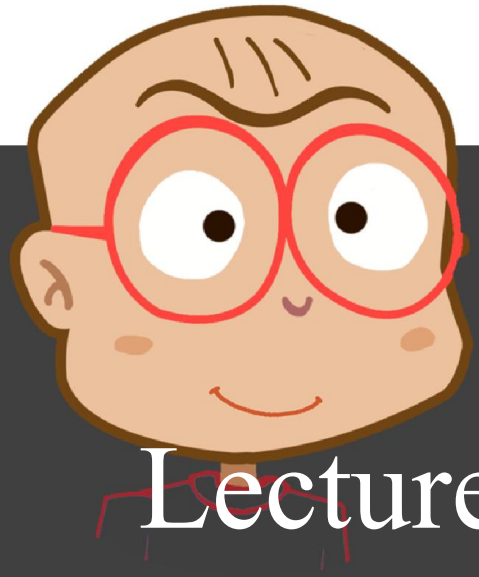


Lecture 03: Java Script Recap

Learning goals

- Understand JavaScript and Syntax
- Practicce well with JavaScript

Table of contents

Trainee's missions

To complete this course and achieve goals, trainees must:

- Read Lecture, Reference
- Do Exercises
- Take quiz
- Complete Assignment

What is JavaScript?

- A lightweight programming language that runs in a Web browser
- (client-side).
- Embedded in HTML files and can manipulate the HTML itself.
- Interpreted, not compiled.
- JavaScript is not Java.
- Developed by Netscape, not Sun.
  - Only executed in a browser.
  - Is not a full-featured programming language.
  - However, the syntax is similar.

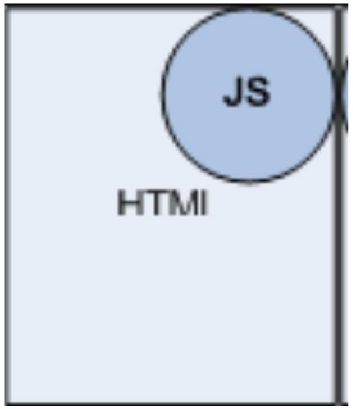
Why use JavaScript?

- 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?

- 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

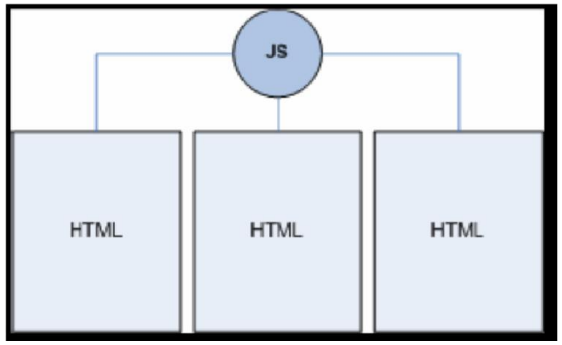


- In the HTML page itself:

```
<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

- JavaScript instructions are usually grouped together in a *function*:

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

- Like a method, procedure, or subroutine.
- Functions are called by *events*.

Events

- JavaScript is event-driven: something has to happen before the JavaScript is executed.

- 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.

- Events are specified in the HTML code.

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 ;
```

The DOM

- Unlike other programming languages, JavaScript

understands HTML and can directly access it.

- JavaScript uses the HTML Document Object

Model to manipulate HTML.

- The DOM is a hierarchy of HTML things.
- Use the DOM to build an “address” to refer to

HTML elements in a web page.

- Levels of the DOM are dot-separated in the

syntax.

Part of the DOM

Part of the DOM

- window (browser window)
- location (URL)
- document (HTML page)
- anchors <a>
- 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" . . .>)

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.

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:

document.writeln("<a href=\"file.html\">");

Write to the browser - Sample

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

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>
. . .
```

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.

Image swap - Sample

```
<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');">
```

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.

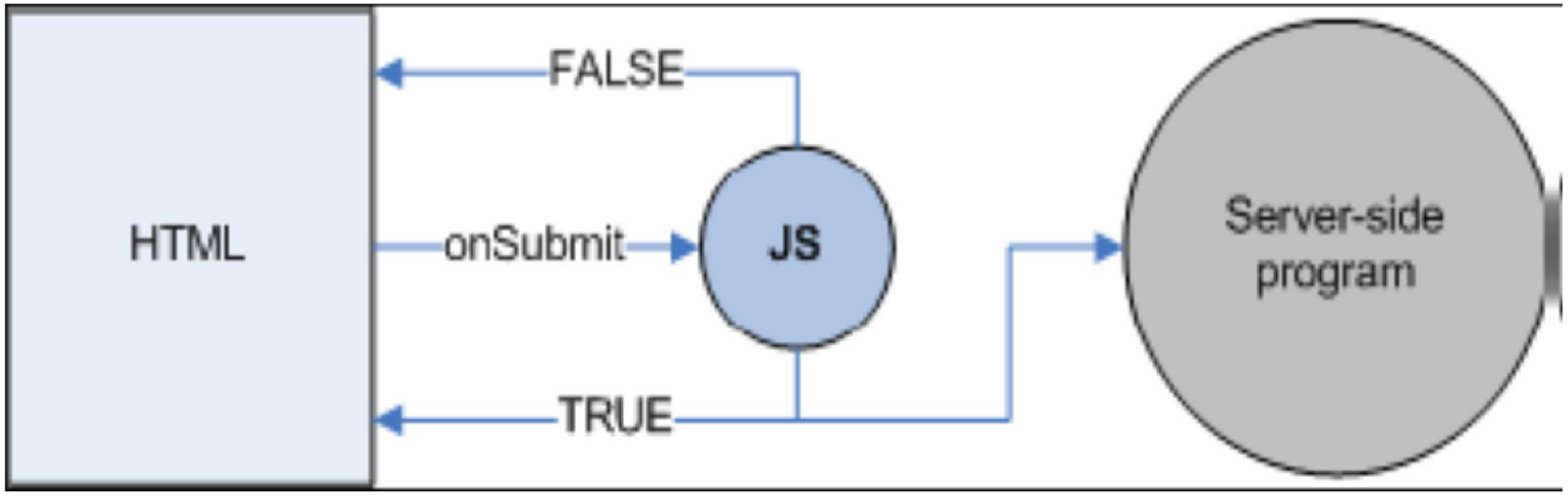
Form validation

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.

– 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



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>
. . .
```

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;
}
```

Cookies

• JavaScript provides some limited, persistent storage, called *cookies*:

– Data is stored in a text file on the client

– *name=value*

–Multiple values are delimited by a semicolon

• Use sparingly. There are limits (generally):

– Up to 300 cookies per browser, 20 cookies per web server, and 4 KB of data per cookie

• Don’t depend on cookies—users can block or delete them.

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.

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.

Tools for debugging JavaScript

- Use Netscape, Mozilla, or Firefox browsers.

– Load the page in the browser.

– Type JavaScript: in the URL window or select Tools Web Development JavaScript Console to bring up the console.

– You can also view cookie content from the browser settings.

- Download a JavaScript debugger: http://www.mozilla.org/projects/venkman/
- The JavaScript debugger for Internet Explorer is available in MS Visual Studio.

Summary

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