INTRODUCTION TO WEB DEVELOPMENT AND HTML

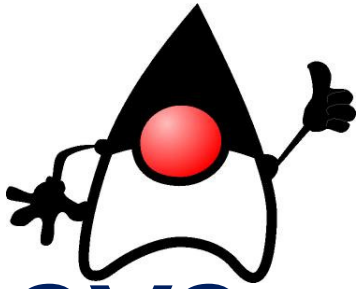
Lecture 13: Intro to JavaScript - Spring 2011

Outline

Intro to JavaScript

What is JavaScript?

JavaScript != Java



Intro to JavaScript

- JavaScript is a lightweight programming language
- There is no way of teaching all about JavaScript in just a few classes.
- However, we will learn to understand the **basics** and be **able to reuse** the thousands of free scripts available on the web such as the Google Libraries API.

Why do we need JavaScript?

JavaScript allows to:

- Read elements from documents and write new elements and text into documents
- Manipulate or move text
- Create pop-up windows
- Perform mathematical calculations on data
- React to events, such as a user's rolling over an image or clicking a button

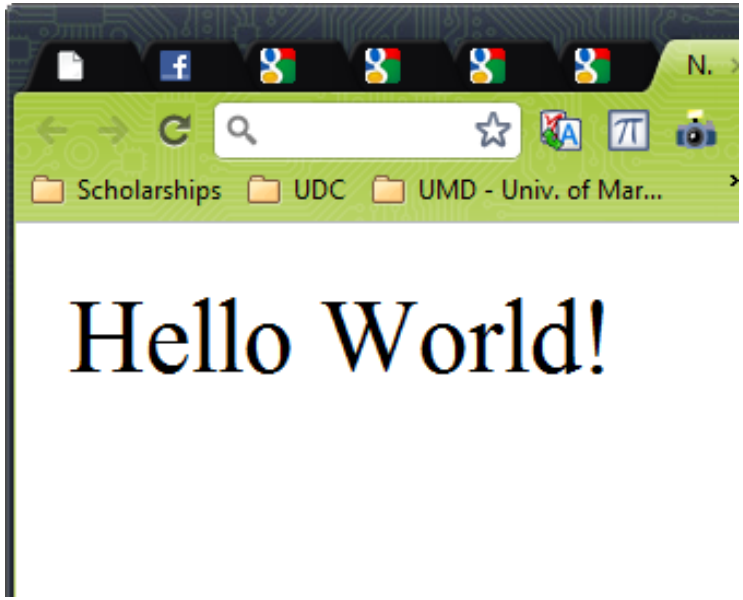
JavaScript allows to:

- Retrieve the current date and time from a user's computer or the last time a document was modified
- Determine the user's screen size, browser version, or screen resolution
- Perform actions based upon conditions such as alerting users if they enter the wrong information into a form or if they press a certain button

Hello World! In JavaScript

Our First Example

```
<html>
<body>
<p>
<script type=”text/javascript”>
   document.write(“Hello World!”)
</script>
</p>
</body>
</html>
```



The write( ) method

- It writes text into the document.

Events

- How about doing things when an event triggers???
- but first, what is an EVENT?
- An event can be something like a **key being pressed**, or a **submit button being clicked**.

Where to put your JavaScript code?

- Three places:

1. **In the <head>**: Embedded on the web page using the <script> element. It will execute when an event triggers.
2. **In the <body>:** This scripts will execute while the page is being loaded.

   <script> /* Your JavaScript Code*/</script></head>

3. **As an external file**: with extension .js (**most recommended**). Same as 1.

   <script type="JavaScript" src="js/your_script.js" />

How to comment your code?

- Use JavaScrip comments.
- Two ways of how to comment:
  1. Single Line:

     ```
     document.write("Hello World");          //this is a comment
     ```

  2. Multiple Lines:

     ```
     /* This is a multiple line comment that
     can take up to as many lines
     you want to use */
     ```

- You should **comment your code as much as possible** to make it clear for other people to read it.

Example

1. Create an external JavaScript file: script.js:

   ```
   document.write("<h1>Hello JavaScript!!!</h1>");
   ```

2. Create an html file with the following in the body

   ```
   <body>
    <script src="script.js" type="text/JavaScript"></script>
   </body>
   ```

3. Test it and comment your output.

DOM: The Document Object Model

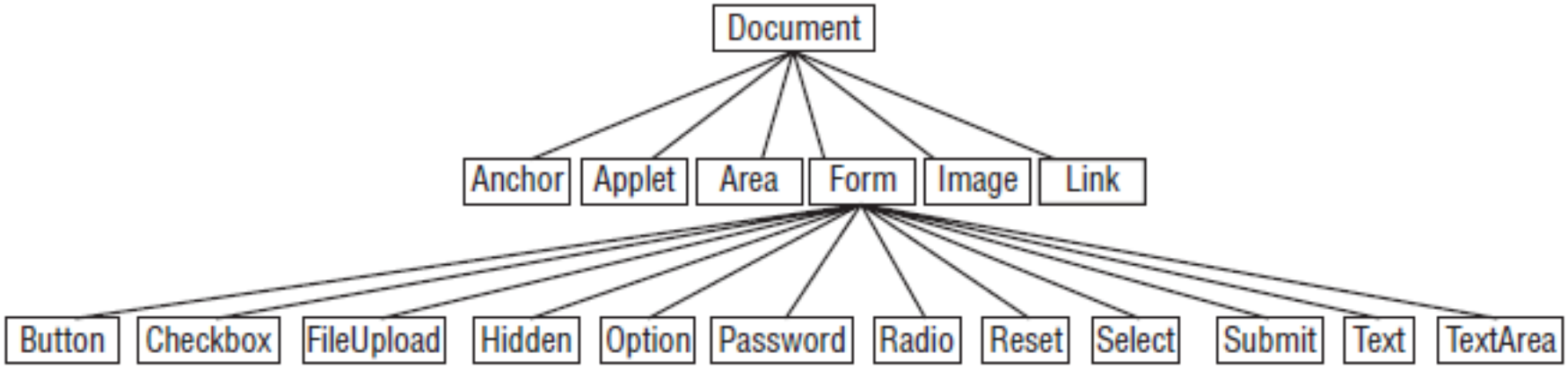

FreeFoto.com

The Document Object Model (DOM)

- Explains what ***properties of a document*** *a script can retrieve and which ones it* can alter
- Also **defines some *methods*** *that can be called to perform an action on the document*

Intro to DOM

- The figure below shows you an illustration of the **Level 0 HTML Document Object Model**
- It specifies how you can retrieve values users have entered into a form. Once you have retrieved these values, you can use JavaScript to ensure the user has entered an appropriate



Intro to DOM (Cont'd)

- The *forms collection contains all the <form> tags in the document.*
- The *image collection represents all the images in a document.*
- The *link collection represents all the hyperlinks within a page.*
- The *anchor collection represents all the anchors in a document (<a> elements with a name* or id attribute rather than an href attribute).
- The *area collection represents all the image maps that use an <area> element in the* document.
- The *applet collection represents all the applets within a document.*

Intro to DOM (Cont'd)

- The forms collection also has child objects to represent each of the different types of form controls that can appear on a form:
  - Button
  - CheckBox
  - FileUpload
  - Hidden
  - Option
  - Password
  - Radio
  - Reset, Select, Submit, Text, and TextArea.

Example 1: retrieve the value of the first link in the document

```
<h1>User Registration</h1>
<form name="frmLogin" action="login.aspx" method="post">
Username <input type="text" name="txtUsername" size="12" />
<br />
Password <input type="password" name="pwdPassword" size="12" />
   <br />
<input type="submit" value="Log In" />
</form>
<p>If you are a new user <a href="register.aspx">Register here</a> |
If you have lost your password you can
<a href="lostPassword.aspx">retrieve your password here</a>.</p>
```

Example 1: retrieve the value of the first link in the document (Cont'd)

- In order to access the first link in the document, you could use something like this:

```
document.links[0].href
```

Example 1: retrieve the value of the first link in the document (Cont'd)

There are four parts of this statement, three of which are separated by periods, to get to the first link:

1. The word document indicates access to the document object.
2. The word links corresponds to the links collection.
3. The [0] indicates that we want the first link in the document.
4. Now, we retrieve the **href** property for this link.

Example 2: retrieve the value of the text in password

```
<h1>User Registration</h1>
<form name="frmLogin" action=”login.aspx” method=”post”>
Username <input type=”text” name=”txtUsername” size=”12” />
<br />
Password <input type=”password” name=”pwdPassword” size=”12” />
   <br />
<input type="submit" value="Log In"/>
</form>
<p>If you are a new user <a href=”register.aspx”>Register here</a> |
If you have lost your password you can
<a href=”lostPassword.aspx”>retrieve your password here</a>.</p>
```

Example 2: retrieve the value of the text in password

- Or:

  document.frmLogin.**pwdPassword**.**value**

- The **document** comes first again as it is the top-level object.
- The name of the form: **frmLogin**.
- This is followed by the name of the form control: **pwdPassword**.
- Finally the property to retrieve is the value of the password box: **value**

Some DOM Properties

- **title**: The title of the page in the <title> element
- **lastModified**: The date the document was last modified. (sent by the web server)
- **referrer**: the URL of the XHTML page that users came from if they click a link. Empty is there is no referrer.

- For example to access the title of a document do:

  document.title

- Or to find out the date a document was last modified:

  document.lastModified

DOM Methods

- write(string) : Allows you to add text or elements into a document
- writeln(string) : same as write() but adds a new line at the end of the output. (like pressing Enter key)

For example:

```
document.write("page last modified on" +
  document.lastModified);
```

The Form Collections

The Forms Collection

- The forms collection holds references corresponding to each of the **<form>** elements in the page.
- 0 for the first form, 1 for the second form, 2 for the third, and so on.
- For example:

  document.forms[0].action
- Or accessing by element's name:

  document.frmLogin.action

Forms: Properties and Methods

Property Name	Purpose	Read/Write
`action`	The `action` attribute of the `<form>` element	Read/write
`length`	Gives the number of form controls in the form	Read only
`method`	The `method` attribute of the `<form>` element	Read/write
`name`	The `name` attribute of the `<form>` element	Read only
`target`	The `target` attribute of the `<form>` element	Read/write

Method Name	Purpose
`reset()`	Resets all form elements to their default values
`submit()`	Submits the form

Form Elements: Properties and Methods

- Each <form> element has an elements[ ] collection object as a property, which represents all of the elements in that form.
- Here are some of the things we can do with the elements in a form:
  - **Text fields:** Read data a user has entered or write new text to these elements.
  - **Checkboxes** and **radio buttons**: Test if they are checked and check or uncheck them.
  - **Buttons**: Disable them until a user has selected an option.
  - **Select boxes**: Select an option or see which option the user has selected.

Form Elements: Properties

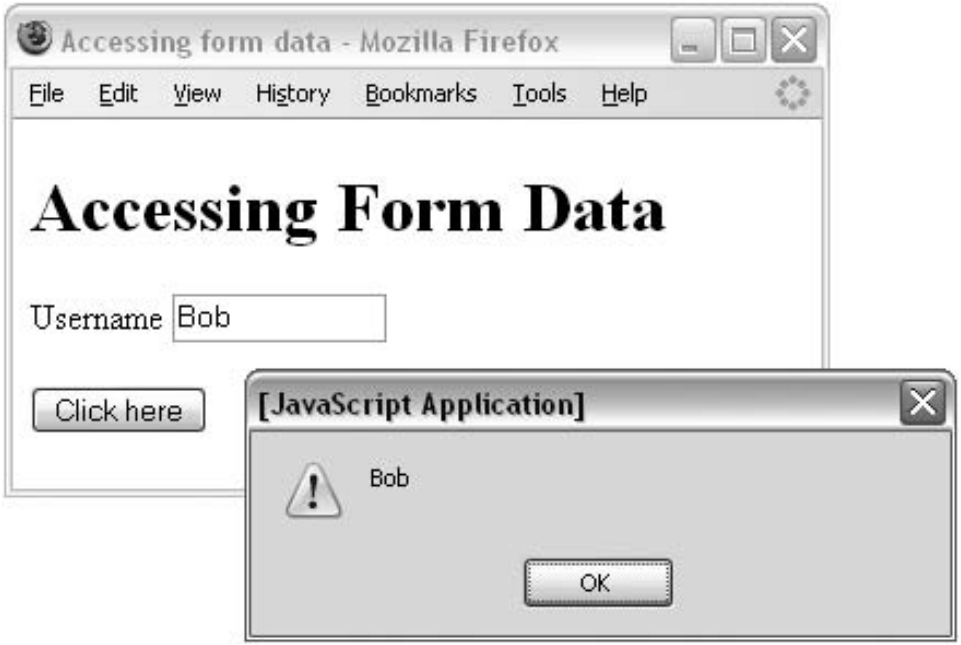
Property	Applies to	Purpose	Read/Write
`checked`	Checkboxes and radio buttons	Returns `true` when checked or `false` when not	Read/write
`disabled`	All except hidden elements	Returns `true` when disabled and user cannot interact with it (supported in IE4 and Netscape 6 and later versions only)	Read/write
`form`	All elements	Returns a reference to the form it is part of	Read only
`length`	Select boxes	Number of options in the `<select>` element	Read only
`name`	All elements	Accesses the name attribute of the element	Read only
`selectedIndex`	Select boxes	Returns the index number of the currently selected item	Read/write
`type`	All	Returns type of form control	Read only
`value`	All	Accesses the value attribute of the element or content of a text input	Read/write

Form Elements: Methods

Property Name	Applies to	Read/Write
`blur()`	All except hidden	Takes focus away from currently active element to next in tabbing order
`click()`	All except text	Simulates the user's clicking the mouse over the element
`focus()`	All except hidden	Gives focus to the element
`select()`	Text elements except hidden	Selects the text in the element

Exercise

- Retrieve the value of a text box and write it into something known as a JavaScript alert box.
- When the user clicks on a submit button of the form, the alert box will be displayed containing the text written in the input box.



The Image Collection

Images Collection

- The images collection provides references to image objects, one representing each image in a document.
- These can again be referenced by name or by their index number in the collection.
- So the src attribute of the first image could be found using the following:
  - document.images[0].src
- Or by the element's name:
  - document.myImage.src
- To create a rollover change the **src** property.

Image Collection: Properties

Property	Purpose	Read/write
`border`	The `border` attribute of the `<img>` element	Read/write
`complete`	Indicates whether an image has loaded successfully	Read only
`height`	The `height` attribute of the `<img>` element	Read/write
`hspace`	The `hspace` attribute of the `<img>` element	Read/write
`lowsrc`	The `lowsrc` attribute of the `<img>` element (indicating a lower resolution version of the image)	Read/write
`name`	The `name` attribute of the `<img>` element	Read/write
`src`	The `src` attribute of the `<img>` element	Read/write

Example: Creating an Image Rollover

- Replace one image with another one, when the user rolls over the image with the mouse.

Summary

- You will come across several types of objects in JavaScript, each of which is responsible for a related set of functionalities.
- Some other types of objects you are likely to come across:
  - **W3C DOM objects:** document, forms, images, etc.
  - **Built-in objects**: objects that are built-in or are part of JavaScript.
  - **Custom objects:** Objects you create by yourself

Questions?