JavaScript and HTML Documents

Xingquan (Hill) Zhu
xqzhu@cse.fau.edu

DHTML

- JS Execution Environment
- The Document Object Model (DOM)
- Element Access in JS
  - Dynamic style & positioning
- Events and Event Handling
  - Handling events from body elements
  - Handling events from button elements
  - Handling events from text and password elements
    - Pattern Matching with Regular Expression
- The navigator Object

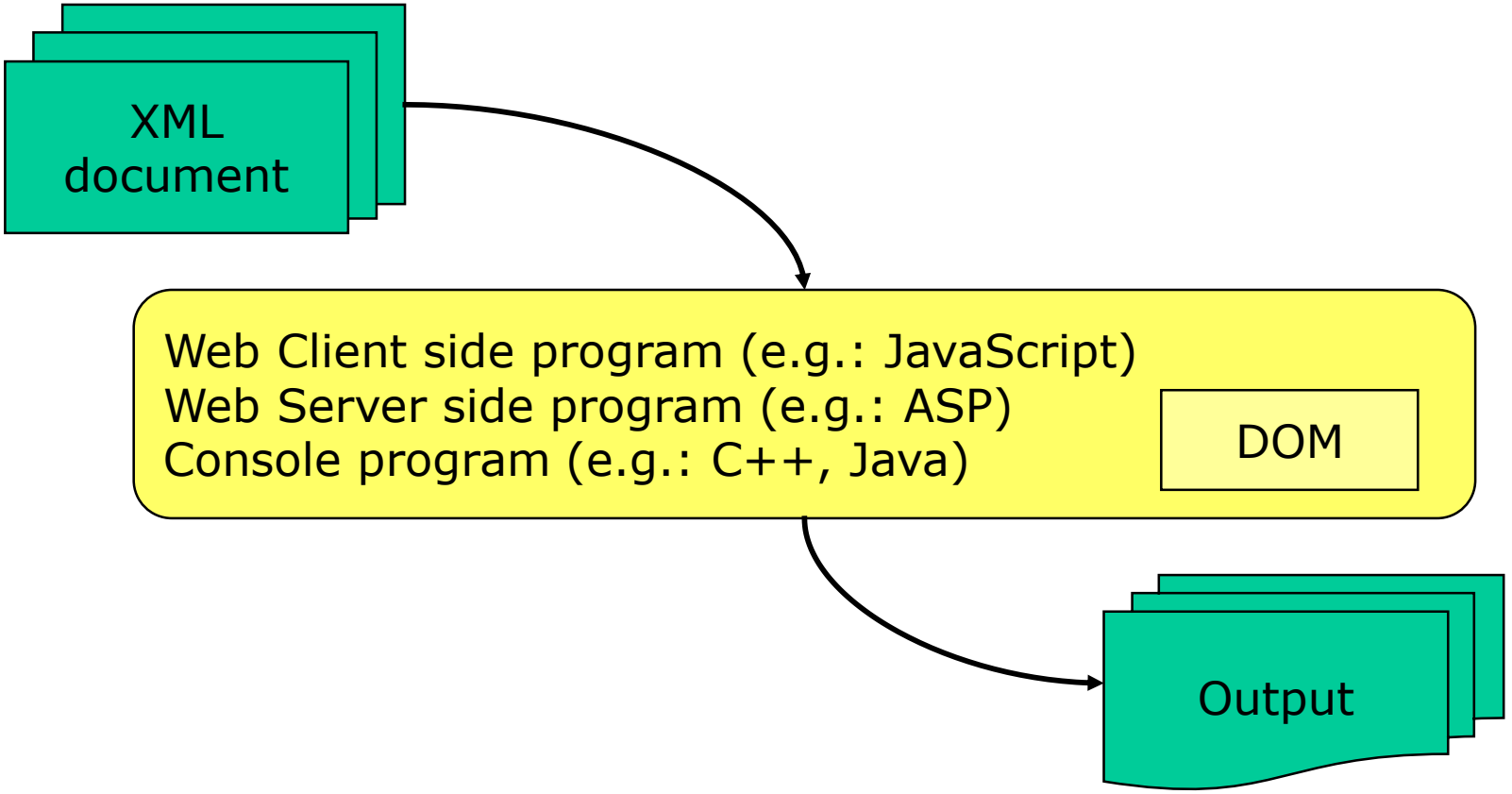
JS Execution Environment

- The JavaScript Window object represents the window in which the browser displays documents
- All JS variables are properties of some object
- The Window object provides the largest enclosing referencing environment for scripts
  - All global variables are properties of Window
- Implicitly defined Window properties:
  - document - a reference to the Document object that the window displays
  - frames - an array of references to the frames of the document
- Every Document object has:
  - forms - an array of references to the forms of the document
    - Each Form object has an elements array, which has references to the form's elements
  - Document also has anchors, links, & images
- There can be more than one Window object
  - A variable declared in one Window object is not a global variable in another Window object
  - Cross reference is available

The Document Object Model (DOM)

- The DOM is a platform- and language-neutral interface
  - Allow programs and scripts to dynamically access and update the content, structure and style of documents.
  - With the DOM, users can code in programming languages to create documents, move around in their structures, change, add, or delete elements and their content
- DOM 0 is supported by all JavaScript-enabled browsers (no written specification)
  - DOM 1 was released in 1998
  - DOM 2 is the latest approved standard (2000)
    - Nearly completely supported by NS7
    - IE6's support is lacking some important things

The Relation Graph



DOM

- A language that supports the DOM must have a binding to the DOM constructs

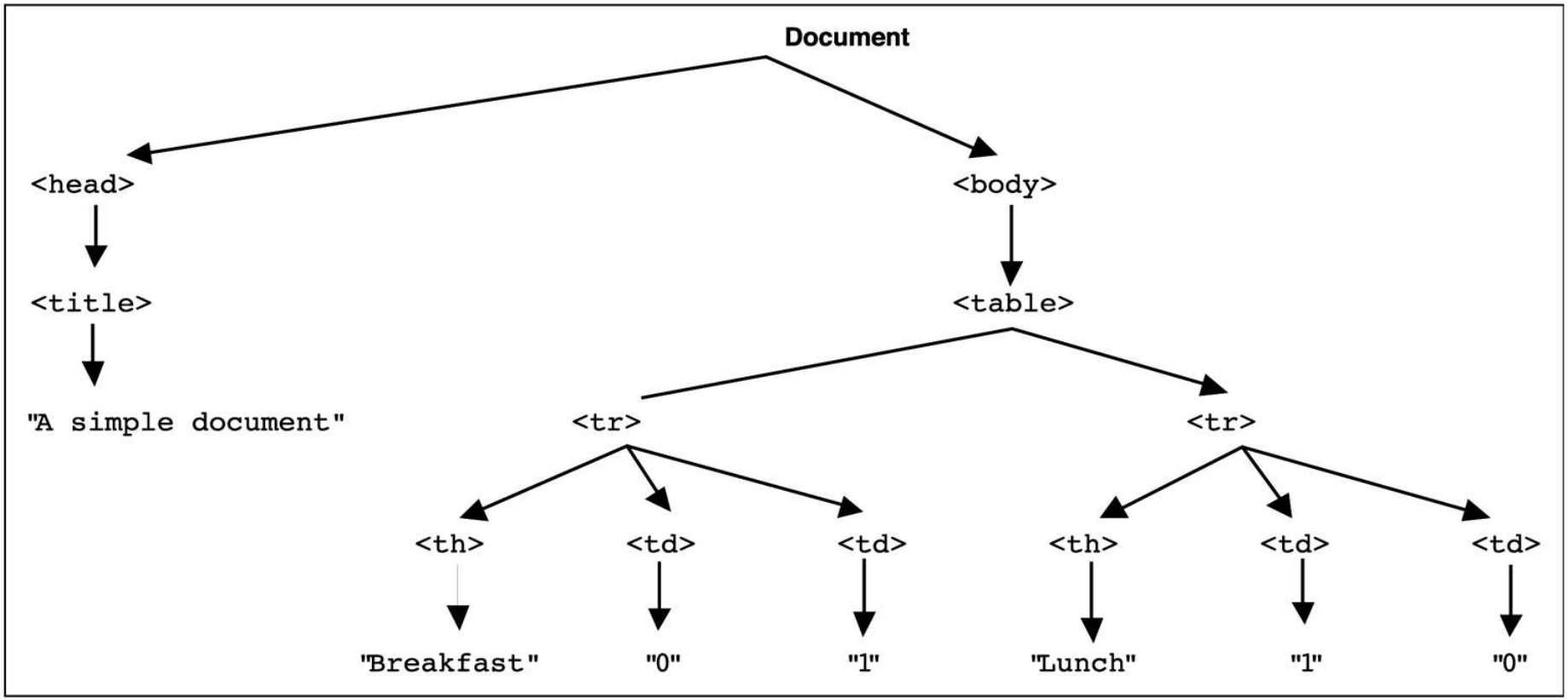
- In the JavaScript binding, HTML elements are represented as objects and element attributes are represented as properties

  e.g., <input type = "text" name = "address">
  would be represented as an object with two properties, type and name, with the values "text" and "address"

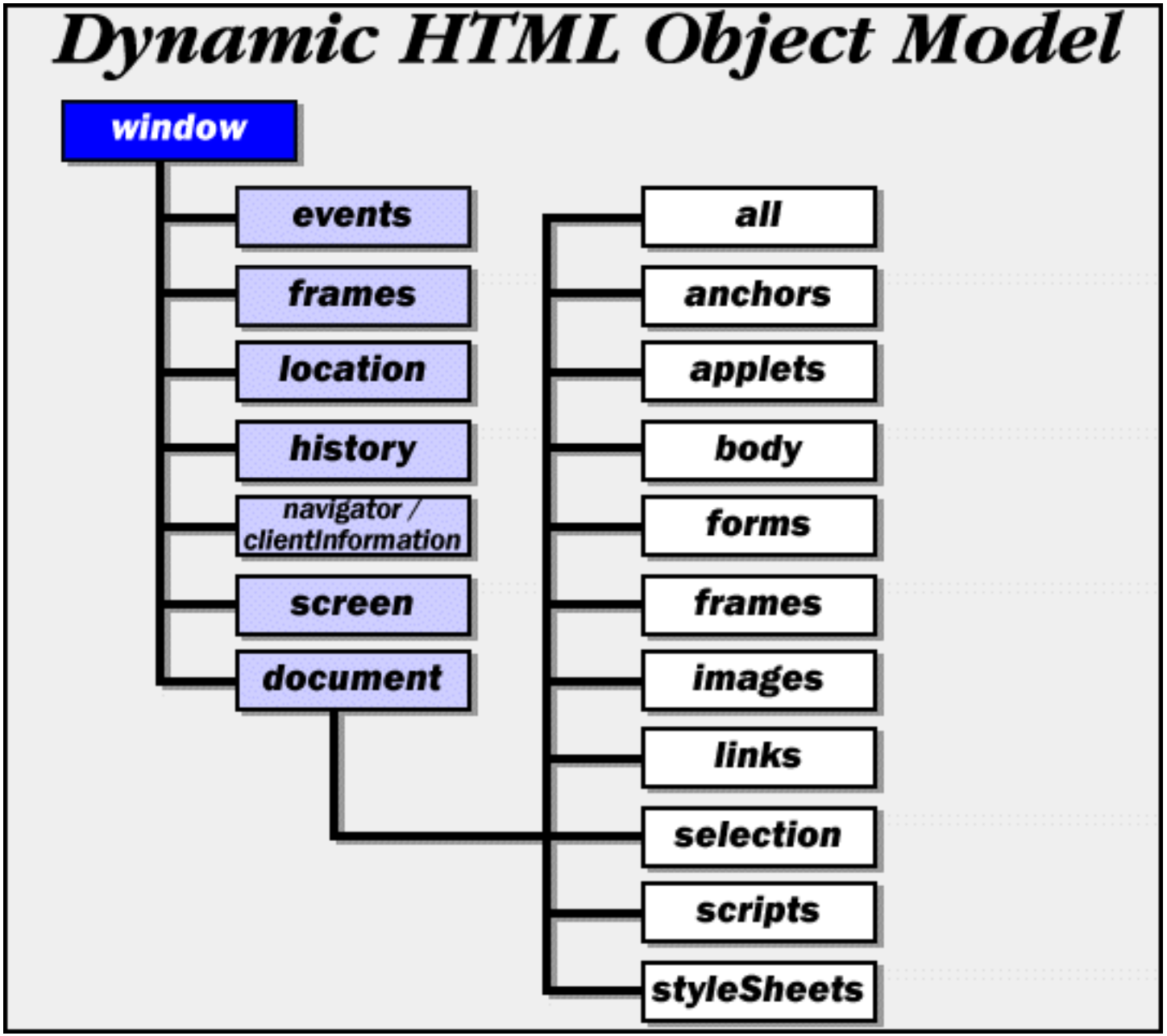
The DOM structure for a simple document

The DOM is an abstract model that defines the interface between HTML documents and application programs—an API

DOM.html



Dynamic HTML Object Model



DHTML

- JS Execution Environment
- The Document Object Model (DOM)
- Element Access in JS
  - Dynamic style & positioning
- Events and Event Handling
  - Handling events from body elements
  - Handling events from button elements
  - Handling events from text and password elements
    - Pattern Matching with Regular Expression
- The navigator Object

Element Access in JS

- **There are several ways to do it**
  Example (a document with just one form and one widget):

```
<form action = "">
        <input type = "button"  name = "pushMe">
</form>
```

- **1. *DOM address***
  - document.forms[0].elements[0]
  - *Problem*: document changes volume.html
- **2. *Element names* – requires the element and all of its ancestors (except body) to have name attributes**

```
<form name = "myForm"  action = "">
        <input type = "button"  name = "pushMe">
</form>

document.myForm.pushMe
```

***Problem*: XHTML 1.1 spec doesn't allow the name attribute on form elements**

Element Access in JS

- 3. getElementById Method (defined in DOM 1)

  Example:

  ```
  <form action = "">
          <input type = "button"  id = "pushMe">
  </form>
  document.getElementById("pushMe")
  ```

  Reference.html

- getElementsByName getelementsbyname.html

- getElementsByTagName
  - **var e=document.getElementsByTagName("div");**

Element Access in JS

- Checkboxes and radio button have an implicit array, which has their name

```
<form id = "topGroup">
      <input type = "checkbox"  name = "toppings" value = "olives" />
                  ...
      <input type = "checkbox"  name = "toppings" value = "tomatoes" />
</form>
      ...
var numChecked = 0;
var dom = document.getElementById("topGroup");
for index = 0; index < dom.toppings.length; index++)
      if (dom.toppings[index].checked]
                  numChecked++;
```

Collections all and children

- Collections
  - Arrays of related objects on a page
  - all
    - all the XHTML elements in a document
    - Even for those without id/name
  - children
    - Specific element contains that element's child elements

All.html Children.html
Innerhtmlvsouterhtml.html

Dynamic Styles

- Element's style can be changed dynamically
- Dynamic HTML Object Model also allows you to change the class attribute

Dynamicstyle.html

Dynamicstyle2.html

Dynamic positioning

- XHTML elements can be positioned with scripting
  - Declare an element's CSS position property to be either absolute or relative
  - Move the element by manipulating any of the top, left, right or bottom CSS properties

Dynamicposition.html Dynamicposition_2.html

DHTML

- JS Execution Environment
- The Document Object Model (DOM)
- Element Access in JS
  - Dynamic style & positioning
- Events and Event Handling
  - Handling events from body elements
  - Handling events from button elements
  - Handling events from text and password elements
    - Pattern Matching with Regular Expression
- The navigator Object

Event and Event Handling

- An *event* is a notification that something specific has occurred, either with the browser or an action of the browser user
  - Event is an object
- An *event handler* is a script that is implicitly executed in response to the appearance of an event
- The process of connecting an event handler to an event is called *registration*
- Avoid using document.write in an event handler, because the output may go on top of the display

simpleclick.html

Event and Event Handling

Event	*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

Because events are JS objects, their names are case sensitive.

The names of all event objects have only lowercase letters.

For example, click is an event but Click is not.

Event attributes and their tags

Attribute	Tag	Description
onblur	<a>	The link loses the input focus.
	<button>	The button loses the input focus.
	<input>	The input element loses the input focus.
	<textarea>	The text area loses the input focus.
	<select>	The selection element loses the input focus.
onchange	<input>	The input element is changed and loses the input focus.

Attribute	Tag	Description
	<textarea>	The text area is changed and loses the input focus.
	<select>	The selection element is changed and loses the input focus.
onclick	<a>	The user clicks on the link.
	<input>	The input element is clicked.
onfocus	<a>	The link acquires the input focus.
	<input>	The input element receives the input focus.
	<textarea>	A text area receives the input focus.
	<select>	A selection element receives the input focus.
onload	<body>	The document is finished loading.
onmousedown	Most elements	The user clicks the left mouse button.
onmousemove	Most elements	The user moves the mouse cursor within the element.
onmouseout	Most elements	The mouse cursor is moved away from being over the element.
onmouseover	Most elements	The mouse cursor is moved over the element.
onmouseup	Most elements	The left mouse button is unclicked.
onselect	<input>	The mouse cursor is moved over the element.
	<textarea>	The text area is selected within the text area.
onsubmit	<form>	The Submit button is pressed.
onunload	<body>	The user exits the document.

Event and Event Handling

- The same attribute can appear in several different tags
  - e.g., The onclick attribute can be in <a> and <input>

  onclick.html

- *A text element gets focus in three ways:*
  - When the user puts the mouse cursor over it and presses the left button
  - When the user tabs to the element
  - By executing the focus method

  onfocusblur.html

Event and Event Handling

- *Event handlers can be registered in two ways*:
  - By assigning the event handler script to an event tag attribute

    <input type="button" name="myButton" onclick= "alert('Mouse click!');"
    - By referring to a JS function (the literal string value of the attribute is the call to the function:

      <input type="button" name="myButton" onclick= "myHandler();"
  - By the assignment to the associated event property on the object
    - Document.myButton.onclick=myHandler;

onerror

- Onerror onerror.html

DHTML

- JS Execution Environment
- The Document Object Model (DOM)
- Element Access in JS
  - Dynamic style & positioning
- Events and Event Handling
  - Handling events from body elements
  - Handling events from button elements
  - Handling events from text and password elements
    - Pattern Matching with Regular Expression
- The navigator Object

Handling Event from Body Elements

- Example: the load event - triggered when the loading of a document is completed

load.html onload.html

Handling Events from Button Elements

- Plain Buttons – use the onclick property
- *Radio buttons*
  - If the handler is registered in the markup, the particular button that was clicked can be sent to the handler as a parameter
  - e.g., if planeChoice is the name of the handler and the value of a button is 172, use

    onclick = "planeChoice(172)"

  radio_click.html
- There is another way of choosing the clicked button

Handling Events from Button Elements

- If the handler is registered by assigning it to a property of the JavaScript objects associated with the HTML elements. As in:

  var dom = document.getElementById("myForm")

  dom.elements[0].onclick = planeChoice;
  - This registration must follow both the handler function and the XHTML form
    - If this is done for a radio button group, each element of the array must be assigned

- In this case, the checked property of a radio button object is used to determine whether a button is clicked
  - If the name of the buttons is planeButton

```
var dom = document.getElementById("myForm");
for (var index = 0; index < dom.planeButton.length; index++) {
        if (dom.planeButton[index].checked) {
                plane = dom.planeButton[index].value;
                break; }
}
```

radio_click2.html

Handling Events from Button Elements

- The disadvantage of specifying handlers by assigning them to event properties is that there is no way to use parameters
- The advantages of specifying handlers by assigning them to event properties are:
  - It is good to keep HTML and JavaScript separate
  - The handler could be changed during use

DHTML

- JS Execution Environment
- The Document Object Model (DOM)
- Element Access in JS
  - Dynamic style & positioning
- Events and Event Handling
  - Handling events from body elements
  - Handling events from button elements
  - Handling events from text and password elements
    - Pattern matching with Regular Expression
- The navigator Object

Handling Events from Textbox and Password Elements

- The Focus Event:
  - Can be used to detect illicit changes to a text box by blurring the element every time the element acquires focus

nochange.html

Handling Events from Textbox and Password Elements

- Checking Form Input
  - A good use of JavaScript, because it finds errors in form input before it is sent to the server for processing
    - So, it saves both:
      - Server time, and
      - Internet time
- *Things that must be done*:
  - Detect the error and produce an alert message
  - Put the element in focus (the focus function)
    - Document.getElementById("phone").focus();
  - Select (highlight) the element (the select function)
    - Document.getElementById("phone").select();
- If an event handler returns false
  - It tells the browser not to perform any default factions

Comparing password

- The form just has two password input boxes to get the passwords and Reset and Submit buttons
  - The event handler is triggered by the Submit button
- *Handler actions*: pswd_chk.html
  - If no password has been typed in the first box, focus on that box and return false
  - If the two passwords are not the same, focus and select the first box and return false if they are the same, return true

Event Bubbling

- Crucial part of the event model
- Process whereby events fired in child elements "bubble" up to their parent elements

bubbling.html

Tracking the Mouse with Event onmousemove

- onmousemove
- Fires repeatedly when the user moves the mouse over the Web page
- Gives position of the mouse

onmousemove.html

- Two more events fired by mouse movements
  - onmouseover
    - Mouse cursor moves over element
  - Onmouseout
    - Mouse cursor leaves element

onmouseoverout.html

Checking form input

- The focus function puts the element in focus, which puts the cursor in the element
  - document.getElementById("phone").focus();
- The select function highlights the text in the element
- To keep the form active after the event handler is finished, the handler must return false

  Problems:
  - With IE6, focus and select only work if the handler is registered by assigning it to the element event property
  - With NS7, select works, but focus does not

Checking the format of a name and phone number

- The event handler will be triggered by the change event of the text boxes for the name and phone number
- If an error is found in either, an alert message is produced and both focus and select are called on the text box element

validator.html

DHTML

- JS Execution Environment
- The Document Object Model (DOM)
- Element Access in JS
  - Dynamic style & positioning
- Events and Event Handling
  - Handling events from body elements
  - Handling events from button elements
  - Handling events from text and password elements
    - Pattern matching with regular expression
- The navigator Object

Pattern Matching using Regular Expression (RE)

- JavaScript provides two ways to do pattern matching:
  - Using RegExp objects: iRegExp.test(string) true or false
  - Using methods on String objects:
    - istring.search(pattern) index of the match or -1
    - istring.match(pattern) array of match or null
- search(pattern) (String Object) search.html
  - Returns the position in the object string of the pattern (position is relative to zero); returns
    -1 if it fails

```
var str = "Gluckenheimer";
  var position = str.search(/n/);
  /* position is now 6 */
```

- test(pattern) (RegExp Object)
  - Tests if the given string matches the Regexp, and returns true if matching, false if not.

```
var regObj = /n/;
if(regObj.test("Gluckenheimer"))
  /* test whether the given string contain the pattern */
```

Pattern Matching

- *Simple patterns*
  - Two categories of characters in patterns:
    - normal characters (match themselves)
    - metacharacters (can have special meanings in patterns--do not match themselves)

      \ | ( ) [ ] { } ^ $ * + ? .

    - A metacharacter is treated as a normal character if it is backslashed
    - period is a special metacharacter - it matches any character except newline
- *Character classes*
  - Put a sequence of characters in brackets, and it defines a set of characters, any one of which matches
    - [abcd]
  - Dashes can be used to specify spans of characters in a class
    - [a-z]
  - A caret at the left end of a class definition means the opposite
    - [^0-9]

Character Classes

- **Character class abbreviations**

Abbr.	*Equiv. Pattern*	*Matches*	
\d	[0-9]	a digit	/\d\.\d\d/
\D	[^0-9]	not a digit	/\D\d\D/
\w	[A-Za-z_0-9]	a word character	/\w\w\w/
\W	[^A-Za-z_0-9]	not a word character	
\s	[\r\t\n\f]	a whitespace character	
\S	[^ \r\t\n\f]	not a whitespace character	

Quantifiers

- Quantifiers in braces quantifier.html

  Quantifier Meaning /\d{3}-\d{4}/
  - {n} exactly n repetitions /ab{4}/
  - {m,} at least m repetitions /ab{4,}/
  - {m, n} at least m but not more than n repetitions /ab{4,6}/
- Other quantifiers (just abbreviations for the most commonly used quantifiers)
  - * means zero or more repetitions
    - e.g., \d* means zero or more digits /ab*/
  - + means one or more repetitions
    - e.g., \d+ means one or more digits /ab+/
  - ? Means zero or one
    - e.g., \d? means zero or one digit/ab?/

Anchors

- The pattern can be forced to match only at the left end with ^; at the end with $
  - e.g., /^Lee/
  - matches "Lee Ann" but not "Mary Lee Ann"
  - /Lee Ann$/
  - matches "Mary Lee Ann", but not "Mary Lee Ann is nice"
- The anchor operators (^ and $) do not match characters in the string--they match positions, at the beginning or end

Pattern Modifiers

- The *i* modifier tells the matcher to ignore the case of letters
  - /oak/i matches "OAK" and "Oak" and …
- The *x* modifier tells the matcher to ignore whitespace in the pattern (allows comments in patterns)

```
/\d+                    # the street number
\s                      # the space before the street name
[A-Z][a-z]+             # the street name
/x
```

Equals to /\d+\s[A-Z][a-z]+/

- The *g* modifier tells whether the matching/replacement is global or for the first hit only

match

- String.match(pattern) The most general pattern-matching method
  - Returns an array of results of the pattern matching operation
- With the g modifier, it returns an array of the substrings that matched
  - var str = "My 3 kings beat your 2 aces";
  - var matches = str.match(/[ab]/);
    - matches is set to ["b", "a", "a"]
- Without the g modifier, first element of the returned array has the matched substring
  - var str = "My 3 kings beat your 2 aces";
  - var matches = str.match(/[ab]/);
    - matches is set to ["b"]

match.html

match

- Parenthesized subexpressions
  - a pattern can consist of parenthesized subexpressions
    - Var str="I have 400 but I need 500";
    - Var matches=str.match(/(\d+)([^\d+])(\d+)/);
    - Matches=["400 but I need 500", "400", "but I need", "500"];

The Navigator Object

- Indicates which browser is being used
- Two useful properties
  - The appName property has the browser's name
  - The appVersion property has the version #
- Microsoft has chosen to set the appVersion of IE6 to 4 (?)
- Netscape has chosen to set the appVersion of NS7 to 5.0 (?)

navigate.html

DHTML

- JS Execution Environment
- The Document Object Model (DOM)
- Element Access in JS
  - Dynamic style & positioning
- Events and Event Handling
  - Handling events from body elements
  - Handling events from button elements
  - Handling events from text and password elements
    - Pattern Matching with Regular Expression
- The navigator Object