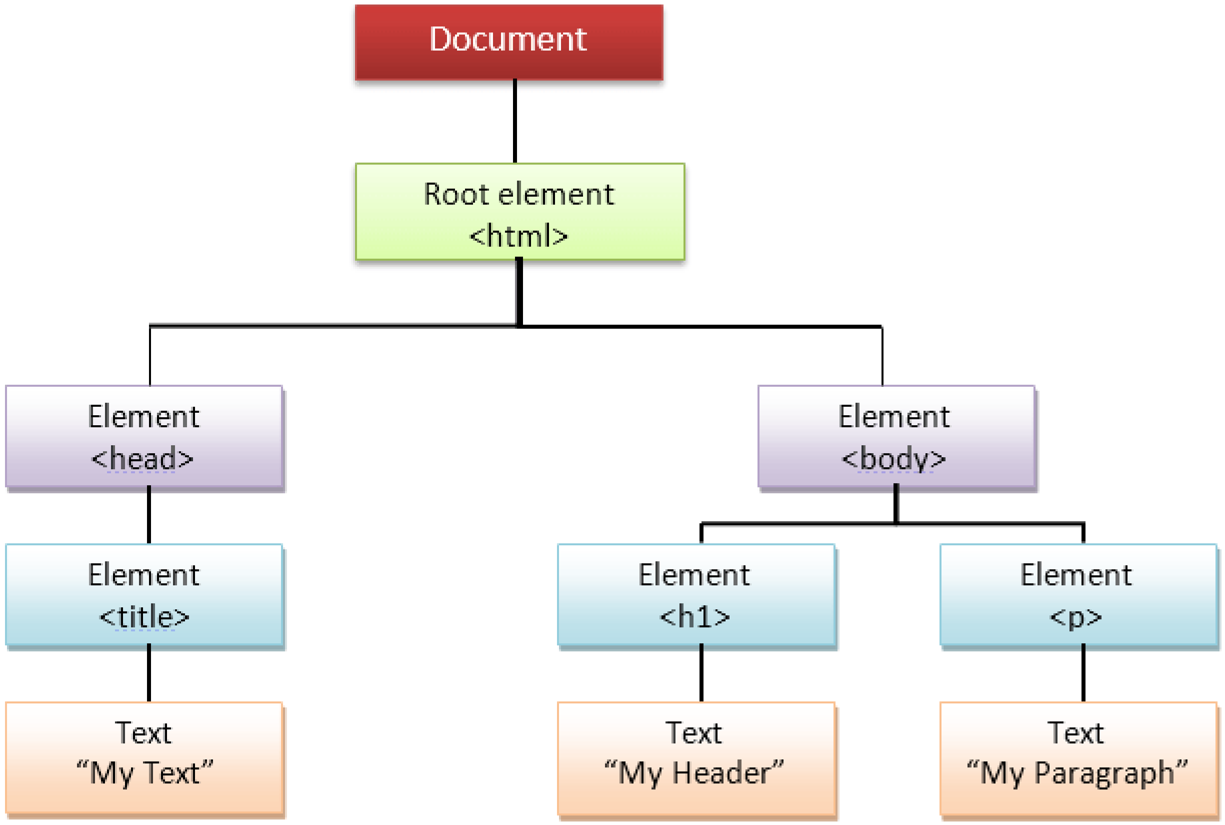
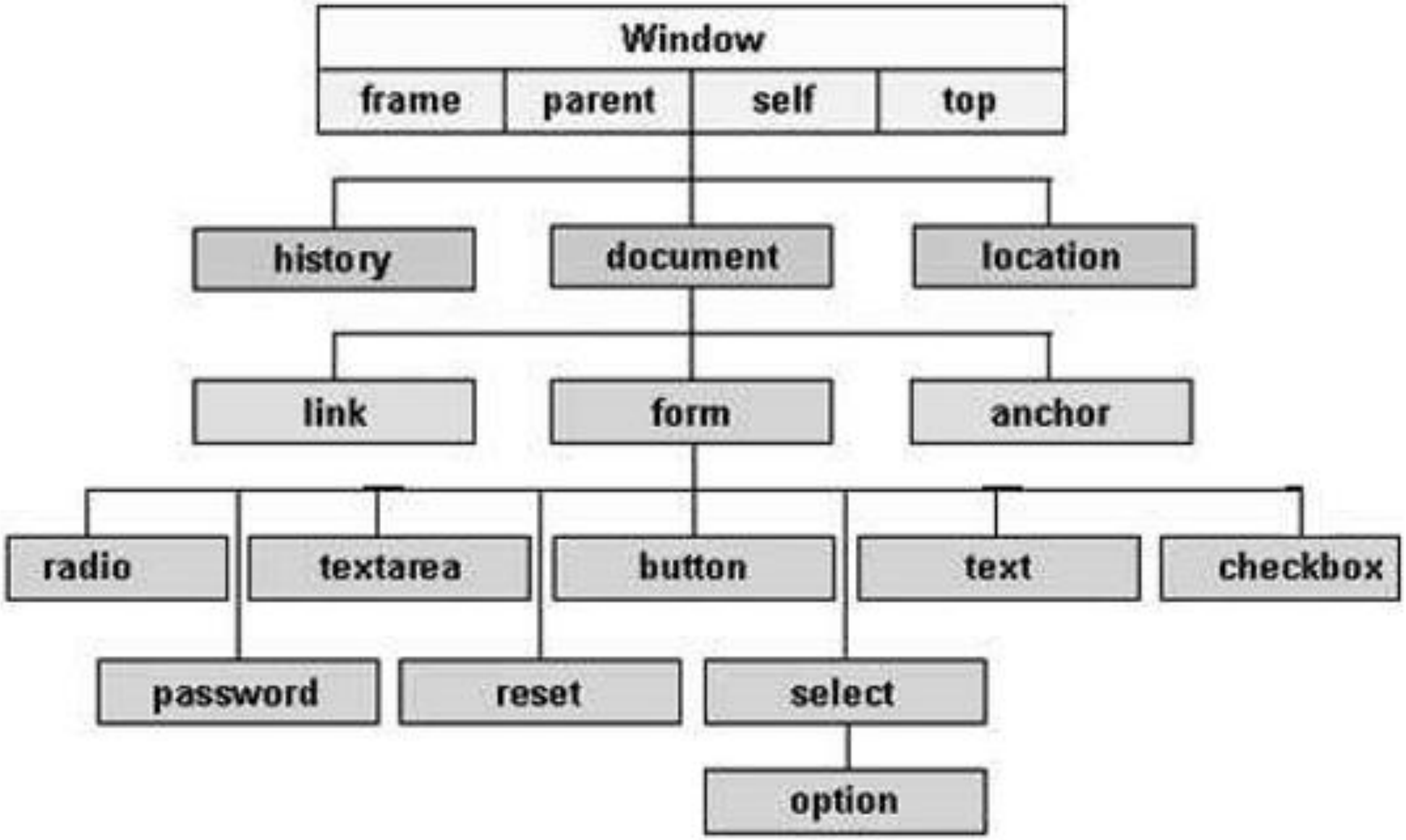
DOM

- *"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 W3C DOM standard is separated into 3 different parts:
- Core DOM - standard model for all document types
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

The HTML DOM (Document Object Model)

- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
- The **HTML DOM** model is constructed as a tree of **Objects**:

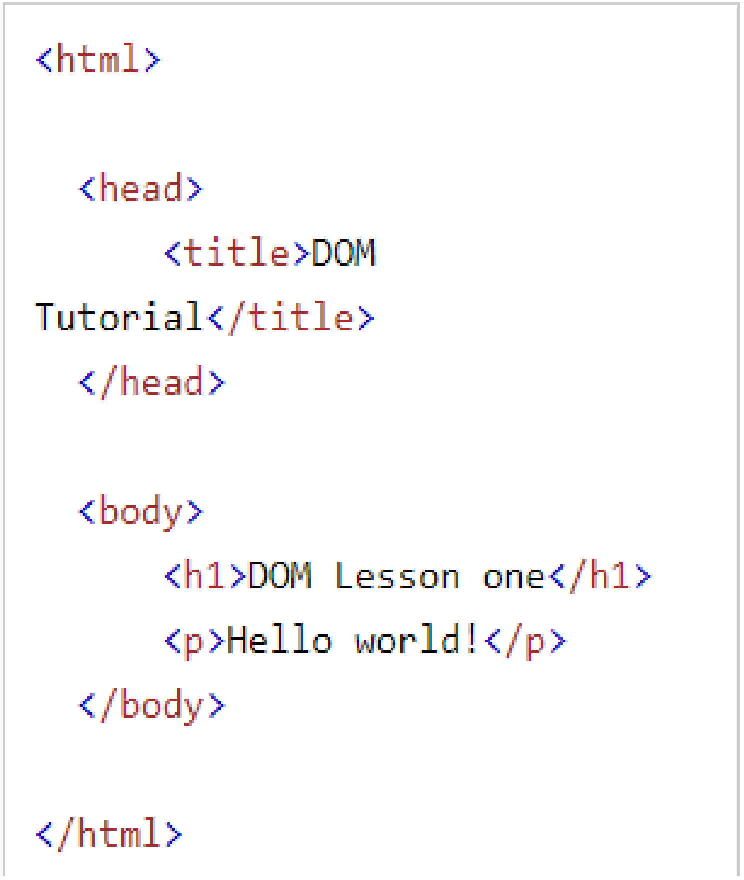




Window
frame
parent
self
top
history
document
location
link
form
anchor
radio
textarea
button
text
checkbox
password
reset
select
option

- Using DOM, JavaScript can perform multiple tasks.
- It can create new elements and attributes, change the existing elements and attributes and even remove existing elements and attributes.
- JavaScript can also react to existing events and create new events in the page.

Node Relationships

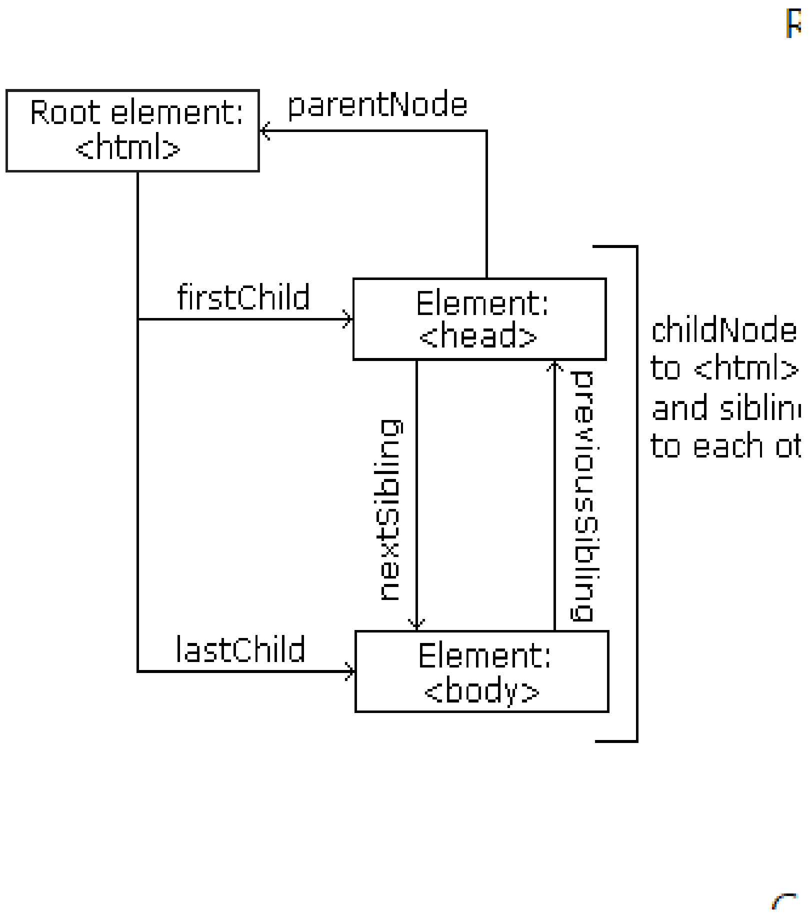


```
<html>

  <head>
      <title>DOM
Tutorial</title>
  </head>

  <body>
      <h1>DOM Lesson one</h1>
      <p>Hello world!</p>
  </body>

</html>
```



Node Relationships

- From the HTML we can say
  - <html> is the root node
  - <html> has no parents
  - <html> is the parent of <head> and <body>
  - <head> is the first child of <html>
  - <body> is the last child of <html>
  - <head> has one child: <title>
  - <title> has one child (a text node): "DOM Tutorial"
  - <body> has two children: <h1> and <p>
  - <h1> has one child: "DOM Lesson one"
  - <p> has one child: "Hello world!"
  - <h1> and <p> are siblings

Navigating Between Nodes

- You can use the following node properties to navigate between nodes with JavaScript:
  - parentNode
  - childNodes[*nodenumber*]
  - firstChild
  - lastChild
  - nextSibling
  - previousSibling

What is the HTML DOM?

- The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:
  - The HTML elements as **objects**
  - The **properties** of all HTML elements
  - The **methods** to access all HTML elements
  - The **events** for all HTML elements

  In other words: **The HTML DOM is a standard for how to get, change, add, or delete HTML elements.**

JavaScript - HTML DOM Methods

- HTML DOM methods are **actions** you can perform (on HTML Elements).
- HTML DOM properties are **values** (of HTML Elements) that you can set or change.

The DOM Programming Interface

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

Example

```
<html>
    <body>
    <p id="demo"></p>

    <script>
    document.getElementById("demo").innerHTML = "Hello World!";
    </script>

    </body>
    </html>
```

- In the example above, getElementById is a **method**, while innerHTML is a **property**.
- The innerHTML property can be used to get or change any HTML element, including <html> and <body>.

Finding HTML Elements

- To find the elements following are different ways:
  - Finding HTML elements by id
  - Finding HTML elements by tag name
  - Finding HTML elements by class name
  - Finding HTML elements by CSS selectors
  - Finding HTML elements by HTML object collections

Finding HTML Elements

Method	**Description**
document.getElementById()	Find an element by element id
document.getElementsByTagName()	Find elements by tag name
document.getElementsByClassName()	Find elements by class name

Changing HTML Elements

Method	**Description**
element.innerHTML=	Change the inner HTML of an element
element.attribute=	Change the attribute of an HTML element
element.setAttribute(attribute,value)	Change the attribute of an HTML element
element.style.property=	Change the style of an HTML element

Adding and Deleting Elements

Method	**Description**
document.createElement()	Create an HTML element
document.removeChild()	Remove an HTML element
document.appendChild()	Add an HTML element
document.replaceChild()	Replace an HTML element
document.write(text)	Write into the HTML output stream

Creating New HTML Elements (Nodes)

```
<html>
<body>

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

<script>
var para = document.createElement("p");
var node = document.createTextNode("This is new.");
para.appendChild(node);
var element = document.getElementById("div1");
element.appendChild(para);
</script>

</body>
</html>
```

```
This is a paragraph.
This is another paragraph.
This is new.
```

Creating New HTML Elements (Nodes)

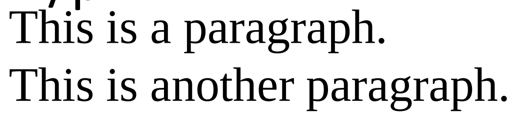
- This code creates a new <p> element:
  - var para = document.createElement("p");
- To add text to the <p> element, you must create a text node first. This code creates a text node:
  - var node = document.createTextNode("This is a new paragraph.");
- Then you must append the text node to the <p> element:
  - para.appendChild(node);
- Finally you must append the new element to an existing element.
- This code finds an existing element:
  - var element = document.getElementById("div1");
- This code appends the new element to the existing element:
  - element.appendChild(para);

Removing Existing HTML Elements

```
<!DOCTYPE html>
<html>
<body>

<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("p2");
parent.removeChild(child);
</script>
</body>
```



```
This is a paragraph.
This is another paragraph.
```

Adding Events Handlers

Method	**Description**
document.getElementById(id).onclick =function(){code}	Adding event handler code to an onclick event

JavaScript HTML DOM Events

- A JavaScript can be executed when an event occurs, like when a user clicks on an HTML element.
- To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute:

onclick=*JavaScript*

- Examples of HTML events:
  - When a user clicks the mouse
  - When a web page has loaded
  - When an image has been loaded
  - When the mouse moves over an element
  - When an input field is changed
  - When an HTML form is submitted
  - When a user strokes a key

Assign Events Using The Event Handler

```
<!DOCTYPE html>
<html>
<body>

<h1 onclick="changeText(this)">Click on this text!</h1>

<script>
function changeText(id) {
   id.innerHTML = "Ooops!";
}
</script>

</body>
</html>
```

Event Handler Function

```
<!DOCTYPE html>
<html>
<body>

<h1 onclick="this.innerHTML='Ooops!'">Click on this text!</h1>

</body>
</html>
```

Assign Events Using the HTML DOM

```
<!DOCTYPE html>
<html>
<body>
<button id="myBtn">Try it</button>
<p id="demo"></p>
<script>
document.getElementById("myBtn").onclick = displayDate;

function displayDate() {
    document.getElementById("demo").innerHTML = Date();
}
</script>
</body>
</html>
```

onclick

```
<html>
</head>
<body>
<script>
function displayDate() {
    document.getElementById("demo").innerHTML = Date();
}
</script>
<p id="demo“></p>
<button onclick=displayDate()>Try it</button>

</body>
</html
```

ondblclick

```
<html>
</head>
<body>
<script>
function displayDate() {
   document.getElementById("demo").innerHTML = Date();
}
</script>
<p id="demo">hyttyu</p>
<button ondblclick=displayDate()>Try it</button>

</body>
</html>
```

onload

```
<html>
<head>
</head>
<body onload=fun()>
<p> This is paragraph</p>
<script>
function fun() {
   document.write("page has been loaded successfully");
}
</script>
<p> This is paragraph</p>
</body>
</html>
```

Mouse Events

- The onmouseover and onmouseout Events
  - The onmouseover and onmouseout events can be used to trigger a function when the user put mouse over, or out of, an HTML element
- The onmousedown, onmouseup and onclick Events
  - The onmousedown, onmouseup, and onclick events are all parts of a mouse-click.
  - First when a mouse-button is clicked, the onmousedown event is triggered, then, when the mouse-button is released, the onmouseup event is triggered, finally, when the mouse-click is completed, the onclick event is triggered.

Event Handler

- An **event handler** typically is a software routine that processes actions such as keystrokes and mouse movements. With Web sites, **event handlers** make Web content dynamic. JavaScript is a common method of scripting **event handlers** for Web content.

onchange

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
  var x = document.getElementById("fname");
  x.value = x.value.toUpperCase();
}
</script>
</head>
<body>

Enter your name: <input type="text" id="fname" onchange="myFunction()">

</body>
```