Dynamic HTML

Code: dyn-html

Author and version

- Daniel K. Schneider
- Email: Daniel.Schneider@tecfa.unige.ch

- Version: 0.6 (modified 18/4/07 by DKS)

Prerequisites

- HTML
- CSS
- Some JavaScript

Objectives

- Introduction to modern Dynamic HTML (DOM/CSS/JavaScript)
- Reuse of some simple JavaScript functions
- Object Referencing and modification of contents
- Dynamic Styles and Dynamic Positioning
- The concept of "tree walking"

Disclaimer

- There may be typos (sorry) and mistakes (sorry again)
- Please also consult a textbook ! (However, the Deitel book is not uptodate for this module!)

Acknowledgement

These slides have been prepared with the help of

- The Textbook and (older) associated slides
- The Gecko DOM Reference and JavaScript reference
  url: **http://developer.mozilla.org/en/docs/Gecko_DOM_Reference**
  url: **http://developer.mozilla.org/en/docs/JavaScript**
- W3C specifications
  url: **http://www.w3.org/TR/DOM-Level-2-Core/**
  url: **http://www.w3.org/TR/DOM-Level-2-HTML/**
  url: **http://www.w3.org/TR/DOM-Level-2-Events/**
  url: **http://www.w3.org/TR/DOM-Level-2-Style/**

Contents

1. Prerequisites

1.1 HTML and/or XHTML

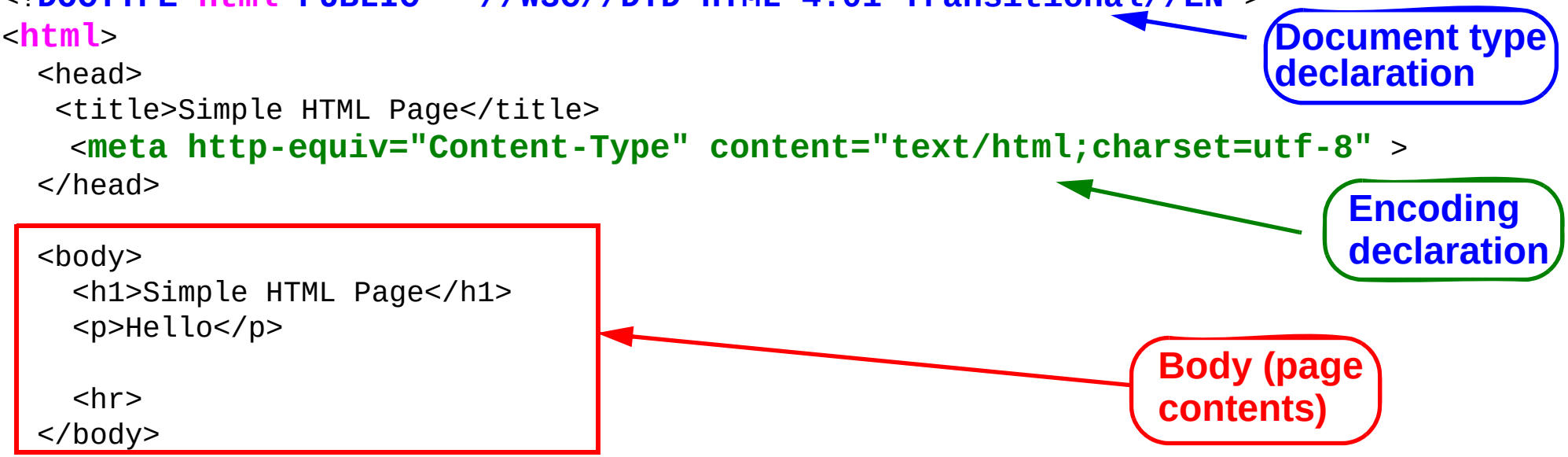
- DOM works with all XHTML versions and all more recent HTML versions (> 4).

Example 1-1: HTML Page (preq-html-page.html)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
   <title>Simple HTML Page</title>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" >
  </head>

  <body>
    <h1>Simple HTML Page</h1>
    <p>Hello</p>

    <hr>
  </body>
</html>
```



Notes:

- Navigators are very forgiving, you may omit the doctype, the encoding declarations, or even the head. But your page will not validate (which is not acceptable in most professional settings). Don't worry just for this class ...

Example 1-2: XHTML Page

- All most recent HTML versions are XHTML.
- XHTML is an XML application (i.e. it uses the XML formalism).
- XHTML is more powerful than HTML, but more strict and somewhat less forgiving ....
- All tags are lower case, all tags must be closed (think "boxes" within "boxes") !!!

```
<?xml version = "1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns = "http://www.w3.org/1999/xhtml">
 <head>
   <title>Object Model</title>
  <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
 </head>
<body>
   <h1>Welcome to our Web page!</h1>
   <p>Enjoy ! </p>
</body>
</html>
```

XML and Document type declaration

Namespace declaration

Encoding declaration

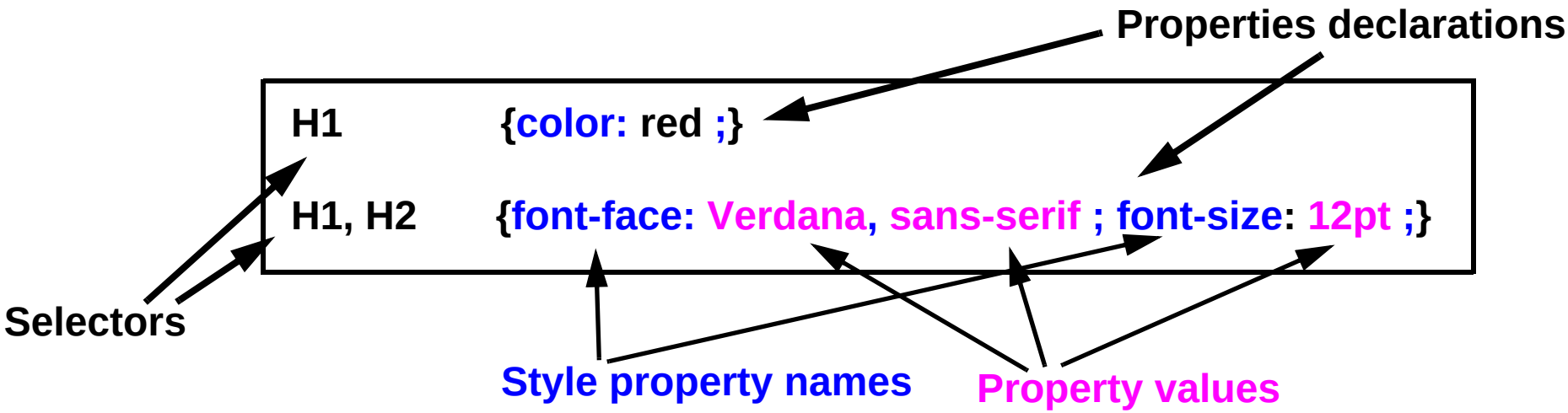
Body (page contents)

1.2 CSS

- Style sheet = set of ***rules*** that describe how to render XML or (X)HTML elements
- Each rule has two parts:
  - The ***selector***: defines to which elements a rule applies
  - The ***declaration***: defines rendering, i.e. defines values for style properties

Example 1-3: A simple HTML example

```
H1 { color: red }
P  { font-face: Verdana, sans-serif ; font-size: 12pt}
H1, H2, H3 { color : blue }
H1.ChapterTOC, H2.PeriodeTOC, H2.ExerciceTOC, H2.SectionTOC  {
        display: block;text-indent: 30pt;
        text-align: left; font-size: 14.000000pt;
        font-weight: Bold; font-family: "Times";
    }
```



1.3 JavaScript

- JavaScript is a programming language that runs inside your navigator
- JavaScript functions are usually defined in the head and within "**script**" tags
- JavaScript functions can be called in various ways from the body of the document
  - JavaScript instructions can be inserted within "**script**" tags (as in the head)
  - JavaScript also can be found within "inline" event handler definitions

```
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Object Model</title>
      <script type = "text/javascript">
         function start()
         {
            var p = document.getElementById("pText");
            alert( "Page contents : " + p.innerHTML );
            p.innerHTML = "Thanks for coming.";
            alert( "Page contents changed to : " + p.innerHTML );
            p.innerHTML = "Cool, isn't it ?";
         }
      </script>
   </head>

   <body onload = "start()">
      <p id = "pText">Welcome to our Web page!</p>
   </body>
</html>
```

Definition of the event handling function "start"

Inline event and call of event handling function

- Note: This is just a FYI-type of slide about JavaScript ....

Note on standardization

- The JavaScript language itself is fairly cross-browser compatible
- However, event handling and DOM is not fully standard (Microsoft DHTML is not a standard and even IE 7 is still quite far from implementing the DOM level 2.)
- Inline events management however is fairly cross-browser compatible
- More advanced stuff is less

Note on JavaScript and XHTML

- Valid XHTML requires JavaScript to be inserted within a commented CDATA section

```
<script language="javascript">
// <![CDATA[
   alert ("hello I am alerting you from a valid XHTML Page");
// ]]>
</script>
```

- An other, better solution is to put the javascript code into an external file. E.g.

```
<script type="text/javascript" src="external.js"></script>
```

2. Introduction

2.1 DHTML and DOM - some definitions

DHTML ?

- Dynamic HTML (DHTML) does not exist as a standard
- DHTML = HTML + CSS + Javascript + Javascript DOM bindings
  - So the difficult part is understanding DOM and JavaScript programming ...

DOM = Document Object Model

A document object model (DOM) is an application programming interface (API) for representing a document (such as an HTML document) and accessing and manipulating the various elements (such as HTML tags and strings of text) that make up that document.

- Allows Web authors to control and dynamically modify the presentation of their pages
- Gives them access to all the elements on their pages
  - e.g. Headers, paragraphs, forms, styles
- All information is represented in an object hierarchy and from which scripts can retrieve and modify properties and attributes
- Non standard extensions give information about navigator and windows ...

A short history

- Only recently, DHTML became standardized as DOM+Javascript.
- Before, DHTML was a nightmare for web developers, i.e. there was total incompatibility between browser brands and lesser incompatibilities between OS and navigator versions.

2.2 History of DOM and DOM levels

- DOM is defined by levels and several specifications per level

Level 0

- Non-standardized things, like Microsoft DHTML or Netscape DOM 4.x, etc.
- Methods that concern window and navigator are still being used and implemented in most browsers ...

Level 1 (octobre 1998)

- DOM Level 1 Core: defines how HTML and XML are represented in the machine and can be navigated
- DOM Level 1 HTML: HTML-specific extensions

Level 2 (2001)

- 6 specifications, adding XML namespace support, filtered views and events.
- DOM Level 2 Core Specification (extension of DOM Level 1 Core)
- DOM Level 2 Views Specification
- DOM Level 2 Events Specification: standardizes management of user generated events
- DOM Level 2 Style Specification: CSS
- DOM Level 2 Traversal and Range Specification
- DOM Level 2 HTML Specification: extensions for HTML

Level 3: contains 6 specifications

- DOM Level 3 Core
- DOM Level 3 Load and Save
- DOM Level 3 XPath
- DOM Level 3 Views and Formatting
- DOM Level 3 Requirements
- DOM Level 3 Validation

Other W3C languages that define DOM extensions

- MathML
- SVG
- SMIL

Non W3C languages also may support DOM

- X3D, ...

2.3 Implementations

- Most modern browsers (2005 +) implement most of DOM level 1 specs.
- Most modern browsers implement at least some of DOM level 2.
- Some proprietary DOM 0 still is needed (e.g. navigator information). These are usually cross-browser compatible.
- Other "old style" DHTML should be avoided if possible (e.g. "innnerHTML")
- Other "old style" DHTML should be avoided in every case (e.g. the MS DHTML model).

Navigator comparison tables:

url: **Comparison of layout engines (DOM)** **(wikipedia)**

url: **http://www.quirksmode.org/dom/w3c_html.html** **(Quirksmode)**

url: **http://www.quirksmode.org/dom/w3c_core.html**

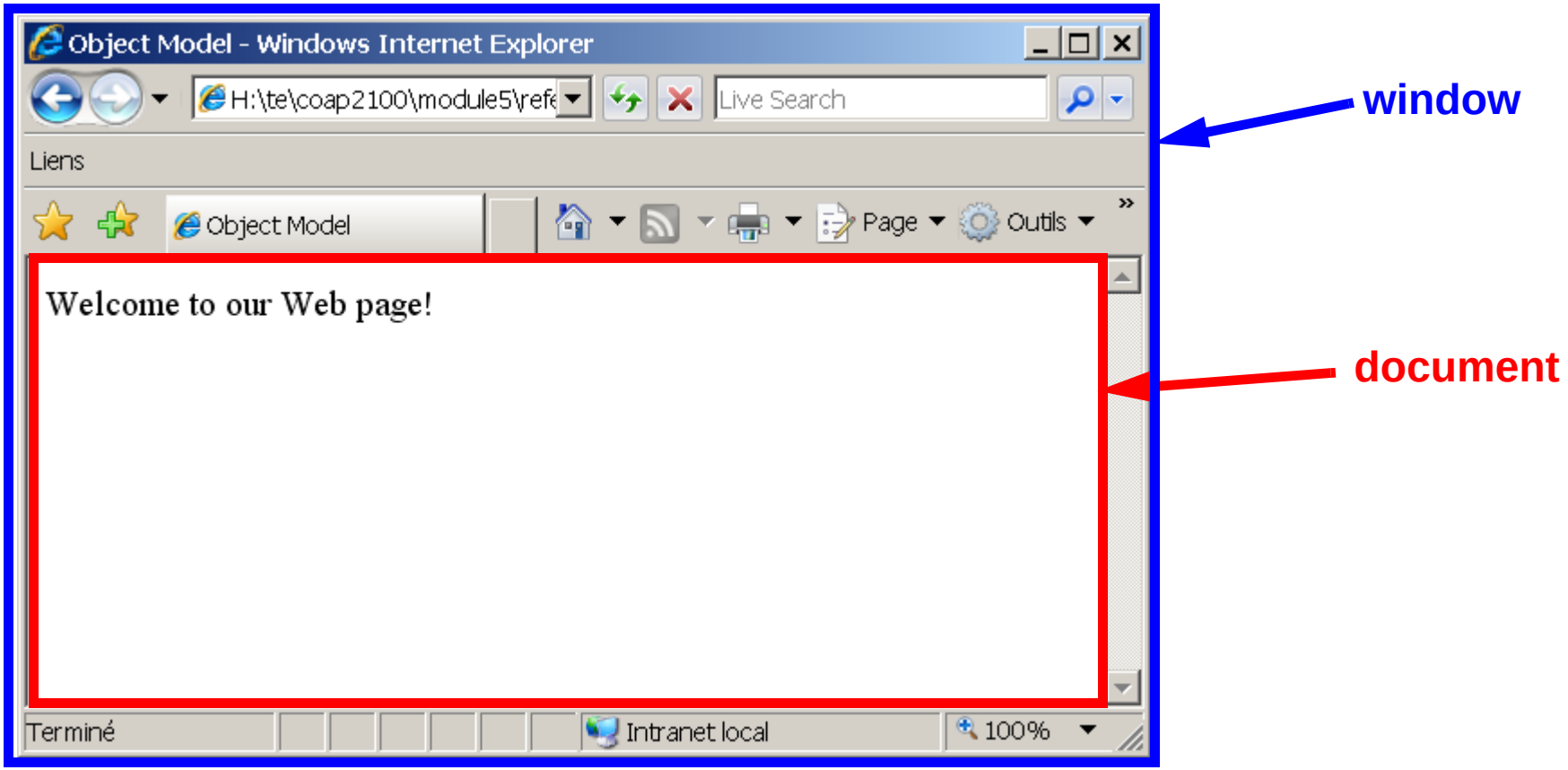
Executive summary regarding DOM/DHTML variants:

- Plan for the future (use standards whenever possible)
- Make sure to test your code with browsers your target population will use

2.4 DOM objects Overview

There are a few system variables that point to objects which allow you to access the current document, the current window and the navigator. The most important ones are:

- **document**: to access and manipulate contents of a page
- **window**: to access information about the window and to create new windows
- ***navigator***: to access information about the navigator (browser).



3. Referencing and changing objects

The principle

In order to do some DHTML your script must be able to refer to HTML elements and then change something. So you need to know 4 things:

- how to refer to a specific object within a page (also called "document")
- how to change the contents of an HTML element
- how to define a JavaScript (JS) function that will do this.
- how to launch (call) this JS function

Summary of the most simple solution:

- Add **`id="....."`** attributes to you HTML elements you later want to use
- Launch the JavaScript function when the page will load

```
<body onload = "start()">
```

means "when the page loads into the browser, launch the start() function

- Write a little function that does the following
  - Use the **`document.getElementById("your_id")`** method to refer to an object and use the **`innerHTML`** method to change contents of an objects

```
var my_para = document.getElementById("p1");
```

means: find the HTML with id=p1 in the page and put this object in the my_para variable

```
alert( "Paragraph contents : " + my_para.innerHTML );
```

means: create a little popup window that will display the "inside" of the referenced HTML element.

Example 3-1: Simple object referencing

url: **reference0.html**

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

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Object Model</title>

      <script type = "text/javascript">
         // Definition of a start function
       function start()
         {
            var my_para = document.getElementById("p1");
            alert( "Paragraph contents : " + my_para.innerHTML );
         } // end of start function
      </script>

   </head>

   <body onload = "start()">
      <p id = "p1">Welcome to our Web page!</p>
   </body>
</html>
```

Example 3-2: Simple object referencing and change

url: **reference.html**

- This example shows how to change the contents of an HTML element
- Once we have a reference to the HTML element ("para") in this case we can change it

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

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head><title>Object Model</title>
      <script type = "text/javascript">
         function start()
         {
            var para = document.getElementById("pText");
            alert( "Page contents : " + para.innerHTML );
            para.innerHTML = "Thanks for coming.";
            alert( "Page contents changed to : " + para.innerHTML );
            para.innerHTML = "Cool, isn't it ?";
         }
      </script>
   </head>
   <body onload = "start()">
      <p id = "pText">Welcome to our Web page!</p>
   </body>
</html>
```

4. Style

- Warning (!!!) "CSS inside the DOM" has slightly different names than CSS properties
- For a list see e.g.

  url: **http://developer.mozilla.org/en/docs/DOM:CSS**

  url: **http://www.w3.org/TR/DOM-Level-2-Style/css.html#CSS-CSS2Properties**
- But most of the time it's easy to guess the DOM name:
  - remove the "-"
  - capitalize words (except the first one)
  - example: CSS property "background-color" becomes DOM/CSS "backgroundColor".
- Some examples:

DOM CSS	CSS property
background	background
backgroundColor	background-color
borderColor	border-color
fontFamily	font-family
fontSize	font-size
marginBottom	margin-bottom
marginLeft	margin-left

- In order to change a style property of an element, use the syntax:

```
object.style.xxx = "yyy";
```

e.g.

```
para.style.fontFamily="Helvetica"
```

Example 4-1: Changing style of background with a prompt

url: **dynamicstyle.html**

```
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Object Model</title>

      <script type = "text/javascript">
         <!--
         function start()
         {
            var inputColor = prompt(
               "Enter a color name for the " +
               "background of this page", "" );
            document.body.style.backgroundColor = inputColor;
         }
         // -->
      </script>
   </head>

   <body onload = "start()">
      <p>Welcome to our Web site!</p>
   </body>
</html>
```

Example 4-2: Change style with a push button

url: **change_style1.html**

```
<script type="text/javascript">
    function changeText() {
    var p = document.getElementById("pid");
     p.style.color = "blue"
     p.style.fontSize = "18pt"
    }
    </script>
.....
<p id="pid"
    onclick="window.location.href = 'http://www.cnn.com/';">News Link</p>
<form>
  <p>
    <input value="Change style" type="button" onclick="changeText();">
 </p>
</form>
```

Example 4-3: Change style forth and back with a push button

url: **change_style2.html**

(code not shown here, look at the file please)

5. HTML Window and Navigator

- The Window interface gives information about the window
- It also includes a reference to the navigator
- (Based on the AbstractView interface of DOM Level 2 Views Specification)

A few examples

window.document

- gives the document object of a window (useful if you manipulate more than one window)

```
doc= window.document;
// print the title
window.dump(doc.title);
```

window.navigator

- gives the navigator object

```
nav = window.navigator;
```

- you then can extract information from the navigator, e.g.

```
alert ("Language of your browser: " + nav.language);
```

- Navigator information is important for writing hairy cross-browser scripts. Here are a few properties of interest (there are more):

```
navigator.appName
navigator.appVersion
navigator.userAgent
navigator.appCodeName
```

- On the Web, you can find good browser detection scripts, e.g.

  url: **http://www.webreference.com/tools/browser/javascript.html**

window.history

```
h = window.history;
alert ("You have " + h.items + " item in your navigation history");
```

window.status

- can read or change the "status bar" (at bottom of window)
  `window.status = "cool website";`
- E.g. following code shows how to trigger status bar change when user moves the mouse over a link (by default most browsers will not allow this).
  `<a href="http://yoursite.org/"> onmouseover="status='Consult this'; return true;">Your site</a>`

window.print()

- Sends content of window to printer
  `window.print();`

Example 5-1: Information about the navigator

url: **navigator-props.html**

```
<html xmlns = "http://www.w3.org/1999/xhtml">
 <head>
  <title>The navigator Object</title>
  <script type = "text/javascript">
    <![CDATA[
    function start () {
     var output1 = "Hello, your " + navigator.appName + " navigator has the following
properties <br/>";
      document.writeln(output1);

      var output2 = "<ul>";
      output2 += "<li>Navigator Application Name: " + navigator.appName + "</li>";
      output2 += "<li>Version = " + navigator.appVersion + "</li>";
      output2 += "<li>User agent = " + navigator.userAgent + "</li>";
      output2 += "<li>Code Name = " + navigator.appCodeName + "</li>";
      output2 += "</ul>";

      document.writeln(output2);

      if (navigator.javaEnabled()) document.write("<li>Java works (enabled)");
      else document.write("<li>Java doesn't work");
    }
    ]]>
    </script>
  </head>
  <body onload = "start()"> </body>
</html>
```

Example 5-2: Redirect to page according to navigator

- File: navigator.html (this script is a bit too simple for real world use)

```
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <title>The navigator Object</title>
    <script type = "text/javascript">
     function start() {
     alert ("Your Navigator is " + navigator.appName + ", Version " +
                                   navigator.appVersion.substring() );
      if ( navigator.appName == "Microsoft Internet Explorer" ) {
         if ( navigator.appVersion.substring( 1, 0 ) >= "4" )
            document.location = "newIEversion.html";
         else document.location = "oldIEversion.html";
         }
      else
      document.location = "NSversion.html";
      }
    </script>
  </head>
    <body onload = "start()">
    <p>Redirecting your browser to the appropriate page,
      please wait...</p>
  </body> </html>
```

url: **http://www.webreference.com/tools/browser/javascript.html**

(rather use a professional script like this)

6. Collections and creation of elements

- To understand coode in this chapter you need some training in programming (which we don't provide here)
- Therefore, just try to grasp some very abstract principles given in class and try to copy/paste some code with only very minor modifications ....

6.1 Collections

- Instead of dealing with just one single object, you may want to write a script that can deal with several objects at the same time. E.g:
  - all objects that have the same "name" attribute (only few HTML elements can have a name attribute)
  - all elements of list
  - all titles (h2) of a page

Example 6-1: Emphasize child elements

- This code will hightlight contents of **`li`** elements within an **`ol`** or **`ul`** HTML element that has **`id="important"`**

  url: **emphasize-collection.html**

```
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Collections</title>

      <script type = "text/javascript">
         function doit()
         {
            var list = document.getElementById("important");
```

```
            var list_els = list.getElementsByTagName("li");
            // deal with each of the subelements
            for (var i = 0; i < list_els.length; i++) {
                // if (list_els[i].nodeType == 1)
                list_els[i].style.color = "red";
             }
        }
    </script>
  </head>
  <body>
   <form>
    <p><input value="Highlight important things" type="button" onclick="doit();">
    </p>
   </form>

   <p>Welcome to our <strong>Web</strong> page!</p>
   <div></div>
   <p>Here we have list of important things:</p>
   <ul id="important">
    <li>This is important</li>
    <li>This is quite important</li>
    <li>This is also important</li>
   </ul>
   <p>Here we have list of less important things:</p>
   <ul>
    <li>This is less important</li>
    <li>This is not so important</li>
    <li>This ain't important either</li>
   </ul>
  </body>
</html>
```

6.2 Tree walking

- Tree walking means to look at an element (for starters) and then examine its children, the children of children etc.
- We start using "real" algorithms here which are not easy to understand. However, you may take this code and just change a few things in order to fit it to your purpose. Make sure to change only 1-2 things
- We also introduce here the concept of ***functions that take arguments***. Basically, we can call a function and give it information it has to deal with.

example 6-2 “Tree walking: Collect element names” [p. 27]

- This is fairly ***uninteresting code*** since it will only display HTML elements used in a document. Just copy/paste all the javascript code and the input button if you want to see it in action in one of your pages.

example 6-3 “Tree walking: Change color style” [p. 28]

- Copy the whole script section in the head plus the lines with the input buttons. Just view the source of the html page (tree-walking2.html). It's all in there for you to grab ...
- Decide which elements in your HTML text you wish to highlight and add a class attribute to each of these, e.g. something like:

```
<p class="important">
```

- Change the class attribute value from "important" to whatever you wish, but don't forget to change the argument when calling the do_document function, e.g. if the CSS class you wish to highlight is "**cool**":

```
<input type=button onClick="do_document('cool','red');" value="Highlight">
```

- You may add as many input buttons you like, but of course there should be a corresponding css class and a **color**.

Example 6-2: Tree walking: Collect element names

url: **tree-walking.html**

```
var node_list = "";
function do_document () {
      // call the mark_tags function with argument "body of the document"
      mark_tags(document.body);
      alert ("This text has the following elements: " + node_list);
    }

function mark_tags(node) {
    // Check if n is an Element Node
    if (node.nodeType == 1 /*Node.ELEMENT_NODE*/)  {
     // Append the node name to the list
     node_list += node.nodeName + " ";
     // Let's see if there are children
     if (node.hasChildNodes()) {
         // Now get all children of n
         var children = node.childNodes;
         // Loop through the children
         for(var i=0; i < children.length; i++) {
                 mark_tags(children[i]);      // Recurse on each one
         }}   } }
</script>
</head><body>
   <h1>Tree walking</h1>
    <input type=button onClick="do_document();" value="Show body node names">
```

Example 6-3: Tree walking: Change color style

url: **tree-walking2.html**

```
<script>

 function do_document (css_class,color) {
      mark_tags(document.body,css_class,color);
    }

 function mark_tags(node,css_class,color) {
   // Check if n is an Element Node
   if (node.nodeType == 1 /*Node.ELEMENT_NODE*/)  {
    // Highlight if needed
    if (node.getAttribute("class") == css_class)
       node.style.color = color;
    // Let's see if there are children
    if (node.hasChildNodes()) {
        // Now get all children of node
        var children = node.childNodes;
        // Loop through the children
        for(var i=0; i < children.length; i++) {
                // Recurse on each child
                mark_tags(children[i],css_class,color);
        } }}
   }
</script>
</head>
```

```
<body>
    <h1>Tree walking</h1>
    <input type=button onClick="do_document('important','red');"
           value="Highlight important stuff">
    <input type=button onClick="do_document('boring','grey');"
    value="Highlight boring stuff">
.....
   <p class="boring">Let's hope you enjoy this</p>
   <p class="important">If you want to reuse this code:</p>
.....
```

Example 6-4: Tree walking: Change any style

url: **tree-walking3.html**

- Students who wish to change other style properties may look at this code ...
- Implementation of a **`do_document(css_class,style_value,style_property)`** function.
- E.g.

  `do_document('boring','none','display')`
  would change the display property of elements with class="boring" to none (hide elements)

6.3 Creation of elements

- Creation of new HTML elements is somewhat ***hairy*** (unless you use the innerHTML method which is less standard and doesn't give you the same power)
- You have to create an element with the **`document.createElement`** "factory" method.
- Then you have to fill in the element, e.g. by creating a TextNode with **`document.createTextNode`**
- Then only you can insert it, e.g. append an element's children with **`heading.appendChild();`**

Example 6-5: Simple append example

url: **insert3.html**

```
<script type="text/JavaScript">
    function insert_things () {
      // heading will contain a new "h1" element (node)
      heading = document.createElement("h1");
      // heading_text will contain a new "text" node
      heading_text = document.createTextNode("I love pressing buttons !");
      // we can now ask the heading node to insert the heading_text
      heading.appendChild(heading_text);
      // Finally we can ask the body to insert this at the end.
      document.body.appendChild(heading);
      }
    </script></head>
<body>
<input type=button onClick="insert_things();" value="DoIt">
```

7. Animation

- Animation means changing some style properties (e.g. position, size, color)
- There are several sorts of animations, e.g.
  - automatic animations
  - animations that react to user input
- Warning (again). This section requires some programming skills. If you don't have these, do not worry. The goal here is to demo that more sophisticated animation can be done with a little bit of scripting. Learning how to script takes several weeks ....

Example 7-1: User-driven object moving

url: **move-object1.html**

- In the body, we use href attributes to call a moveitBy javascript function
- This function is called with 3 arguments: ***id***, ***move_right***, ***move_down***
  - "id" is the value of and id attribute
  - move_right and move_down are positive or negative numbers (pixels to shift).

```
<div id="image" style="position: relative; left: 0px; top: 0px;">
  <img src="tecfa-affiche.gif" alt="logo">
</div>
<ul>
<li><a href="javascript:moveitBy('image',20,0);">Move by 20px (right)</a></
li>
<li><a href="javascript:moveitBy('image',-20,0);">Move by 20px (left)</a></
li>
```

- The moveitBy function
  - firstly takes the first argument (an id) and retrieves the div object within which we placed the image
  - it then modifies "left" and "top" values of this div by adding or substracting 20.

```
function moveitBy(img, x, y){
  var obj          =  document.getElementById(img);
  obj.style.left   =  parseInt(obj.style.left)+x+"px"
  obj.style.top    =  parseInt(obj.style.top)+y+"px"
  }
```

- The function moevitTo will reposition an object at given left/top coordinates

```
function moveitTo(img, x, y){
  var obj          = document.getElementById(img);
  obj.style.left   = x+"px"
  obj.style.top    = y+"px"
 }
</script>
```

Example 7-2: Automatic font, size and position animation

url: **dynamicposition.html**

- also requires some programming skills to understand ....

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

<!-- Dynamic Positioning, originally from Deitel -->
<!-- adapted to more real DOM + less difficult code by Daniel K. Schneider -->

<html xmlns = "http://www.w3.org/1999/xhtml">
 <head>
  <title>Dynamic Positioning</title>

  <script type = "text/javascript">
   var speed = -5;
   var count = 10;
   var firstLine = "Text growing";
   var pText;

  function start() {
    pText = document.getElementById("pText");
    window.setInterval( "run()", 100 );
    }

  function run() {
```

```
    count += speed;

    if ( ( count % 200 ) == 0 ) {
      speed *= -1;
      pText.style.color = ( speed < 0 ) ? "red" : "blue" ;
      firstLine =   ( speed < 0 ) ? "Text shrinking" : "Text growing";
      }

    size = count / 3 ;
    pText.style.fontSize = size + "px";
    pText.style.left = count + "px";
    pText.innerHTML = firstLine + "<br /> Font size: " + count + "px";
     }
    // -->
</script>
   </head>

<body onload = "start()">
    <p id = "pText" style = "position:absolute; left:0; font-family:serif;
color:blue">
     Welcome!</p>
   </body>
</html>
```

8. Next steps

8.1 Reading

Textbook, Chapter 13 - Dynamic HTML: Object Model and Collections

Textbook, Chapter 14 - Dynamic HTML: Event Model

- Warning:
  - the Textbook introduces proprietary old-style Microsoft Dynamic HTML
  - It will only work with IE browsers. Although easier to use, it is not as powerful.
  - It will not prepare you for DOM scripting with XML (even in "IE"/Microsoft)
- Therefore, reading is quite optional ....

8.2 Next module

- Introduction to XML

8.3 Homework

1. Create an HTML page that includes some elements (tags) some of which you consider important.
   - Add a class attribute to these with some value, e.g.

```
<ul class="important"> <li>Before you start, insert the key</li> </ul>
```

   - Implement at least one push button that will change styling of these elements
   - Optional: Implement other push buttons that highlight other elements
   - Look at example 6-3 “Tree walking: Change color style” [p. 28]

2. For those with some programming skills:
   - Change other style elements, e.g. make elements visible or hidden, change position, etc.
   - Look at example 6-4 “Tree walking: Change any style” [p. 29]
- Homework will not be evaluated. However, doing it will help you pass the exams ...