IT4409: Web Technologies and e-Services

2020-2

Document Object Model (DOM)

Instructor: Dr. Thanh-Chung Dao
Slides by Dr. Binh Minh Nguyen

Department of Information Systems
School of Information and Communication Technology
Hanoi University of Science and Technology

Outline

- Introduction of DOM
- Overview of DOM
- DOM Examples
- How the DOM Really works?
- Advantages and Disadvantages
- DOM or SAX
- Summary

Before going to the DOM

- **HTML** – How to Display the Data in the Webpage.
- **XML** – How to Describe the Data .
- **DHTML** - How to Add Movement or Animation to an HTML Document.
- **JAVASCRIPT** - How to make Web Content Dynamic.

World Wide Web Consortium-W3C

World Wide Web Consortium-W3C

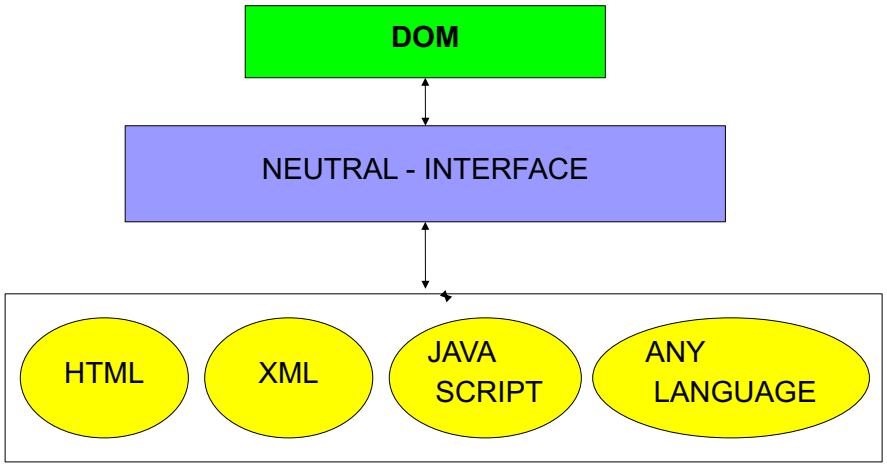
- To Promote Open Standard For world wide web.
- W3C is a vendor Organization.
- Main Vendors are Netscape and Microsoft.
- Some W3C Standards are HTTP,HTML,XML,CSS.
- DOM is also Recommend by W3C.

Overview

The Document Object Model (DOM) is an API that allows programs to interact with HTML (or XML) documents

- In typical browsers, the JavaScript version of the API is provided
- W3C recommendations define standard DOM

What is the DOM?



7

7

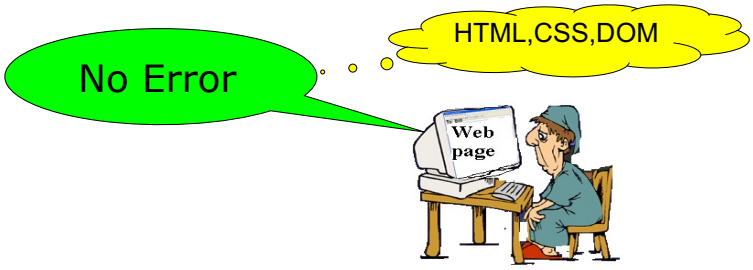
W3C

8

8

Five Basic Levels Of W3C :

Recommendation:- It is the Final outcome from W3C.All the Web functions are working properly.

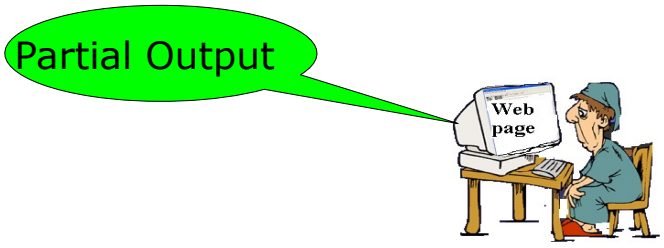


9

9

Proposed Recommendation:-

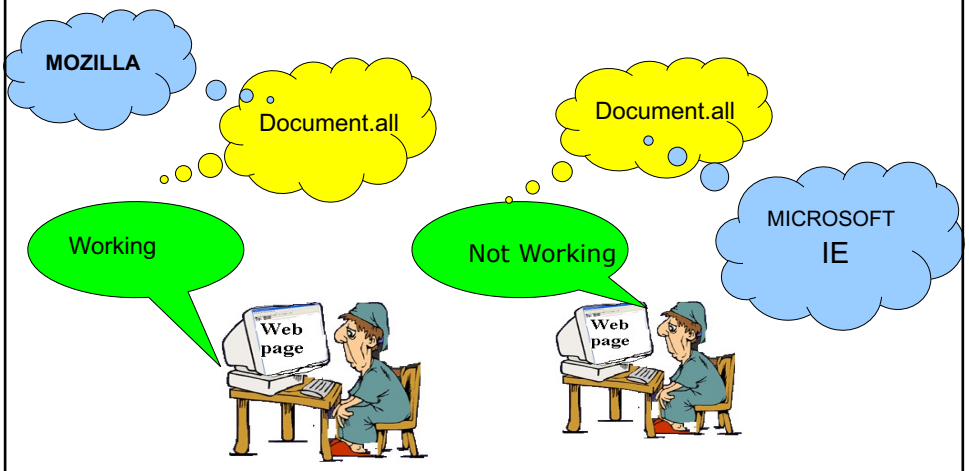
- In this layer the work is mostly complete .But some minor changes is occur.



10

10

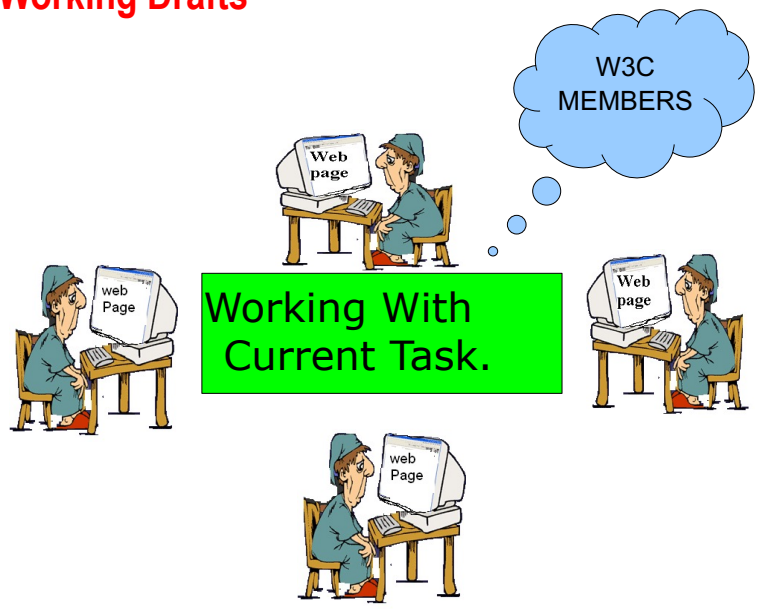
Candidate Recommendation:-



11

11

Working Drafts



12

12

Status Of The DOM

13

DOM Level 1:

W3C recommendation, 1 Oct. 1998.

Interfaces for representing an XML and HTML document.

1)**Document**

2)Node

3)Attr

4)Element

5)and Text interfaces.

14

DOM Level 2:

W3C recommendation, 13 Nov. 2000.

It contains six different specifications:

1)DOM2 Core
2)Views
3)Events
4)Style
5)Traversal and Range
6)and the DOM2 HTML.

15

DOM Level 3:

W3C candidate recommendation, 7 Nov. 2003

It contains five different specifications:

1)DOM3 Core
2)Load and Save
3)Validation
4)Events
5)and XPath

16

Overview of DOM

DOM Introduction

Example: "Rollover" effect

Cursor not over image

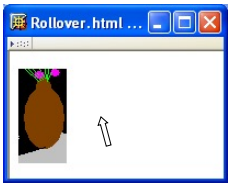
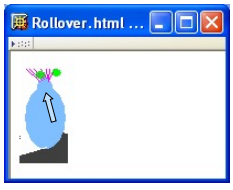


Image changes when cursor moves over



DOM Introduction

```
<!DOCTYPE html
       PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
    <title>Rollover.html</title>
    <script type="text/javascript" src="rollover.js">
    </script>
    <meta http-equiv="Content-Script-Type" content="text/javascript" />
  </head>
  <body>
    <p>
      <img id="img1" src="CFP2.png" alt="flower pot"
        height="86" width="44"
        onmouseover="show('img1', 'CFP22.png');"
        onmouseout="show('img1', 'CFP2.png');" />
    </p>
  </body>
</html>
```


DOM Introduction

```
<!DOCTYPE html
       PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
    <title>Rollover.html</title>
    <script type="text/javascript" src="rollover.js">
    </script>
    <meta http-equiv="Content-Script-Type" content="text/javascript" />
  </head>
  <body>
    <p>
      <img id="img1" src="CFP2.png" alt="flower pot"
        height="86" width="44"
        onmouseover="show('img1', 'CFP22.png');"
        onmouseout="show('img1', 'CFP2.png');" />
    </p>
  </body>
</html>
```

Import JavaScript code

DOM Introduction

```
<!DOCTYPE html
       PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
    <title>Rollover.html</title>
    <script type="text/javascript" src="rollover.js">
    </script>
    <meta http-equiv="Content-Script-Type" content="text/javascript" />
  </head>
  <body>
    <p>
      <img id="img1" src="CFP2.png" alt="flower pot"
        height="86" width="44"
        onmouseover="show('img1', 'CFP22.png');"
        onmouseout="show('img1', 'CFP2.png');" />
    </p>
  </body>
</html>
```

Default language for scripts specified as attribute values

DOM Introduction

```
<!DOCTYPE html
       PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
    <title>Rollover.html</title>
    <script type="text/javascript" src="rollover.js">
    </script>
    <meta http-equiv="Content-Script-Type" content="text/javascript" />
  </head>
  <body>
    <p>
      <img id="img1" src="CFP2.png" alt="flower pot"
        height="86" width="44"
        onmouseover="show('img1', 'CFP22.png');"
        onmouseout="show('img1', 'CFP2.png');" />
    </p>
  </body>
</html>
```

Calls to JavaScript `show()` function when mouse moves over/away from image

DOM Introduction

```
<!DOCTYPE html
       PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
    <title>Rollover.html</title>
    <script type="text/javascript" src="rollover.js">
    </script>
    <meta http-equiv="Content-Script-Type" content="text/javascript" />
  </head>
  <body>
    <p>
      <img id="img1" src="CFP2.png" alt="flower pot"
        height="86" width="44"
        onmouseover="show('img1', 'CFP22.png');"
        onmouseout="show('img1', 'CFP2.png');" />
    </p>
  </body>
</html>
```

Notice that `id` of image is first argument to `show()`

DOM Introduction

```
// rollover.js

function show(eltId, URL) {
  var elt = window.document.getElementById(eltId);
  elt.setAttribute("src", URL);
  return;
}
```

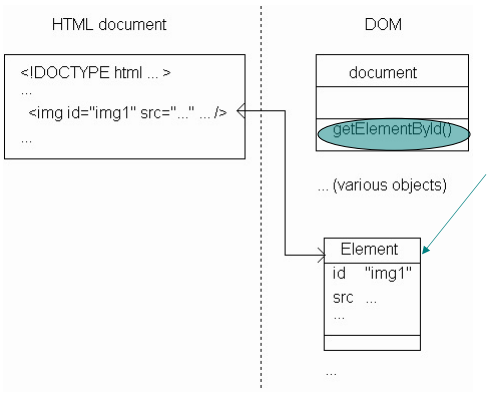
DOM Introduction

```
// rollover.js

function show(eltId, URL) {
  var elt = window.document.getElementById(eltId);
  elt.setAttribute("src", URL);
  return;
}
```

DOM method returning Object

DOM Introduction



Returns instance of `Element` (DOM-defined host object) representing HTML element with given `id`

DOM Introduction

```
// rollover.js

function show(eltId, URL) {
  var elt = window.document.getElementById(eltId);
  elt.setAttribute("src", URL);
  return;
}
```

Method inherited by `Element` instances for setting value of an attribute

DOM Introduction

```
// rollover.js

function show(eltId, URL) {
  var elt = window.document.getElementById(eltId);
  elt.setAttribute("src", URL);
  return;
}
```

Effect: `src` attribute of HTML element with specified `eltId` is changed to specified URL

DOM Introduction

```
<!DOCTYPE html
       PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
    <title>Rollover.html</title>
    <script type="text/javascript" src="rollover.js">
    </script>
    <meta http-equiv="Content-Script-Type" content="text/javascript" />
  </head>
  <body>
    <p>
      <img id="img1" src="CFP2.png" alt="flower pot"
        height="86" width="44"
        onmouseover="show('img1', 'CFP22.png');"
        onmouseout="show('img1', 'CFP2.png');" />
    </p>
  </body>
</html>
```

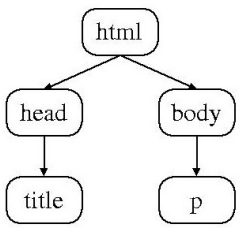
Image `src` changed to `CFP22.png` when mouse is over image, `CFP2.png` when leaves

29

Document Tree

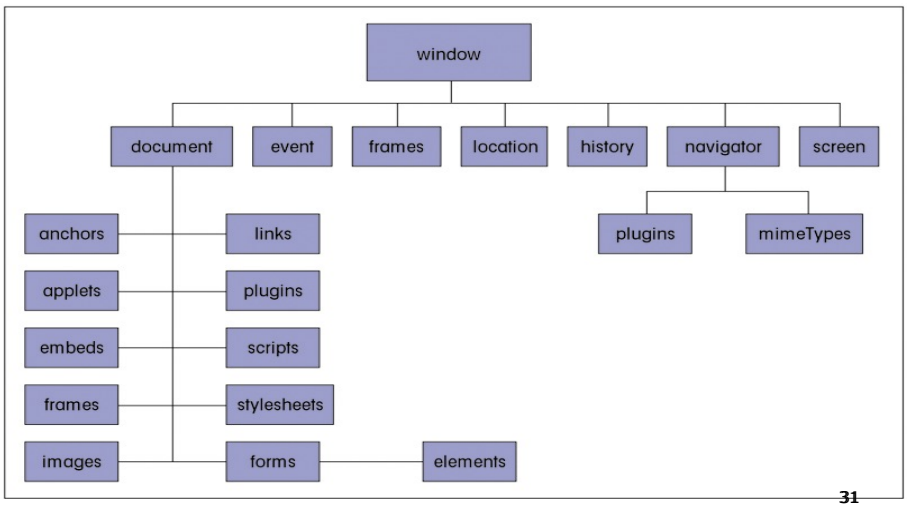
Recall that HTML document elements form a tree structure, *e.g.*,

DOM allows scripts to access and modify the document tree



30

The Document Tree Example



31

31

Referencing Objects-Each Object is Identified by Object Name.

Object Name	Description
window	The browser window
document	The Web document displayed in the window
document.body	The body of the Web document displayed in the browser window
event	Events or actions occurring within the browser window
history	The list of previously visited Web sites within the browser window
location	The URL of the document currently displayed in the browser window
navigator	The browser itself
screen	The screen displaying the document

32

32

How To Use Referencing Objects

Object Names

- General form is *object1.object2.object3..*

TO Access The History

```
window.history
```

To Access The Body

```
document.body
```

33

The DOM structure model

- It is a Hierarchy of *Node objects*

- node object

Element

Attribute

etc

34

Document Tree: Node

Example HTML document

```
<body>
  <p>
    Text within a "p" element.
  </p>
  <ol>
    <li>First element of ordered list.</li>
    <li>Second element.</li>
  </ol>
  <!-- Call function producing an outline of this document's
       element tree -->
  <form action="">
    <p><input type="button" name="button" value="Click to see outline"
                  onclick="window.alert(treeOutline());" /></p>
  </form>
</body>
```

Function will use `Node` methods and properties to produce string representing `Element` tree

Document Tree: Node

String produced by `TreeOutline()`:

An Example of DOM XML

```
<?xml version = "1.0"?>
<message from = "Paul" to = "Name">
   <body>Hello!</body>
</message>
```

Node is created for **message** element:

- **message** element has child element: **body**.
- **body** element has **Text** "Hello!"
- Attributes: **from** and **to** also are nodes in DOM tree.

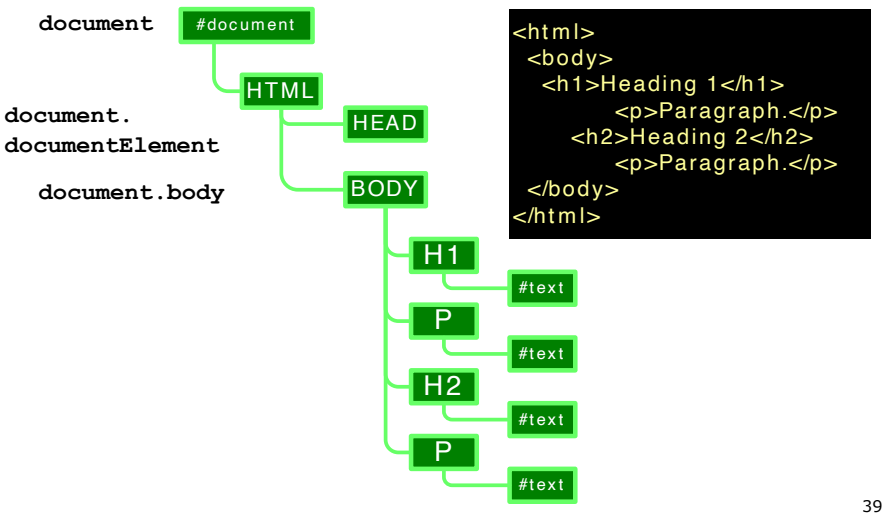
The DOM Interface

- The DOM has many interfaces to handle various node objects.
- Every interface has its "Attributes" and "Methods".
  - Compare with Object Oriented Programming (OOP).

DOM	OOP
Interface	Object Class
Attribute	Property
Method	Method

38

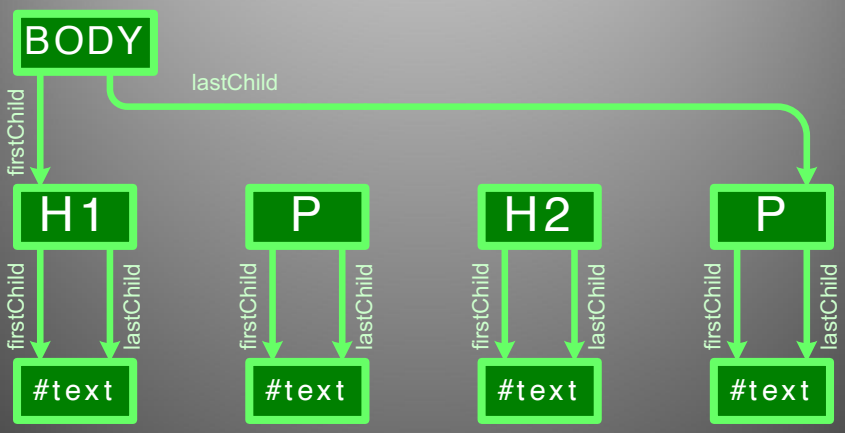
Document Tree Structure



39

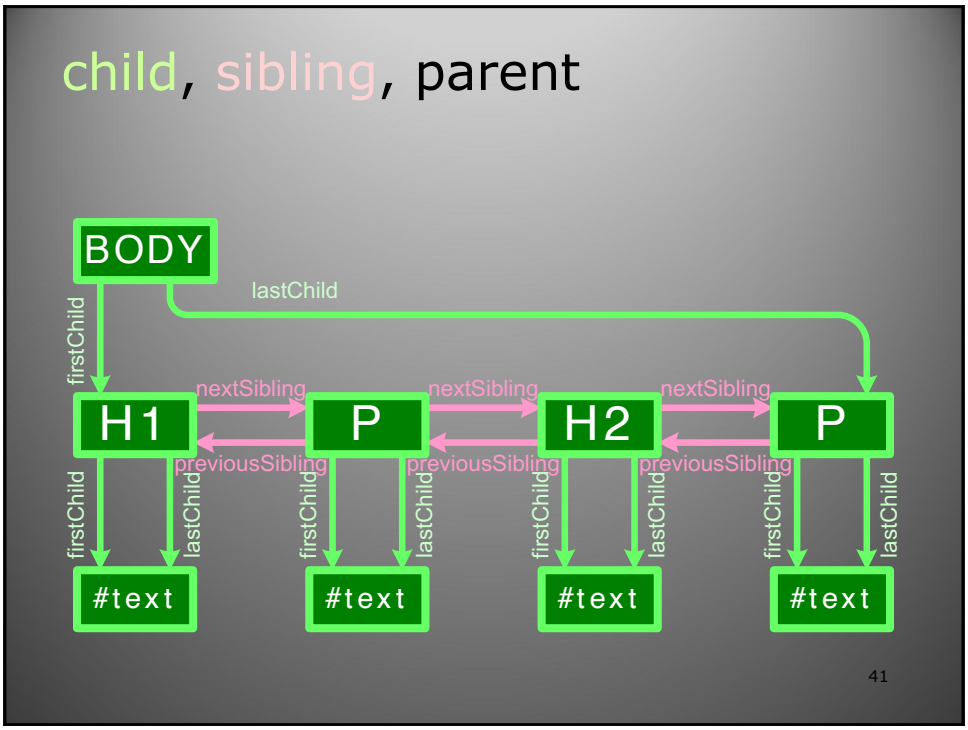
39

child, sibling, parent

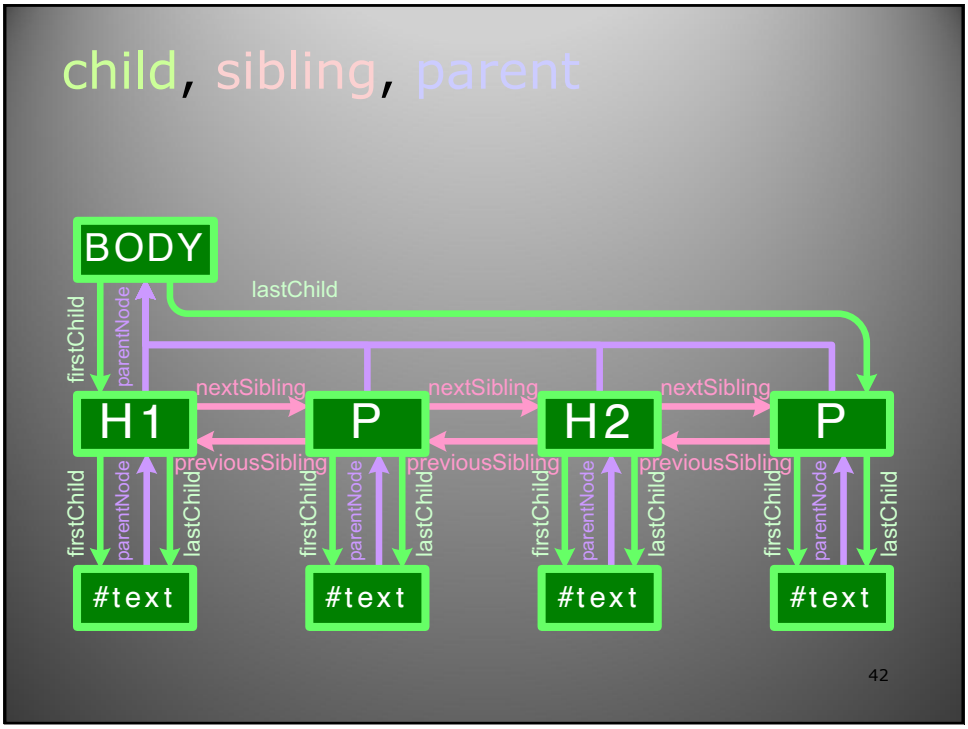


40

40


child, sibling, parent
BODY
lastChild
firstChild
nextSibling
nextSibling
nextSibling
H1
P
H2
P
previousSibling
previousSibling
previousSibling
firstChild
lastChild
firstChild
lastChild
firstChild
lastChild
firstChild
lastChild
#text
#text
#text
#text
41

41

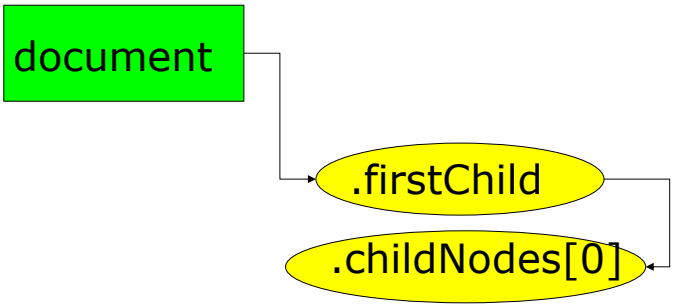

child, sibling, parent
BODY
lastChild
firstChild
parentNode
nextSibling
nextSibling
nextSibling
H1
P
H2
P
previousSibling
previousSibling
previousSibling
firstChild
parentNode
lastChild
firstChild
parentNode
lastChild
firstChild
parentNode
lastChild
firstChild
parentNode
lastChild
#text
#text
#text
#text
42

42

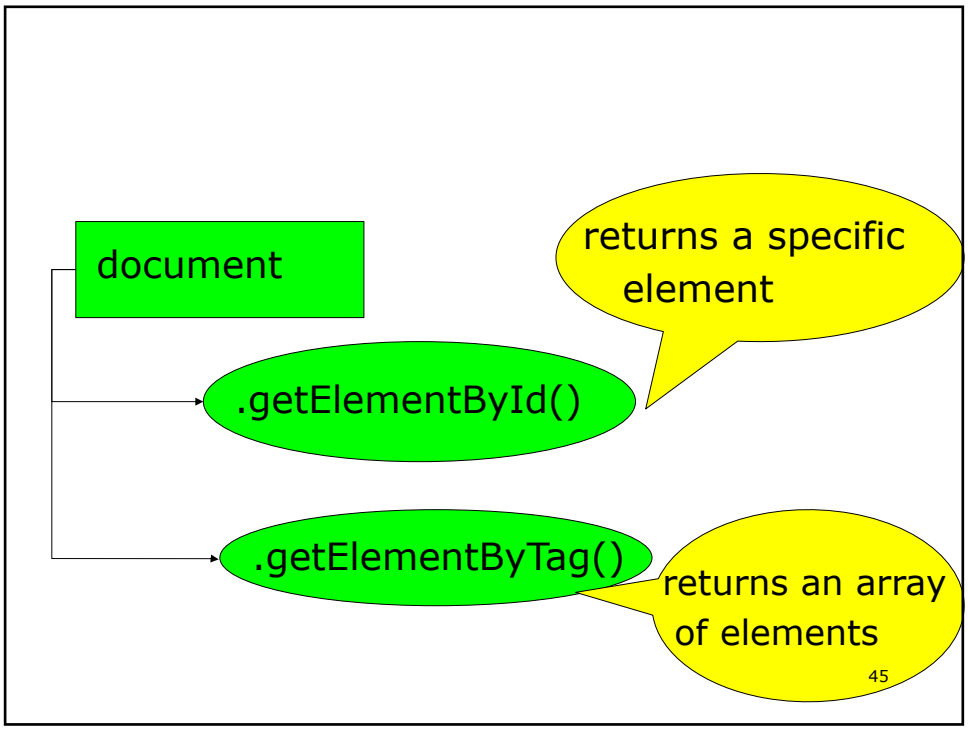
DOM NODE Methods

Method Name	Description
`appendChild`	Appends a child node.
`cloneNode`	Duplicates the node.
`getAttributes`	Returns the node's attributes.
`getChildNodes`	Returns the node's child nodes.
`getNodeName`	Returns the node's name.
`getNodeType`	Returns the node's type (e.g., element, attribute, text, etc.).
`getNodeValue`	Returns the node's value.
`getParentNode`	Returns the node's parent.
`hasChildNodes`	Returns `true` if the node has child nodes.
`removeChild`	Removes a child node from the node.
`replaceChild`	Replaces a child node with another node.
`setNodeValue`	Sets the node's value.
`insertBefore`	Appends a child node in front of a child node.

43

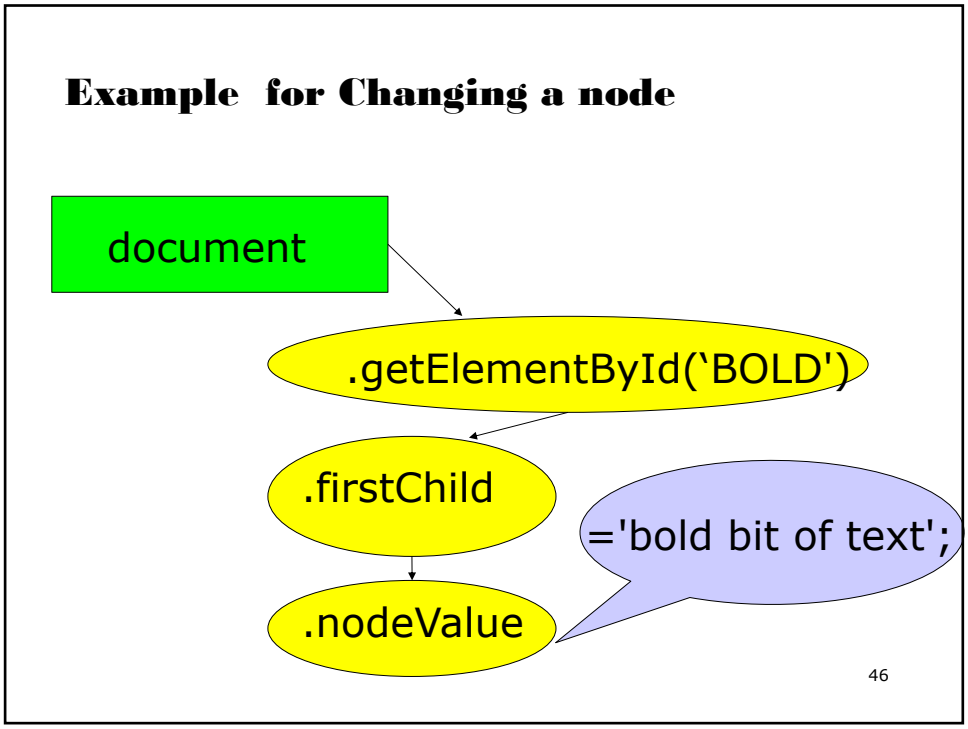


44



45

Example for Changing a node



46

Example Source Code For Document Method

document.body.style.backgroundColor

https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_doc_body

47

NAVIGATOR

48

NAVIGATOR :-Some properties are **read-only**

Browser Property	Description
navigator.appName	The name of the browser
navigator.appVersion	The major version number of the browser (may also include a compatibility value and the name of the operating system)
navigator.appMinorVersion	The minor version number of the browser
navigator.appCodeName	The name of the browser's code
navigator.userAgent	The name of the browser associated user agent
navigator.platform	The operating system under which the browser is running
navigator.cpuClass	The type of CPU in use with the browser
navigator.systemLanguage	The language used by the browser
navigator.cookieEnabled	A Boolean value indicating whether cookies are enabled

49

Example Source Code For Navigator Method

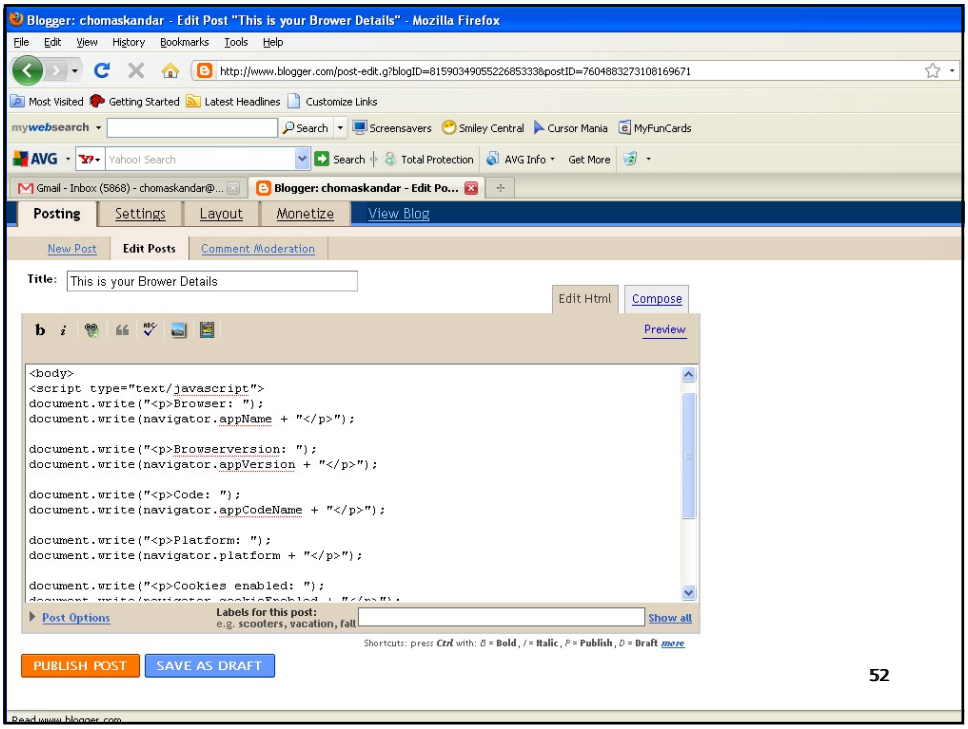
```
navigator.appName
navigator.appVersion
navigator.appCodeName
navigator.platform
navigator.cookieEnabled
```

50

Example For NAVIGATOR

```
<html><body><script type="text/javascript">
document.write("<p>Browser: ");
document.write(navigator.appName + "</p>");
document.write("<p>Browserversion: ");
document.write(navigator.appVersion + "</p>");
document.write("<p>Code: ");
document.write(navigator.appCodeName + "</p>");
document.write("<p>Platform: ");
document.write(navigator.platform + "</p>");
document.write("<p>Cookies enabled: ");
document.write(navigator.cookieEnabled + "</p>");
document.write("<p>Browser's user agent header: ");
document.write(navigator.userAgent + "</p>");
</script></body></html>
```

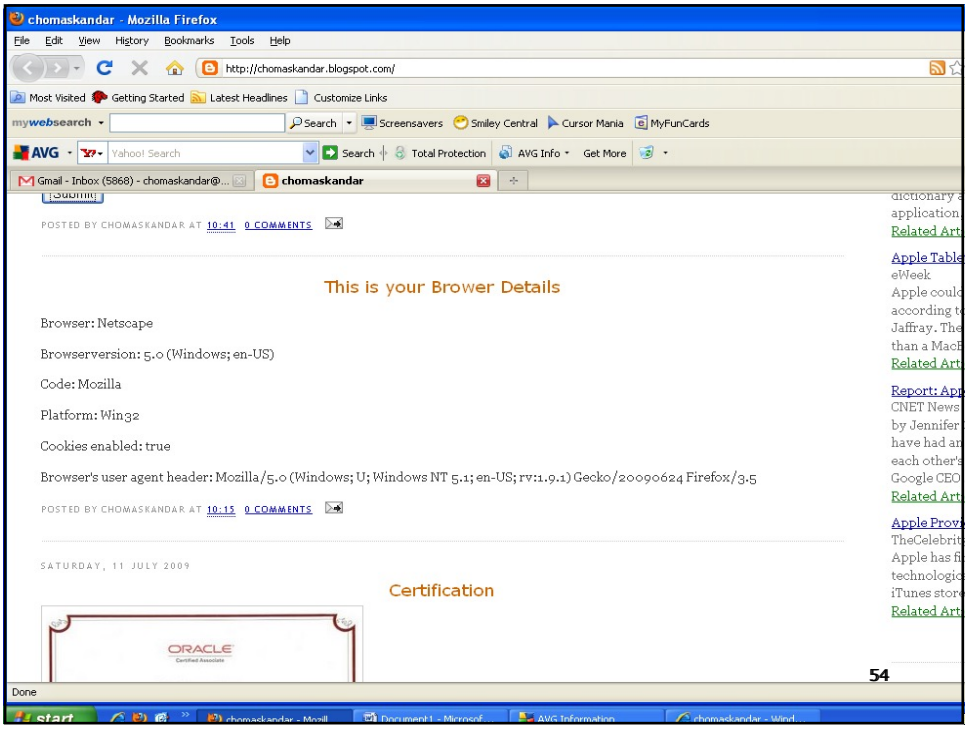
51



MOZILLA

53

53



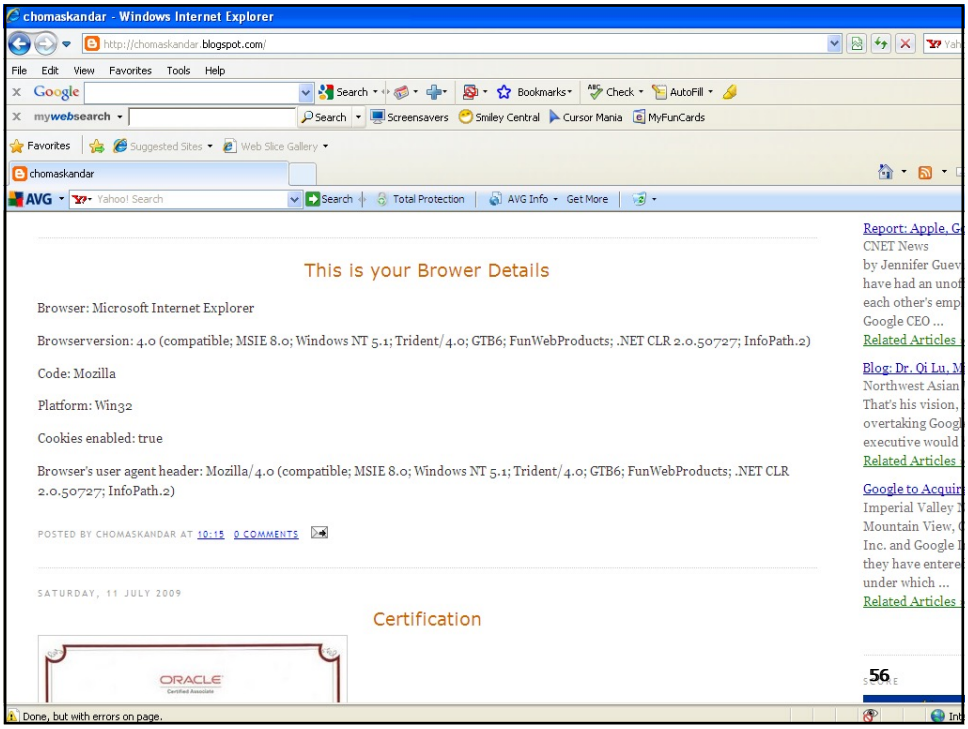
54

54

MICROSOFT INTERNET EXPLORER

55

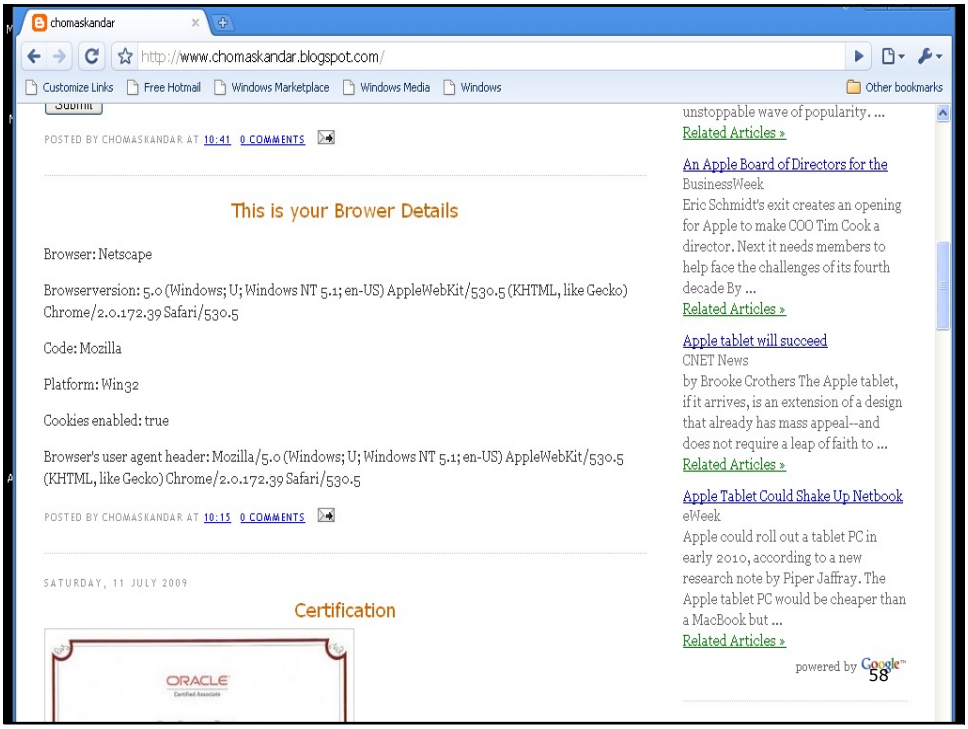
55



56

56

GOOGLE CHORME



W3C example

https://www.w3schools.com/jsref/obj_navigator.asp

DOM’s other examples

https://www.w3schools.com/jsref/dom_obj_all.asp

DOM Event Handling

Note: IE6 has a different event model

`Event` instance created for each event

`Event` instance properties:

- `type`: name of event (click, mouseover, *etc.*)
- `target`: Node corresponding to document element that generated the event (*e.g.*, `button` element for click, `img` for mouseover). This is the event target.

DOM Event Handling

JavaScript event listener: function that is called with `Event` instance when a certain event occurs

An event listener is associated with a target element by calling `addEventListener()` on the element

DOM Event Handling

```
var button = window.document.getElementById("msgButton");
button.addEventListener("click", sayHello, false);

function sayHello(event) {
  window.alert(
    "Hello World!\n\n" +
    "Event type: " + event.type + "\n" +
    "Event target element type: " + event.target.nodeName);
  return;
}
```

63

DOM Event Handling

Event target

```
var button = window.document.getElementById("msgButton");
button.addEventListener("click", sayHello, false);

function sayHello(event) {
  window.alert(
    "Hello World!\n\n" +
    "Event type: " + event.type + "\n" +
    "Event target element type: " + event.target.nodeName);
  return;
}
```

64

DOM Event Handling

```
var button = window.document.getElementById("msgButton");
button.addEventListener("click", sayHello, false);
function sayHello(event) {
  window.alert(
    "Hello World!\n\n" +
    "Event type: " + event.type + "\n" +
    "Event target element type: " + event.target.nodeName);
  return;
}
```

Event type

DOM Event Handling

DOM event types:

- All HTML intrinsic events except keypress, keydown, keyup, and dblclick
- Also has some others that are typically targeted at the `window` object:

Event	Cause
`error`	An error (problem loading an image, script error, etc.) has occurred.
`resize`	View (window or frame) of document is resized.
`scroll`	View (window or frame) of document is scrolled.

DOM Event Handling

```
var button = window.document.getElementById("msgButton");
button.addEventListener("click", sayHello, false);
```

Event handler

Definition of event handler

```
function sayHello(event) {
  window.alert(
    "Hello World!\n\n" +
    "Event type: " + event.type + "\n" +
    "Event target element type: " + event.target.nodeName);
  return;
}
```


DOM Event Handling

```
var button = window.document.getElementById("msgButton");
button.addEventListener("click", sayHello, false);

function sayHello(event) {
  window.alert(
    "Hello World!\n\n" +
    "Event type: " + event.type + "\n" +
    "Event target element type: " + event.target.nodeName);
  return;
}
```

`Event` instance

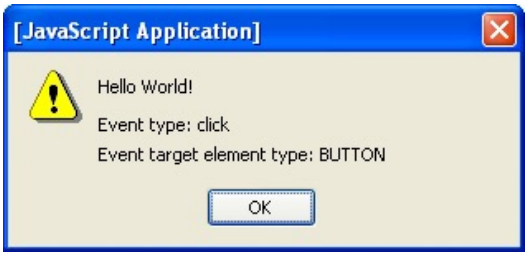
DOM Event Handling

```
var button = window.document.getElementById("msgButton");
button.addEventListener("click", sayHello, false);

function sayHello(event) {
  window.alert(
    "Hello World!\n\n" +
    "Event type: " + event.type + "\n" +
    "Event target element type: " + event.target.nodeName);
  return;
}
```

Normally false
(more later)

DOM Event Handling

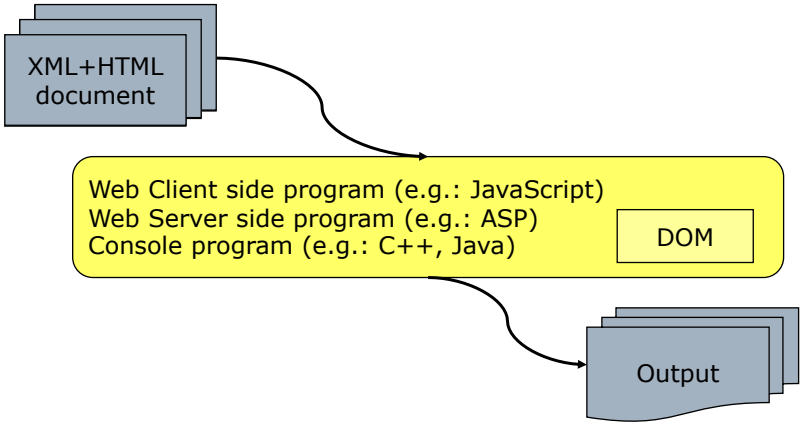


W3C example

https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_onclick_addeventlistener

How the DOM Really works?

The Relation Graph



73

DOM in Programming Languages

- Java
- C++
- C#
- VB.Net, etc.

74

DOM Advantages & Disadvantages

DOM Advantages & Disadvantages

ADVANTAGES

- Robust API for the DOM tree
- Relatively simple to modify the data structure and extract data

Disadvantages

- Stores the entire document in memory
- As Dom was written for any language, method naming conventions don’t follow standard java programming conventions

Some DOM Supporting Browsers

Konqueror

Opera

Camino

Safari

SUMMARY

Summary

- DOM is a tree representation of an XML document in memory
- Dom provides a robust API to easily Modify and extract data from an XML document
- JAXP provides a vendor –neutral interface to the underlying DOM or SAX Parser

79

References

- www.w3.org/DOM
- http://developer.mozilla.org/en/Gecko_DOM_Reference
- www.corewebprogramming.com
- http://www.w3schools.com

80

email: chungdt@soict.hust.edu.vn

Q&A