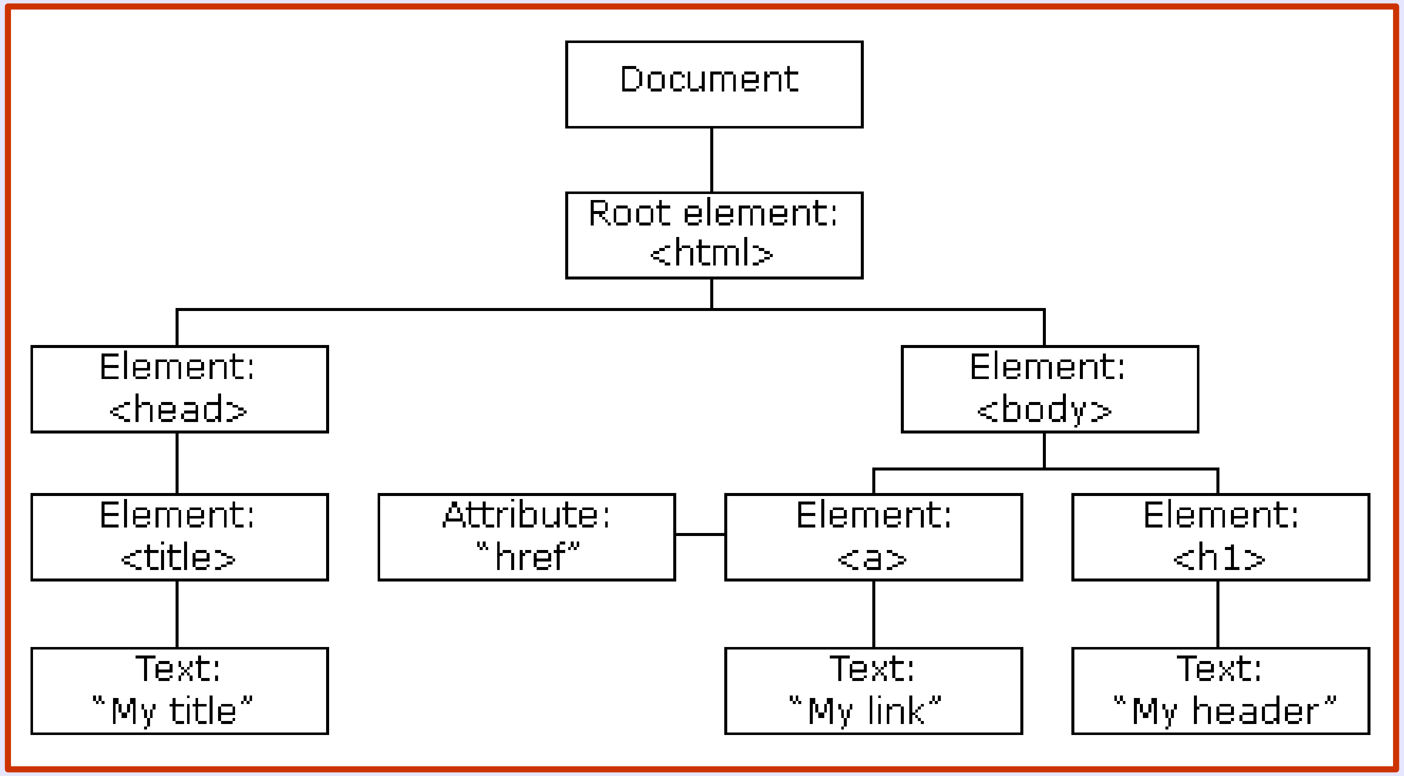
The HTML DOM

When a web page is loaded, the browser creates a **Document Object Model** of the page

The HTML DOM model is constructed as a tree of Objects:



The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

The HTML DOM can be accessed with JavaScript!

JavaScript can :

- change all the HTML elements in the page
- change all the HTML attributes in the page
- change all the CSS styles in the page
- remove existing HTML elements and attributes
- add new HTML elements and attributes
- react to all existing HTML events in the page
- create new HTML events in the page

JavaScript HTML DOM Navigation

According to the W3C HTML DOM standard,
everything in an HTML document is a node:

- The entire document is a document node
- Every HTML element is an element node
- The text inside HTML elements are text nodes
- Every HTML attribute is an attribute node
- All comments are comment nodes

With the HTML DOM, all nodes in the node tree can be accessed by JavaScript.

Nodes can be created, modified or deleted.

Node Relationships

- The nodes in the node tree have a hierarchical relationship to each other.
- The terms parent, child, and sibling are used to describe the relationships.
- In a node tree, the top node is called the root (or root node)
- Every node has exactly one parent, except the root (which has no parent)
- A node can have a number of children
- Siblings (brothers or sisters) are nodes with the same parent

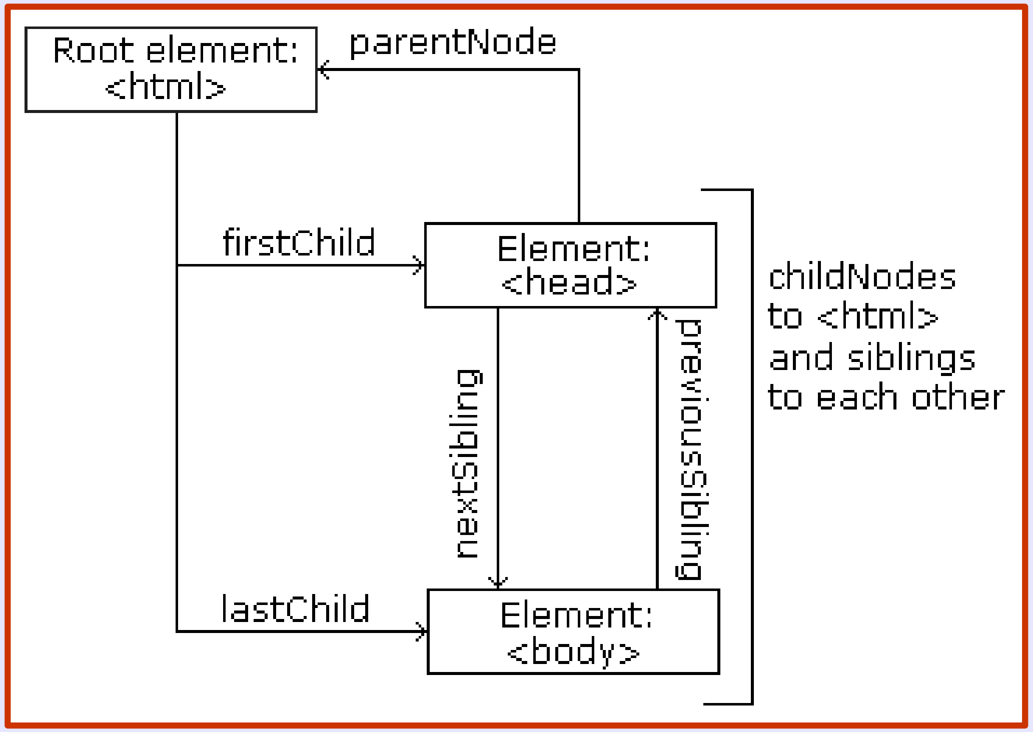
For example:

```
<html>

  <head>
    <title> Presentazione </title>
  </head>

  <body>
    <h1> Presentazione HTML5  </h1>
    <p> Hello world! </p>
  </body>

</html>
```



Navigating Between Nodes

Properties to navigate between nodes with JavaScript:

- parentNode
- childNodes[nodenumber]
- firstChild
- lastChild
- nextSibling
- previousSibling

Using them you can get access to existing elements through the DOM.

But you can also use the DOM to **create new elements**, and then, **add them to the DOM**.

How to create a new element

(HTML)

```
<!doctype html>


<head>

<title> Playlist </title>

<meta charset="utf-8">

<script src="playlist.js"></script>

<link rel="stylesheet"
href="playlist.css">

</head>
```

```
<body>

<form>

<input type="text" id="songTextInput"
size="40" placeholder="Song name">

<input type="button" id="addButton"
value="Add Song">

</form>

<ul id="playlist">

</ul>

</body>

</html>
```

How to create a new element (JavaScript)

```
window.onload = init;

function init() {

var button =
document.getElementById("add
Button");

button.onclick =  function()  {

var textInput =
document.getElementById("son
gTextInput");

var songName = textInput.value;

if (songName == "") {

alert("Please enter a song");

}

else {

var li = document.createElement("li");

li.innerHTML = songName;

var ul =
document.getElementById("playlist");

ul.appendChild(li); }

}
```

...and add it to the DOM

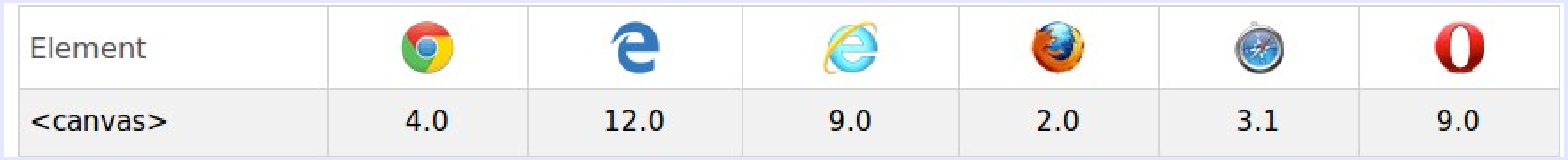
What is HTML Canvas?

The HTML <canvas> element is used to **draw graphics** on a web page **via scripting** (usually JavaScript).

The <canvas> element is only a container for graphics. You must use a script to actually draw the graphics.

Canvas has several methods for drawing paths, circles, text, and adding images.

Browser Support



Element						
<canvas>	4.0	12.0	9.0	2.0	3.1	9.0

What is HTML Canvas?

A canvas is a rectangular area on an HTML page.
By default, a canvas has no border and no content.

The markup looks like this:

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

Always specify an id attribute (to be referred to in a script), and a width and height attribute to define the size of the canvas.

Border and other details can be added using CSS.

For example:

HTML:

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

JavaScript:

```
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.fillStyle = "green";
ctx.fillRect(10, 10, 100, 100);
```