HUMAN CAPITAL
NATIONAL COHESION STRATEGY

EUROPEAN UNION
EUROPEAN
SOCIAL FUND

JavaScript: Events and DOM

Maciej Grzenda & Michał Okulewicz
Warsaw University of Technology
Faculty of Mathematics and Information Science
M.Grzenda@mini.pw.edu.pl
M.Okulewicz@mini.pw.edu.pl

JS events and HTML

Event handlers

- A very common application of JS code is to provide the HTML documents with the procedures that will be executed by the web browser whenever an event on an HTML document item is triggered. Sample solutions created in this manner include:
  - when the cursor leaves the surname field of the form, check if there is any text typed in the field,
  - when a user tries to submit the HTML form verify if all mandatory fields are not empty,
  - when the document is closed prompt the user to confirm.
- How to specify the event handlers?
- Example:

```
<input type="button" value="Check" onclick="myCheck(this.form);">
```

Typically the role of an event handler is to call a function defined in the header section of the document or in a separate js file. In both cases SCRIPT tag defines the code. Many SCRIPT tags can be used in the same document.

Event handlers

- Key events are:
  - click - click on form element or link,
  - change - change value of text, text area or select,
  - load - user loads the page in a browser,
  - mouseover - move mouse pointer over a link,
  - submit - submit a form,
  - unload - user exits a page.
- The JS handler names are: on<EventName> e.g. `onclick, onload`

A list of supported events

Attribute	The event occurs when...
onblur	An element loses focus
onchange	The content of a field changes
onclick	Mouse clicks an object
ondblclick	Mouse double-clicks an object
onerror	An error occurs when loading a document or an image
onfocus	An element gets focus
onkeydown	A keyboard key is pressed
onkeypress	A keyboard key is pressed or held down
onkeyup	A keyboard key is released
onload	A page or image is finished loading
onmousedown	A mouse button is pressed
onmousemove	The mouse is moved
onmouseout	The mouse is moved off an element
onmouseover	The mouse is moved over an element
onmouseup	A mouse button is released
onresize	A window or frame is resized
onselect	Text is selected

Source: http://www.w3schools.com/jsref/dom_obj_event.asp

Event handlers in JS

- Example:
  - `<input type="button" value="Check" onclick="myCheck(this.form);">`
  - By specifying the onClick handler code we inform the browser that whenever someone clicks on this button, the myCheck() function should be called.
  - The function body should be defined in the SCRIPT tag i.e. either embedded in HTML document or any of linked JS files (see above examples on combining HTML with JavaScript).
- Use

  `return false;`

  in the event handler to cancel default action e.g. not to submit the form you have found to be incomplete.
- Notice: complex code can be used as a code of the handler. However, in order to easily maintain the HTML document, it is highly recommended to place the JS code in separate JS functions.

DOM and JavaScript

- DOM stands for Document Object Model and came into being as a result of web community efforts to provide common interfaces for accessing the content of HTML, XHTML and XML documents programmatically.
- The complete specification of DOM can be found at http://www.w3.org and contains:
  - core interfaces that are sufficient to manipulate XML and HTML documents programmatically in products conforming to DOM specification, in particular web browsers. See IDL definitions for detailed examples.
  - additional interfaces for HTML documents that provide extra functionality for HTML documents.

DOM – why?

- One of the key assumptions of DOM is that using DOM interfaces web developer can programmatically create or modify the whole document (in particular HTML document), thus define both tags, their hierarchical structure, content and attributes.
- In particular that feature can be used to generate or modify the HTML document on-line, within a browser – without any help of web server and server-side scripting.
- This means that server-side processing can be avoided as long as server resources are not necessary.

Why DOM+JavaScript?

- Consequently, JavaScript can be used to call DOM methods and manipulate the document content.
- This combination of DOM APIs, HTML, CSS and JavaScript is referred to as DHTML (Dynamic HTML) as it concentrates on dynamic modification and creation of documents. Typical applications of DOM interfaces are:
  - accessing the attributes of document attributes, in particular to check the form field values,
  - adding new tags and attributes - for instance to extend the document by adding an extra input field,
  - changing the document display by manipulating attributes and styles (in particular visibility and location of document parts),
  - opening and defining the content of new browser windows. The following sections give an overview of sample methods and their applications in the code.

JavaScript - disadvantages

- Unfortunately, the JS implementation in different web browsers differs.
- Thus programming in JS for many browsers (e.g. Internet Explorer and Mozilla Firefox) is considered by many to be a kind of an art.
- In some cases, one must detect browser version and prepare different pieces of code to obtain the same functionality in the two browsers.
- Still it remains feasible to prepare JS solution working properly in all leading browsers
- One of the techniques is to use JS libraries largely simplifying the development of JS code, such as jQuery and prototype library

DOM – sample methods

- One of the most popular DOM methods is:
  `Element getElementById(String ID);`
- It allows the following examples:

```
<input type="text" id="ident" name="surname">
...
<input type="submit" onclick="if
(document.getElementById('ident').value=='')
{ alert("No data in surname"); return false;} ...">
```

- In other words the method can be used to retrieve the reference to the element with a certain ID – no matter where the element is in the document tree. Then all defined attributes can be accessed. However:
  - Behaviour is not defined if more than one element share the same ID,
  - The document tree must be properly formatted in terms of tag nesting, ending tags, attribute values etc. Otherwise the behaviour of this and other DOM methods may be unpredictable.

DOM – method definitions

getElementsByTagName

Returns a `NodeList` of all descendant `Elements` with a given tag name, in document order.

Parameters

`name` **of type** `DOMString`

The name of the tag to match on. The special value "*" matches all tags.

Return Value

`NodeList` A list of matching `Element` nodes.

createElement

Creates an element of the type specified. Note that the instance returned implements the `Element` interface, so attributes can be specified directly on the returned object.

In addition, if there are known attributes with default values, `Attr` nodes representing them are automatically created and attached to the element.

To create an element with a qualified name and namespace URI, use the `createElementNS` method.

Parameters

`tagName` **of type** `DOMString`

The name of the element type to instantiate. For XML, this is case-sensitive, otherwise it depends on the case-sensitivity of the markup language in use. In that case, the name is mapped to the canonical form of that markup by the DOM implementation.

Return Value

`Element` A new `Element` object with the `nodeName` attribute set to `tagName`, and `localName`, `prefix`, and `namespaceURI` set to `null`.

Source: DOM specification. The specification contains detailed description of individual attributes and methods.

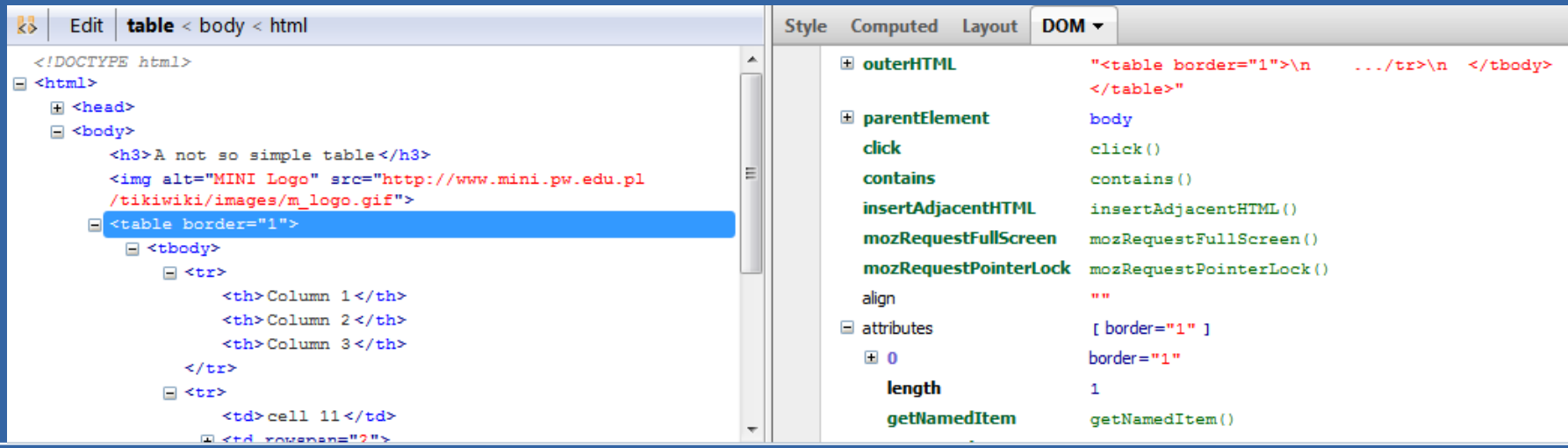
Case study – form validation

- One of the key applications of JavaScript is to validate the content of an HTML form,
- This includes:
  - Checking the content of the HTML form supplied by a user for completeness and validity before it is sent to a server,
  - Checking the data at a textbox level,
  - Formatting the data that is being typed in,
  - …

lab_5_task_2.html

DOM – tree node

- DOM-compliant browser while reading the document creates a tree node out of its content
- Nodes correspond to HTML/XML tags
- For each node a list of attributes is created
- Thus, document tree contains complete document representation



DOM methods - continued

- In case of HTML, a separate method has been defined to access a collection (possibly empty) of all the nodes with a certain value of NAME attribute by calling for instance:

```
document.getElementsByName('SURNAME')
```

- Moreover, it is possible to call for instance

```
document.getElementsByTagName('INPUT')
```

to retrieve a collection of all the nodes with a certain node name.

- Another useful node property is `childNodes`. It provides a read-only list of the children of the node.

- Similarly `attributes` is a table of all defined attributes of a node.

Maciej Grzenda & Michał Okulewicz http://www.mini.pw.edu.pl/~grzendam http://www.mini.pw.edu.pl/~okulewiczm

Adding new attributes

- It is enough to reference the element to access and possibly change one of its existing attributes
- A special method must be called to create a new attribute in a tag. The method is:

  `setAttribute(String name, String value)`

  and is called on an Element object.
- Similarly `removeAttribute(String name)` can be used to remove an existing attribute.
- Notice: an attempt to set the value of previously undefined attribute may generate an error!

`lab_5_task_4.html`

innerHTML

- Programmatic changes in HTML document are not reflected in document source.
- Thus, there is a need to check the actual document code incl. modifications made through JavaScript code.
- One of the properties used in this context is named `innerHTML`
- It allows the developer to observe the actual HTML code based on the original content and JavaScript-based modifications.

Document structure

- Additional insight into document structure as seen by the browser is provided by the following properties:
  - `innerText` – inner text of a node
  - `innerHTML` – inner HTML code of a node
  - `outerText` – outer text of a node
  - `outerHTML` – outer HTML code (includes tag definition)
- Manipulating innerHTML might lead to unexpected results, one should always use:
  - document.createElement(tag name as string)
  - document.createTextNode(content);

The investigation of the changes made to document content

```
…
<p>just testing</p>
<div    id="test"><strong>This    is    sample    HTML
   content</strong></div>
<script type="text/javascript">
    alert("InnerText is: "+test.innerText);
    alert("InnerHTML is: "+test.innerHTML);
    alert("outerText is: "+test.outerText);
    alert("outerHTML is: "+test.outerHTML);
    test.setAttribute("myattrib","myvalue");
    alert("InnerText is: "+test.innerText);
    alert("InnerHTML is: "+test.innerHTML);
    alert("outerText is: "+test.outerText);
    alert("outerHTML is: "+test.outerHTML);
</script>
…
```

js_example_inner_outer.html

New attribute that is added by setAttribute() is reflected in the output of the last alert command using outerHTML.
Notice the differences between IE and FF browsers.

DHTML – sample solution

```
<STYLE type="text/css">
   DIV {position:relative;}
</STYLE>
<SCRIPT type="text/javascript">
var pos=0;
function move_window()
{
   document.getElementById('mydiv').style.left=pos+'px';
   document.getElementById('mydiv').style.top=pos*2+'px';

   pos = pos+10;
   if (pos>300)
        pos=0;
   window.setTimeout('move_window();',100);
}
</SCRIPT>
</head>
<BODY onLoad="move_window();">
<DIV ID='mydiv'>The text</DIV>
</BODY>
```

js_example_animation.html

- **onLoad starts a function that will move DIV object on the screen**
- **style property of a node is used to access CSS properties (left is an example of CSS property)**

Summary

- JavaScript is present in virtually all web sites
- The HTML document content can be fully redefined by JavaScript code accessing DOM interfaces,
- The role of JavaScript is even larger in modern web sites, when rich user interface is expected,
- To simplify JavaScript development and hide underlying differences between web browser JS dialects libraries such as jQuery or prototype can be used