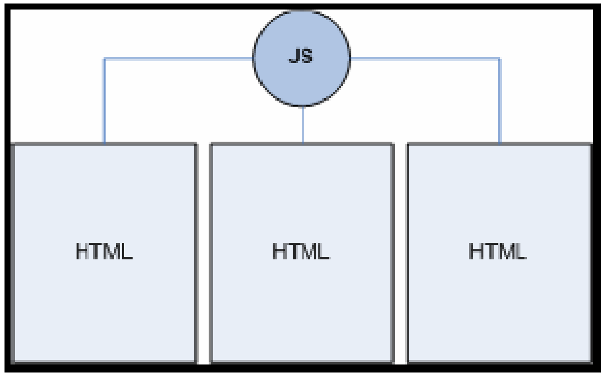
- **JavaScript** can be placed in the <**body**> and the <**head**> sections of an HTML page.
- **In the HTML page itself:**

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



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

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



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>
```

- ***In which:***
  - ✓ Function **parameters** are the **names** listed in the function definition.
  - ✓ Function **arguments** are the real **values** received by the function when it is invoked.

Events

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

Event example

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

Variables

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

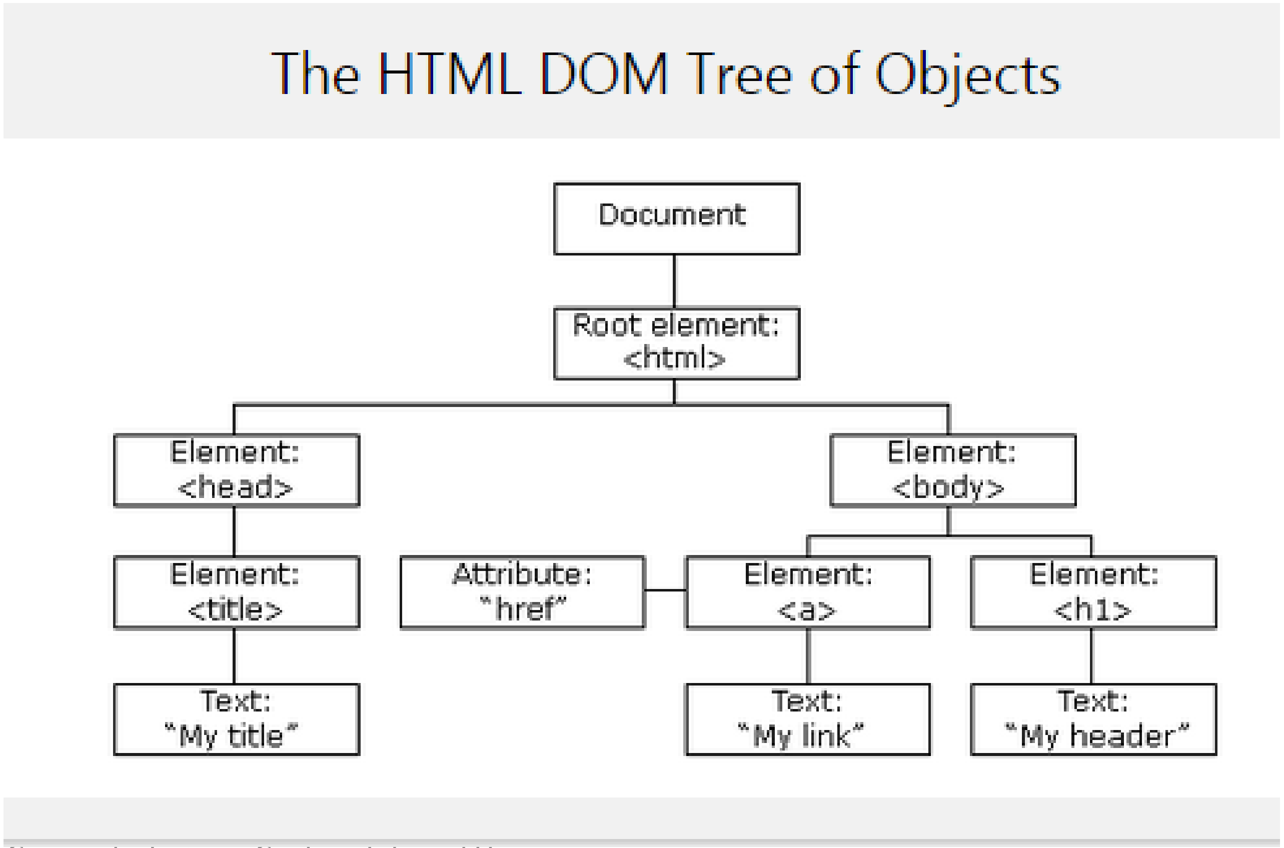
```
var num = 1;
var name = "Mel";
var phone ;
```

Section 2

DOCUMENT OBJECT MODEL

The HTML DOM

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



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>

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”>)

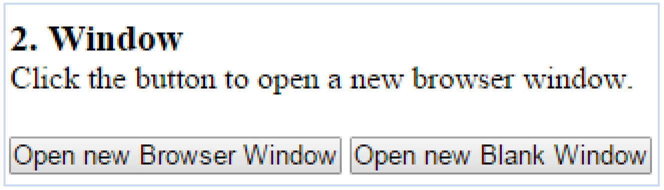
  window.document.catform.fname

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

window

- **Example:**

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



2. Window

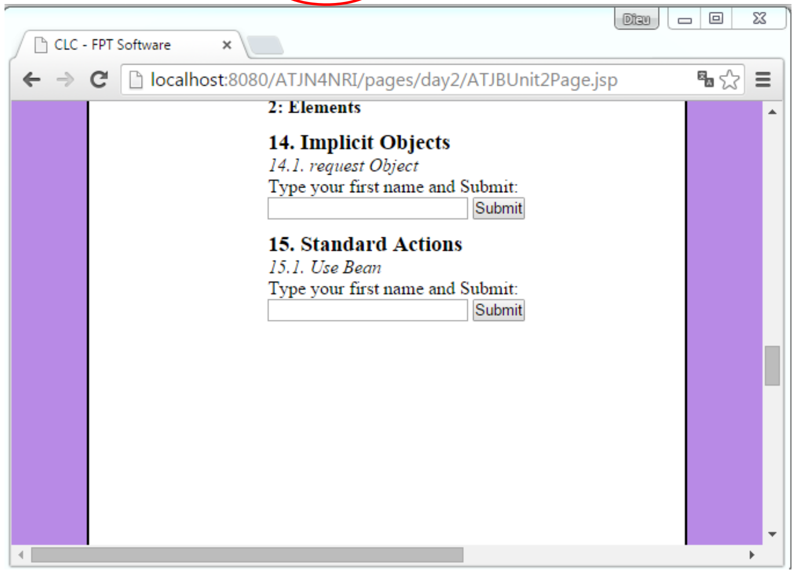
Click the button to open a new browser window.

document (HTML page) - body <body>

- **Example:**

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

Before



CLC - FPT Software

localhost:8080/ATJN4NRI/pages/day2/ATJBUnit2Page.jsp

2: Elements

14. Implicit Objects

14.1. request Object

Type your first name and Submit:

Submit

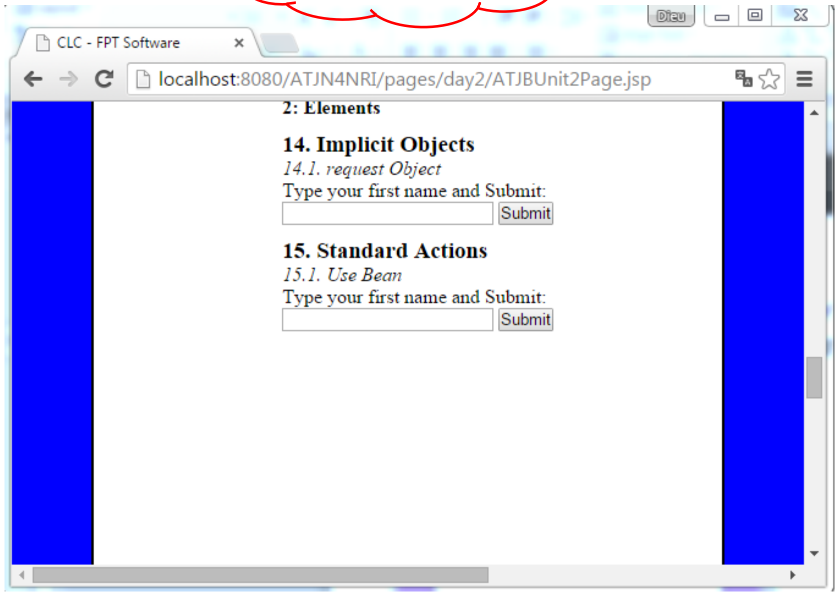
15. Standard Actions

15.1. Use Bean

Type your first name and Submit:

Submit

After



CLC - FPT Software

localhost:8080/ATJN4NRI/pages/day2/ATJBUnit2Page.jsp

2: Elements

14. Implicit Objects

14.1. request Object

Type your first name and Submit:

Submit

15. Standard Actions

15.1. Use Bean

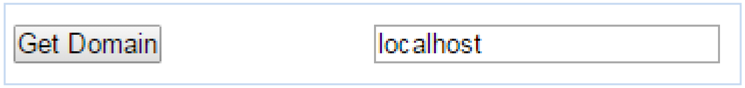
Type your first name and Submit:

Submit

location

❖ **Example:**

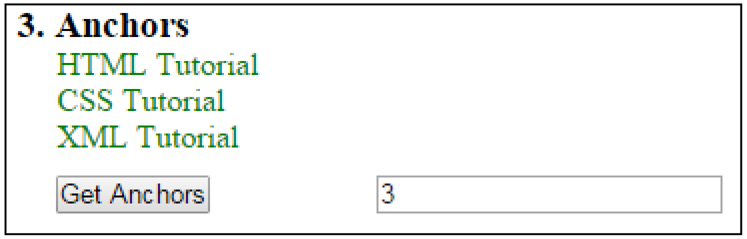
```
function getDomain() {
    document.getElementById("myText").value = document.domain;
    // or
    document.getElementById("myText").value =
                document.lastModified;
    var theText = document.getElementById("myText");
    theText.value = document.lastModified;
}
```



anchors

- **Example:**

```
<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

images

- **Examples:**

```
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";
}
```

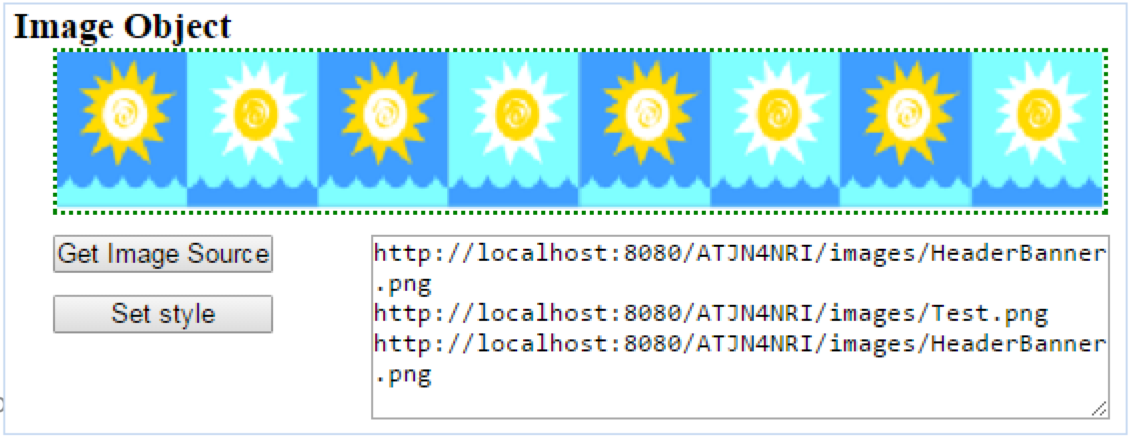


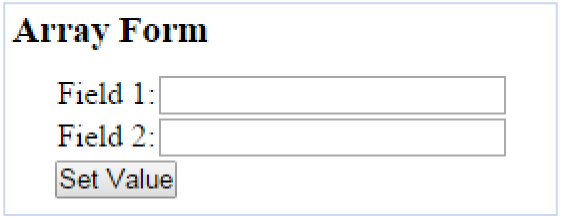
Image Object

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

forms <form>

- **Example:**

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



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 (hạn chế).

Alerts Sample

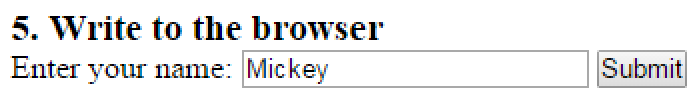
```
<html>
<head>
<script language="javascript">
function showAlert(text) {
    alert(text);
}
</script>
</head>
<body onload="showAlert
          ('This alert displays when the page is loaded!');">
. . .
//OR
<body onload="alert('This alert…');">
```

Write to the browser

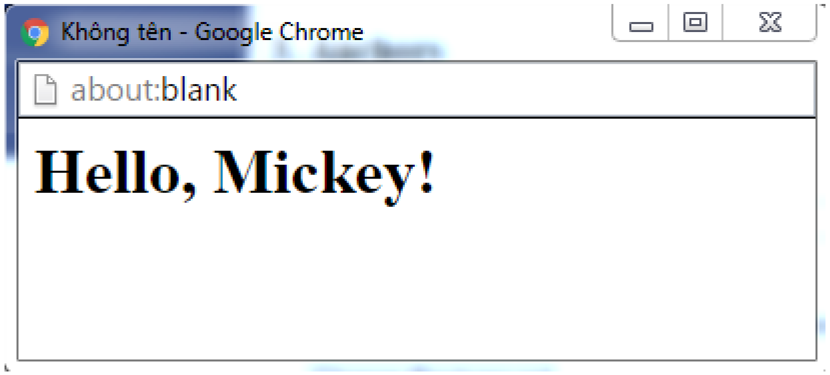
- **JavaScript** can dynamically generate a **new HTML page.** Use **document.writeln(**“*text*”);
  - ✓ **Cannot** add to the current page.
- When you’re done, use **document.close();**
- This flushes the buffer, and the generated document is then loaded into the browser.
- If the HTML code you’re generating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

Write to the browser - Sample

```
<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



Hello, Mickey!

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*”;

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>
```

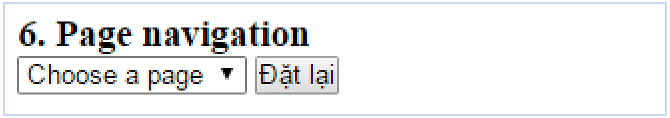

6. Page navigation

Image Swap

- 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');">
```

Practical time (1)

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>
```

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.

Section 3

FORM VALIDATION

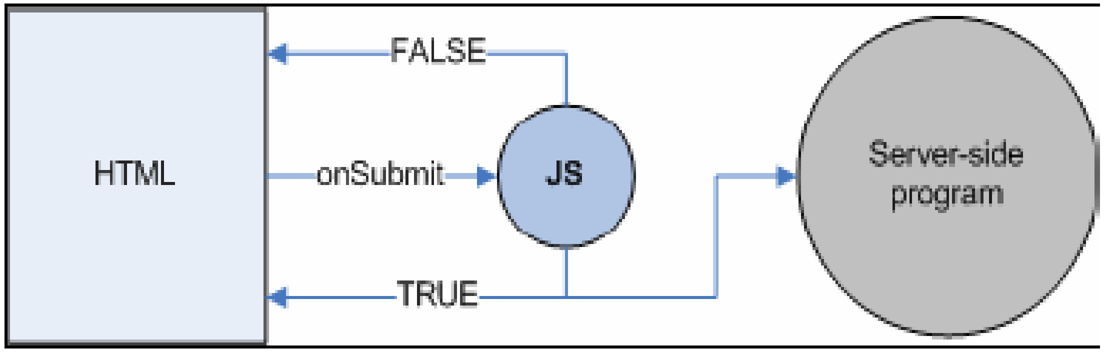
Form validation

- 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

❖ **The steps:**

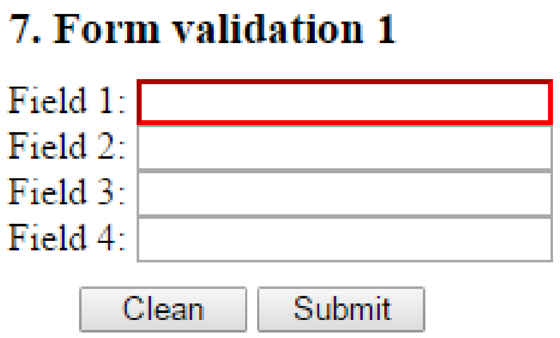
1. Add an **onSubmit** event for the form.
2. Use the **return** keyword to get an answer back from JavaScript about whether the data is valid or not.
   a) ***return false***: server-side program is not called, and the user must fix the field(s).
   b) ***return true***: the valid data is sent to the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



Form validation - Sample

- **All fields:** HTML code

```
...
<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 – Sample

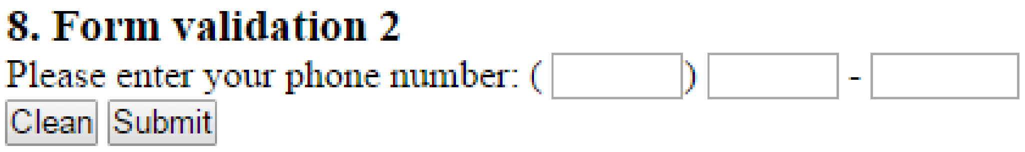
All fields: JavaScript code

```
<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 - Sample

Phone number: HTML code

```
. . .
<form onsubmit="javascript: return validPhone();”
    action=“/cgi-bin/getphone" method="post" name="phone">
    <p>Please enter your phone number:
    (<input type="text" name="area" size="3" maxlength="3">)
    <input type="text" name="pre" size="3" maxlength="3"> -
    <input type="text" name="last" size="4" maxlength="4">
    </p>
    <input type="reset">
    <input type="submit" value="Submit”>
</form>
. . .
```



8. Form validation 2

Please enter your phone number: ( ) -

Clean Submit

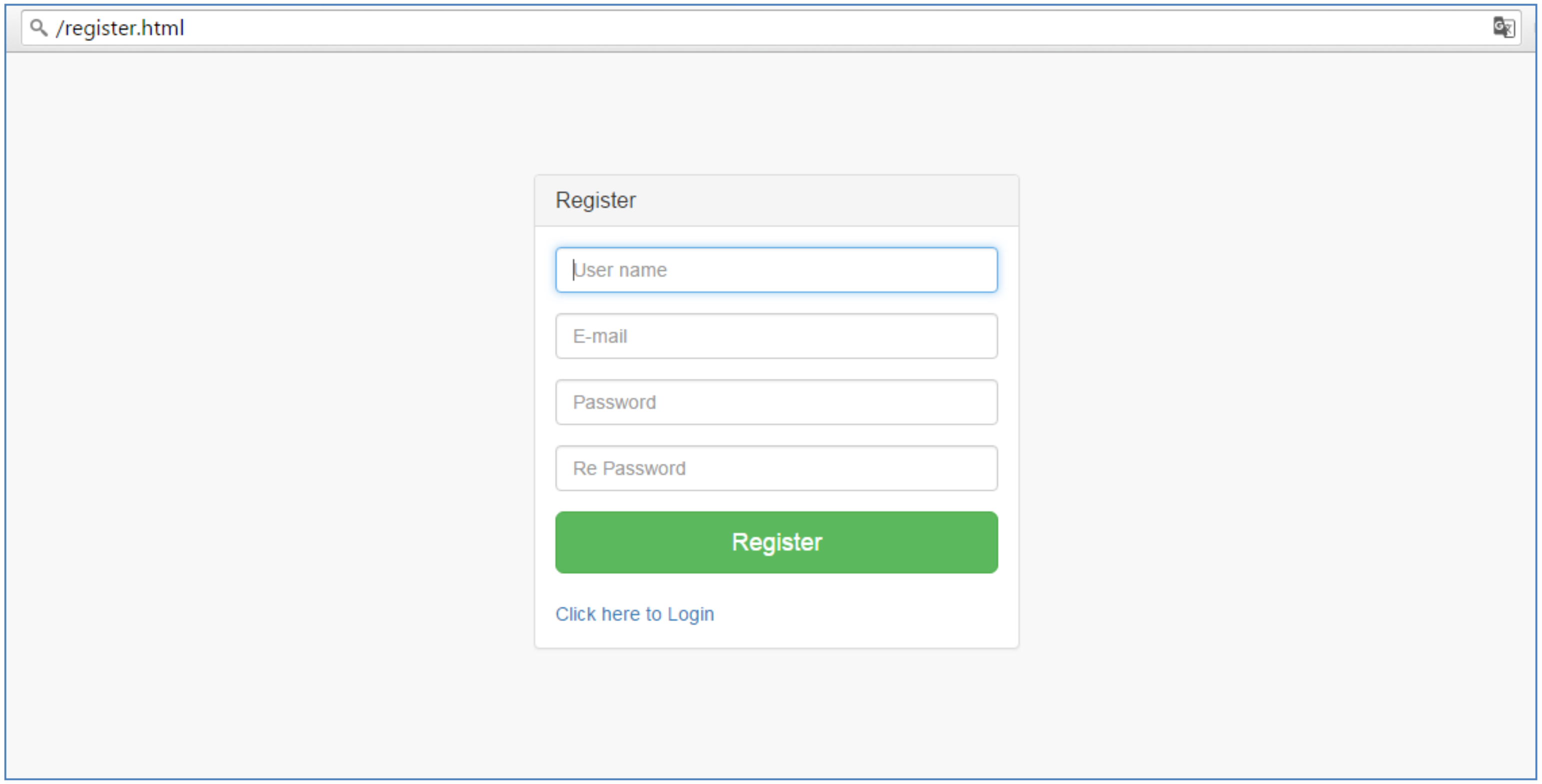
Form validation – Sample

Phone number: JavaScript code

```
function validPhone() {
    var phNum = document.phone.area.value +
    document.phone.pre.value + document.phone.last.value;
    // Check for numbers only
    for (i = 0; i < phNum.length; i++) {
    if (phNum.charAt(i) < "0" || phNum.charAt(i) > "9") {
     alert(“Please enter only numbers.”);
     return false;
    }
    }
    // Check for 10 digits
    if (phNum.length < 10) {
     alert("Please enter your 10-digit phone number.");
     return false;
    }
    return true;
}
```

Practical time

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


/register.html

Register

User name

E-mail

Password

Re Password

Section 4

COOKIES

Cookies

- 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 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

- **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.
- **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.

Create and Read a Cookie with JS

- JavaScript can **create**, **read**, and **delete** cookies with the **document.cookie** property.
- **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";
```

  - ✓ 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;
```

JavaScript Cookie Example

- 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.
- 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

JavaScript Cookie Example

- **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;
}
```

- **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 "";
}
```

JavaScript Cookie Example

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

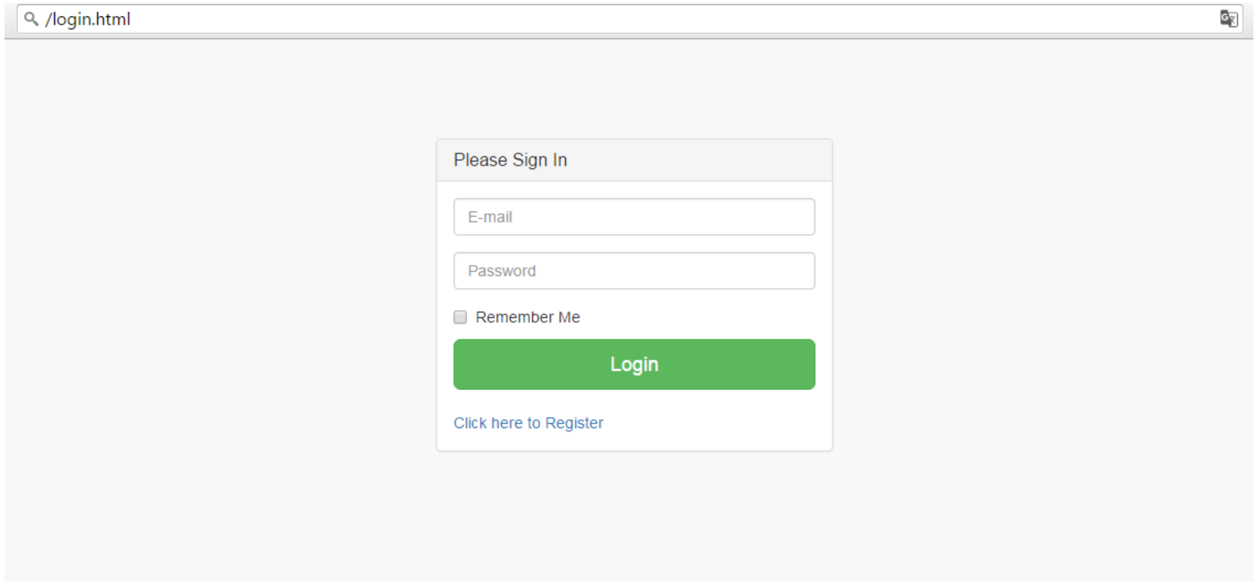
```
function checkCookie() {
    var username=getCookie("username");
    if (username!="") {
        alert("Welcome again " + username);
    }else{
        username = prompt("Please enter your name:", "");
        if (username != "" && username != null) {
            setCookie("username", username, 365);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

Cookies - Sample

❖ **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>
```

Practical time


/login.html

Please Sign In

Tips for debugging JavaScript

- 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 alerts 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.

Summary

- Understand Javascript
- Practice basic syntax in Javascript
- Practice with DOM in Javascript

Thank you