Faculty of Science,
Technology
and Medicine

Web Programming

Volker Müller
University of Luxembourg

Example: Accessing Form Information

```
<form name = "demo">
<select name = "w" size="2"
    onChange = "chooseOption (  )">
<option>Option1</option>
<option>Option2</option>
</select>
```

chooseOption will be explained on next slide

Value of selectbox can be accessed as:

document.forms[0].elements[0] (if first form in doc.)

document.demo.w

Example: Form Information (2)

```
function chooseOption ( )
{
  var n = document.demo.w.selectedIndex;
  alert ("Choosen index : " + n);
  document.demo.textfield1.value = n + " --> " +
        document.demo.w.options[n].value) ;
}
```

Sets value of "textfield1" in form

Example: Changing Event Handlers

Most properties are even writable, including functions:

```
function init( )   // called as   <body onload="init()">

{     document.onmouseup = setStatus     }
```

onmouseup defines default function called for "mouse up" event

Event now handled by user-defined function setStatus

Extension of this Idea

Find a possibility for JS Scripts to access all possible parts of an HTML file, not only forms, images, links, …

Find a way to also move / delete / add HTML tags

Solve ambiguity of "name" attribute

→ Define how a document can be represented as an object, then we can manipulate that object

Document Object Model (DOM)

Platform- and language-neutral interface

Allows programs to dynamically access and update the content, structure and style of documents

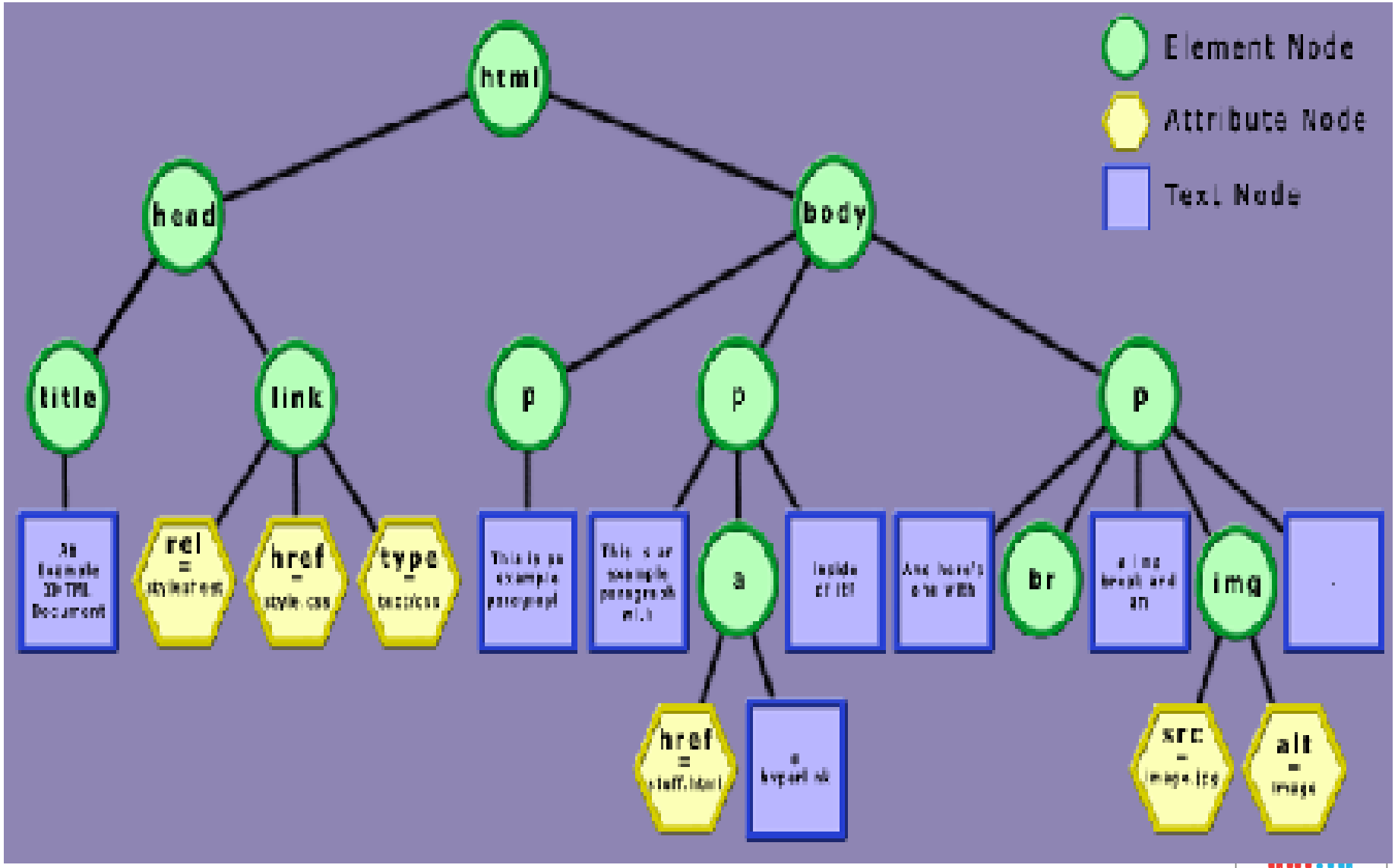
Focused on XML documents, but can also be used with HTML

Specification: http://w3.org/DOM

An Example HTML File

```
<html>

<head> <title> An Example HTML document</title>

<link ref = "stylesheet" href="style.css" type="text/css">

</head>

<body><p>This is an example paragraph.</p>

<p>This is an example paragraph with <a href="stuff.html">a
   hyperlink</a> inside it!</p> <p>And here's one with<br/>a line
   break and an <image src="image.jpg" alt="image">.</p>

</body></html>
```

DOM Tree for Example HTML



Main Idea of Using DOM

DOM specifies objects for representing XML/HTML documents, i.e. attributes and methods

Access of DOM subtrees / sub-objects possible

Object attributes can be changed

DOM tree can also be manipulated with methods:
change existing node, add new node, move nodes to another position, find parent node, ...

Some Methods for Documents

Element createElement (in DOMString tagName)

Text createTextNode (in DOMString data)

Attr createAttribute (in DOMString name)

NodeList getElementsByTagName (in DOMString tagname)

Element getElementById (in DOMString elementId)

Example of Using DOM with JS

```
function addParagraph ( )
{
  let p = document.getElementById ("par1");
  let newP = document.createElement ("p");
  let newText = document.createTextNode ("New");
  newP.appendChild (newText);
  p.parentNode.appendChild(newP);
}
```

DOM Nodes

createElement ('element_name')

createTextNode ('text content')

Methods for nodes:

- attributes[ ]
- getAttribute('attributeName')
- removeAttribute('attributeName')
- setAttribute('attribute_name','attribute_value')

Attributes / Methods for Nodes & NodeList

parentNode

childNodes

firstChild, lastChild

previousSibling, nextSibling

getAttribute (name), setAttribute (name, value)

length

item(i)

See http://www.howtocreate.co.uk/tutorials/javascript/domstructure

Example: DOM Events

```
var oDiv = document.getElementById ('thediv');

oDiv.addEventListener('click', function (e) {

  alert('1st event handler'); });

oDiv.addEventListener('click', function (e) {

  alert('2nd event handler');});
```

Comments on DOM Event Handling

Can add several event handlers for the same event on same object, but order is undefined

There exists optional third boolean parameter that determines order of event handler calls (of same type) for inheritance chain of objects

Default ("bubbling" phase): event handlers called in up going inheritance order

DOM and Styles

DOM Level 2 defines access to CSS styles

Can also manipulate styles directly:

```
t = document.getElementById ("p1");

t.style.backgroundColor = "green";
```

Note: CSS property names slightly changed

background-color → backgroundColor

font-variant → fontVariant

JS DOM Problems

DOM API provides a nice framework for implementing very general "dynamic HTML"

Focus still on programmers, not always easy to develop scripts for “amateurs” → use simple JS Frameworks

Ajax

Ajax = “Asynchronous JavaScript and XML”

Cross-platform technique

Exchanging asynchronously small amounts of data with the server behind the scenes

Basis technology behind "Web 2.0"

Combination of HTML, CSS, Javascript and a predefined JS object XMLHttpRequest

XMLHttpRequest Object

Used for JS script to send requests to server and receive response (without explicit user action)

Any type of data can be received from server, not only XML (in practice: text, html, JSON, XML)

Subject to same-origin policy

API:
https://developer.mozilla.org/en/docs/Web/API/XMLHttpRequest

Usage Example: Sending Request

```
xhr = new XMLHttpRequest ( );

xhr.onreadystatechange  = function ( )

{   if (xhr.readyState  == 4 && xhr.status  == 200)

        document.write (xhr.responseText);

};

xhr.open ("GET", "http://localhost/data.php",  true);

xhr.send (null);
```

3rd parameter: asynchr.(true) or synchronous (false)

Possible States

Attribute readyState holds the current status of the XMLHttpRequest. Changes from:

0: Request not yet initialized

1: Server connection established (open() called)

2: Response headers received

3: Processing response (downloading response data)

4: Response ready

Usage Example (3): Sending POST Request

```
...
xhr.open("POST","http://localhost/data.php", true);
xhr.setRequestHeader ("Content-type","application/x-
    www-form-urlencoded");
xhr.send("fname=Henry&lname=Ford");
```

HTTP header determines format provided in request body

Alternative: multipart/form-data for binary or large payload

Server Side for Ajax

Code on server side is exactly the same as for a normal web page

Input via GET parameters or POST (in request body)

→ Ajax independent of used server side technology (code can be written in any suitable language)

Often "RESTful Web Service" on server side

Usage Example (2): Receive XML

Replace output responseText with

```
var doc = xhr.responseXml;

var element =
  doc.getElementsByTagName("root").item(0);

// output element.firstChild.data
```

Access to complete XML data possible using DOM

Distinguishing Text and XML Response

```
var cType = xhr.getResponseHeader ("Content-Type");


if (cType == 'text/xml')
    {   /* XML response  */    }
else if (cType == 'text/plain')
    {    /* plain text response */  }
```

XML Documents and DOM

Exactly the same DOM functions as explained before can be also used with XML

Tag names can vary → script must be consistent with specific XML file

Be careful with white space characters in XML file → show up in DOM tree

Disadvantage: quite some overhead due to XML tags for small amount of data

JSON

JSON = JavaScript Object Notation (RFC 4627)

Lightweight computer data interchange format

Text-based, human-readable

Subset of Javascript

Basic types: Number, String, Boolean, Array, Object, null

JSON Example

```
{
    "firstName": "Volker",
    "lastName": "Müller",
    "address": {
        "streetAddress": "rue XYZ, 12",
        "city": "Luxembourg"
    },
    "phoneNumbers": [ "6534",  "+49 651 12345" ]
}
```

Exactly equal to definition of objects in JS (remember last week)

Using JSON Data on Client Side

Set desired format in request:

```
request.responseType = "json";
```

Assign response to an object, conversion to JS object will be done (almost) automatically:

```
xhr.onreadystatechange = function ( ) {

  if (xhr.readyState == 4 && xhr.status == 200)

     {   object = JSON.parse(xhr.response);   ..... }
```

jQuery (jquery.com)

Fast, relatively small and concise JavaScript Library

Simplifies HTML document traversing, event handling, animating, and Ajax interactions

CSS-like syntax for access of elements in combination with many predefined operations and data structures → special focus on designers

Cross-browser

Some Examples

```
$("div.contentToChange p").size();

$("div.contentToChange
p.firstparagraph:hidden").slideDown("slow");

$("div.contentToChange
p:not(.alert)").append("<strong class='added'> Text
</strong>");

$("div.contentToChange p.par3").hide("slow");
```

Basic Idea of jQuery

Access operator $(...) can be used with HTML tag names, CSS selectors or DOM specifiers, returns list of all elements of that type

Large set of predefined functions available that can be applied to that list

Functions can also be chained since each function returns changed list

Event handlers can be directly defined with JS closures

Example: Simple Sliding Menus

```
$(document).ready( function() {
    $("#drop_down").hide( );
    $("#drop_down").animate({  opacity:0.5  });
  $("a:contains('Google')").click(
            function( ) {  $("#drop_down").show("slow"); });
  $("#drop_down").mouseout (
            function( ){  $(this).hide('slow');  });
});
```

```
<div id="drop_down"> stores
 menu
```

More jQuery Features

jQuery event handlers can be easily written

jQuery has very easy to use Ajax support (for client side)

jQuery UI components exist

Large list of jQuery plugins exist

jQuery also exists in variant for mobile apps

Next Meeting

Dedicated Javascript APIs

Javascript Tools

TypeScript