HTML DOM
Objectives

• Study the Document Object Model (DOM)
• Learn how to get, change, create, add, and delete HTML elements
What is the DOM?

• Document Object Model (DOM)
  – "The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

• W3C standard for accessing any type of XML document, including XHTML

• View document as a collection of nodes in a tree structure
Examples of using DOM

• Dynamically add or remove HTML elements
  – e.g. used by AJAX technology to reduce page refresh
• Combine with events to respond to user interaction or time-based events
  – e.g. change images for slide show
HTML DOM

- HTML DOM defines the objects and properties of all HTML elements, and the methods to access them
- HTML DOM is the standard for how to get, change, add, or delete HTML elements
- (In JavaScript, there are also other ways, but these are not necessarily compatible with other programming languages)
- View HTML document as a tree structure.
HTML DOM Tree Example

```html
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <a href="link">My link</a>
    <h1>My header</h1>
  </body>
</html>
```
Tree Structure

- Trees are always depicted up-side-down!
- Top node is called the root node
  - `<html>` element
- Each node can have any number of children
  - e.g. `<head>` and `<body>` are children of `<html>`
- Each child has exactly 1 parent (except root)
- Nodes without children are leaf nodes
- Siblings are nodes with the same parent
DOM Nodes

- 4 types of nodes in the DOM TREE:
  - Entire document is the `document` node
  - Every HTML element is an `element` node
  - The text in the HTML elements are `text` nodes
  - Every HTML attribute is an `attribute` node

- A node is a JavaScript `object`
Detailed Example

<html>
<head>
  <title>DOM Tutorial</title>
</head>
<body>
  <h1>DOM Lesson one</h1>
  <p style="color:grey;font-size:20px">Hello world!</p>
  <p style="color:red">And here is more text</p>
</body>
</html>
<!DOCTYPE html>
<html><head>
<title>DOM Tutorial</title>
</head>
<body>
<h1>DOM Lesson one</h1>
<p style="color: grey; font-size: 20px;">Hello world!</p>
<p style="color: red;">And here is more text</p>
</body></html>
DOCTYPE: HTML

HTML

HEAD

#text:

#text:

#text: TITLE

#text: DOM Tutorial

#text:

BODY

#text:

H1

#text: DOM Lesson one

#text:

P style="color: grey; font-size: 20px;"

#text: Hello world!

#text:

P style="color: red;"

#text: And here is more text

#text:
<html>
<head>
<title>DOM Tutorial</title>
</head>
<body>
<h1>DOM Lesson one</h1>
<p style="color: grey; font-size: 20px;">
Hello world!
</p>
<p style="color: red;"
And here is more text
</p>
</body>
</html>
DOCTYPE: HTML

HTML

HEAD

#text:
#text:
#text: "DOM Tutorial"

#text:

BODY

#text:

H1

#text: "DOM Lesson one"

#text:

P style="color: grey; font-size: 20px;" style="color: red;"

#text: "Hello world!"

#text:

#text: "And here is more text"

#text:
text nodes – most are empty!

DOM Tutorial

DOM Lesson one

Hello world!

And here is more text
Traversing the Tree

document.documentElement  access to root node
document.body gives direct access to the <body> node

node.parentNode returns a parent node
node.childNodes returns array of nodes
node.attributes returns array of attribute nodes

node.firstChild returns first child
node.lastChild returns last child
node.nextSibling returns next sibling

etc....
DOM Lesson one

Hello world!

And here is more text
Node property: `nodeName`

`node.nodeName` – returns name of node

- `nodeName` is read-only
- `nodeName` of an `element` node is the same as the tag name
- `nodeName` of an attribute node is the `attribute` name
- `nodeName` of a `text` node is always `#text`
- `nodeName` of the `document` node is always `#document`
Node property: `nodeValue`

`node.nodeValue` - returns value of node
- `nodeValue` for `element` nodes is undefined
- `nodeValue` for `text` nodes is the text itself
- `nodeValue` for `attribute` nodes is the attribute value
DOCTYPE: HTML

HTML

HEAD

#text:

#text:

#text:

TITLE

  #text: DOM Tutorial

#text:

#text:

BODY

#text:

H1

  #text: DOM Lesson one

#text:

#text:

P style="color: grey; font-size: 20px;"

  #text: Hello world!

#text:

P style="color: red;"

  #text: And here is more text

#text:
Accessing Example

document.body.childNodes[1].
    firstChild.nodeValue

will have value: DOM Lesson one
Accessing Element Text: the `innerHTML` Property

- W3C has not officially approved the `innerHTML` property as part of the DOM
Modifying Elements

document.body.childNodes[1].firstChild.nodeValue = "..";

or

document.body.childNodes[1].innerHTML = ".."
Direct Node Access

• Impractical to traverse entire tree, inflexible to changes
• Can use unique identifiers to directly access certain elements:

```html
<h1 id="...">
id has to be unique!
```
Direct Node Access

```
node.getElementById(id)
```

- Returns the first element in a document with a matching id attribute

```
node.getElementsByTagName(tagname)
```

- Returns all elements that have the same tagname in an array
Example

• Accessing the **text node** of a `p` node

...<p id="intro">Hello World!</p>...

```javascript
<script type="text/javascript">
  txt=document.getElementById("intro").childNodes[0].nodeValue;

  document.write("<p>The text from the intro paragraph: " + txt + "</p>"网球);
</script>
```
Example (cont)

txt=document.getElementById("intro").firstChild.nodeValue;

- Note that the text node is a child of the element node
- Equivalent to:

txt=document.getElementById("intro").innerHTML;
Attribute Nodes

- Each **element node** can contain **attribute nodes**
  - e.g. src, href, value, id, type
- Only element nodes can contain attributes
- Accessing attributes:
  
  ```javascript
  node.attributes
  ```
  returns array containing all attribute *nodes (objects)*

  ```javascript
  node.getAttribute(name)
  ```
  returns attribute *value (!)*

  ```javascript
  node.setAttribute(name, value)
  ```
  adds or changes value of attribute
Example 1

...<img id="image1" src="slide0.jpg"/>
...

// change existing attribute
document.getElementById("image1").setAttribute("src","slide1.jpg");

// add a new attribute
document.getElementById("image1").setAttribute("name","slideName");
Example 2

...<p id="par1">Some Text</p>...

// add a style attribute
document.getElementById("par1").setAttribute("style","color:green");
Form Elements (1)

- **Reading the user input from a text element**

```html
<form id="form1"><input type="text" value="enter your name"/></form>

...  

username =
    document.forms[0].elements[0].value;

username =
    document.getElementById("form1").elements[0].value;

username =
    document.getElementById("form1").childNodes[1].value;
```
Form Elements (2)

• Note that the following returns the default value, and not the value entered by the user

```javascript
// incorrect:
username = document.getElementById("form1").childNodes[1].getAttribute("value");
```
Exercise

Create the `showName()` function that reads the text field, and changes the `<h1>` text to display the name using the DOM

```html
<html>
<head>
    <title>Manipulating the DOM</title>
</head>
<body>
    <h1>Please enter your name</h1>
    <form>
        <input type="text"/>
        <input type="button" value="Press Me" onclick="showName()"/>
    </form>
</body>
</html>
```
Why is this useful?

• We would like to dynamically change the HTML document
• Need to create new elements
• Add and remove element nodes from structure
Creating a New Node

- Creating **element nodes**

  ```javascript
  node = document.createElement(tag);
  example:
  document.createElement("p");
  ```

- Creating **text nodes**

  ```javascript
  node = document.createTextNode(text);
  example:
  document.createTextNode("Hello World!");
  ```
Adding and Removing nodes

- Insert a child `nodeChild` to `nodeParent`

  ```javascript
  nodeParent.appendChild(nodeChild)
  ```

- Remove a child `nodeChild` from `nodeParent`

  ```javascript
  nodeParent.removeChild(nodeChild)
  ```
Example

// create p element
par1=document.createElement("p");

// create text node
text1=document.createTextNode("Hello World!");

// add text to p element
par1.appendChild(text1);

// add an attribute to p element
par1.setAttribute("style","color:green");

// finally add node to document body
document.body.appendChild(par1);
Summary

• DOM presents an HTML document as a tree-structure
• All nodes can be accessed through the tree.
• Their contents can be modified or deleted, and new elements can be created.
To Conclude

• DOM is not covered properly in book!
• Some information in Chapter 10
• See also W3C School Tutorials:
  http://www.w3schools.com/html/dom/default.asp
  http://www.w3schools.com/Dom/default.asp
• Try it yourself: Live DOM viewer
  http://livedom.validator.nu/