Today

Setting attributes

Finding elements

Node relationships

Creating children

Getting started

Document Object Model and JavaScript

Dennis Olsson

Lecture #8

1 Setting attributes

2 Finding elements

3 Node relationships

4 Creating children

5 Getting started

Document Object Model

Objects in trees haven't been handled, but objects in a tree can be seen as forks leading to one or several branches. Each branch can be either a primitive value or another object.

The idea is that, since the document has a hierarchy, get the elements (for instance tags) as objects with attributes.

DOM usage

The DOM can be used to read, write and manipulate the entire XHTML-tree.

New nodes can be added, removed, changed, moved, etc.

On the next slide an example where a link is updated by a minimal manipulation in the DOM-tree.

Example

Code from example-domexample.html

```
/* JavaScript code */
var comicID = 39;
function changeLink() {
  comicID++;
  var link = document.getElementById("comiclink");
  link.href = "http://www.explosm.net/comics/" + comicID;
  link.innerHTML = "Comic ID: " + comicID;
}

<!-- XHTML-code -->
<a href="#" id="comiclink">No comic set</a>
<a href="#" onclick="changeLink();">Next Comic</a>
```

Each node as an object

In the above example, the anchor was identified as an anchor by the browser, enabling us to direct access the attribute **href**.

This is the clean way to do it, but since **href** is an attribute, we could just as well change that attribute:

```
link.setAttribute("href",
  "http://www.explosm.net/comics/" + comicID);
```

It might be easier to remember. All XHTML-attributes can be changed the same way.

Event handling continued

As mentioned in the previous lecture, the **OnLoad**-attribute should be left out of the HTML-code, since it's cleaner.

This attribute can be changed using either the **setAttribute**-function, or by a function pointer. An example follows.

Example

Code from example-onclickattribute.html

```
/* JavaScript code */
function onloadhandler() {
  var link = document.getElementById("hotlink");
  link.setAttribute("onclick", "alert('Fire!');");
// The above row could be replaced by the following.
  //link.onclick = function(){ alert('Fire!'); };
}
window.onload = onloadhandler;

<!-- XHTML-code -->
<a href="#" id="hotlink">Click this hot link</a>
```

Using setAttribute

Why not to use:

```
link.onclick = function(){ alert('Fire!'); };
```

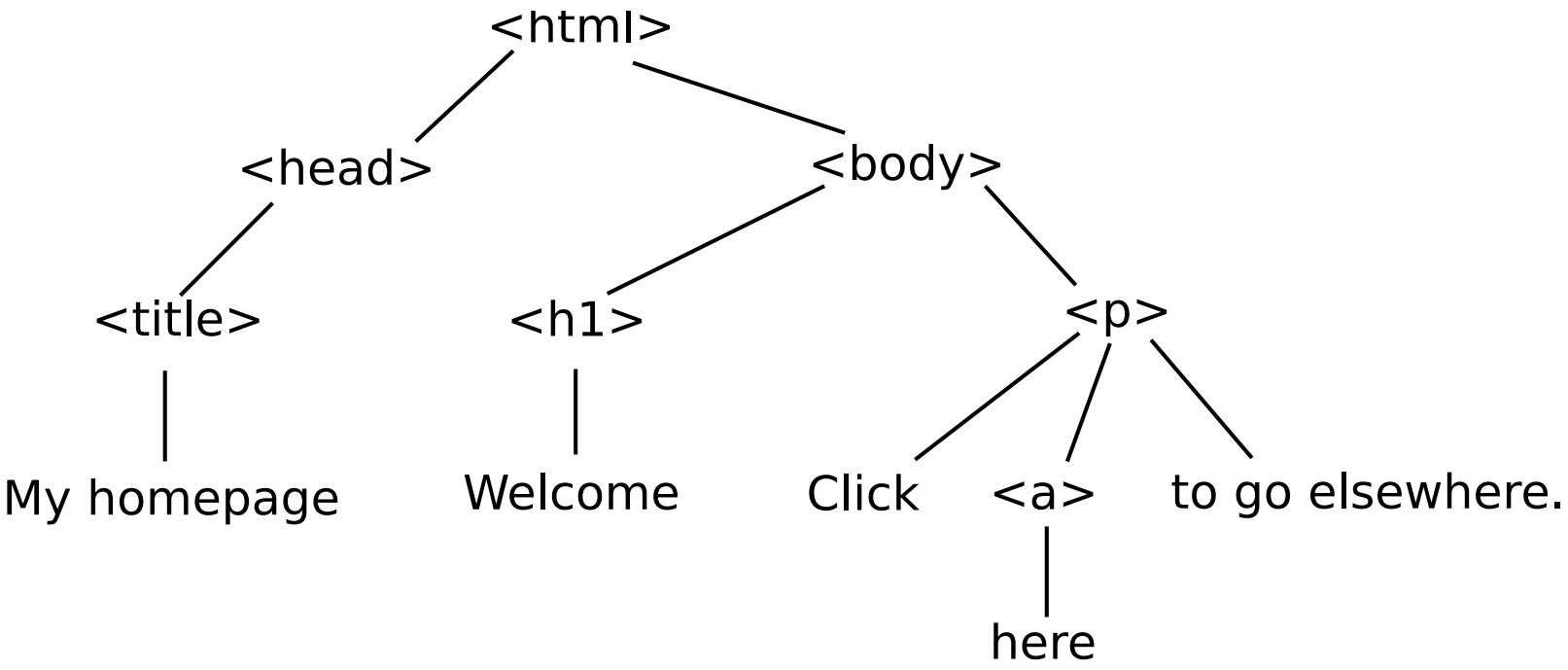
The difficulties is not so much about doing it, but to understand why. Also it'll result in code harder to read for the untrained eye.

Therefore we recommend you to use **setAttribute**. It's better to get used to a few thing at a time, and **setAttribute** will be used for a lot of other things as well.

Where to look

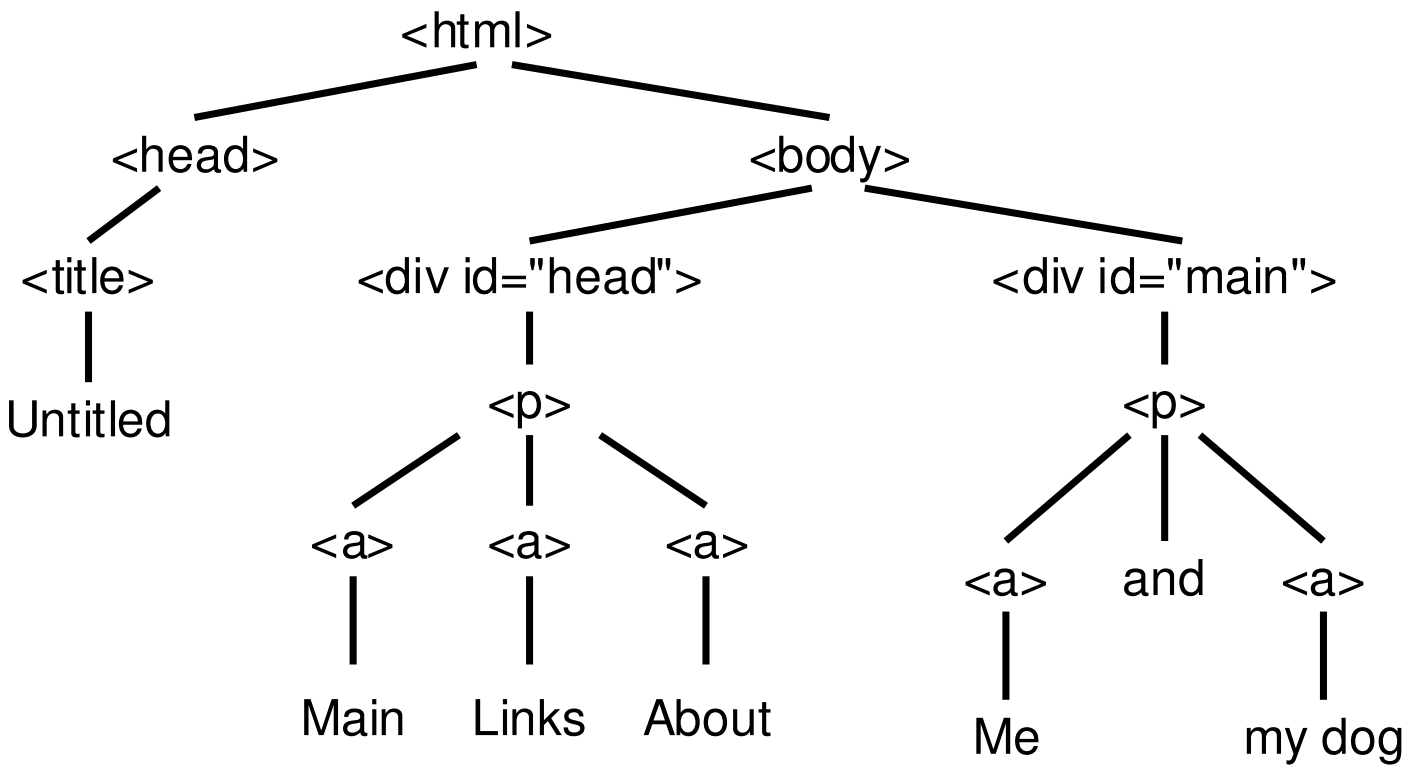
We've already used **getElementById**. A document is supposed to only contain unique IDs, so **getElementById** only return one object at most. This is also the reason why we only have to use **document** as the parent when making the call to **getElementById**.

Remember this image:



The hierarchy

Here's another, more suiting, example:



Let's say we'd like to change the target of all links in the div *head* to named anchors with the same name as the text.

Finding nodes

There are two usable functions for finding all nodes of a certain kind:

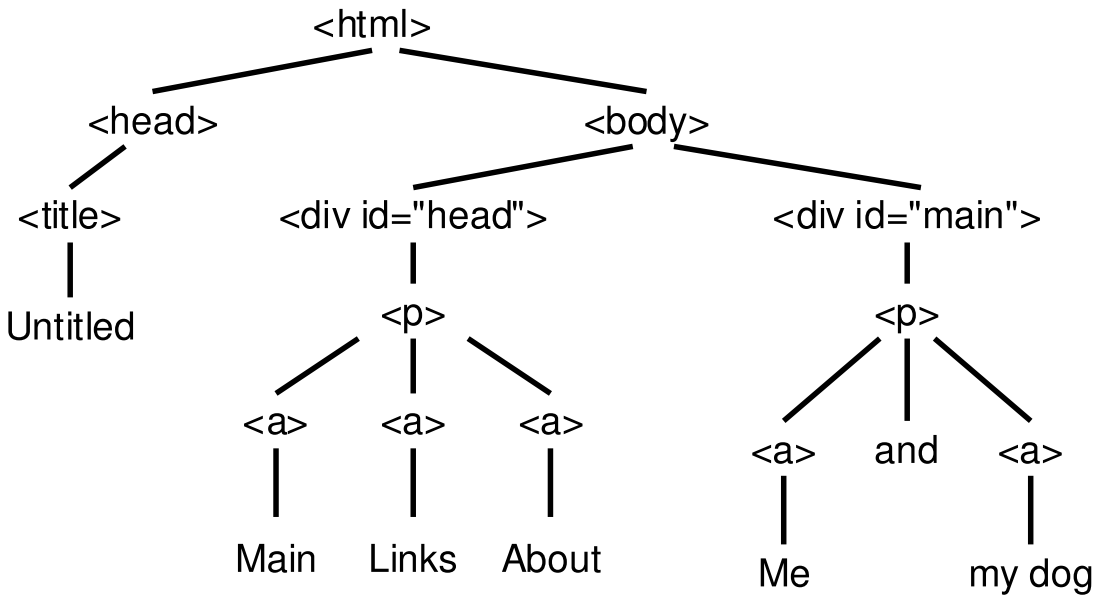
- getElementById(...) – Find an nodes with the given ID
- getElementsByTagName(...) – Find all nodes of this type

So, by joining these, we could first find the div-node, and then get a list of all anchors.

Code from example-domfindelements.html

```
var allA =
  document.getElementById("head").getElementsByTagName("a");
for (var i = 0 ; i < allA.length ; i++)
  allA[i].setAttribute("href", "#" + allA[i].innerHTML);
```

The attribute **innerHTML** is the content of the tag, including other tags.



The family tree

The DOM-tree has a family relationship. Each node (except the window) has a parent. All nodes have zero or more children. No child have more than one parent, though.

A nodes parent can be reached via the **parentNode**-attribute, and the children are listen in an array named **childNodes**. Similarly, **attributes** is an array with all XHTML-attributes of the object.

The name of a node's tag can be accessed via it's **tagName**-attribute.

During event-handling, the node itself can be referred to as **this** in the attribute. See the following example.

Example parent access

Code from `example-parenttype.html`

```
/* The JavaScript code */
function findmyparent(link) {
  alert("The parent is of the type: " +
    link.parentNode.tagName);
}

<!-- The XHTML code -->
<p><a href="#" onclick="findmyparent(this);">
  Link within a paragraph</a></p>
<div><a href="#" onclick="findmyparent(this);">
  Link within a div</a></div>
```

Creating children

To keep it simpler – one general way to handle the creation will be presented.

A child is a descendant from the document. Therefore the document is asked to create it, using;

```
var newA = document.createElement('a');
```

The node can then be manipulated as before:

```
newA.setAttribute('href', 'http://www.asdf.com/');
```

Finally, the element can be inserted in the DOM-tree, by using **appendChild**.

```
alreadyExistingDiv.appendChild(newA);
```

An example follows.

Example - generate links

Code from example-generatelinks.html

```
var theparent = document.getElementById('xkcdlist');
for(var i = 1 ; i < 276 ; i++) {
  var newA = document.createElement('a');

  newA.setAttribute('href', i
      'http://www.xkcd.com/c' + i + '.html');

  newA.setAttribute('id', 'xkcdlink' + i);

  newA.innerHTML = 'xkcd - strip: ' + i;

  theparent.appendChild(newA);
  // break the line
  theparent.appendChild(document.createElement('br'));
}
```

Inserting **into** the child list

The above example inserts all children into the list. It could need some separation of the links, for instance a `<hr />`-tag every five nodes.

Instead of doing it while inserting the links, we will, for demonstration purposes, do it afterwards.

To insert an element before another, use the **insertBefore**. To insert after the element, get the next sibling from the attribute **nextSibling** and insert it before the next.

Code from `example-generatelinks2.html`

```
for (var i = 6 ; i < 276 ; i += 5) {
  var thelink = document.getElementById('xkcdlink' + i);
  thelink.parentNode.insertBefore(
      document.createElement('hr'), thelink);
}
```

Getting started with JavaScript

To start using JavaScript and start parsing the DOM-tree, the best way is trial-and-error. Use Firefox in conjunction with Firebug. to see the complete DOM-tree.

the JavaScript console in Firebug includes tab completion, and the DOM-tab can explore all (for Firefox) known attributes.

What ever you decide to do, always test the JavaScripts in several browsers before deploying a product/site. There are a lot of non-standard ways to do things, even in Firefox. And even if you follow every standard known to man, not all browsers supports them completely.

Finally, assignment 4 only have to work in Firefox.

Summary

- **Setting attributes** – object attribute, XHTML attribute
- **Finding elements** – the DOM tree, get element by id/tag
- **Node relationship** – childNodes, parentNode
- **Creating Children** – creating nodes, attaching to tree