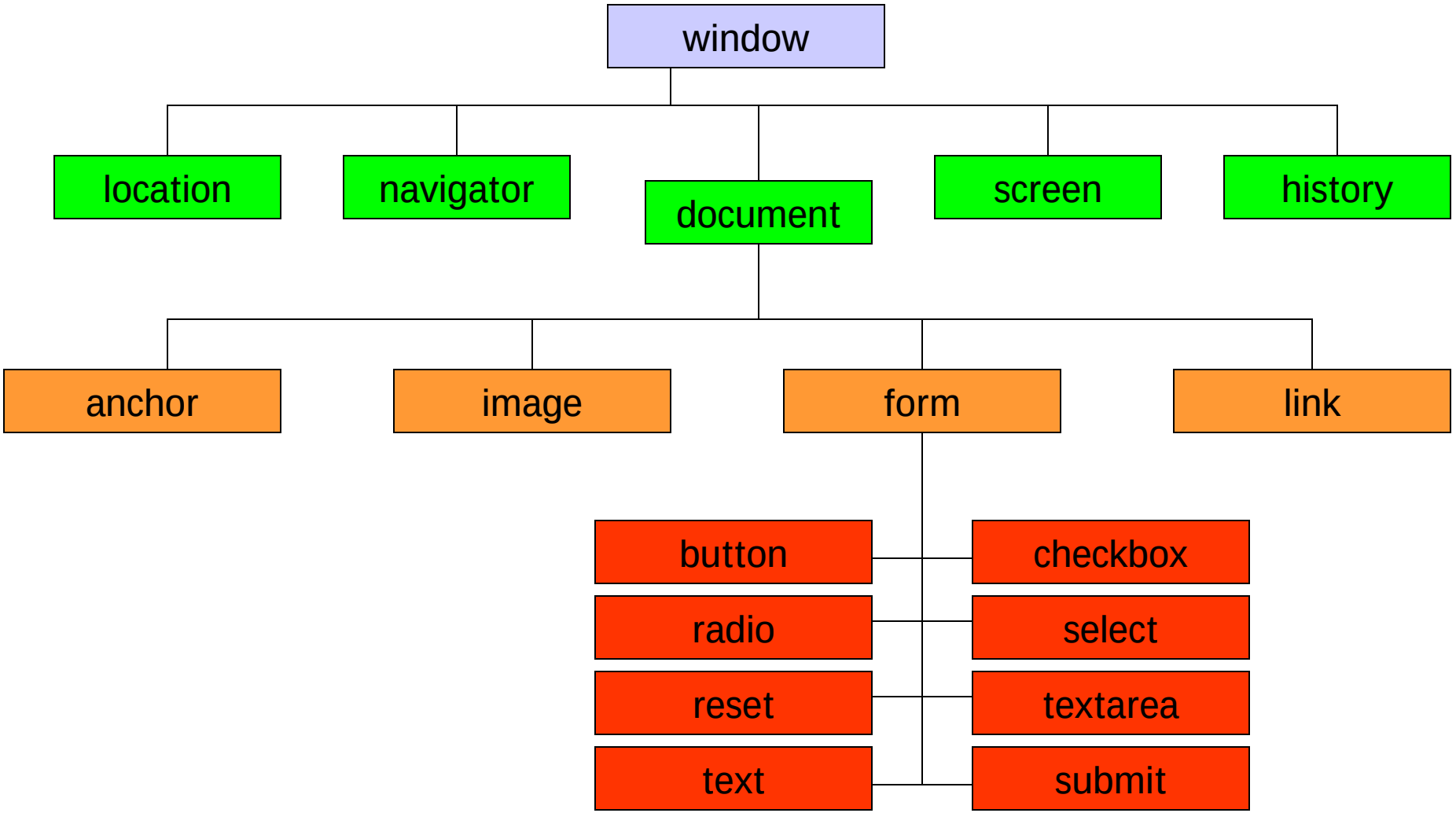
JavaScript
Browser Objects and DOM

Browser objects and their hierarchy model



The “window” Object

- Represents the browser

- The default object (the object is always "there")
  - Writing

```
document.write("a test message");
alert("Hello");
foo = "bar";
```

has the same meaning as writing

```
window.document.write("a test message");
window.alert("Hello");
window.foo = bar;
```

Some of "window" properties and methods

- alert(), prompt(), confirm()
- open()
  - Create a new window
- close()
  - close the current window
- setTimeout(expression, time)
  - Evaluate "expression" after "time" (in millisecond)

Example: Opening a Window

- var winObj = window.open(url, window_name, attributes)
- attributes is a string for specifying the following attributes

Attribute	Description
toolbar	**Creates the standard toolbar**
location	**Creates the location entry field**
directories	**Creates standard directory buttons**
status	**Creates the status bar**
menubar	**Creates the menu bar at the top of a window**
scrollbars	**Creates scrollbars when the document exceeds the window size**
resizable	**Enables the user to resize the window**
width	**Specifies the width of the window**
height	**Specifies the height of the window**

```
<!-- Opening a window with specified characteristics -->
<html><head><script type="text/javascript">
var myWin;
function open_close_win() {
  if (!myWin)  // if not yet opened, open a new window
    myWin = window.open(
      "http://www.w3schools.com",       // Document URL
      "my_new_window",                  // Window Name
      "toolbar=yes,location=yes,directories=no," +
      "status=no,menubar=yes,scrollbars=yes," +
      "resizable=no,copyhistory=yes,width=400,height=400"
    );
  else { // Otherwise close the opened window
    myWin.close();
    myWin = null;
  }
}
</script></head><body><form>
<input type="button" value="Open/close Window"
       onclick="open_close_win()">
</form></body></html>
```

Properties in the "window" object

- location
  - Represents the URL loaded into the window
- navigator
  - Contains info about the browser (Its version, OS, etc.)
- document
  - Holds the real content of the page
- screen
  - Contains info about the client's display screen
- history
  - Contains the visited URLs in the browser window

The “document” Object

- The document object represents a web document or a page in a browser window/frame.
- Useful properties
  - `cookie, URL, images[], forms[], anchors[], …`
- Useful methods
  - `write(), writeln(), getElementById(), getElementsByTagName(), open(), close(), …`

```
<!-- Create new document content -->
<html>
<head>
<script type="text/javascript">
function docOpen()
{
  document.open();                // Old contents are gone
  document.write("<h3>Hello World!</h3>");
  document.close();
}
</script>
</head>

<body>
test
<form>
<input type="button" onclick="docOpen()"
       value="Open a new document">
</form>
</body>

</html>
```

The “form” Object

- The "form" object belongs to the "document" object.
- Contains other objects that represent the form elements (text input field, radio buttons)
- Useful properties
  - action, method, target, elements[]
- Useful methods
  - reset(), submit()
- Form elements can be accessed as
  - `document.forms[idx]` or `document.forms[form_name]` or `document.form_name` or `document.getElementById(form_id)`

```
<!– Validate the range of input in a text field -->
<html><head>
<script type="text/javascript">
function validate() {
  var x = document.myForm;
  var txt = x.myInput.value;
  if (txt >= 1 && txt <= 5)
    return true;
  else {
   alert("Must be between 1 and 5");
   return false;
  }
}
</script>
</head>
<body>
<form name="myForm" action="tryjs_submitpage.htm"
 onsubmit="return validate()">
 Enter a value (1-5):
 <input type="text" name="myInput" size="20">
 <input type="submit" value="Submit">
</form></body></html>
```

Document Object Model (DOM)

- The Document Object Model (DOM) is the model that describes how all elements in an HTML page, like input fields, images, paragraphs etc., are related to each other.

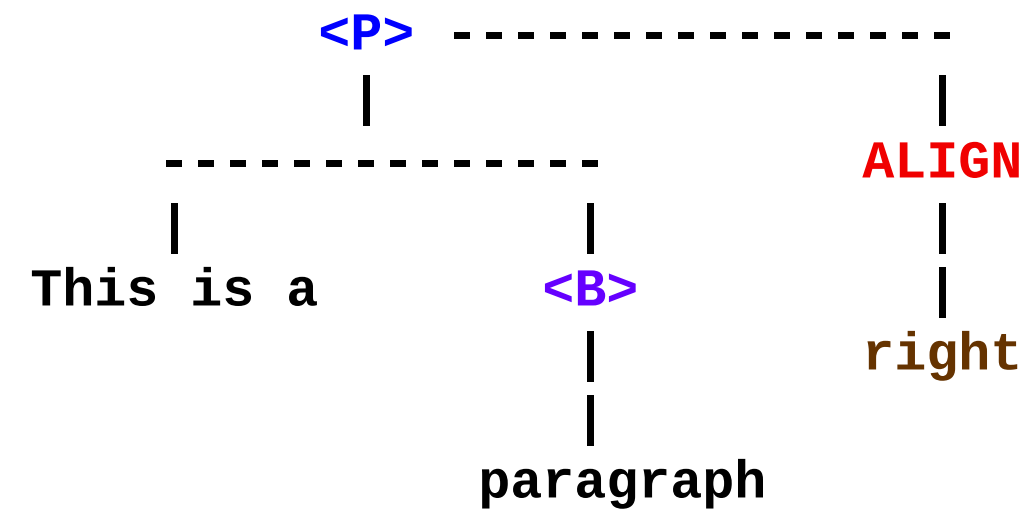
- By calling the element by its proper DOM name, we can access and modify the element.

Nodes

In the Level 1 DOM, each object, whatever it may be exactly, is a *Node*

```
<P ALIGN="right">This is a <B>paragraph</B></P>
```

would give something like



In an HTML document, element P would also have a parent.

Walking Through The DOM Tree

- Each node is modeled as an object.
- Each node (except the root) has a parent
  - **`x.parentNode`**
- Each node has zero or more children nodes
  - To get the # of child nodes
    - **`x.childNodes.length`** // childNodes is an array
  - To get the i^{th} child
    - **`x.childNodes[i-1]`** // First child has index 0
  - To get the first child, you can also write
    - **`x.firstChild`** or **`x.childNodes[0]`**
  - To get the last child
    - **`x.lastChild`**

Getting An Element

- To get an array of all the <p> elements
  - `document.getElementsByTagName("p")`
- To get the first <p> element
  - `var x = document.getElementsByTagName("p")[0];`

  or
  - ```
    var parray = document.getElementsByTagName("p");
    var x = parray[0];
    ```
- If you assigned an "id" attribute to the first <p> element like "<p id="someId">", then you can get the <p> element as
  - `var x = document.getElementById("someId");`
    ```

Nodes Properties and Methods

- A node that represents an element is called an element node.
- A node that represents only the text is called a text node

- An element node object has methods to
  - set/get attributes
  - add / insert / remove / replace child nodes
  - and more …

- An element node has properties which you can access/modify directly
  - id
  - innerHTML
  - and more …

- See http://www.quirksmode.org/?dom/contents.html for complete listing of W3C DOM Model

Methods for changing the structure of the document

- **`newNode = document.createElement("name")`**
  - Creates a new element node with the tag name "name"
- **`newNode = document.creatTextNode("string")`**
  - Creates a new text node with the node value of string
- **`node.appendChild(newNode)`**
  - Adds **`newNode`** as a new child node to node, following any existing children of node
- **`newNode = node.cloneNode(deep_copy)`**
  - Creates **`newNode`** as a copy of node.
  - If **`deep_copy`** is true, the clone includes clones of all the child nodes and attributes of the original

Methods for changing the structure of the document

- **`node.insertBefore(newNode, oldNode)`**
  - Inserts **`newNode`** as a new child node of node before **`oldNode`**
- **`node.removeChild(oldNode)`**
  - Removes the child **`oldNode`** from node
- **`node.replaceChild(newNode, oldNode)`**
  - Replaces the child node **`oldNode`** of node with **`newNode`**

The **`innerHTML`** Property

- A property (of type **`string`**) of an element node that represents the content of the element.
- Kept available for backward compatibilities
- Easier to use but less efficient

References

- W3C DOM -Introduction
  - http://www.quirksmode.org/?dom/intro.html
- HTML DOM Objects
  - http://www.w3schools.com/js/js_obj_htmldom.asp
- JavaScript/DOM Object Quick Reference
  - http://www.dannyg.com/ref/jsquickref.html