Javascript

Code: js-intro

Author and version

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

- Version: 0.6 (modified 18/3/08 by DKS)

Prerequisites

- HTML
- CSS

Objectives

- Introduction to JavaScript
- Reuse of some simple JavaScript functions

Disclaimer

- There may be typos (sorry) and mistakes (sorry again)
- Please also consult the textbook !

Source code of examples

url: **http://tecfa.unige.ch/guides/js/ex/coap/**

- These examples complement the ones in the textbook used or are almost the same
  - Negrino & Smith (2007). JavaScript and Ajax for the Web, Visual QuickStart Guide, Sixth Edition.
- You also can consult the book's examples on its web site or download a zip file:

  url: **http://www.javascriptworld.com/**

Acknowledgement

These slides have been prepared with the help of

- The Textbook
- 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

url: http://tecfa.unige.ch/guides/js/ex/coap/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>
```

Document type declaration

Encoding declaration

Body (page contents)

Notes:

- Navigators are very forgiving, you may omit the doctype, the encoding declarations, or even the head. But your page will not validate (not acceptable in most professional settings).

Example 1-2: XHTML Page

url: http://tecfa.unige.ch/guides/js/ex/coap/preq-xhtml-page.html

- 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 simpler, stricter and somewhat less forgiving ....
- All tags are lower case, all tags must be closed (think “boxes” within “boxes”) !!!
- <html> must include xmlns, i.e. a namespace declaration
- The XML declaration on top is not absolutely mandatory, but the DOCTYPE is !

```
<?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

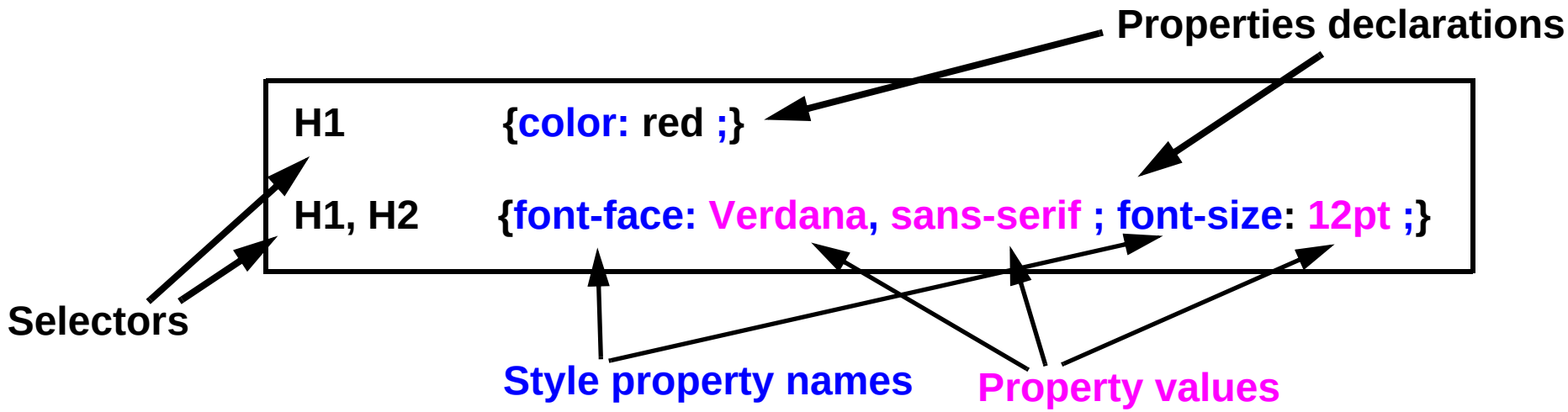
Body (page contents)

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";
    }
```



2. Why and where JavaScript ?

2.1 Origin

- JavaScript was developed by Netscape (under the name LiveScript)
- Microsoft adopted it under the name JScript
- The name “JavaScript” reflects certain syntactic similarities with JAVA (but JavaScript and Java are **very** different !)

2.2 Principal use of JavaScript

- Interactive forms
- Interactive applications with HTML forms (par ex. tests or quizzes)
- Verification of HTML forms before sending them off for “server-side” treatment

Interactive pages (DHTML)

- Richer HTML pages (ex. “highlighting”, menus, etc.)
- Applications that look like desktop applications (e.g. Google applications)
- Animations

User-centered contents

- Generation of HTML pages according to user profile
- Plugin detection, etc.

2.3 Versions of JavaScript

JavaScript has a long and complicated history:

- 1994: Mocha, then Live script - Netscape 1.x
- December 1995: renamed to JavaScript - Netscape 2.0
- August 1996: JScript - Internet Explorer 3.0
- June 1997: First edition of ECMA-262
- .....
- Nov 2006: JScript 5.7 - IE 7.0 based on ECMA-262 3rd ed.
- 2007: JavaScript 1.7 - Firefox 2.0

Fore more information, you may consult Wikipedia, e.g. start from:

url: **http://en.wikipedia.org/wiki/ECMAScript**

2.4 Other uses of ECMA/Java/J-Script

- ActionScript 3 is based on the latest ECMA standard
- Server-side scripting in .NET
- Client-side scripting for other formats (SVG, VRML, X3D, PDF, etc.)
- Some dashboard widgets (MacOS X, Yahoo, ...)
- Some desktop programs (e.g. the user interface of Firefox or Adobe CS3) can be scripted with some JS-like language
- ....

2.5 Tools

- To code Javascript, use a programming editor (and not Word!)
  - If you can, try to find an editor that does syntax highlighting and indentation, e.g. Notepad++
- To test small bits of code in Firefox:
  type the URL: "**`JavaScript:"`** or use the menu: Tools->Error Console
  - This will open the error console, you then can enter JS code in the small input window on top

  ```
  alert("hi there")
  ```
- To test small bits of code in IE
  use the "javascript:" protocol identifier followed by code, e.g:

  ```
  javascript:alert("hi there")
  ```
- For debugging always open the error console !!!!
  - in Firefox, see above
  - in IE: Tools->Internet Options ->Advanced (tick something like "Display notifications .." under the navigation header) ... sorry I don’t have an English Windows version at hand.
- Firefox: You should install several Add-ons like "DomInspector", "WebDevelopper", "GreaseMonkey.

3. JavaScript programming bare bones

- Donc panic, the idea is to give you a rough picture of what happens in the examples ...
- Please consult the textbook for more details !

3.1 Built-in methods

- Tell the browser window to write out some text

```
document.write("Hello, COAP 2130!");
```

url: **http://tecfa.unige.ch/guides/js/ex/coap/1-1-hello.html**

- Popup with a little message

```
alert("Hello, world!");
```

url: **http://tecfa.unige.ch/guides/js/ex/coap/1-3-alert.html**

Tip: Try this out in the Firefox error console or with a javascript URL in IE

- Prompting and confirming

```
prompt("Please, could you tell me a secret ?","");
confirm("Are you sure that you really wanted to give your secret away?"))
```

url: **http://tecfa.unige.ch/guides/js/ex/coap/1-4-prompt-confirm.html**

3.2 Variables

- Result of prompting is stored in a variable for later reuse

```
var answer = prompt("Please, could you tell me a secret ?","");
```

url: **http://tecfa.unige.ch/guides/js/ex/coap/1-4-prompt-confirm.html**

3.3 Functions

- Functions are little programs that can be reused (like recipes)
- Syntax for a simple function that is not given any information to deal with (no arguments):

```
function name () {
  // instructions
}
```

- Example: This function when called will simply execute the line alert(....);

```
function popupMessage() {
  alert("Hello, world!");
}
```

url: **Most examples in http://tecfa.unige.ch/guides/js/ex/coap/**

3.4 Associate events with functions

- window is an object that represents the navigator's display area, you can tell it to execute a function when it loads a page (assign its event handler onload property a function name)

```
window.onload = writeMessage;

function writeMessage() {
    document.getElementById("helloMessage").innerHTML = "Hello, world!";
}
```

url: **http://tecfa.unige.ch/guides/js/ex/coap/1-2-external-code.html**

3.5 DOM

- To change content of an HTML element, give it first a unique id, i.e. the HTML file:

```
<p id = "pText">Welcome to our Web page!</p>
```

- Then, in JavaScript, you can program somthing like:

```
var p = document.getElementById("pText");
p.innerHTML = "Heh I changed your text";
```

or in a single line:

```
document.getElementById("pText").innerHTML = "Heh I changed your text";
```

url: **see the previous example**

3.6 Conditional

- If this is true do this, else do that ....

```
var answer = prompt("Please, could you tell me a secret ?","");

if (answer) {
    alert("You told me: " + answer);
  }
else {
    alert("You refused to answer");
  }
```

url: **http://tecfa.unige.ch/guides/js/ex/coap/1-4-prompt-confirm.html**

4. JavaScript and HTML

Programming in JavaScript usually means two things:

1. Define functions, i.e. executable recipes that do something like ask questions from the user, popup information windows, compute an animation
2. Trigger (call) these functions after some user action. E.g. user loads the page, user moves the mouse, user clicks on a button ....

Usually, according to this logic, you put JavaScript code in two locations:

1. JavaScript ***functions*** are defined in the <head> .. </head> and within "**script**" tags
   - More complex code is usually defined in external files and linked like this:

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

   - The principle: Definitions of functions and variables etc. belong in the head within the script tag or in some external file (loaded also through the script tag).
2. JavaScript functions can be ***called*** or triggered in various ways from the body of the HTML document, e.g.:
   - JavaScript instructions can be inserted within ***"script"*** tags (as in the head)
   - JavaScript functions can be triggered by ***"HTML inline" event handler definitions*** (old style)
   - Event handling also may be defined purely within JS code (new DOM style) by assigning function names to event handling properties (new style).

   ... don't panic here. we will see the operational details later.

In the next four slides we will show some examples, do not worry if you do not (yet) understand details

All examples can be found here: **http://tecfa.unige.ch/guides/js/ex/coap/**

4.1 A popup/DOM/onload example

url: http://tecfa.unige.ch/guides/js/ex/coap/1-6-dom-change-modern.html

```
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Object Model</title>
      <script type = "text/javascript">
         window.onload = start;

         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>
      <p id = "pText">Welcome to our Web page!</p>
   </body>
```

loading event will call event handling function start.

Definition of the event handling function "start"

Remember:

- function definitions belong in the head
- You may trigger a function at page load (like above)

4.2 A popup/DOM/onload example - old style

Most often you will rather find old DOM 0, not as good since HTML is mixed with JS.

url: **http://tecfa.unige.ch/guides/js/ex/coap/1-6-dom-change.html**

```
<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 association to event handling function

4.3 A popup/DOM/button example - new style

url: http://tecfa.unige.ch/guides/js/ex/coap/1-7-dom-change-button-modern.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Simple DOM/innerHTML/button demo</title>
    <script type = "text/javascript">
      window.onload=init;
      function init() {
        document.getElementById("hitme").onclick = start;
      }
      function start() {
        var p = document.getElementById("pText");
        alert( "Page contents : " + p.innerHTML );
        p.innerHTML = "Thanks for coming.";
        alert( "Page contents temporarly changed to : " + p.innerHTML );
        p.innerHTML = "Cool, isn't it ?";
      }
    </script>
  </head>
  <body>
    <p id = "pText">Welcome to our Web page!</p> <hr/>
     <input id="hitme" type="button" value="Press me"/>
  </body> </html>
```

Associate event handling function start with a click event

Definition of the event handling function "start"

4.4 A popup/DOM/button example - old style

- This is old DOM 0 style - does not work with IE 7
- http://tecfa.unige.ch/guides/js/ex/coap/1-7-dom-change-button.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Simple DOM/innerHTML demo</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 temporarly changed to : " + p.innerHTML );
      p.innerHTML = "Cool, isn't it ?";
      }
    </script>
  </head>
  <body>
    <p id = "pText">Welcome to our Web page!</p>
    <hr/>
    <input onclick="start();" type="button" value="Press me"/>
  </body>
</html>
```

Definition of the event handling function "start"

Inline event and call of event handling function

4.5 Standardization etc.

Note on standardization

- The JavaScript language itself (ECMAScript) is fairly cross-browser compatible
- Event handling (DOM 2) and DOM 1 (querying and modifying the document) is not fully standard nor fully implemented by all browsers
- More advanced stuff (parts of DOM 2 and DOM 3) is less frequently implemented

Note on JavaScript and XHTML

- Valid XHTML requires JavaScript to be inserted within a commented CDATA section
- Many textbooks do not teach valid XHTML ....

```
<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.