Javascript and the DOM

1 – Introduction to The DOM

JS and the DOM

When a web page is loaded, the browser creates a Document Object Model of the page, which is as a tree of Objects.

Every element in a document—the document as a whole, the head, tables within the document, table headers, text within the table cells—is part of its DOM, so they can all be accessed and manipulated using the DOM and a scripting language like JavaScript.

- With the object model, JavaScript can:
- change all the HTML [elements, attributes, styles] in the page
- add or remove existing HTML elements and attributes
- react to HTML events in the page
- create new HTML events in the page

Using languages

implementations of the DOM can be built for any language:

e.g.

Javascript

Java

Python

…

But Javascript is the only one that can work client-side.

Javascript and the DOM

2 – Fundamental datatypes

Fundamental datatypes - 1

- Document (is-a Node)
  - the root document object itself.
- Node

  Every object located within a document is a node. In an HTML document, an object can be an element node but also a text node or attribute node.
- Element (is-a Node)

  The most general base class from which all element objects (i.e. objects that represent elements) in a Document inherit.

Fundamental datatypes - 2

- Attr (is-a Node)

  An object reference that exposes a special interface for attributes. Attributes are nodes in the DOM just like elements are.

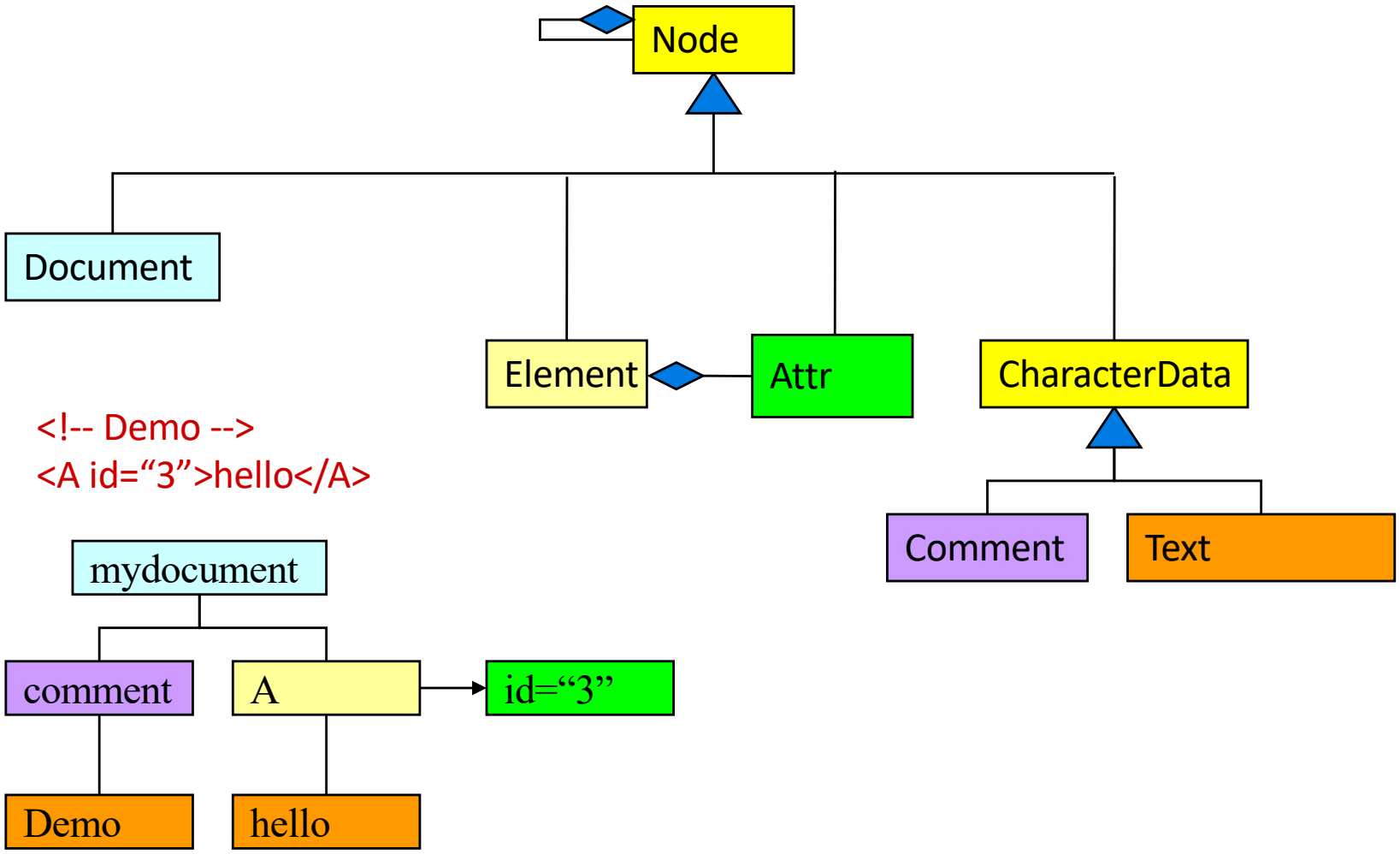
- NodeList

  A nodeList is an array of elements. Items in a nodeList are accessed by index: list.item(1) or list[1]

- NamedNodeMap

  A namedNodeMap is an associative array, where the items are accessed by name. They can also be accessed by index using the item() method (but nodes they are in no particular order in the list). You can also add and remove items from a namedNodeMap.

The Node hierarchy



Node: WARNING!

The implied semantic of this model is WRONG!

You might deduce that a comment might contain another comment, or a document, or any other node!

The integrity is delegated to a series of Node's attributes, that the programmer should check.

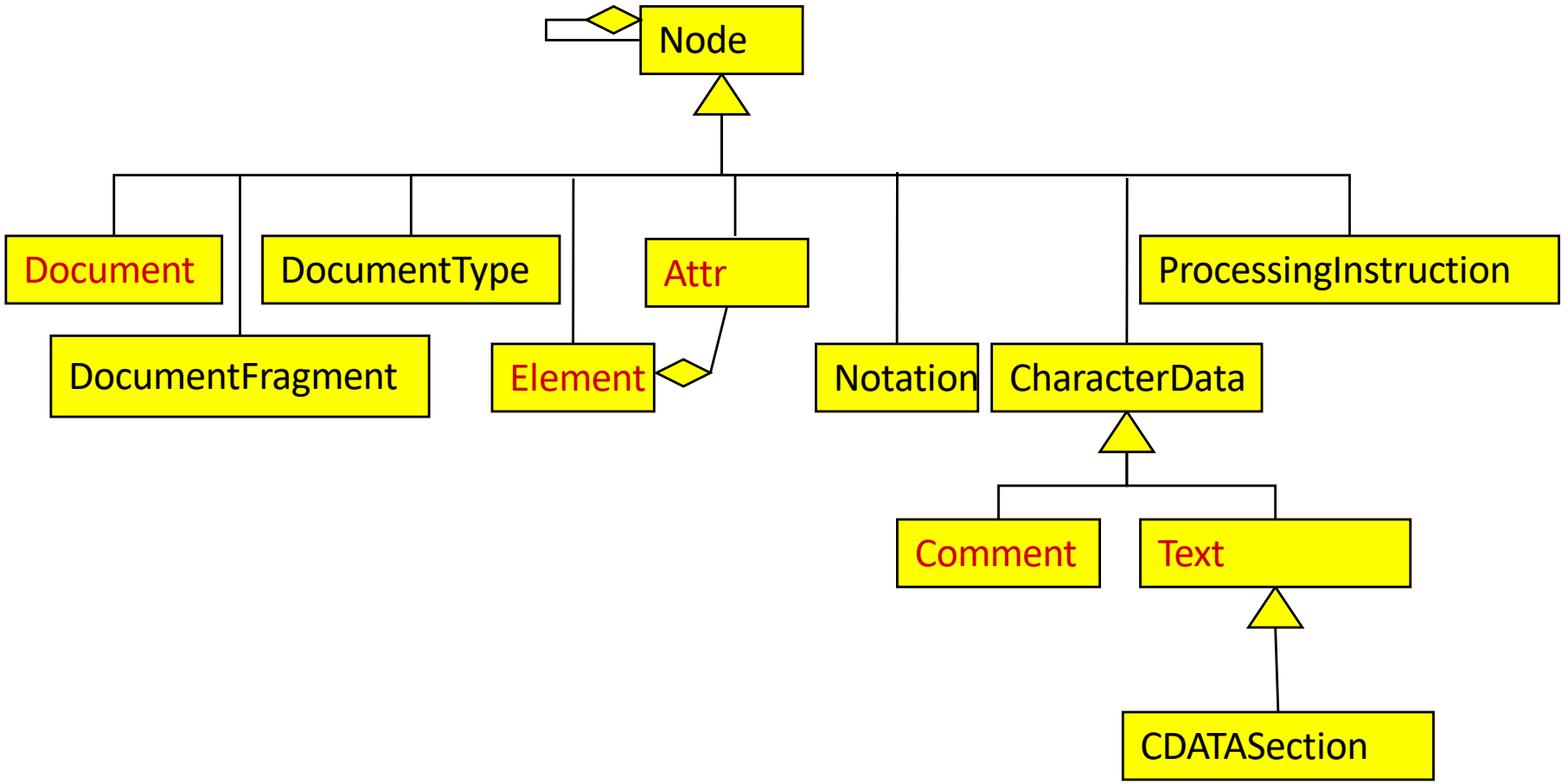
Node

- All of the following types inherit the Node interface's methods and properties:

  Document, Element, Attr, CharacterData (which Text, Comment, and CDATASection inherit), ProcessingInstruction, DocumentFragment, DocumentType, Notation

see https://developer.mozilla.org/en-US/docs/Web/API/Node

The Node hierarchy



Node properties

- Node.nodeType - Read only ==>
- Node.nodeName - Read only

  (An HTMLElement will contain the name of the corresponding tag, like 'audio' for an HTMLAudioElement, a Text node will have the '#text' string, or a Document node will have the '#document' string).
- Node.baseURI - Read only
- Node.textContent – read/write

Node Type	
ELEMENT_NODE	1
ATTRIBUTE_NODE	2
TEXT_NODE	3
CDATA_SECTION_NODE	4
ENTITY_REFERENCE_NODE	5
ENTITY_NODE	6
PROCESSING_INSTRUCTION_NODE	7
COMMENT_NODE	8
DOCUMENT_NODE	9
DOCUMENT_TYPE_NODE	10
DOCUMENT_FRAGMENT_NODE	11
NOTATION_NODE	12

Node: Read only navigation properties

- Node.childNodes
- Node.firstChild
- Node.lastChild

- Node.nextSibling
- Node.previousSibling

- Node.parentNode
- Node.parentElement
- Node.ownerDocument

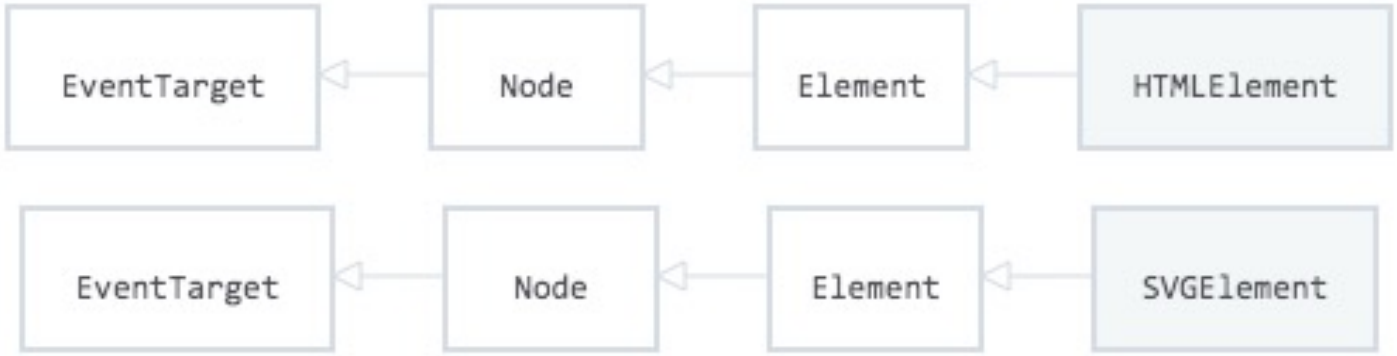
Element

- **Element**

  the most general base class from which all element objects (i.e. objects that represent elements) in a Document inherit.

  It only has methods and properties common to all kinds of elements. More specific classes inherit from Element.

  For example, the HTMLElement interface is the base interface for HTML elements, while the SVGElement interface is the basis for all SVG elements.



see https://developer.mozilla.org/en-US/docs/Web/API/Element

Some Element properties

- Element.id
- Element.attributes
- Element.className
- Element.innerHTML
- Element.outerHTML

- Element.clientHeight - the inner height of the element.
- Element.clientLeft - the width of the left border of the element.
- Element.clientTop - the width of the top border of the element.
- Element.clientWidth - the inner width of the element.

Element main properties

- element.innerHTML
- element.style.left
- element.setAttribute()
- element.getAttribute()
- element.addEventListener()

Element "name" property

- **name** gets or sets the name property of an element in the DOM. It only applies to the following elements: <a>, <applet>, <button>, <form>, <frame>, <iframe>, <img>, <input>, <map>, <meta>, <object>, <param>, <select>, and <textarea>.
- **Note:** The name property doesn't exist for other elements; unlike tagName and nodeName, it is not a property of the Node, Element or HTMLElement interfaces.
- name can be used in the document.getElementsByName() method, a form and with the form elements collection. When used with a form or elements collection, it may return a single element or a collection.

JS output

Writing into an HTML element, using:

- innerHTML *(Element property)*

```
<div onmouseover="this.innerHTML='How are you?';">
  Hello</div>
```

- innerText *(HTMLElement property)*

```
<div onmouseover="this.innerText='How are you?';">
  Hello</div>
```

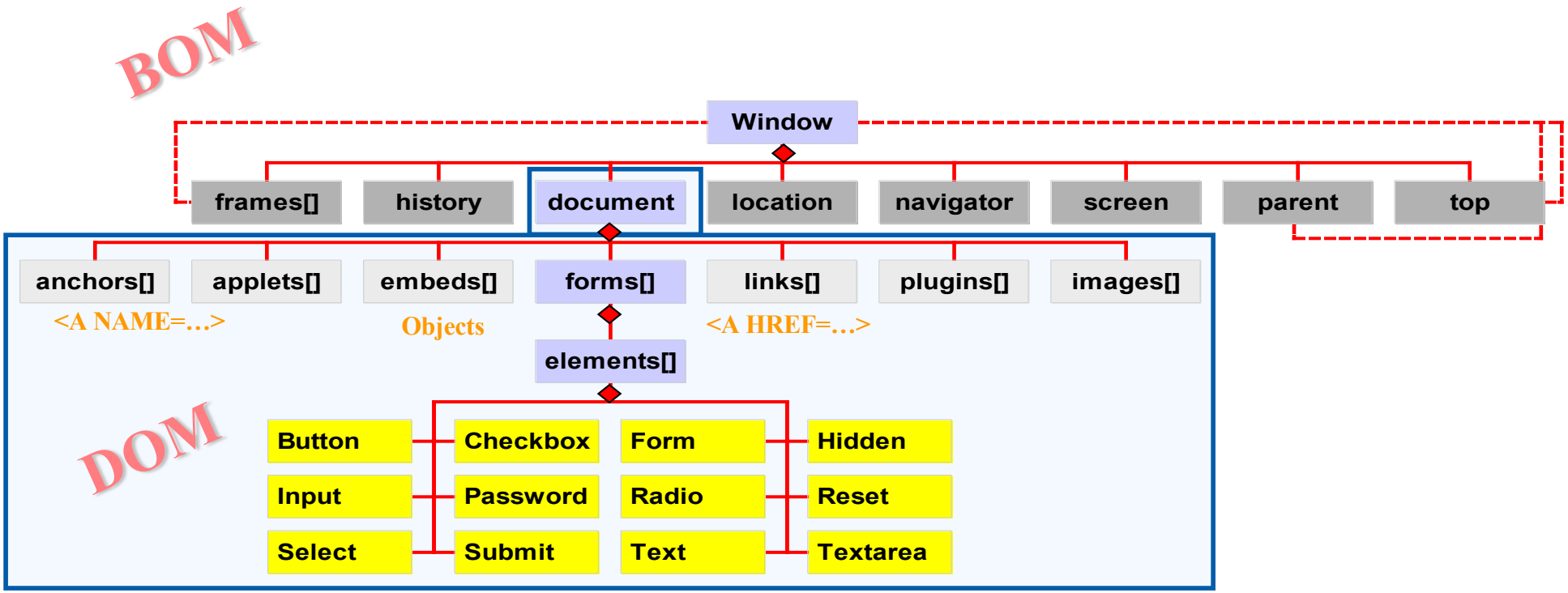
- textContent *(Node property)*

```
<div onmouseover="this.textContent='How are you?';">
  Hello</div>
```

Javascript and the DOM

3 – A DOM subset: the BOM

Object hierarchy

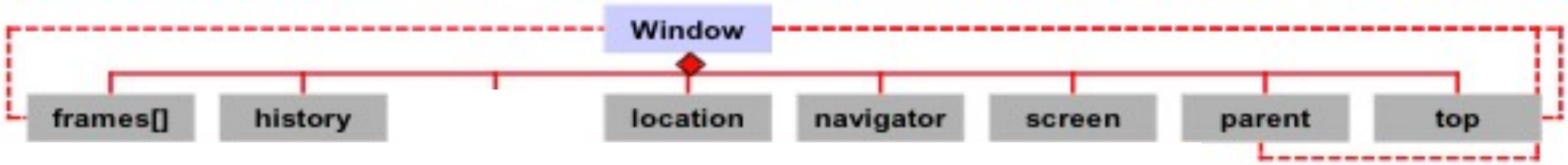


Symbol ◆ means containment (has-a)

Dashed line means "is an instance of"

BOM

Window



"A web browser window or frame"

Main properties

Objects

history

frames[]

document

location

navigator

screen

parent – top

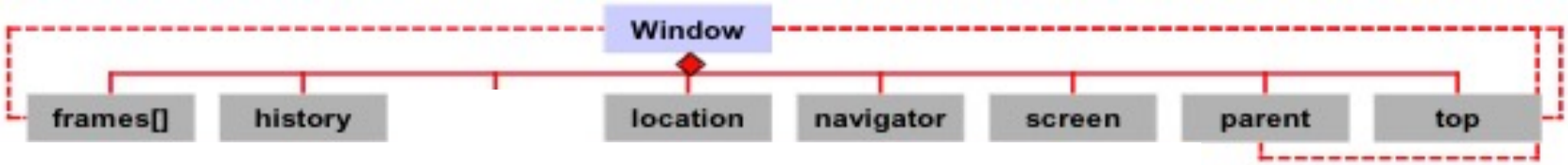
Other properties

status – defaultStatus

name

BOM

Window



Main methods

alert(), prompt(), confirm()

focus(), blur()

moveBy(), moveTo()

resizeBy(), resizeTo()

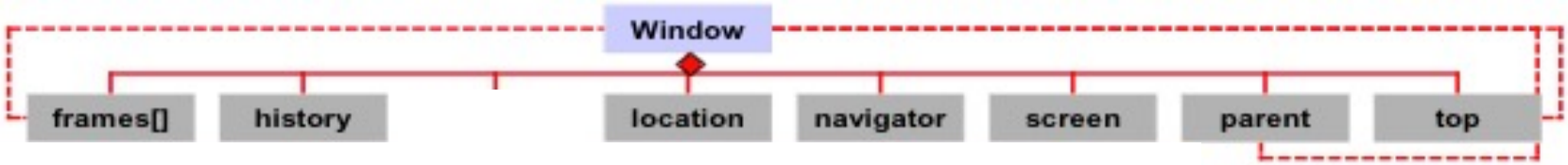
scroll(), scrollBy(), scrollTo()

setInterval(), clearInterval()

setTimeout(), clearTimeout()

BOM

Screen



"Information about the display"

Main properties

availHeight, availWidth

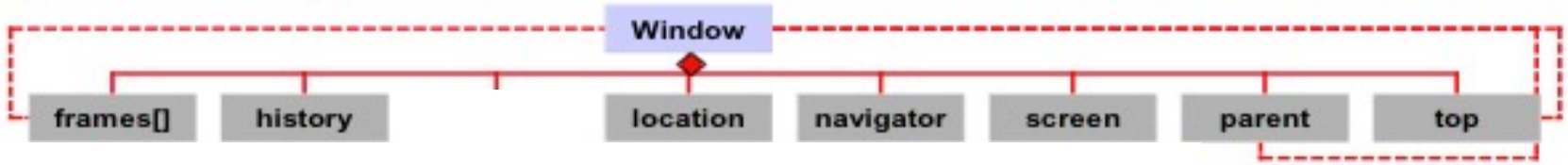
height, width

colorDepth, pixelDepth

hash

BOM

Navigator



"Information about the browser in use"

Main properties

appName

appVersion

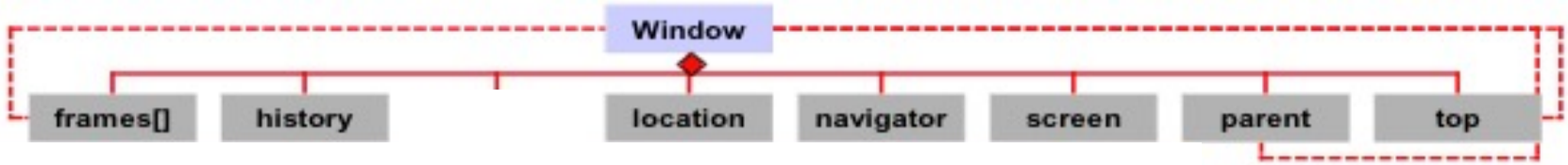
Platform

Main methods

javaEnabled()

BOM

History



"The URL history of the browser"

Main properties

lenght

Main methods

back()

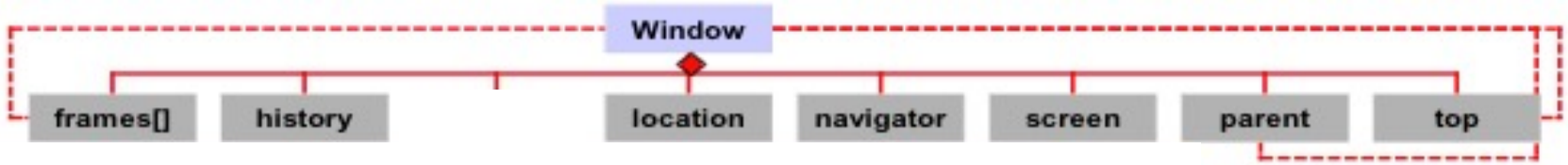
forward()

go(+/-n)

go(target_substring)

BOM

Location

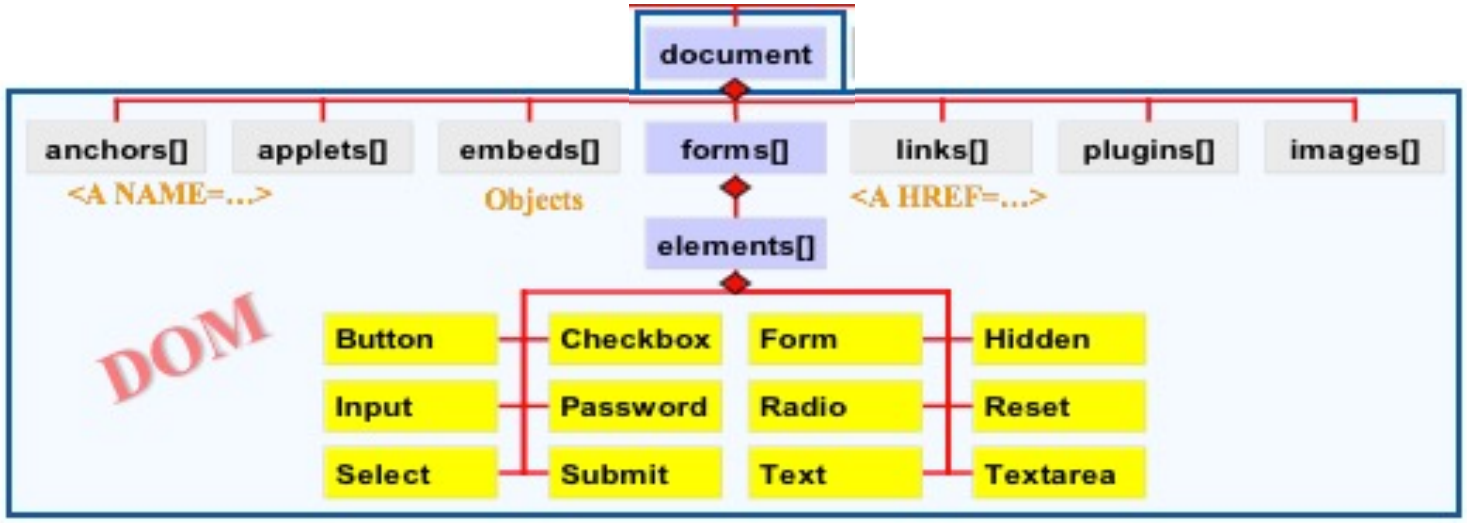


"The specification of the current URL"

Main properties	Main methods
href	reload()
protocol, hostname, port	replace()
search	
hash	

Javascript and the DOM

4 – Document and (some of) its components



Document

"An HTML document"

Main properties

Arrays of Component Objects

anchors[]

applets[]

embeds[]

forms[]

links[]

plugins[]

Other properties

bgColor, fgColor, linkColor, vlinkColor

lastModified

title, URL, referrer, cookie

Main methods

open()

close()

clear()

write()

see https://developer.mozilla.org/en-US/docs/Web/API/Document

Element selection

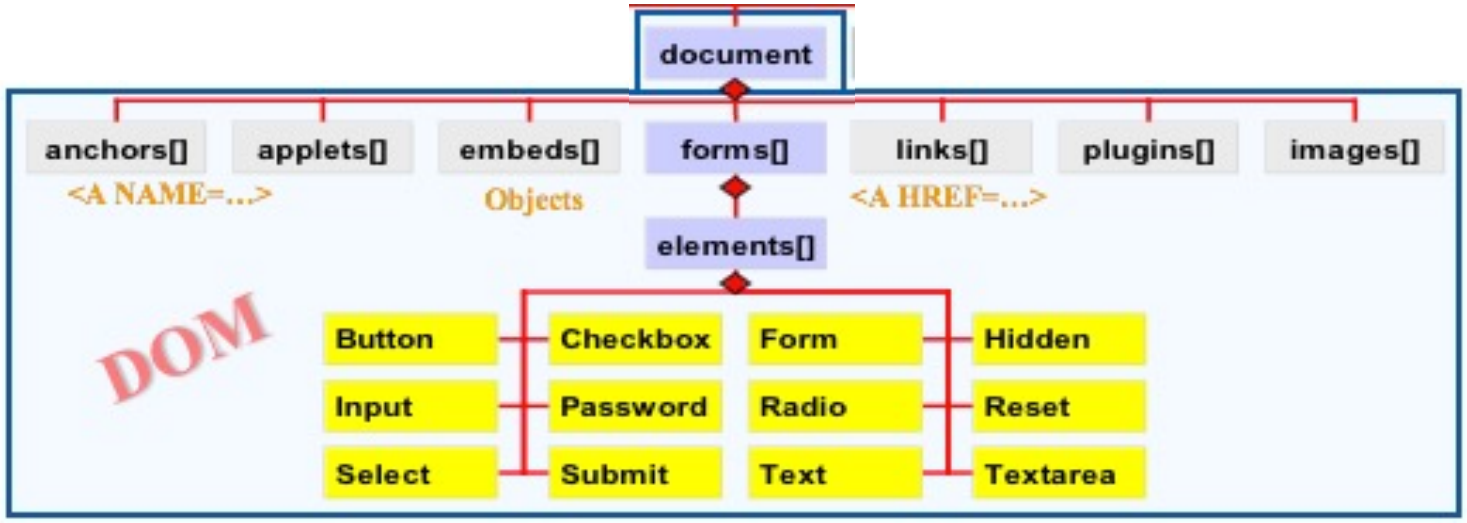
- document.getElementById(*id*)
- document.getElementsByName(*name*)
- document.getElementsByTagName(*name*)
- document.getElementsByClassName(*name*)


```
<!DOCTYPE html>
<html>
    <head>
        <style>.rosso { background: red;}</style>
    </head>
    <body>
        <div id="pippo">Il mio id è Pippo</div>
        <b name="pluto">Il mio nome è Pluto</b>
        <div class="rosso">La mia classe è "rosso"</div>
        <script>window.alert("ByName :"+
                document.getElementsByName("pluto")[0].innerText);</script>
        <script>window.alert("ByTagName :"+
                document.getElementsByTagName("div")[0].innerText);</script>
        <script>window.alert("ByClassName :"+
                document.getElementsByClassName("rosso")[0].innerText);</script>
        <script>window.alert("ByID :"+
                document.getElementById("pippo").innerText);</script>
    </body>
</html>
```

DOM Structure modification

- document.createElement(*element*)
- document.removeChild(*element*)
- document.appendChild(*element*)
- document.replaceChild(*newElement, oldElement*)

```
<!DOCTYPE html>
<html>
    <body>
        <div id="1">UNO</div>
        <div>TRE</div>
        <script>
            var node = document.createElement("DIV");
            //var textnode = document.createTextNode("DUE");
            //node.appendChild(textnode);
            node.innerText="DUE";
            document.getElementById("1").appendChild(node);
         </script>
    </body>
</html>
```



Image

"An image embedded in an HTML document"

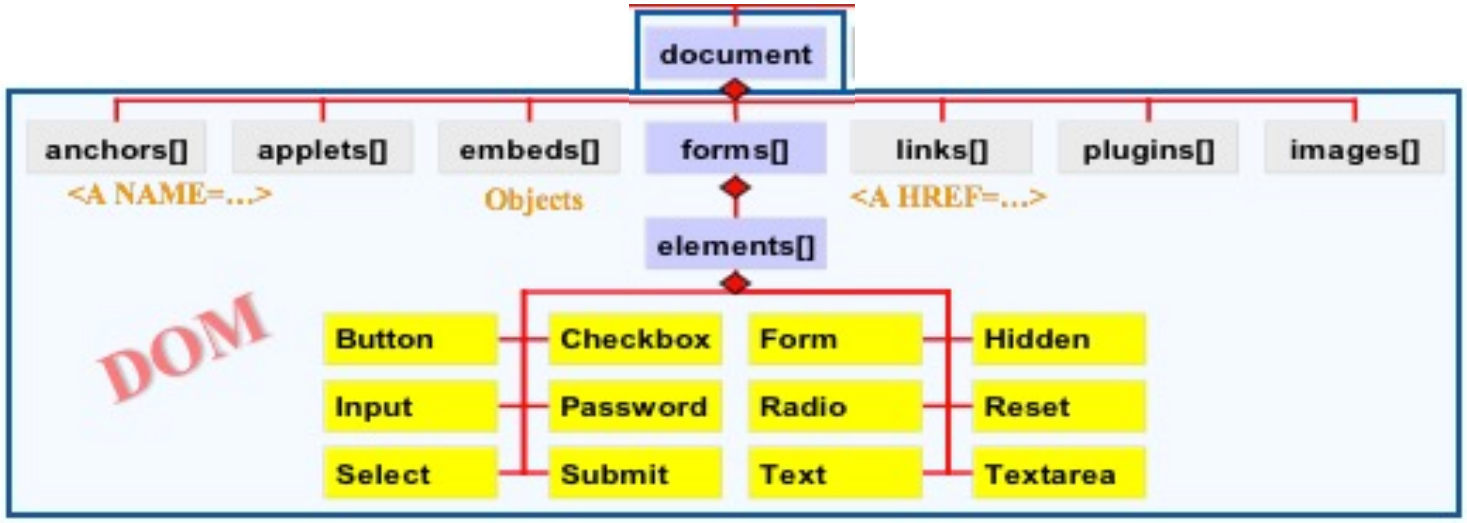
Main properties

border *[width in pixels]*

height

width

src *[URL of the image to be displayed]*



Applet

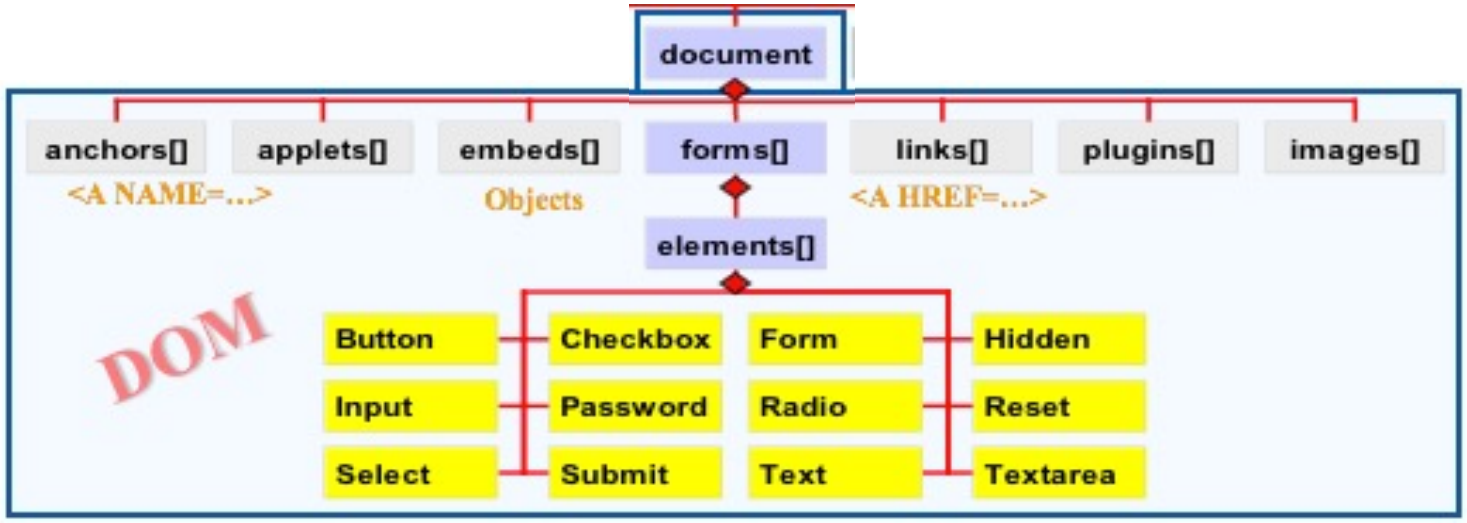
"An applet embedded in a Web page"

Properties

Same as the public fields of the Java applet

Methods

Same as the public methods of the Java applet



Form

"An HTML input form"

Main properties

action *[destination URL]*

method *[get/post]*

name *[name of Form]*

name *[destination Window]*

Elements[] *[list ;of contained elements]*

Main methods

reset()

submit()

Javascript and the DOM

5 – Events

Event

- The Event interface represents an event which takes place in the DOM.
- An event can be triggered by the user action(e.g. clicking the mouse button) or generated by APIs to represent the progress of an asynchronous task.
- It can also be triggered programmatically, such as by calling the HTMLElement.click() method of an element, or by defining the event, then sending it to a specified target using EventTarget.dispatchEvent().

- There are many types of events, some of which use other interfaces based on the main Event interface. Event itself contains the properties and methods which are common to all events.

Event

UIEvent:

- Input https://www.w3schools.com/jsref/obj_inputevent.asp
- Keyboard https://www.w3schools.com/jsref/obj_keyboardevent.asp
- Focus https://www.w3schools.com/jsref/obj_focusevent.asp
- Mouse https://www.w3schools.com/jsref/obj_mouseevent.asp
- Drag&drop https://www.w3schools.com/jsref/obj_dragevent.asp
- …

Generic Events

- Events https://www.w3schools.com/jsref/obj_event.asp
- Animation https://www.w3schools.com/jsref/obj_animationevent.asp
- Clipboard https://www.w3schools.com/jsref/obj_clipboardevent.asp
- …

Event

- Events can be managed by EventHandlers on*Event*="*javascript code*"

```
<!DOCTYPE html>
<html>
<head><script>
function displayDate() {
  document.getElementById("demo").innerHTML = Date();
}
</script></head>
<body>
<p onclick="displayDate()" id="demo">
Click me  to display the date.</p>
</body>
</html>
```

Javascript and the DOM

6 – A simple example

Events example 1

```
<!DOCTYPE html>
<head>
   <title>Form Example</title>
   <script>
      function setColor() {
         var choice;
         choice = document.colorForm.color.selectedIndex;
         switch(choice) {
            case 0: document.bgColor = "FF0000"; break;
            case 1: document.bgColor = "00FF00"; break;
            case 2: document.bgColor = "0000FF"; break;
            case 3: document.bgColor = "FFFFFF"; break;
            case 4: document.bgColor = "FFFF00"; break;
            case 5: document.bgColor = "FF00FF"; break;
         }
      }
   </script>
</head>
```

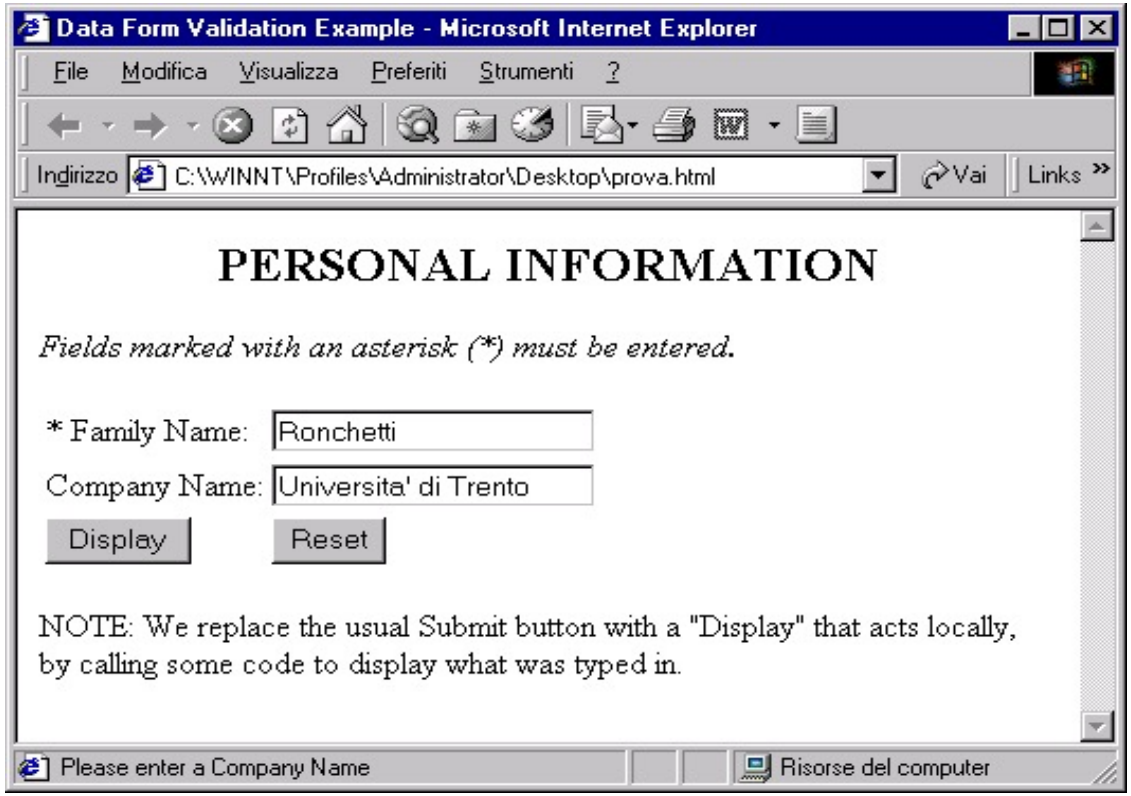
Events example 1

```
<body>
    <h1>Color Changer</h1>
    <br><br>
    Select Your Favorite Background Color:
    <form name="colorForm">
        <select name="color" onChange=setColor() >
            <option value="red">Red</option>
            <option VALUE="green">Green</option>
            <option VALUE="blue">Blue</option>
            <option VALUE="white">White</option>
            <option VALUE="yellow">Yellow</option>
            <option VALUE="purple">Purple</option>
        </select>
    </form>
</body>
</html>
```

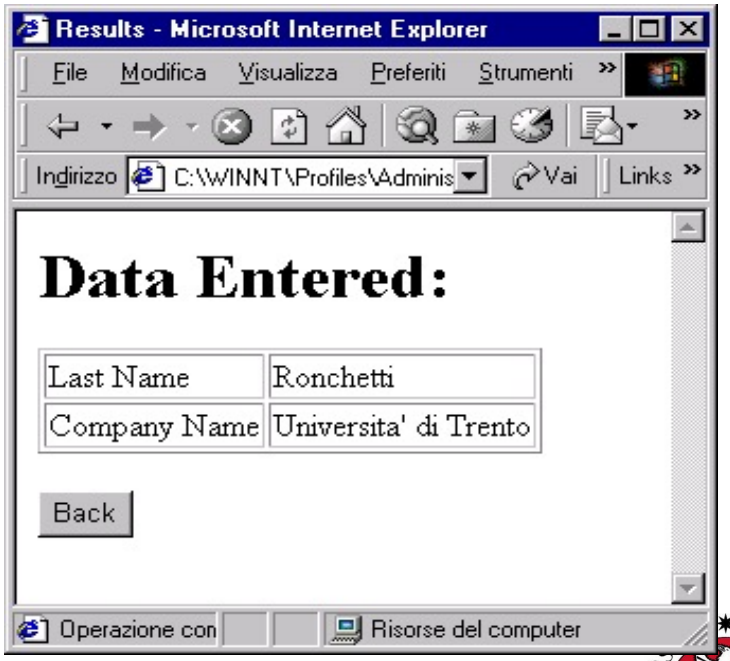
Javascript and the DOM

7 – A more complex example

A more complex example -1



A simple data entry validation page



A more complex example -2

Start of file "FormValidation.html"

```
<HTML>
<HEAD>
<TITLE>Data Form Validation Example</TITLE>

<SCRIPT LANGUAGE="JavaScript1.1" SRC="FormCheck.js"></SCRIPT>
```

Load file "FormCheck.js",
which contains several JavaScript functions

A more complex example -6

```
<BODY BGCOLOR="#ffffff">
<CENTER><H2>PERSONAL INFORMATION </H2></CENTER>
<P><P><I>Fields marked with an asterisk (*) must be entered.</I>
<FORM NAME="PersonalInfo">
<TABLE>
<TR>
   <TD>* Family Name:</TD>
   <TD><INPUT TYPE="text" NAME="LastName"
     onChange="checkString(this,sLastName)" ></TD>
</TR>
<TR>
   <TD>Company Name:</TD>
   <TD><INPUT TYPE="text" NAME="Company" ></TD>
</TR>
```

First Field

Second Field

Start of “BODY” portion of “FormValidation.html”

A more complex example -3

```
function isEmpty(s)
{   return ((s == null) || (s.length == 0))
}
```

Check that the string "s" is not empty

```
function warnEmpty (theField, s)
{
   var mPrefix = "You did not enter a value into the ";
   var mSuffix = " field. This is a required field. Please enter it now.";
   theField.focus();
   alert(mPrefix + s + mSuffix);
   return false;
}
```

Issue a warning message

All this is contained in the file "FormCheck.js"

A more complex example -4

```
function validatePersonalInfo(form)
{   return (
      checkString(form.elements["LastName"],sLastName)
    )
}

function checkString (theField, s)
{
   if (isEmpty(theField.value)) return warnEmpty (theField, s);
   else return true;
}
```

Validate the form

(should run over all fields
And perform suitable checks)

Check that "theField"
is not empty

All this is contained in the file "FormCheck.js"

A more complex example -5

```
<SCRIPT>
var sCompany="Company Name";  var sLastName="Last Name";  var
      form="PersonalInfo";


function displayPersonalInfo(form)
{     var outputTable = "<HTML><HEAD><TITLE>Results</TITLE></HEAD>" +
     "<BODY><H1>Data Entered:</H1><TABLE BORDER=1>" +
     "<TR><TD>" + sLastName + "</TD><TD>" + form.elements["LastName"].value +
      "</TD></TR>" +
     "<TR><TD>" + sCompany + "</TD><TD>" + form.elements["Company"].value +
      "</TD></TR></TABLE><FORM>“ +
     "<INPUT TYPE=\"BUTTON\" NAME=\"Back\" VALUE=\"Back\"
      onClick=\"location.reload()\"> </FORM></BODY></HTML>"
     document.writeln(outputTable)
     document.close()
     return true
}    </SCRIPT>
</HEAD>
```

Global variables

Value-printing function

Add a Button to reload the page

End of “HEAD” portion of “FormValidation.html”

A more complex example -7

```
<TR>
   <TD>
   <INPUT TYPE="BUTTON" NAME="fakeSubmit" VALUE="Display"
     onClick="if (validatePersonalInfo(this.form)) displayPersonalInfo(this.form); ">
   </TD>
   <TD><INPUT TYPE = "reset" VALUE = "Reset">
   </TD>
</TR>
</TABLE>
<P> NOTE: We replace the usual Submit button with a "Display" that acts locally,
<BR>by calling some code to display what was typed in.
</FORM>
</BODY>
</HTML>
```

First Button

Second Button

End of file “FormValidation.html”

References

Standard ECMA-262 ECMAScript Language Specification:

http://www.ecma-international.org/publications/standards/Ecma-262.htm

Books:

- D.Flanagan "Javascript. The definitive guide" O'Reilly.
- D.Goodman "Dynamic HTML. The definitive reference" O'Reilly

Server-Side JavaScript

A substitute for CGI.

Server-dependent technology to process the Web page ***before*** passing it to the client.

(An approach which started long ago (Netscape SSJS)

Then mostly forgotten, later revived by Rhino (a bridge between JS and Java) and recently by Node.js

Node.js

Node.js is an open-source, cross-platform JavaScript
run-time environment for executing JavaScript code server-side.
Node.js has an event-driven architecture capable of asynchronous I/O.

Optimize throughput and scalability

- in Web applications with many input/output operations,
- for real-time Web applications

https://www.w3schools.com/nodejs/default.asp