CHAPTER 5

JavaScript and XHTML Documents

Note

- All examples used for this chapter are at

http://swe.umbc.edu/~zzaidi1/is448/chap5-examples

Chapter 5

- Introduction to the Document Object Model (DOM)
  - a set of JS objects representing the XHTML elements on the page

- In this chapter, we'll learn how to:
  - make our event handlers interact with HTML elements on the page
  - examine the state of the elements, e.g., whether a checkbox is checked
  - change the state of elements, e.g., putting text into a text area
  - dynamically change the styles of those elements while the page is on screen

Document Object Model

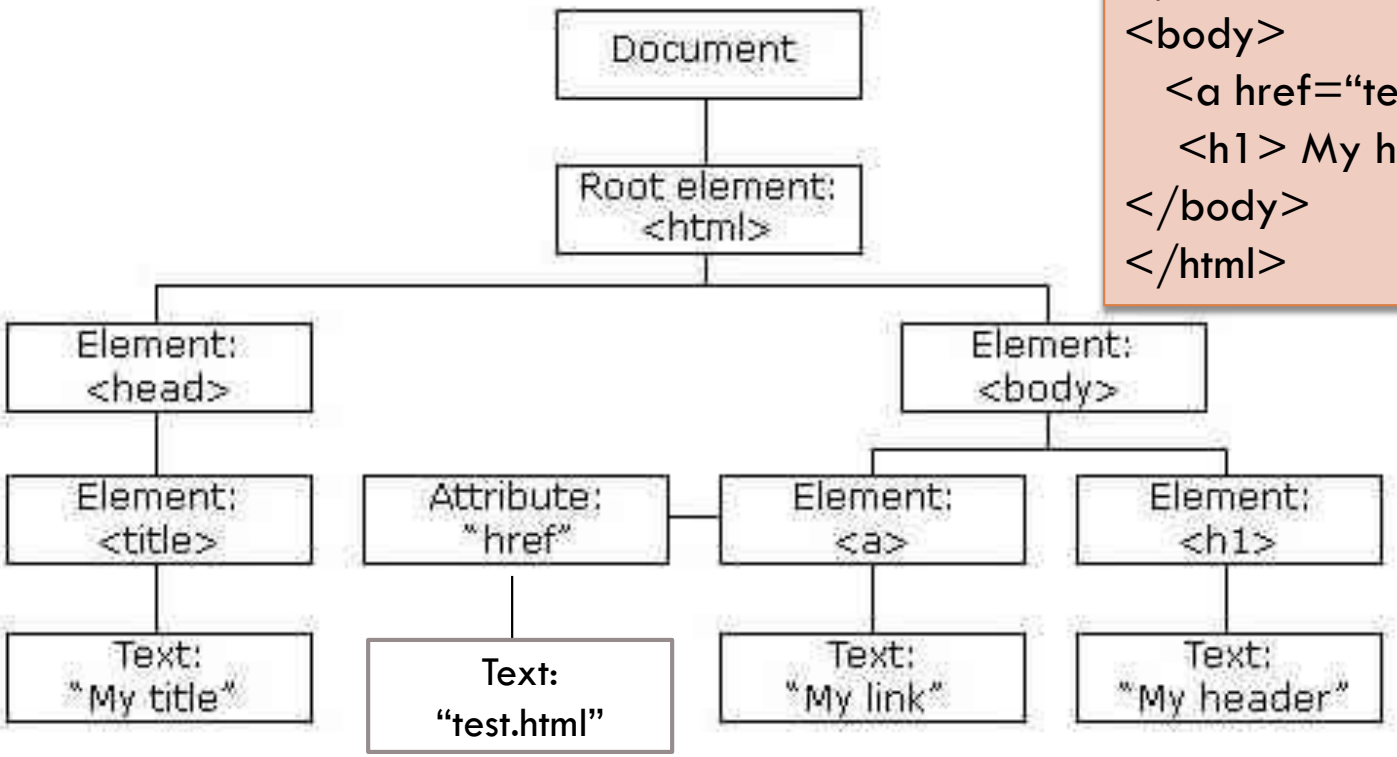
- The DOM defines an interface between XHTML documents and application programs (in this case, JavaScript programs)

- Most JavaScript code manipulates elements on an XHTML page
  - example: clicking a button makes text bold

Example DOM

- HTML DOM views the HTML page as a node tree

```
<html>
<head>
  <title> My title </title>
</head>
<body>
   <a href="test.html"> My link </a>
    <h1> My header </h1>
</body>
</html>
```



The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

Nodes in the DOM

- Everything in HTML is a node
  - The entire document is a document node
  - Every HTML tag is an element node
  - The text in the HTML elements are text nodes
  - Every HTML attribute is an attribute node
  - Comments are comment nodes

- Element nodes in the DOM correspond to a JavaScript DOM object

JavaScript binding to DOM

- Each tag in a page corresponds to a JavaScript DOM object
- JavaScript code can talk to these objects to examine elements' state
  - e.g. see whether a box is checked
- We can use JavaScript to change state
  - e.g. insert some new text into a div
- We can use JavaScript to change styles (Chapter 6)
  - e.g. make a paragraph red

5.3 How to get DOM elements in JavaScript

- Elements in XHTML document correspond to objects in JavaScript
- Objects can be addressed in several ways:
  - forms and elements array defined in DOM 0
    - Individual elements are specified by index
    - Disadvantage: the index may change when the form changes
  - Using the name attributes for form and form elements
    - A name on the form element causes validation problems
    - Names are required on form elements providing data to the server
  - Using getElementById method with id attributes
    - id attribute value must be unique for an element

5.3 How to get DOM elements in JavaScript: Using forms array

- Consider this simple form:

```
<form action = “test.php">
  <input type = "button"  name = "pushMe“/>
</form>
```

- The input button element can be referenced by

```
var element1 = document.forms[0].elements[0];
```

5.3 How to get DOM elements in JavaScript: Using name Attribute

- All elements from the reference element up to, but not including, the body must have a name attribute

- Example

```
<form name = "myForm"  action = “test.php">
  <input type = "button"  name = "pushMe“/>
</form>
```

- The input button element can be referenced by

```
var element1 = document.myForm.pushMe;
```

5.3 How to get DOM elements in JavaScript: Using id Attribute

- Preferred method
- Set the id attribute of the input element

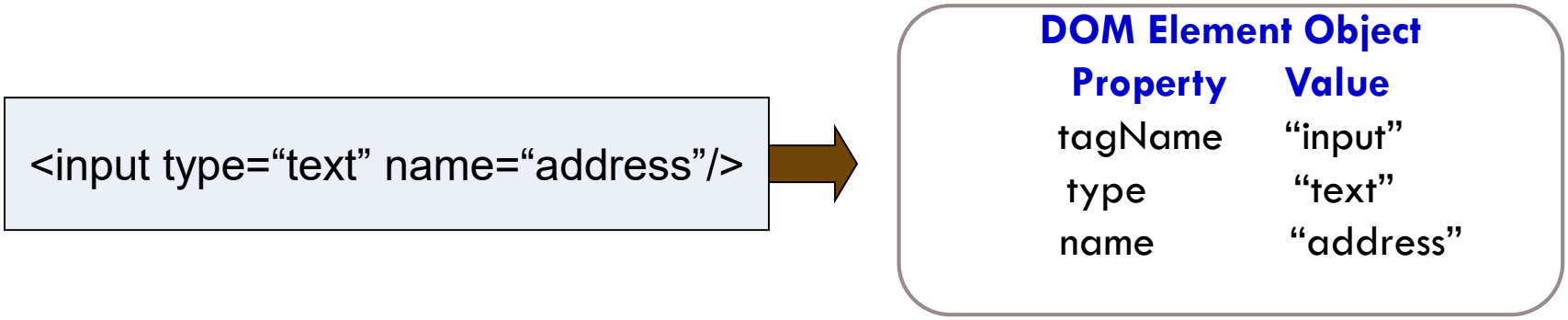
```
<form action = “test.php”>
  <input type="button"  id=“turnItOn“ name=“pushMe” />
</form>
```

- Then, use getElementById to reference the element

```
var element1 = document.getElementById(“turnItOn");
```

JavaScript binding to DOM

- JavaScript binding to the DOM: represent the current page as a set of JavaScript objects
  - e.g., each XTHML tag is represented as an object with both data properties and method properties



- We can access these objects in several ways in JavaScript
  - asking for an element's DOM object by its id
  - by traversing the page as a tree-like structure

What's inside a DOM object?

- For starters, inside the DOM object the HTML attributes are accessible as properties
- The HTML:

```
<img src="images/cat.jpg" alt="moody cat"/>
```

- Has a DOM object (lets call it catImg in JavaScript) with these properties

  catImg.src set by browser to images/cat.jpg

  catImg.alt set by browser to moody cat

Properties for Form Controls

Commonly applied to Form Control	Property	Description	Example Usage
textbox or select box	value	set or return text/value chosen/entered by user	Example to get value of textbox: var uname = document.getElementById("username").value;
check box or radio button	checked	sets or returns checked state of form control	Example to get checked state of form control var yesOrNo = document.getElementById("flex").checked
checkbox or radio button or select box	disabled	sets or returns whether form control is disabled	Example to set value of form control: document.getElementById("flex").disabled
textbox	readOnly	Sets or returns whether the text field is read only, or not	Example to set readOnly for form control document.getElement.ById("username").readOnly = true;

DOM Object's Properties

```html
<div id="main" class="foo bar">
  <p>See our <a href="sale.html" id="saleslink">Sales</a> today!</p>
  <img id="icon" src="images/borat.jpg" alt="Borat" />
</div>
```
HTML

```js
var mainDiv = document.getElementById("main");
var icon    = document.getElementById("icon");
var theLink = document.getElementById("saleslink");
```
JS

Property	Description	Example
tagName	element's HTML tag	mainDiv.tagName is "DIV"
className	CSS classes of element	mainDiv.className is "foo bar"
innerHTML	content in element	mainDiv.innerHTML is "\n <p>See our <a hr...
src	URL target of an image	icon.src is "images/borat.jpg"
href	URL target of a link	theLink.href is "sale.html"

The innerHTML property

All DOM elements have an innerHTML property that has contents of HTML tag as a string

```
<button onclick="addText();">Click me!</button>
<span id="output">Hello </span>
```
HTML

```
function addText() {
  var span = document.getElementById("output");
  span.innerHTML += " bro";
}
```
JS

Click me! Hello

output

- can change the text inside most elements by setting the `innerHTML` property

https://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_innerhtml

The value property

- Assume you have a textbox defined as follows

```
<form action="" method="get">
        User : <input type="text" id="username" />
</form>
```

- Then, you can use the getElementById to reference the element, and add the .value to get the value entered in the element

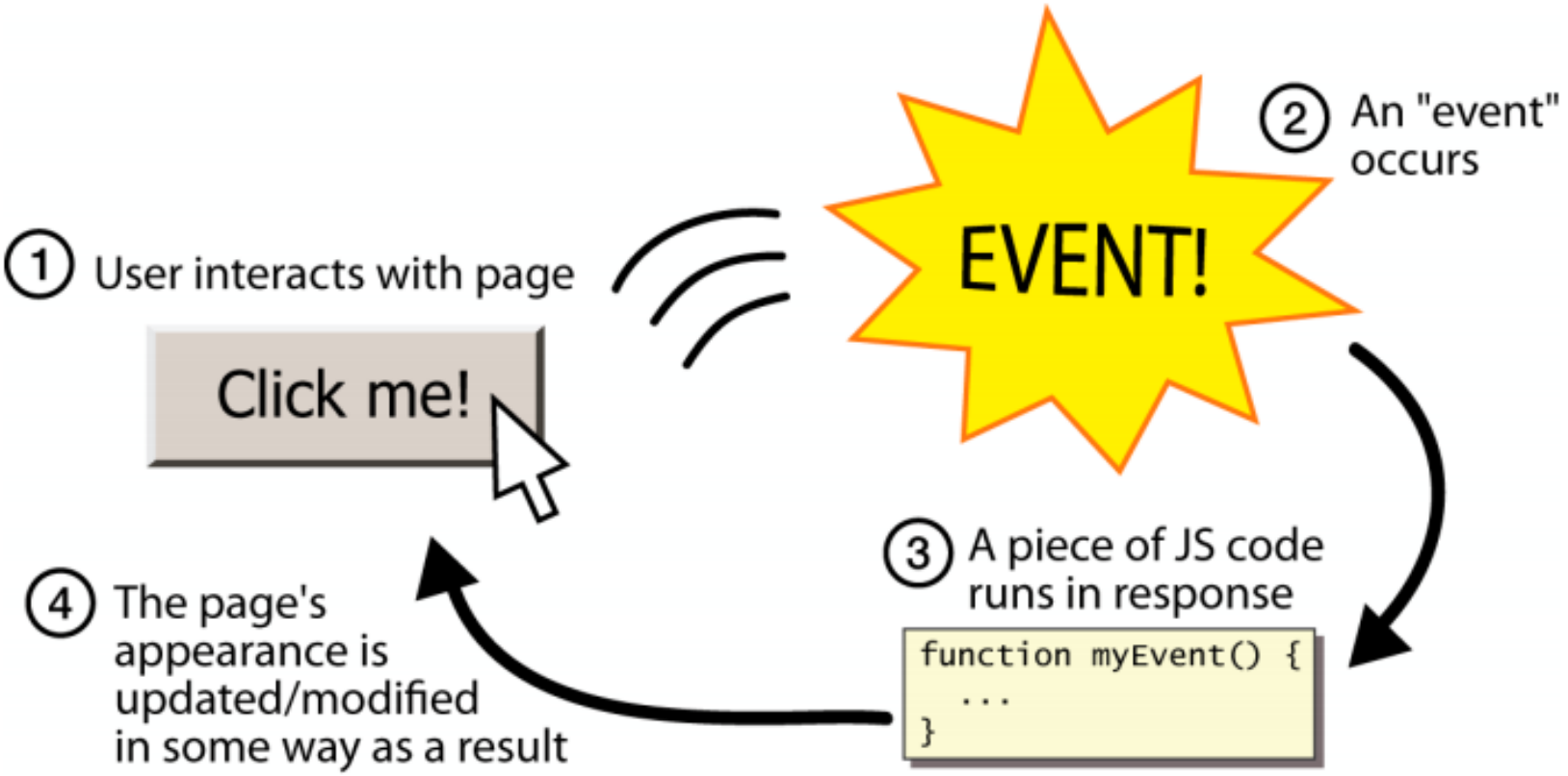
```
var name_entered = document.getElementById("username").value;
```

- See example simpleform.html, simpleform.js

Versions of DOM

- DOM 0 model
  - supported by most browsers (Firefox and IE)
- DOM 2 model
  - not supported by IE7, IE8. supported in IE9
- Our discussion in class is concentrated on methods and properties of the DOM 0 model
  - refer to Section 5.8 and W3 schools link to learn about DOM 2 model

Event-driven programming



JavaScript programs have no *main* method; they respond to user actions called events
Event-driven programming: writing programs driven by events

5.4 Events and Event Handling

- Event-driven programming is a style of programming in which pieces of code, event handlers, are written to be activated when certain events occur
- Events represent activity in the environment including, especially, user actions such as moving the mouse or typing on the keyboard
- An event handler is a program segment designed to execute when a certain event occurs
- Events are represented by JavaScript objects
- Registration is the activity of connecting JavaScript code to a type of event
  - Assign an event attribute an event handler
  - Assign a DOM node to an event handler

5.4 Events, Attributes and Tags

- Particular events are associated to certain HTML attributes

- The attribute for one kind of event may appear on different tags allowing the program to react to events affecting different components
  - Example: A textbox has an onclick attribute and a radio button has an onclick attribute. Can set each to perform a different action.

5.4 Events, Attributes and Tags

For this Event	Set this XHTML Tag Attribute
`blur`	`onblur`
`change`	`onchange`
`click`	`onclick`
`focus`	`onfocus`
`load`	`onload`
`mousedown`	`onmousedown`
`mousemove`	`onmousemove`
`mouseout`	`onmouseout`
`mouseover`	`onmouseover`
`mouseup`	`onmouseup`
`select`	`onselect`
`submit`	`onsubmit`
`unload`	`onunload`

5.4 Registering an Event Handler

- Using an attribute, a JavaScript command can be specified:

```
<input type="button" name="myButton" onclick="alert('You clicked the button!')"/>
```

OR

- A function call can be used if the handler is longer than a single statement

```
<input type="button" name="myButton" onclick="myHandlerFunction()"/>
```

This is called **obtrusive JavaScript** because we are mixing up HTML with JavaScript code.

Attaching an event handler in JS code: Step 1 (Unobtrusive JavaScript)

```
<button id="ok"> OK </button>
```

Example: see button.html, button.js

```
function pageLoad(){
  var okButton = document.getElementById("ok");
  okButton.onclick = actionOnClick;
}

function actionOnClick(){
    alert('button was clicked");
}
```

Attach event handler to elements' DOM objects in JS code

Notice that you do not put parenthesis after the function's name

This is a better style than registering in HTML code

Attaching an event handler in JS code: Step 2 (Unobtrusive JavaScript)

```
<button id="ok"> OK </button>
```

```
window.onload = pageLoad();

function pageLoad(){
  var okButton = document.getElementById("ok");
  okButton.onclick = actionOnClick;
}

function actionOnClick(){
    alert('button was clicked");
}
```

There is a global event called window.onload that happens when everything on page is loaded

Attach a function to the window.onload event, to run after everything on page is loaded

Example: see button.html, button.js

Common Errors

- event names are all lowercase
  - window.onLoad is incorrect
  - window.onload is correct
- You shouldn't write () when attaching the event handler
  - ok.onclick = actionOnClick() is incorrect
  - ok.onclick = actionOnClick; is correct
- Can't call functions directly when attaching
  - ok.onclick = alert("clicked") is incorrect
  - ok.onclick = actionOnClick;

    function actionOnClick() { alert("clicked"); } Is correct

5.5 Handling Events from Body Elements

- Load event
  - See load.html, load.js example
- Click event
  - See clickdisplay.html, clickdisplay.js
- Click event, change image
  - See changeimage.html, changeimage.js

5.6 Handling Events from Button Elements

- An event can be registered for an HTML tag by using an event attribute

```
<input type="button" name="freeOffer" id="freeButton"
   onclick="freeButtonHandlerFunction()"/>
```

- Note
  - This is a line of HTML code and the registration occurs where the form is defined
  - This line assigns the function freeButtonHandlerFunction to the **HTML attribute** onclick of the HTML button element
- Examples:
  - See radio_click.html, radio_click.js
  - See verify.html, verify.js

Event-driven programming

To make a responsive button or other UI control:

1. choose the control (e.g. button) and event (e.g. mouse click) of interest
2. write a JavaScript function to run when the event occurs
3. attach the function to the event on the control

Lab

- Download colors.html which has a textbox and five radio buttons, labeled red, blue, green, yellow and orange
- Augment your HTML page to reference a JavaScript file that you will be creating
- In the JavaScript file, write JavaScript code that does the following
  - Define an event handler for these buttons that displays an alert message displaying the user's name and their chosen favorite color
  - The event handler must be implemented as a function
  - Attach event handler to radio buttons clicking using unobtrusive JavaScript
- Use the following examples to help you when working on this lab
  - simpleform.html, simpleform.js
  - button.html, button.js
  - clickdisplay.html, clickdisplay.js
  - changeimage.html, changeimage.js
  - radio_click.html, radio_click.js
  - verify.html, verify.js

5.7 Validating Form Input

- Validating data using JavaScript provides quicker interaction for the user
  - Validity checking on the server requires a round-trip for the server to check the data and then to respond with an appropriate error page
- After identifying the invalid data
  - Put the focus in the field in question (the *focus* method)
  - Highlight the text for easier editing (the *select* method)
- See pswd_chk.html, pswd_chk.js: illustrates validity checking

5.7 Validating Form Input (2 ways)

- See validator.onclick.html, validator.onclick.js, display_info,php
  - Event registration occurs with the *onclick* attribute of the *submit* button
- See validator.onsubmit.html, validator.onsubmit.js, display_info.php
  - Event registration occurs with the *onsubmit* attribute of the *form* tag (*onsubmit* can only be used with the *form* tag)
- In both these examples
  - Note, the use of the *return statement* where function is called
  - Note, the use of the *return false* or *return true* statements at various points in the body of the function in the .js file
  - Note the use of regular expressions to validate text input
    - The name is "first last" /^[a-z]+\s+[a-z]+$/i
    - The phone is (ddd)ddd-dddd where d is a digit /^\(\d{3}\)\d{3}-\d{4}$/
    - Each regular expression uses the ^ and $ anchors to make sure the entire string matches
- All files are in eg6-validator-onclick folder

Lab – Part 1

- Download the file forms.html, form_proc.css
- Open Developer Tools in your browser
- Create a a new JavaScript file and define a function to perform the following validation to form input when user submits the data
  - User first name must contain only letters and must not be empty

Hint: use regular expressions (from slides 7.pdf)

- Display an alert box when the error is encountered
- Use the following files as examples as you work on this lab
  - pswd_chk.html, pswd_chk.js
  - validator-example

Lab - Part 2

- Add code to the same JS function from part 1 to check for the following:
  - last name must contain only letters and must not be empty

Hint: use regular expressions (from slides 7.pdf)

- Display an alert box when the error is encountered
- Use the following files as examples as you work on this lab
  - pswd_chk.html, pswd_chk.js
  - validator-example
- Look for any errors using Developer Tools

Lab – Part 3

- Add code to the same JS function from part 2 to check for the following:
  - User middle initial must be a single upper-case letter

Hint: use regular expressions (from slides 7.pdf)

- Display an alert box when the error is encountered
- Use the following files as examples as you work on this lab
  - pswd_chk.html, pswd_chk.js
  - validator-example
- Look for any errors using Developer Tools

Lab – Part 4

- Add code to the same JS function from part 3 to check for the following
  - Address line must contain one or more digits, followed by one or more spaces, followed by the one or more characters. E.g., 1000 Hilltop Circle

Hint: use regular expressions (from slides 7.pdf)

- Display an alert box when the error is encountered
- Use the following files as examples as you work on this lab
  - pswd_chk.html, pswd_chk.js
  - validator-example
- Look for any errors using Developer Tools

Lab – Part 5

- Add code to the same JS function from part 4 to check for the following
  - The only two entries allowed for City are ‘baltimore’ or ‘columbia’

Hint: may use regular expressions (from slides 7.pdf)

- Display an alert box when the error is encountered
- Use the following files as examples as you work on this lab
  - pswd_chk.html, pswd_chk.js
  - validator-example
- Look for any errors using Developer Tools