CS603 – Web Engineering

PREPARED BY: DR. REEMA PATEL

Document Object Model

- The Document Object Model (DOM) is perhaps the single greatest innovation on the Web since HTML was first used to connect related documents together over the Internet.
- The DOM gives developers unprecedented access to HTML, enabling them to manipulate and view HTML as an XML document.
- The DOM represents the evolution of Dynamic HTML, pioneered by Microsoft and Netscape, into a true cross-platform, language-independent solution.

Document Object Model

- The Document Object Model (DOM) is a programming API for HTML and XML documents.
- It defines the logical structure of documents and the way a document is accessed and manipulated.
- With the DOM, programmers can
  - create and build documents,
  - navigate their structure, and add, modify, or delete elements and content.
- The DOM is a W3C (World Wide Web Consortium) standard

Document Object Model

- **Note that the DOM is a language-independent API, meaning that it is not tied to Java, JavaScript, or any other language for implementation.**

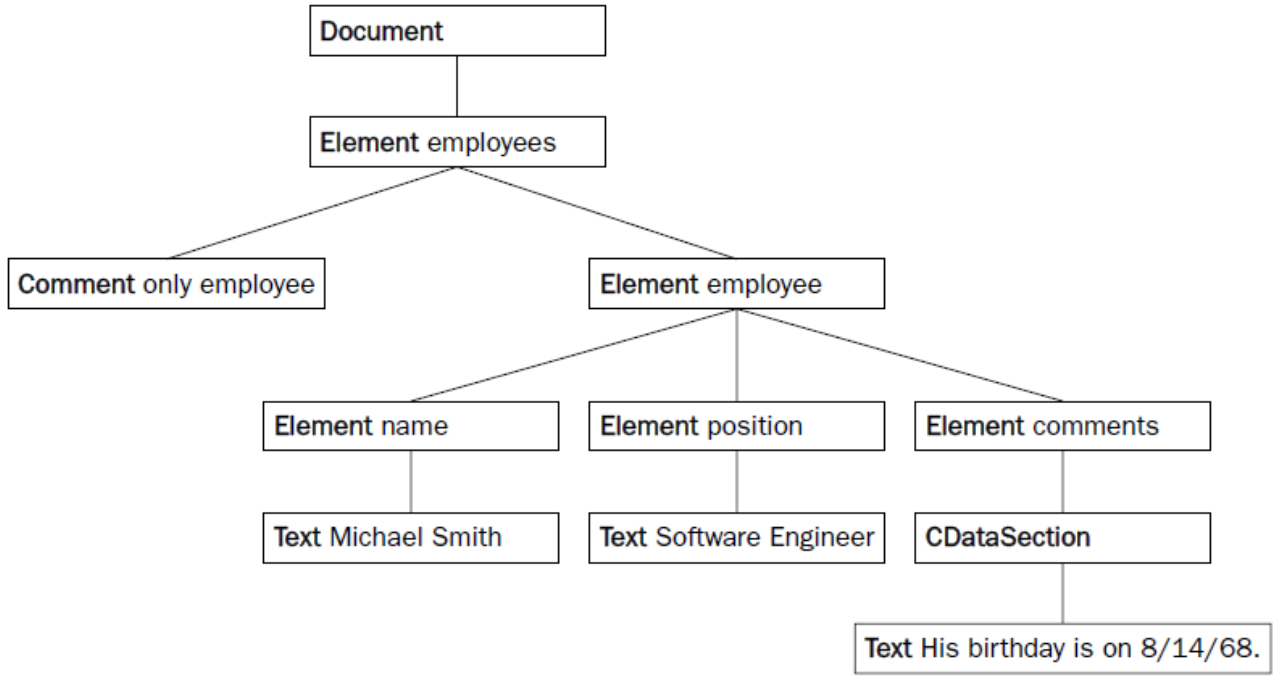
Document Object Model

- Document Object Model
  - Your web browser builds a *model* of the web page (the *document*) that includes all the *objects* in the page (tags, text, etc)
  - All of the properties, methods, and events available to the web developer for manipulating and creating web pages are organized into objects
  - Those objects are accessible via scripting languages in modern web browsers

Document Object Model

- A document is made up of a hierarchy of any number of these nodes. Consider the following XML code:

```
<?xml version="1.0"?>
<employees>
    <!-- only employee -->
    <employee>
        <name>Michael Smith</name>
        <position>Software Engineer</position>
        <comments><![CDATA[
          His birthday is on 8/14/68.
        ]]></comments>
    </employee>""
</employees>
```



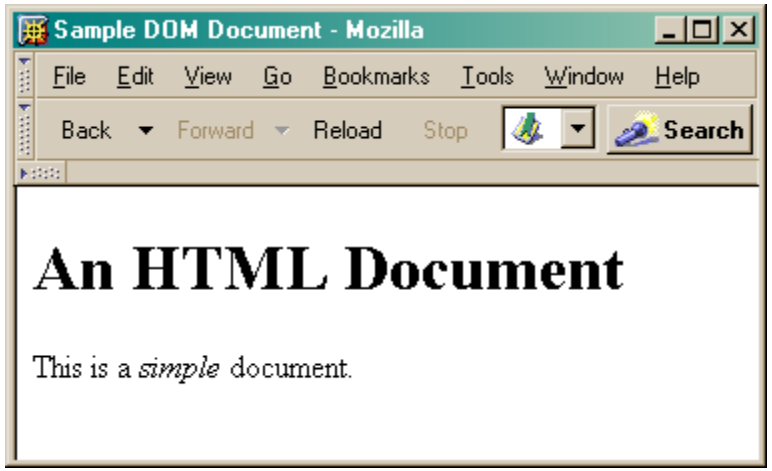
Document Object Model

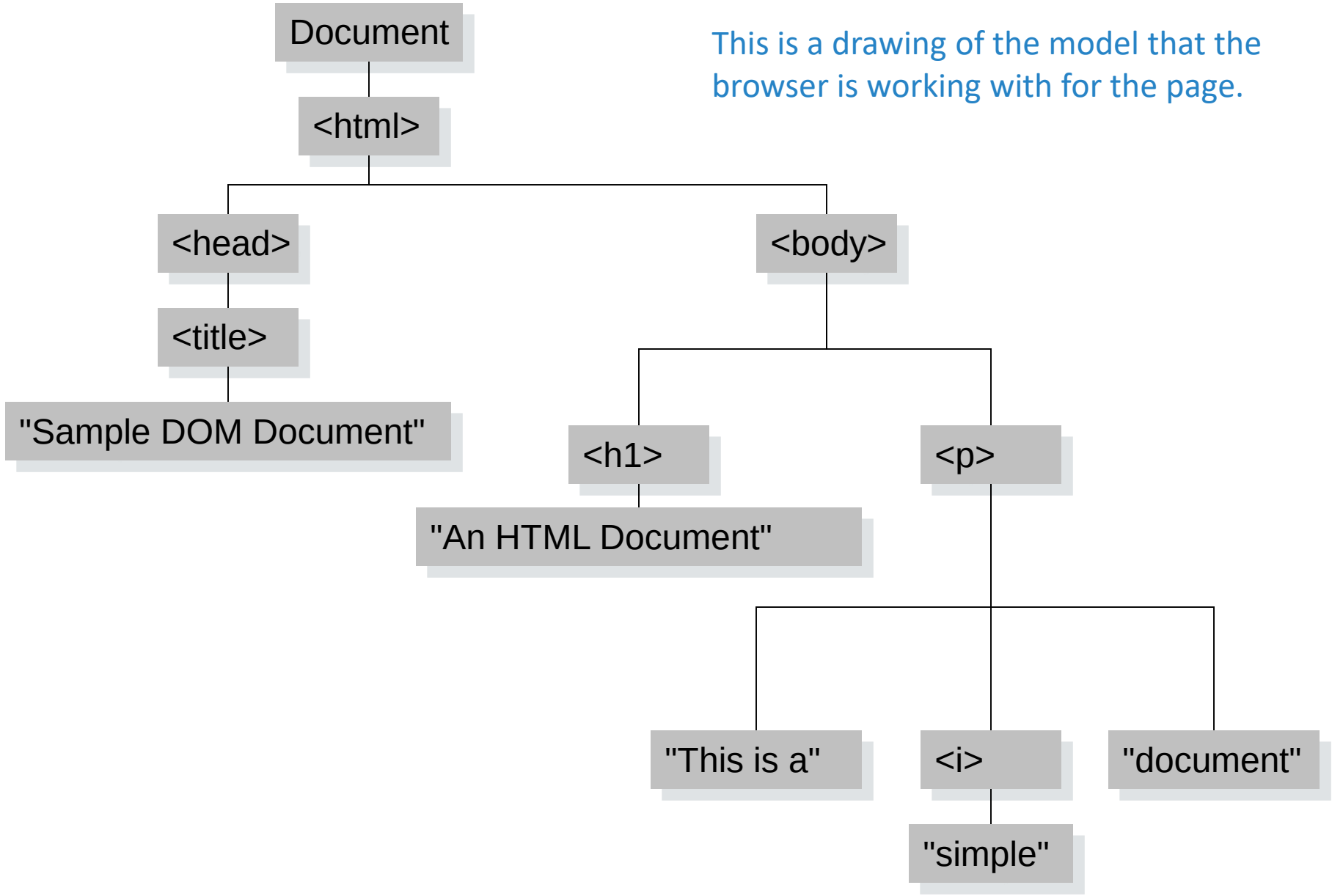
- In the HTML DOM (Document Object Model), everything is a **node**:
  - The document itself is a document node
  - All HTML elements are element nodes
  - All HTML attributes are attribute nodes
  - Text inside HTML elements are text nodes
  - Comments are comment nodes

This is what the browser reads

```
<html>
  <head>
    <title>Sample DOM Document</title>
  </head>
  <body>
    <h1>An HTML Document</h1>
    <p>This is a <i>simple</i> document.
  </body>
</html>
```

This is what the browser displays on screen.





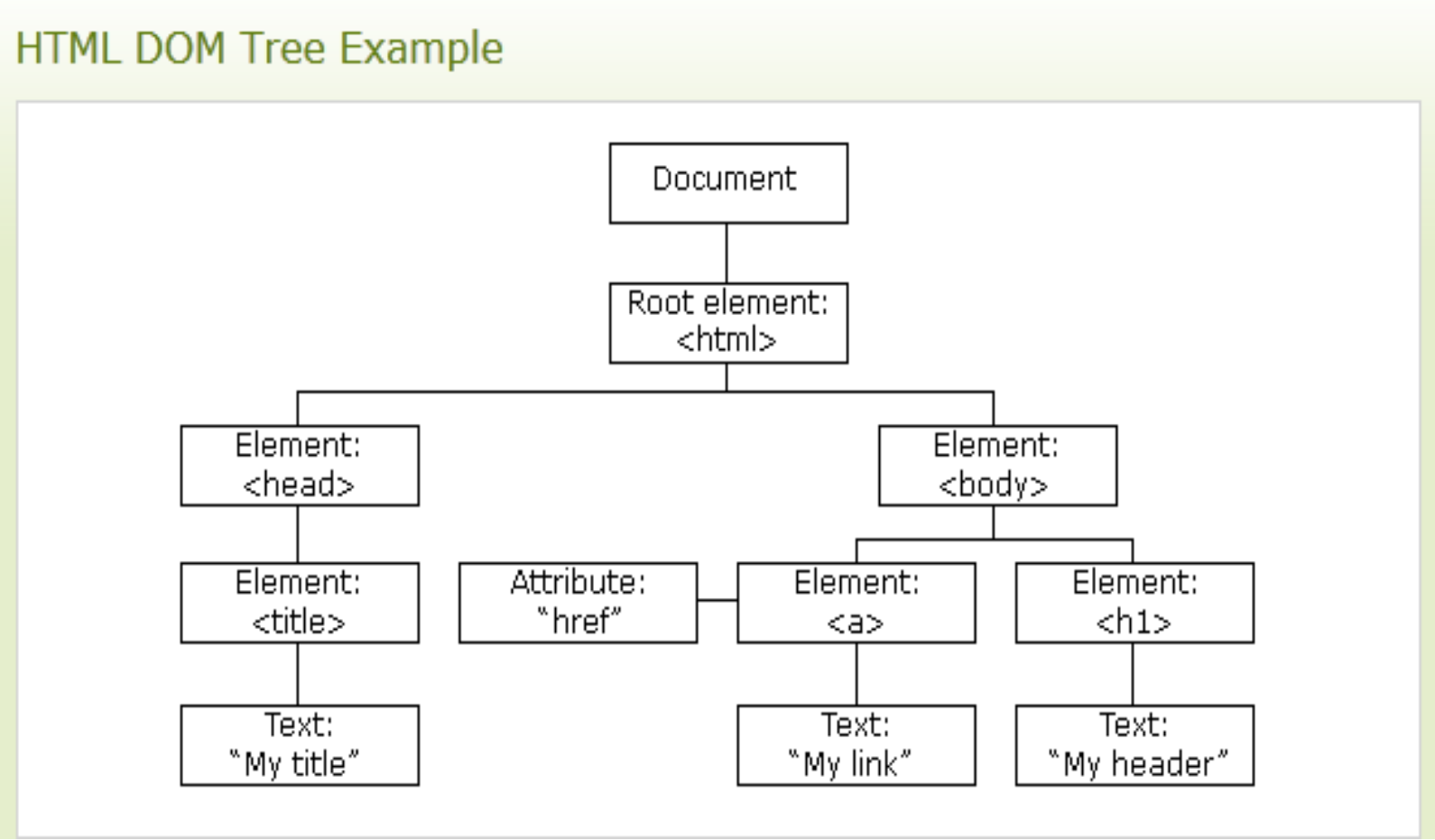
This is a drawing of the model that the browser is working with for the page.

Introduction

- The W3C DOM standard is separated into 3 different parts:

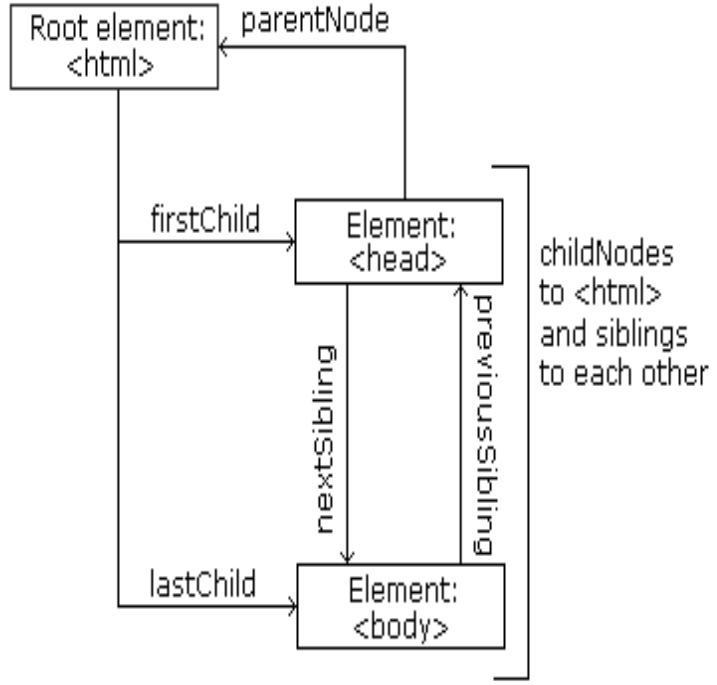
1. **Core DOM** - standard model for any structured document
2. **XML DOM** - standard model for XML documents
    1. XML DOM defines the **objects** and **properties** of all XML elements, and the **methods** to access them
3. **HTML DOM** - standard model for HTML documents
    1. The HTML DOM defines the **objects** and **properties** of all HTML elements, and the **methods** to access them.

HTML DOM


HTML DOM Tree Example
Document
Root element:
<html>
Element:
<head>
Element:
<body>
Element:
<title>
Attribute:
"href"
Element:
<a>
Element:
<h1>
Text:
"My title"
Text:
"My link"
Text:
"My header"

HTML DOM

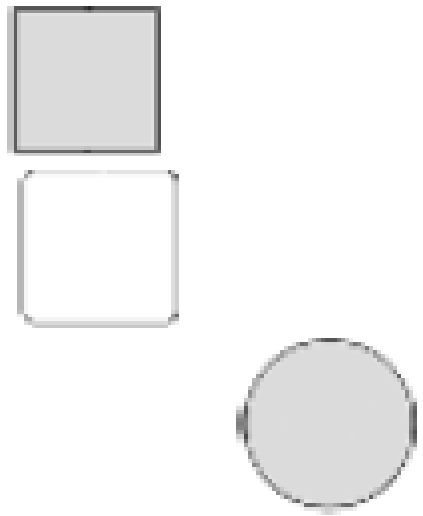
- In a node tree, the top node is called the root
- Every node has exactly one parent, except the root (which has no parent)
- A node can have any number of children
- Siblings are nodes with the same parent



Types of DOM nodes

```HTML
<p>
This is a paragraph of text with a
<a href="/path/page.html">link in it</a>.
</p>
```

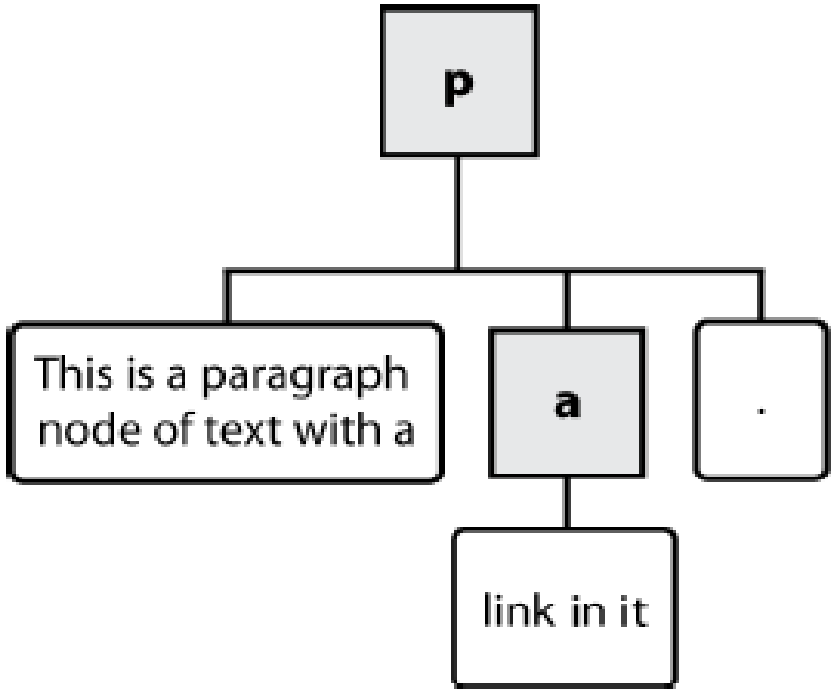
- element nodes (HTML tag)
  - can have children and/or attributes
- text nodes (text in a block element)
- attribute nodes (attribute/value pair)
  - text/attributes are children in an element node
  - cannot have children or attributes
  - not usually shown when drawing the DOM tree



Types of DOM nodes

```
<p>
This is a paragraph of text with a
<a href="/path/page.html">link in it</a>.
</p>
```

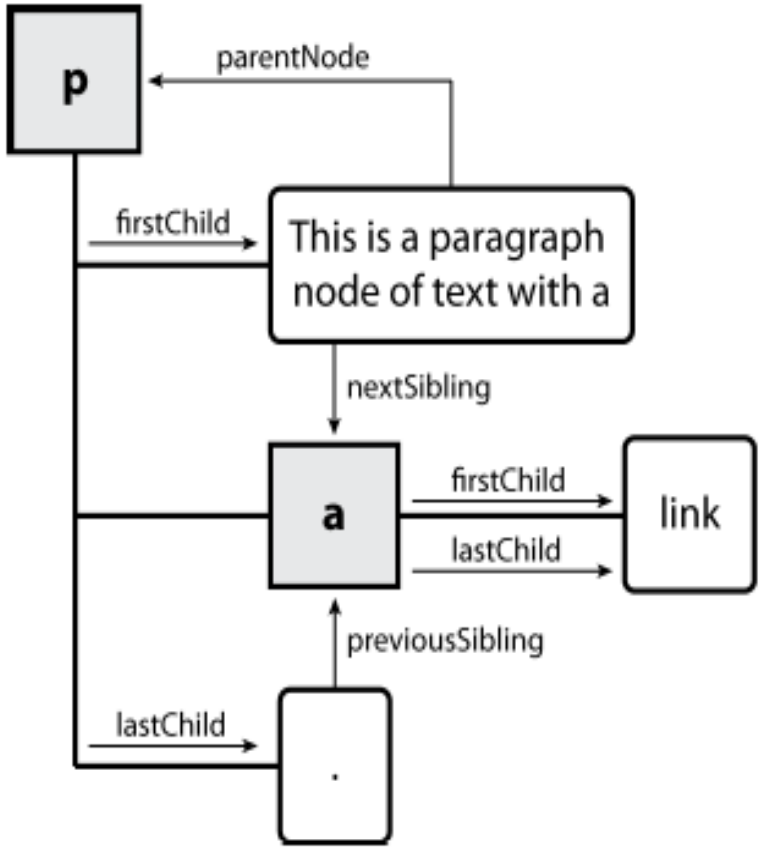
HTML



DOM tree traversal example

```
<p id="foo">This is a paragraph of text with a
<a href="/path/to/another/page.html">link</a>.</p>
```

HTML



Elements vs text nodes

```HTML
<div>
	<p>
		This is a paragraph of text with a
		<a href="page.html">link</a>.
	</p>
</div>
```

- Q: How many children does the div above have?
- A: 3
  - an element node representing the <p>
  - two text nodes representing "\n\t" (before/after the paragraph)
- Q: How many children does the paragraph have?

HTML DOM

- The HTML DOM can be accessed with JavaScript (and other programming languages).
- All HTML elements are defined as objects, and the programming interface is the object methods and object properties .
- A **method** is an action you can do (like add or modify an element).
- A **property** is a value that you can get or set (like the name or content of a node).

HTML DOM

- **Some commonly used HTML DOM methods:**
  - getElementById(*id*) - get the node (element) with a specified **id**
  - appendChild(*node*) - insert a new child node (element)
  - removeChild(*node*) - remove a child node (element)

HTML DOM

- **Some commonly used HTML DOM properties:**
  - innerHTML - the text value of a node (element)
  - parentNode - the parent node of a node (element)
  - childNodes - the child nodes of a node (element)
  - attributes - the attributes nodes of a node (element)

Method	Description
getElementById()	Returns the element that has an ID attribute with the a value
getElementsByTagName()	Returns a node list (collection/array of nodes) containing all elements with a specified tag name
getElementsByClassName()	Returns a node list containing all elements with a specified class
appendChild()	Adds a new child node to a specified node
removeChild()	Removes a child node
replaceChild()	Replaces a child node
insertBefore()	Inserts a new child node before a specified child node
createAttribute()	Creates an Attribute node
createElement()	Creates an Element node
createTextNode()	Creates a Text node
getAttribute()	Returns the specified attribute value
setAttribute()	Sets or changes the specified attribute, to the specified value

HTML DOM - The innerHTML Property

```
<html>
<body>

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

<script>
var txt=document.getElementById("intro").innerHTML;
document.write(txt);
</script>

</body>
</html>
```

HTML DOM - Modifying

- Changing HTML content
- Changing CSS styles
- Changing HTML attributes
- Creating new HTML elements
- Removing existent HTML elements
- Changing event(handlers)

HTML DOM

- HTML DOM allows JavaScript to react to HTML events.
- HTML DOM events allow **JavaScript** to register different **event handlers** on elements in an HTML document.
- **Events** are normally used in combination with **functions**, and the function will not be executed before the event occurs (such as when a user clicks a button).

Event	Description
onclick	The event occurs when the user clicks on an element
oncontextmenu	The event occurs when the user right-clicks on an element to open a context menu
ondblclick	The event occurs when the user double-clicks on an element
onmousedown	The event occurs when the user presses a mouse button over an element
onmouseenter	The event occurs when the pointer is moved onto an element
onmouseleave	The event occurs when the pointer is moved out of an element
onmousemove	The event occurs when the pointer is moving while it is over an element
onmouseover	The event occurs when the pointer is moved onto an element, or onto one of its children
onmouseout	The event occurs when a user moves the mouse pointer out of an element, or out of one of its children
onmouseup	The event occurs when a user releases a mouse button over an element

Example-1

- **Adding Some Text To A Page**

There are five steps:

1. Create a new Element
2. Create new Text
3. Append the new Text to the new Element
4. Find an existing Element
5. Append the new Element to the existing Element

1. Create New Element Node

- Let us, say create a new <p> tag (element) so that we can attach some text to it
- For convenience, we can put the new object into a variable

```
var newNode;
newNode = document.createElement(“p”)
```

2. Create a Text Node

- Next, create a text node
- Again, for convenience, we can put the new text node into a variable

```
var newText;
newText = document.createTextNode(“Some text.”)
```

3. Attach the New Text Node to the New Element

- To put the text into the page, we have to attach the text node to the new HTML element:

```
newNode.appendChild(newText);
```

4.Find an Existing Element

- The new element with our text node attached to it is still floating around in a Javascript world.
- We need to find an existing element so that we can attach it
- For convenience, we shall put this existing element into a variable

```
var docElement;
docElement = document.getElementById(“thisLocation”);
```

5. Append the New Element to the Existing Element

- To insert our text into the page, we now have to append the new element to the existing element

```
docElement.appendChild(newNode);
```

Hands On: Try out this code

DOM1.html

```
<head>
<script language="javascript" type="text/javascript">
var myText;
myText = "This new text to be added to the page dynamically.";
function addText(location) {
    var newNode;
    var newText;
    var docElement;
    newNode = document.createElement("p");
    newText = document.createTextNode(myText);
    newNode.appendChild(newText);
    docElement = document.getElementById(location);
    docElement.appendChild(newNode);
}
</script>
</head>
<body>
<p><a href="#" onclick="addText('thisLocation');">Click to add new text to the page</a></p>
<p id="thisLocation">New text will appear below here</p>
<p>Some further text in the page</p>
</body>
```

Output

Click to add new text to the page

New text will appear below here

Some further text in the page

Click to add new text to the page

New text will appear below here

This is new text to be added to the page dynamically.

Some further text in the page

Example – 2 (Remove a Node)

- To remove a node, we use the element method removeChild (*name of node to be removed*)
- For example:

```
function remText(location) {
    var docElement;
    docElement = document.getElementById(location);
    docElement.removeChild(docElement.lastChild);
}
```

Hands On

Modify your HTML page so that the user can click on some text to remove the text that was inserted

Hands On: Try out this code

```
<head>
<script language="javascript" type="text/javascript">
var myText;
myText = "This is new text to be added to the page dynamically.";
function addText(location) {
Same code as mentioned in previous example-1
}
function remText(location) {
      var docElement;
      docElement = document.getElementById(location);
      docElement.removeChild(docElement.lastChild);
}
</script>
</head>
<body>
<p><a href="#" onclick="addText('thisLocation');">Click to add new text to the page</a></p>
<p id="thisLocation">New text will appear below here</p>
<p><a href="#" onclick="remText('thisLocation');">Click to Remove The Last Added Paragraph</p>
<p>Some further text in the page</p>
</body>
```

getElementsByTagName()

- getElementById() allows you to work with elements by their individual id but often you will want to work with a group of elements

- getElementsByTagName() allows you to work with groups of elements. This method returns an array

getElementsByTagName()

- **Hands On**
- Open the file DOM1.html

- Insert this code at the bottom of the document body:

```
<script language="javascript" type="text/javascript">
theseElements = new Array;
theseElements = document.getElementsByTagName("li");
alert(theseElements.length);
</script>
```

Core interfaces in the DOM

- The following is a brief list of common APIs in web and XML page scripting using the DOM.
- **document.querySelector(selector)**
- **document.querySelectorAll(name)**
- document.createElement(name)
- parentNode.appendChild(node)
- element.innerHTML
- element.style.left
- element.setAttribute()
- element.getAttribute()
- element.addEventListener()
- window.content
- GlobalEventHandlers/onload
- window.scrollTo()