

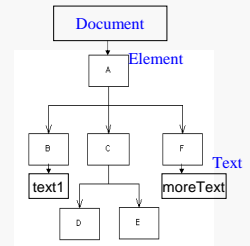
Document Object Model (DOM)

- W3C Standard
 - Implemented differently by every browser
- Interface between document displayed by browser and application programs
 - Object oriented
- Platform-neutral and language-neutral collection of interfaces
 - Can program with a variety of languages (C, Java, VB)
- Documents have treelike structures
 - One interface for every XML node type
 - Element, attribute, text, PI, comment
- Create documents, move around document structure (parse), and change, add, or delete elements.

1

DOM Representation of XML

```
<A>
  <B>text1</B>
  <C>
    <D>child of C</D>
    <E>another child of C</E>
  </C>
  <F>moreText</F>
</A>
```



2

How the DOM is Defined

- The DOM is defined using Interface Definition Language (IDL)
 - Use for inter-process communication
 - API that allows a client X to call services provided by server Y
 - Define object, functions, and function parameters
 - IDL was originally defined for use in CORBA
 - Java RMI and Microsoft DCOM can also use IDL

3

Example IDL Definition

```
interface Element : Node {
    wstring  getTagName();
    wstring  getAttribute(in wstring name);
    void     removeAttribute(in wstring name);
    Attribute getAttributeNode(in wstring name);
    void     normalize();
    .....
};
```

4

JavaScript Language Binding

- Correspondence between constructs in the language and elements in the DOM
 - Elements are represented by objects
 - Attributes are represented by properties

5

Example JavaScript Binding

```
function Element() {
    // properties and methods of the Node object
    .....
    // properties and methods of Element object
    var tagName ;
    getAttribute = elementGetAttribute(name);
    setAttribute = elementSetAttribute(name, value);
    removeAttribute = elementRemoveAttribute(name);
    getAttributeNode = elementGetAttributeNode(name);
    setAttributeNode = elementSetAttributeNode(new Attr);
    .....
}
```

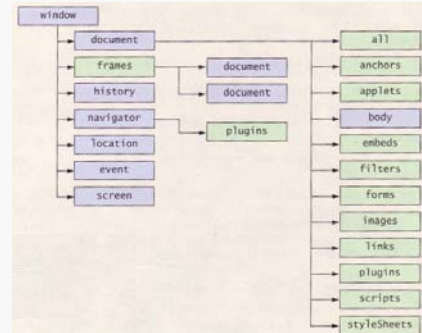
6

DOM Parts

- Core
 - Minimal set of objects and interfaces for accessing and manipulating document objects (mainly XML oriented)
- DHTML
 - Extends the Core API to describe objects and methods specific to HTML documents
 - Convenience methods and properties that more appropriate to script writers.
 - These enhancements are not applicable to general XML documents

7

DHTML Object Model



8

Key Interfaces

- Document
- Element
- Event

9

Document

- The central interface is Document
- Create new elements, attributes and text nodes
- Access existing elements
 - getElementByName(stringName)
 - getElementById(stringId)

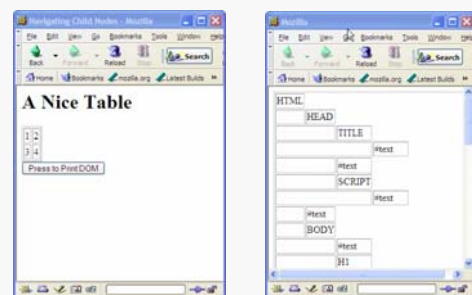
10

Element

- All tags in document inherit from Element
- Navigate document
- Change document tree structure

11

Example: DOM Tree



12

Event Driven Programming

Most, if not all, GUI systems and toolkits are designed to be event driven, meaning that the main flow of your program is not sequential from beginning to end. If you've never done GUI programming, this is one of the trickiest paradigm shifts.

— Robin Dunn, speaking on GUI programming at OSCON2004

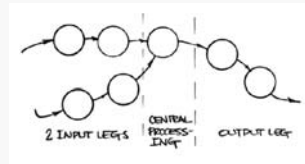
Hollywood Principle: "Don't call us; we'll call you." ... You implement the interfaces, you get registered. You get called when the time is right. This requires a distinctly different way of thinking to that which is taught in introductory programming where the student dictates the flow of control.

— Dafydd Rees, <http://c2.com/cgi/wiki?HollywoodPrinciple>

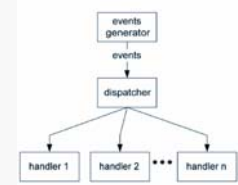
13

Structured vs. Event Driven

Structured Program



Event Driven Architecture



14

Event-Driven Execution

- JavaScript programs are typically event-driven.
- Execution is triggered by various *events* that occur on the Web page, usually as a result of something the user does.
 - *onClick*, *onDbClick*, *onKeyDown*, *onLoad*, *onMouseOver*, *onSubmit*, *onResize*, ...
- Events are associated with the various objects that make up a Web page, for example, an *onClick* event might be associated with clicking a radio button element on a form.

15

Partial List of Events

- Clipboard
 - *oncopy*, *oncut*, *onpaste*
- Keyboard
 - *onkeydown*, *onkeyup*, *onkeypress*
- Mouse
 - *onmousedown*, *onmouseup*, *onmousemove*
- Other
 - *onfocus*, *onblur*,

16

Associating Events with Elements

- In the HTML
 - As value of attributes
- ```

```
- In a script
    - Explicit reference to object's event handler

```
document.onmouseover = functionFoo;
```

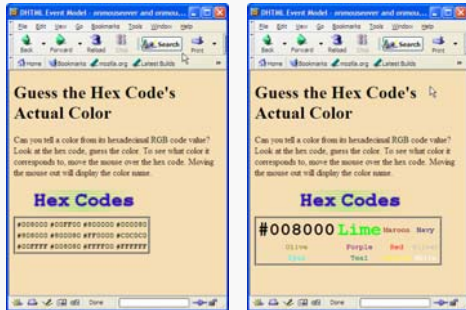
17

## Event Bubbling

- Event "fired" by child elements "bubble" up to their parent elements.
- Event delivery order
  - First to element that fired event
  - Then to parent
- To cancel bubbling, set event property `event.cancelBubble = true`

18

## Example: Event Bubbling



19

## Referencing Objects

```
<body>
 <form name="myForm">
 <input type="button" name="turnItOn" id="b1">
 </form>
</body>
```

- Location in tree
  - document.forms[0].element[0]
- Element name
  - document.getElementsByName(turnItOn)
  - document.myForm.turnItOn
- ID
  - document.getElementById("b1")

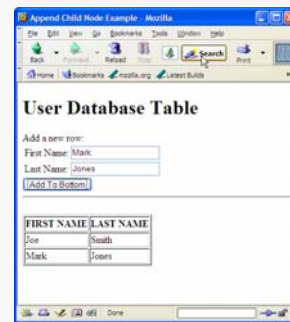
20

## Modifying DOM

- Creating new nodes
  - Document
    - createElement(tag) // Create an element of type tag
    - createTextNode(string) // Creates text node with string
  - Element
    - appendChild(N) // Add the N to the end of child list
    - insertBefore(N,E) // Insert N in child list before E
    - cloneNode(deep) // Copy node. If deep=true copy all descendants
- Removing nodes
  - Node
    - removeChild(N) // Removes N from child list

21

## Example: Adding Table Rows



22

## Changing Style Attributes

- CSS1 and CSS-P (CSS - Positioning) are scriptable from JavaScript
  - allows HTML elements to float around and grow and shrink.

23

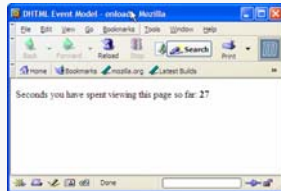
## Example: Move Image on Screen



24

## onload & timers

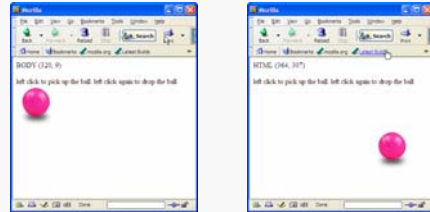
- onload
  - Fires when element (an all children) finish loading
  - Used in the <body> to execute script after page has been rendered
- Example: Count how many seconds have passed since page finish rendering



25

## Tracking Mouse Movements

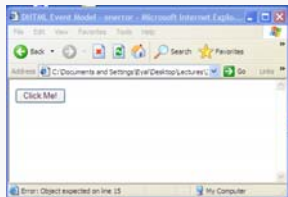
- Track mouse position on screen
- Drag and drop ball on click
- Events onmousemove and onclick



26

## Error Handling

- Specify function to handle script errors
  - window.onerror = errorHandler



27

## Error Handling

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C/DTD XHTML 1.0 Transitional/EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">

<html>
<head>
<title>DHTML Event Model - onerror</title>
<script type = "text/javascript">
// Set error handler
window.onerror = errorHandler;

function doThis() {
alert("hi"); // alert misspelled, creates an error
}

// Error handler gets error type, file URL, line number
function errorHandler(errType, errURL, errLineNum) {
// Writes to the status bar
window.status = "Oops, Error: " + errType + " on line " + errLineNum;

// Returning a value of true cancels the browser's reaction.
return true;
}
</script>
</head>
<body>
<input id = "mybutton" type = "button" value = "Click Me!" onclick = "doThis()" />
</body>
</html>
```

28