THE DOM

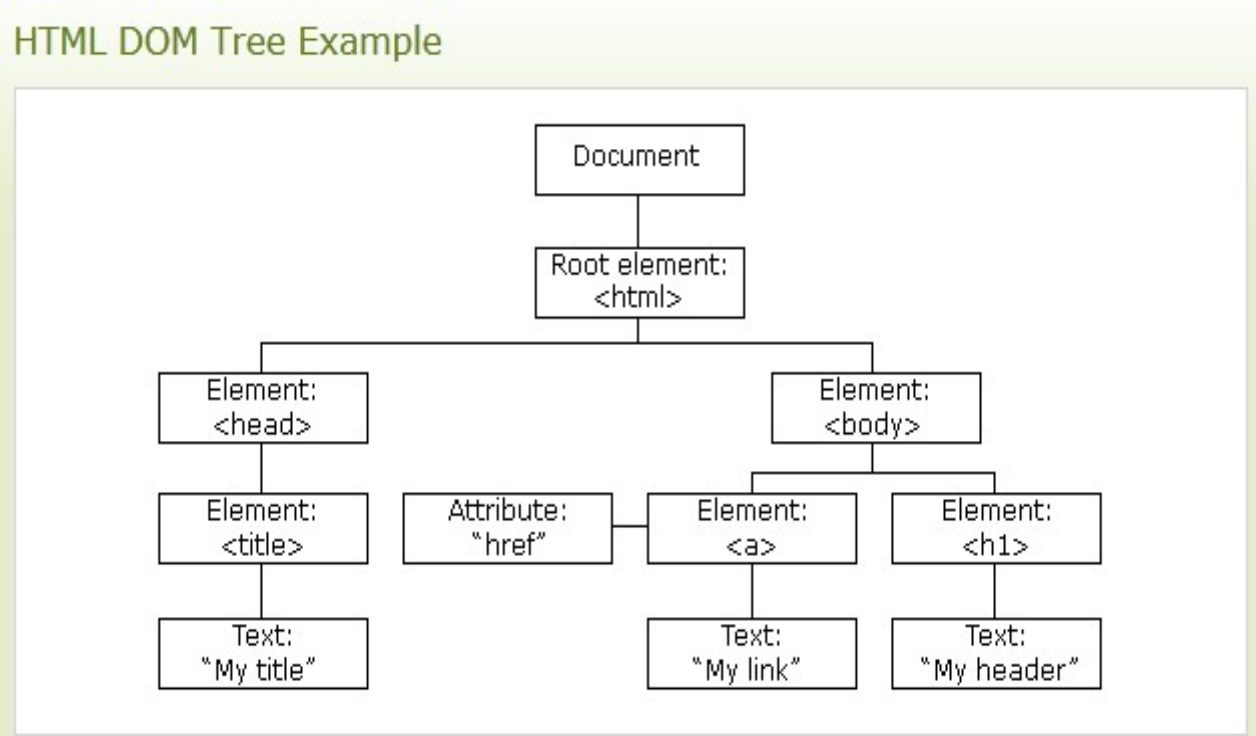
What is the DOM?

- 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.
- The HTML DOM defines a standard way for accessing and manipulating HTML and XML documents.
- 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

The W3C DOM standard is separated into 3 different parts:

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

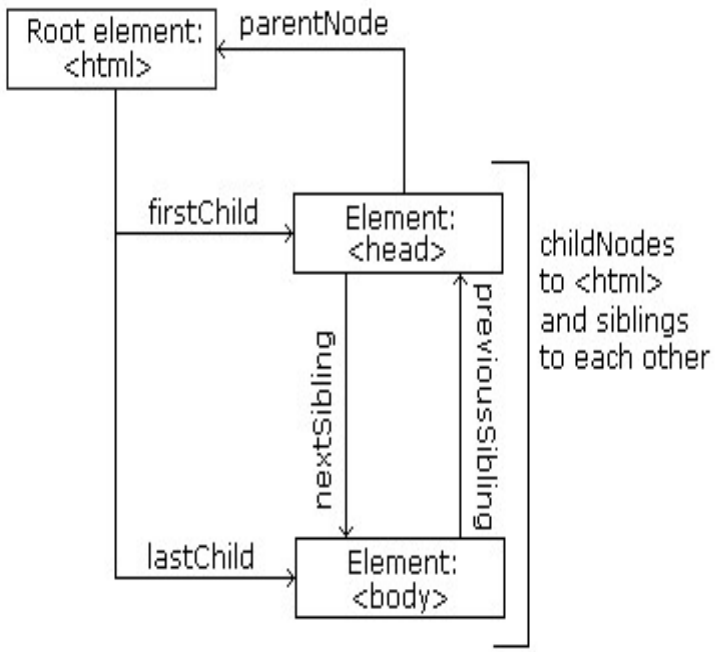
- **In the HTML DOM, everything is a node. The DOM is HTML viewed as a node tree.**



HTML DOM

Node Parents, Children, and Siblings

- 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



HTML DOM - Modifying

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

Programming Interface

- 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 cont.

- **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)
- **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

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

Adding Some Text To Existing <p> (Cont..)

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

Adding <p> and text to a Page (Cont..)

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

Add button dynamically

```
<head>
<script language="javascript" type="text/javascript">
function add(t) {
        var element1 = document.createElement("button");
        element1.value = t;
        element1.name = "b";
        element1.setAttribute("style","font-size:50px;background-color:red;");
        element1.onclick = function() {     alert("new button created");  };
        var f1 = document.getElementById("f");
        f1.appendChild(element1);
}</script>
</head><body><input type="button" id="btnAdd" value="Add Button"
onClick="add('ADD BUTTON')"><p id="f">Fields:</p>
</body>
```

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);
}
```

Try:

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

Remove a Node- Example

```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
var parent = document.getElementById("div1");
var child = document.getElementById("p1");
parent.removeChild(child);
</script>
```

DOM Events

- HTML DOM 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).
- When an event occurs, a code segment is executed in response to a specific event is called “event handler”.

Types of Events

- Keyboard
- Frame
- Drag and drop
- Form
- Clipboard
- Media
- Transition
- Server
- Touch

Event Handlers

Event	Value	Description
onchange	script	Script runs when the element changes
onsubmit	script	Script runs when the form is submitted
onreset	script	Script runs when the form is reset
onselect	script	Script runs when the element is selected
onblur	script	Script runs when the element loses focus
onfocus	script	Script runs when the element gets focus
onkeydown	script	Script runs when key is pressed
onkeypress	script	Script runs when key is pressed and released
onkeyup	script	Script runs when key is released
onclick	script	Script runs when a mouse click
ondblclick	script	Script runs when a mouse double-click
onmousedown	script	Script runs when mouse button is pressed
onmousemove	script	Script runs when mouse pointer moves
onmouseout	script	Script runs when mouse pointer moves out of an element
onmouseover	script	Script runs when mouse pointer moves over an element
onmouseup	script	Script runs when mouse button is released

Event	Description	DOM
onclick	The event occurs when the user clicks on an element	2
oncontextmenu	The event occurs when the user right-clicks on an element to open a context menu	3
ondblclick	The event occurs when the user double-clicks on an element	2
onmousedown	The event occurs when the user presses a mouse button over an element	2
onmouseenter	The event occurs when the pointer is moved onto an element	2
onmouseleave	The event occurs when the pointer is moved out of an element	2
onmousemove	The event occurs when the pointer is moving while it is over an element	2
onmouseover	The event occurs when the pointer is moved onto an element, or onto one of its children	2
onmouseout	The event occurs when a user moves the mouse pointer out of an element, or out of one of its children	2
onmouseup	The event occurs when a user releases a mouse button over an element	

EventListener

The addEventListener() method attaches an event handler to the specified element.

```
<button id="myBtn">Try it</button>

<p id="demo"></p>

<script>

document.getElementById("myBtn").addEventListener("click", displayDate);

function displayDate() {

  document.getElementById("demo").innerHTML = Date();

}</script>
```

EventListener- Example

- Add an event listener that fires when a user resizes the window:

```
window.addEventListener("resize", function(){
   document.getElementById("demo").innerHTML = “hello”;
});
```

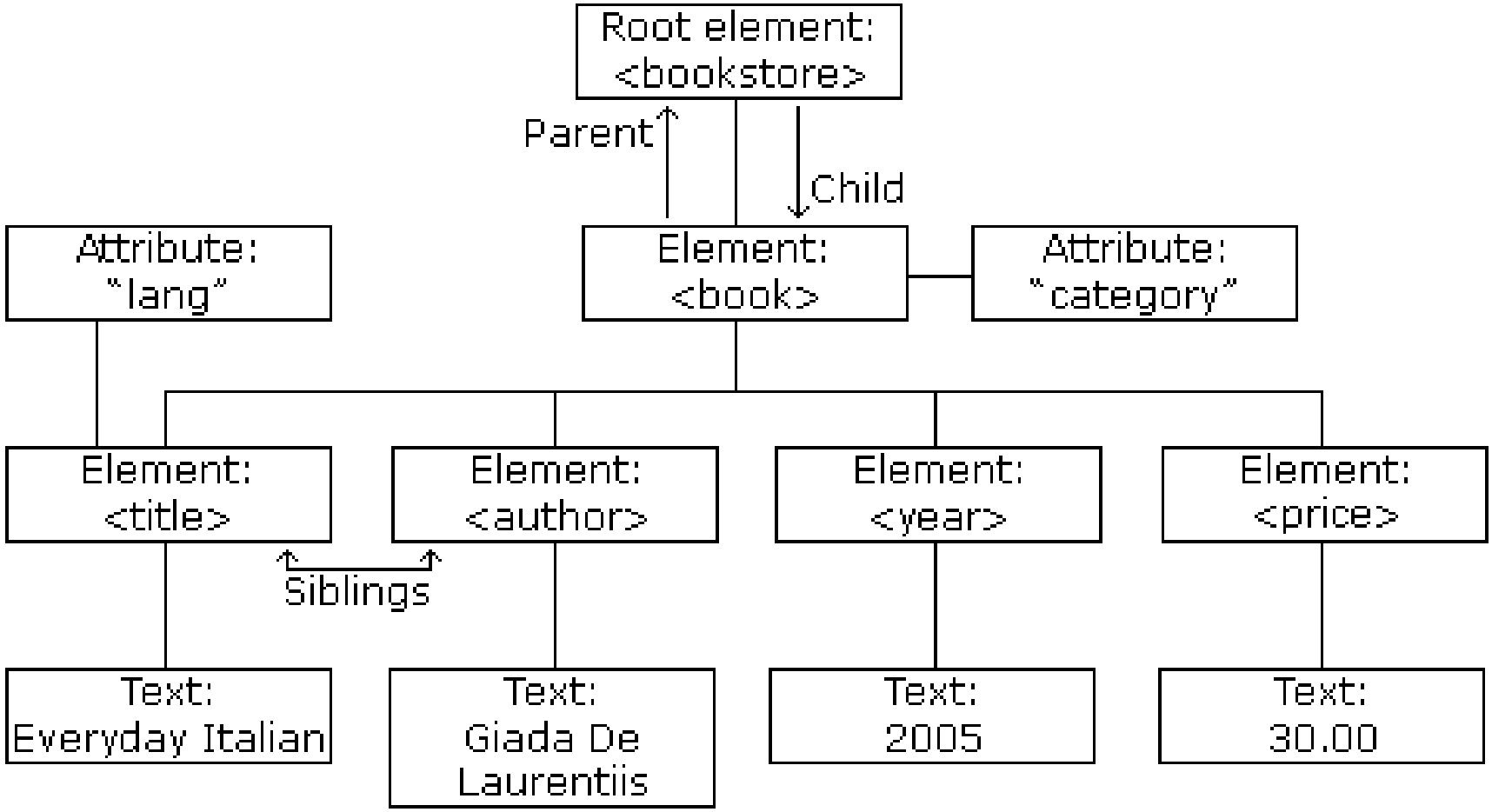
- The removeEventListener() method removes event handlers that have been attached with the addEventListener() method:
- **Example**
  - *element*.removeEventListener("mousemove", myFunction);

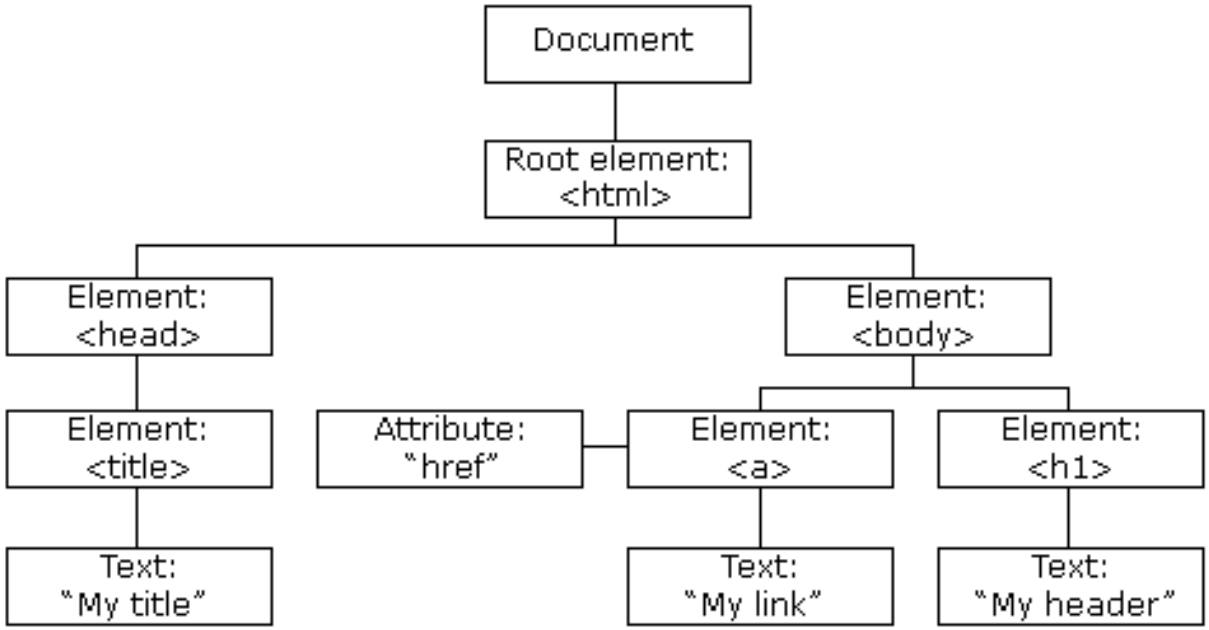
XML DOM

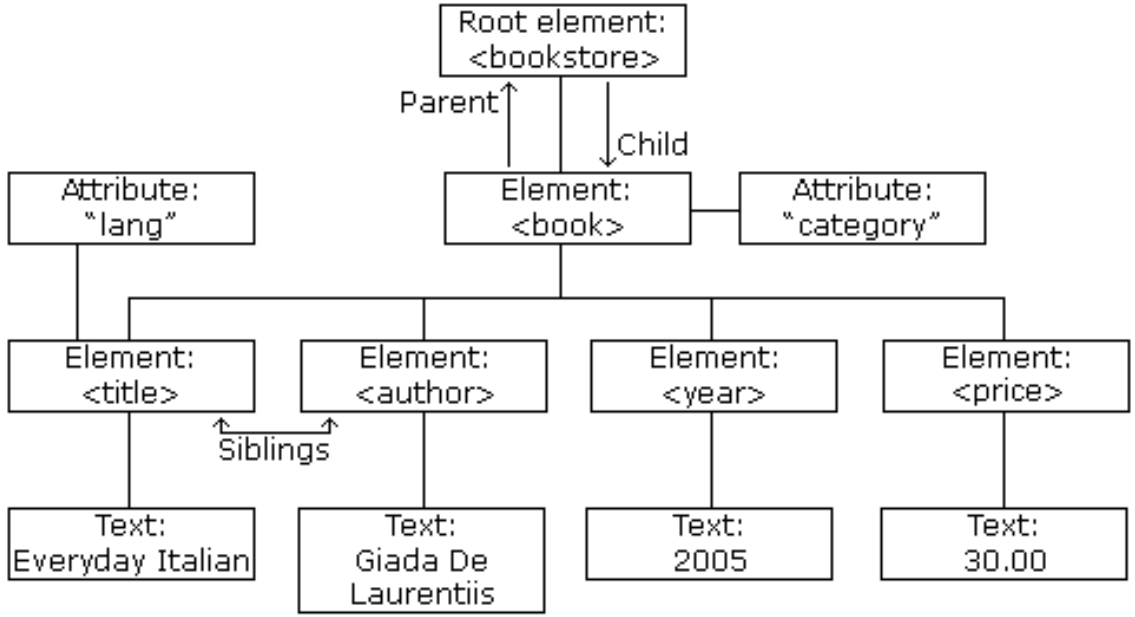
- The entire document is a document node
- Every XML element is an element node
- The text in the XML elements are text nodes
- Every attribute is an attribute node
- Comments are comment nodes

What is the XML DOM?

- A standard object model for XML
- A standard programming interface for XML
- Platform- and language-independent
- A W3C standard
- The XML DOM is a standard for how to get, change, add, or delete XML elements.


Root element:
<bookstore>
Parent
Child
Attribute:
"lang"
Element:
<book>
Attribute:
"category"
Element:
<title>
Element:
<author>
Element:
<year>
Element:
<price>
Siblings
Text:
Everyday Italian
Text:
Giada De
Laurentiis
Text:
2005
Text:
30.00


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"


Root element:
<bookstore>
Parent
Child
Attribute:
"lang"
Element:
<book>
Attribute:
"category"
Element:
<title>
Element:
<author>
Element:
<year>
Element:
<price>
Siblings
Text:
Everyday Italian
Text:
Giada De
Laurentiis
Text:
2005
Text:
30.00