Course Module 1: Course Unit 7: JavaScript (Document Object Model)

Week 9

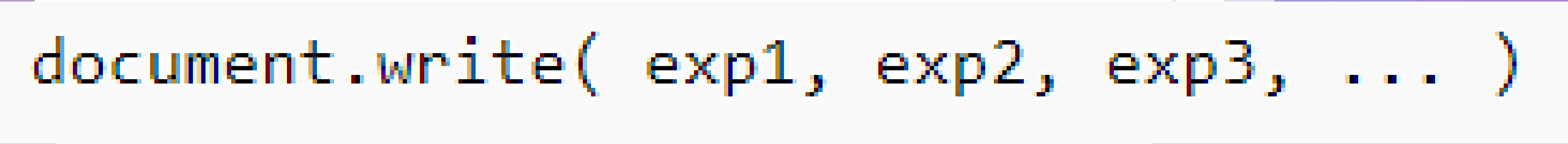
Methods of Document Object Model (DOM)

Methods of DOM

- write(“string”): Writes the given string on the document.
- getElementById(): returns the element having the given id value.
- getElementsByName(): returns all the elements having the given name value.
- getElementsByTagName(): returns all the elements having the given tag name.
- getElementsByClassName(): returns all the elements having the given class name.

DOM write() Method

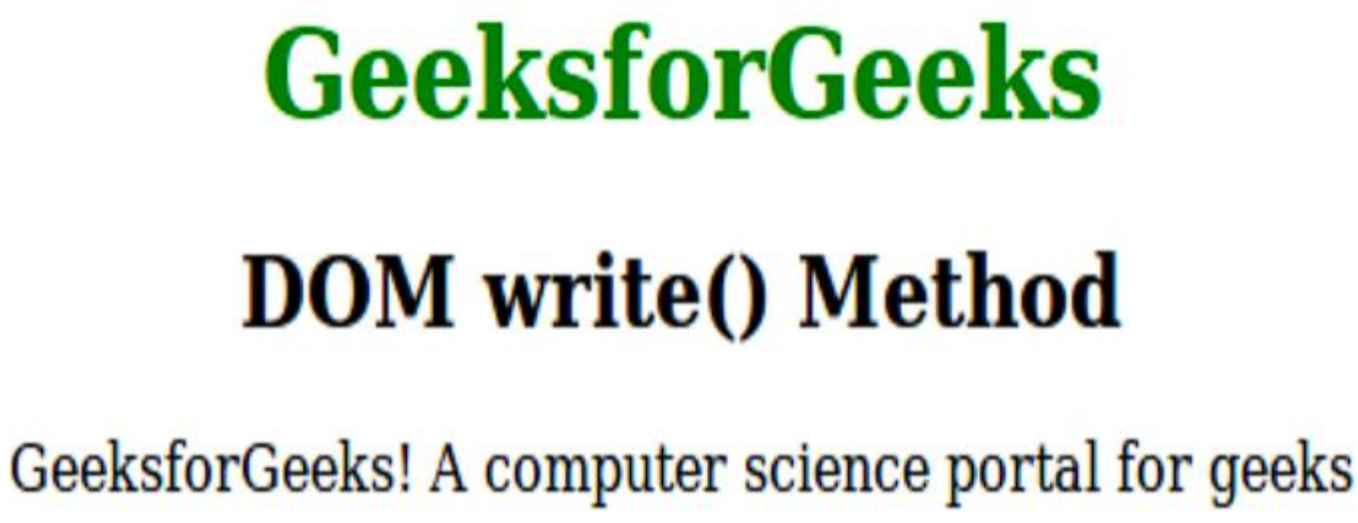
- The write() method in HTML is used to write some content or JavaScript code in a Document. This method is mostly used for testing purpose. It is used to delete all the content from the HTML document and inserts the new content. It is also used to give the additional text to an output which is open by the document.open() method. This method is quite similar to writeln() method.



```
document.write( exp1, exp2, exp3, ... )
```

DOM write() Method

```
<!DOCTYPE html>
<html>
    <head>
        <title>DOM write() Method</title>
        <style>
            body {
                text-align:center;
            }
            h1 {
                color:green;
            }
        </style>
    </head>
    <body>
        <h1>GeeksforGeeks</h1>
        <h2>DOM write() Method</h2>
        <script>
            document.write("GeeksforGeeks! ");
            document.write
                ("A computer science portal for geeks")
        </script>
    </body>
</html>
```



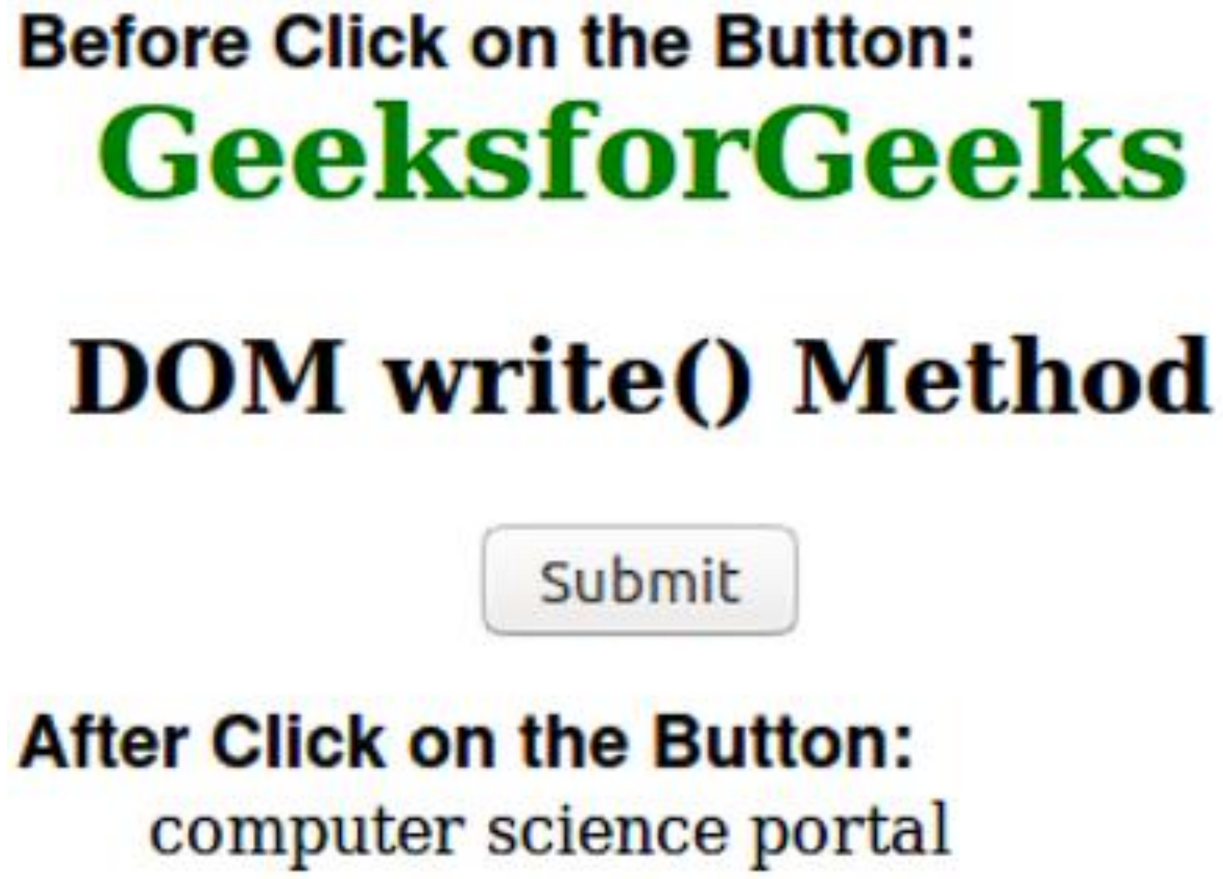
GeeksforGeeks

DOM write() Method

GeeksforGeeks! A computer science portal for geeks

DOM write() Method

```
<!DOCTYPE html>
<html>
    <head>
        <title>DOM write() Method</title>
        <style>
            body {
                text-align:center;
            }
            h1 {
                color:green;
            }
        </style>
    </head>
    <body>
        <h1>GeeksforGeeks</h1>
        <h2>DOM write() Method</h2>
        <button type="button" onclick="Geeks()">
            Submit
        </button>
        <script>
        function Geeks() {
            document.write
                ('<center>computer science portal</center>');
        }
        </script>
    </body>
</html>
```



Before Click on the Button:

GeeksforGeeks

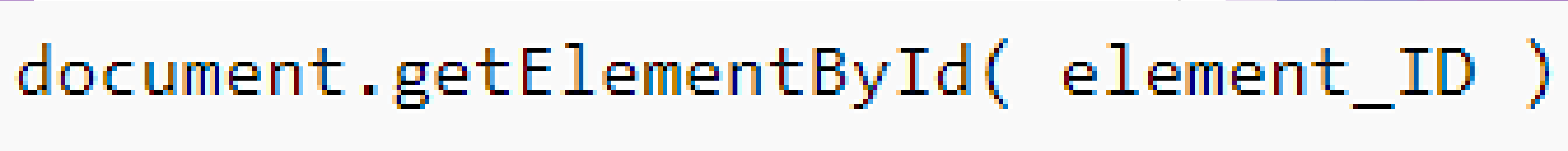
DOM write() Method

After Click on the Button:

computer science portal

DOM getElementById() Method

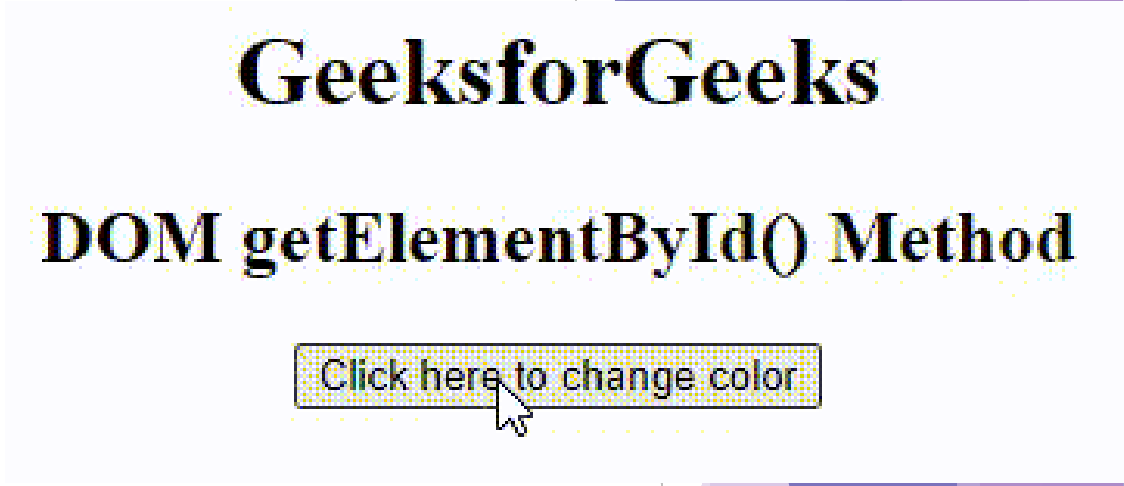
- The getElementById() method returns the elements that have given an ID which is passed to the function. This function is a widely used HTML DOM method in web designing to change the value of any particular element or get a particular element. If the passed ID to the function does not exist then it returns null. The element is required to have a unique id, in order to get access to that specific element quickly, & also that particular id should only be used once in the entire document.

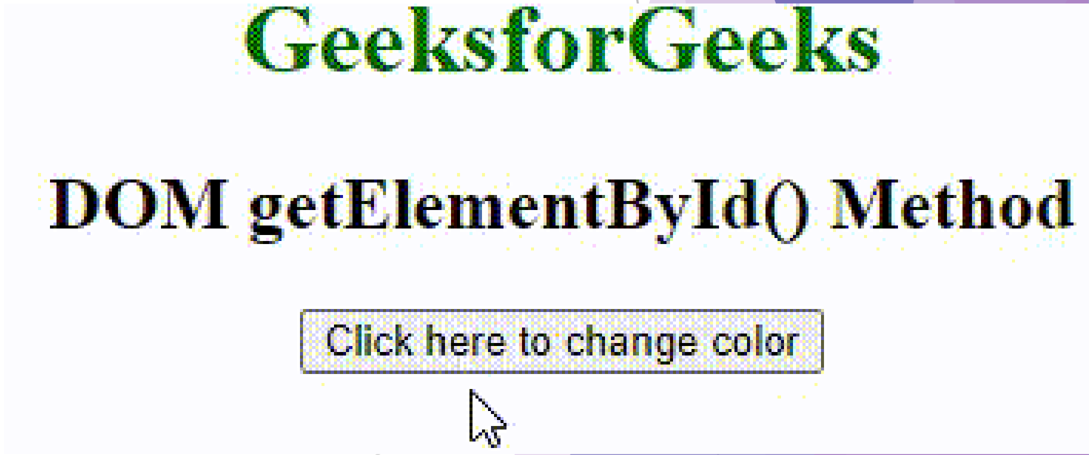


```
document.getElementById( element_ID )
```

DOM getElementById() Method

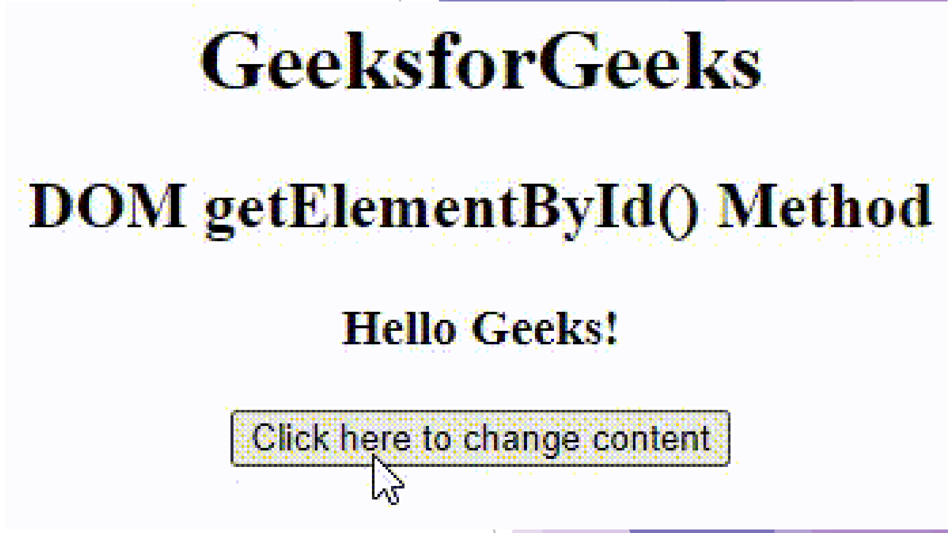
```
<!DOCTYPE html>
<html>

<head>
    <title>
        DOM getElementById() Method
    </title>

    <script>

        // Function to change the color of element
        function geeks() {
            var demo = document.getElementById("geeks");
            demo.style.color = "green";
        }
    </script>
</head>

<body style="text-align:center">
    <h1 id="geeks">GeeksforGeeks</h1>
    <h2>DOM getElementById() Method</h2>

    <!-- Click on the button to change color -->
    <input type="button"
           onclick="geeks()"
           value="Click here to change color" />
</body>

</html>
```





DOM getElementById() Method

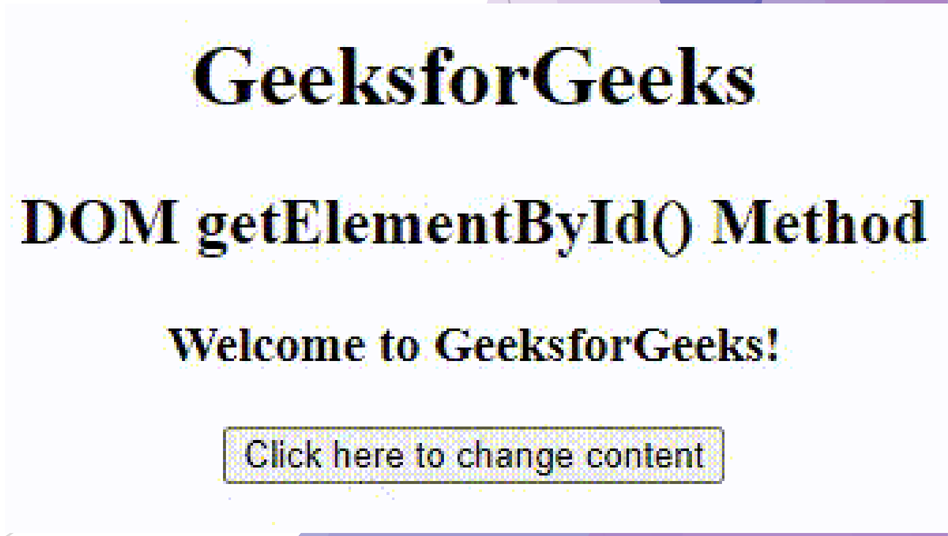
```
<!DOCTYPE html>
<html>

<head>
    <title>
        DOM getElementById() Method
    </title>

    <script>

        // Function to change content of element
        function geeks() {
            var demo = document.getElementById("geeks");
            demo.innerHTML = "Welcome to GeeksforGeeks!";
        }
    </script>
</head>

<body style="text-align:center">
    <h1>GeeksforGeeks</h1>
    <h2>DOM getElementById() Method</h2>
    <h3 id="geeks">Hello Geeks!</h3>

    <!-- Click here to change content -->
    <input type="button"
           onclick="geeks()"
           value="Click here to change content" />
</body>

</html>
```



GeeksforGeeks

DOM getElementById() Method

Hello Geeks!



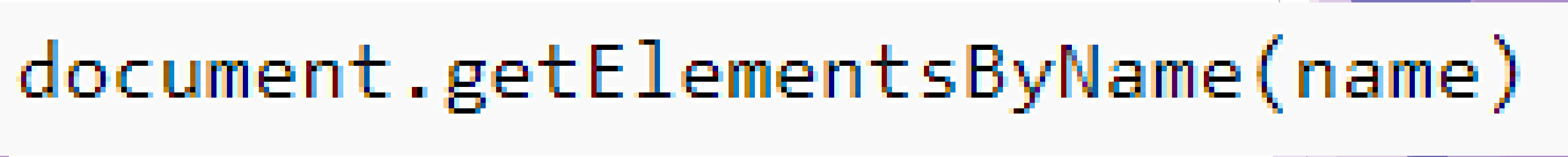
GeeksforGeeks

DOM getElementById() Method

Welcome to GeeksforGeeks!

DOM getElementsByName() Method

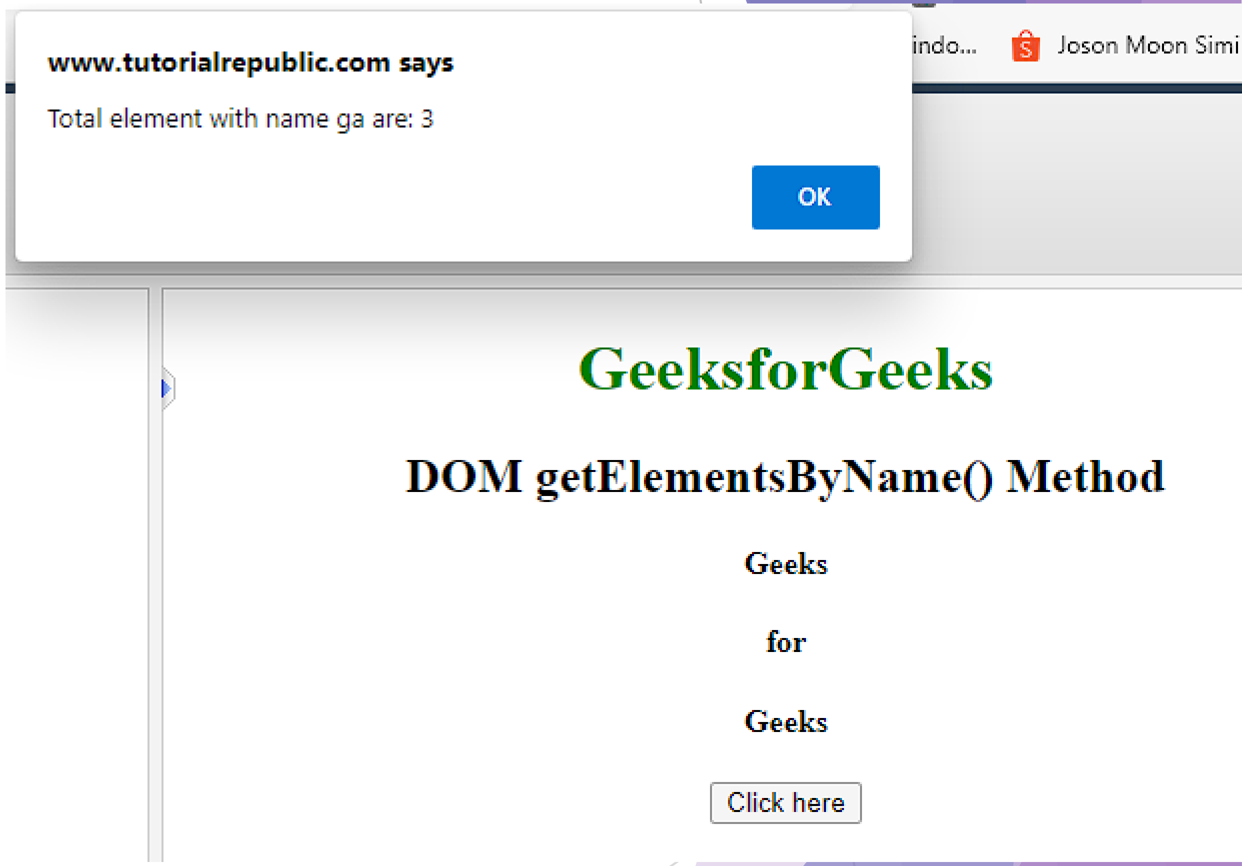
- The getElementsByName() method returns collection of all elements of particular document by name. This collection is called node list and each element of the node list can be visited with the help of the index.



```
document.getElementsByName(name)
```

DOM getElementsByName() Method

```
<!DOCTYPE html>
<html>

<head>
    <title>DOM getElementsByName()</title>
    <style>
        body {text-align: center;}
        h1 {color: green;}
    </style>
    <script>
        // creating geeks function to display
        // number of elements at particular name
        function geeks() {
            // taking list of elements under name ga
            var x = document.getElementsByName("ga");
          // printing number of elements inside alert tag
          alert("Total element with name ga are: " + x.length);
        }
    </script>
</head>

<body>
    <h1>GeeksforGeeks</h1>
    <h2>DOM getElementsByName() Method</h2>
    <!-- creating tag with name ga -->
    <h4 name="ga">Geeks</h4>
    <h4 name="ga">for</h4>
    <h4 name="ga">Geeks</h4>
    <input type="button" onclick="geeks()"
                         value="Click here" />
</body>

</html>
```



www.tutorialrepublic.com says

Total element with name ga are: 3

GeeksforGeeks

DOM getElementsByName() Method

Geeks

for

Geeks

DOM getElementsByTagName() Method

- The getElementsByTagName() method in HTML returns the collection of all the elements in the document with the given tag name. To extract any info just iterate through all the elements using the length property.

```
var elements = document.getElementsByTagName(name);
```

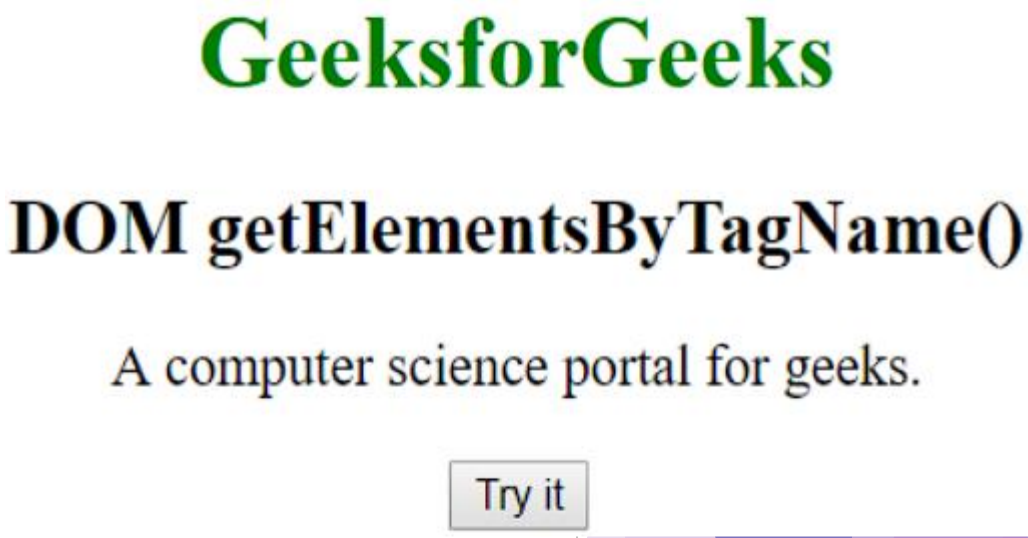
Where:

•**elements** is a collection of all the found elements in the order they appear with the given tag name.

•**name** is a string representing the name of the elements. The special string “*” represents all elements.

DOM getElementsByTagName() Method

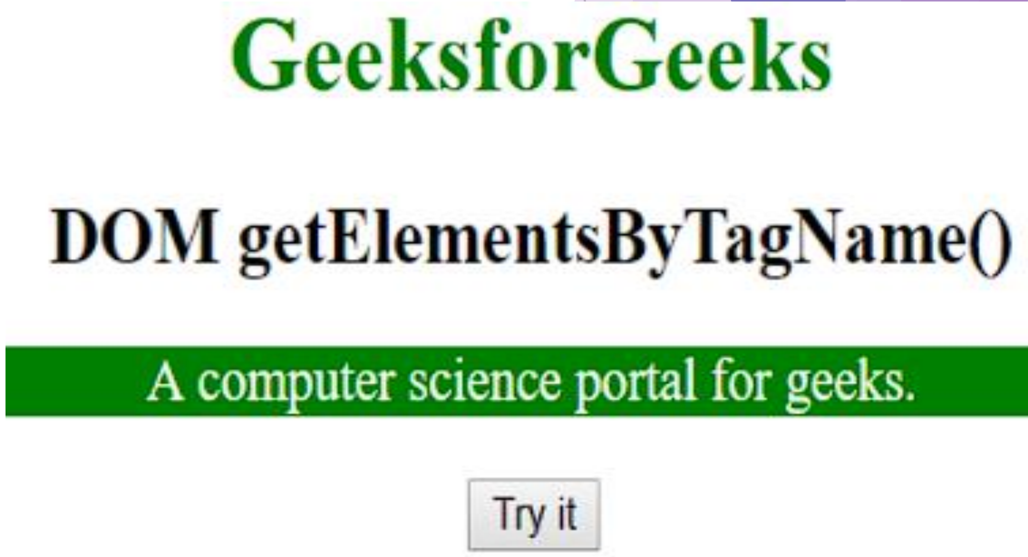
```
<!DOCTYPE html>
<html>
    <head>
        <title>DOM getElementsByTagName() Method</title>
    </head>
    <body style = "text-align: center">
        <h1 style = "color: green;">
            GeeksforGeeks
        </h1>

        <h2 >
            DOM getElementsByTagName()
        </h2>

        <p>A computer science portal for geeks.</p>

        <button onclick="geek()">Try it</button>
        <script>
        function geek() {
          var doc = document.getElementsByTagName("p");
          doc[0].style.background = "green";
          doc[0].style.color = "white";
        }
        </script>

    </body>
</html>
```



GeeksforGeeks

DOM getElementsByTagName()

A computer science portal for geeks.



GeeksforGeeks

DOM getElementsByTagName()

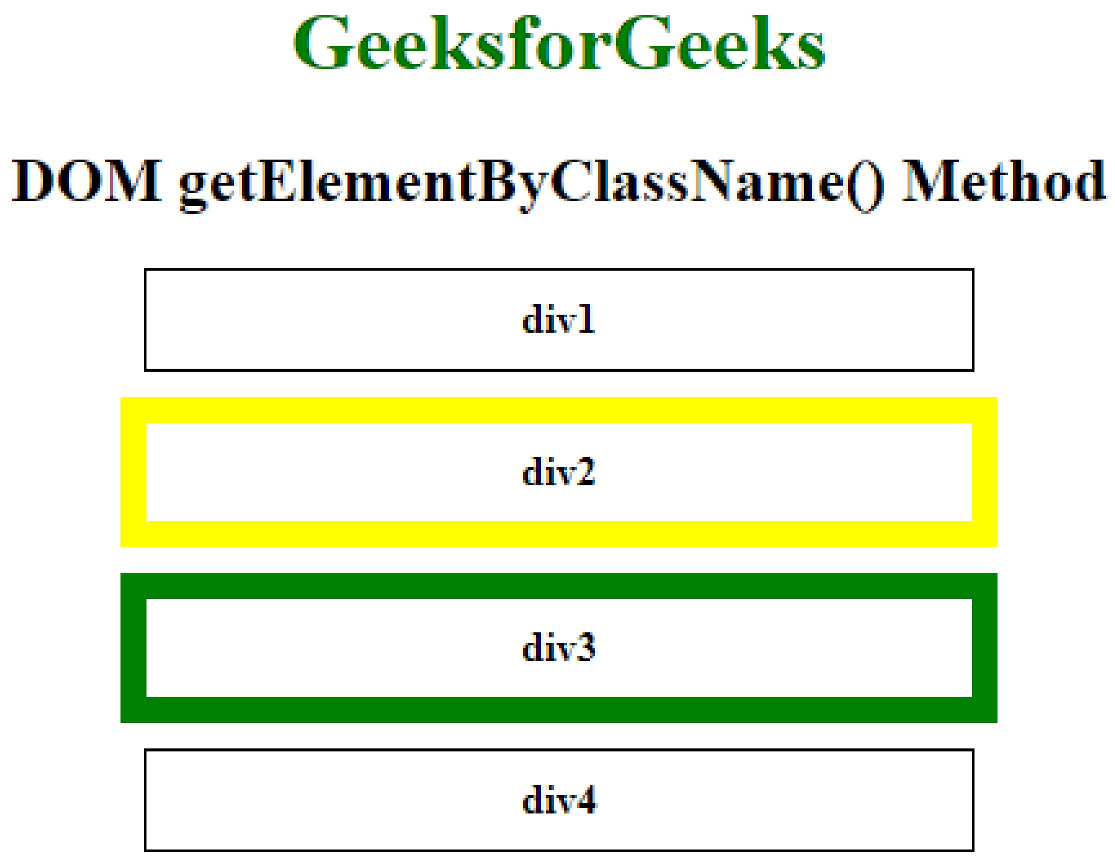
A computer science portal for geeks.

DOM getElementsByClassName() Method

- The getElementsByClassName() method in Javascript returns an object containing all the elements with the specified class names in the document as objects. Each element in the returned object can be accessed by its index. The index value will start with 0. This method can be called upon any individual element to search for its descendant elements with the specified class names.

```
document.getElementsByClassName(classnames);
```

```
        margin: auto;
        margin-top: 10px;
        border: 1px solid black;
        width: 300px;
    }
    </style>
</head>

<body>
    <h1>GeeksforGeeks</h1>
    <h2>DOM getElementByClassName() Method</h2>
    <div>
        <h4 class="example">div1</h4>
        <h4 class="yellowBorder example">div2</h4>
        <h4 class="greenBorder example">div3</h4>
        <h4 class="example">div4</h4> </div>
    <script>
    document.getElementsByClassName('greenBorder example')
[0].style.border =
    "10px solid green";
    document.getElementsByClassName('yellowBorder example')
[0].style.border =
    "10px solid yellow";
    </script>
</body>
</html>
```



Document Object Model (DOM) Architecture

DOM Architecture

- The DOM Architecture is divided into various modules. Each module addresses a particular domain. Domains covered by the current DOM API are XML, HTML, Cascading Style Sheets (CSS), and tree events. Future domains can be the rendered content (that is, the content displayed on the screen which might differ from the input document), user agent function, etc.

DOM Architecture

DOM Core

- The DOM Core defines a tree-like representation of the document, also referred as the DOM tree, enabling the user to traverse the hierarchy of elements accordingly.

- Refer also to the DOM Range and Traversal modules to manipulate the tree elements/structure defined in the DOM Core.

DOM Architecture

DOM XML

- The XML DOM extends the Core platform for specific XML 1.0 needs, such as processing instructions, CDATA, and entities.

DOM Architecture

DOM HTML

- The HTML DOM defines a set of convenient easy to use ways to manipulate HTML documents. The initial HTML DOM only describes methods, such as how to access an identifier by name, or a particular link. The HTML DOM is sometimes referred to as DOM Level 0 but has been imported into DOM Level 1.

DOM Architecture

DOM Events

- This part defines XML-tree manipulation oriented events with tree mutation and user-oriented events such as mouse, keyboard, and HTML-specific events.

DOM Architecture

DOM Cascading Style Sheets

- The DOM CSS defines a set of convenient, easy to use ways to manipulate CSS style sheets or the formatting of documents.

DOM Architecture

DOM Load and Save

- Loading an XML document into a DOM tree or saving a DOM tree into an XML document is a fundamental need for the DOM user. This module includes a variety of options controlling load and save operations.

DOM Architecture

DOM Validation

- This module defines a set of methods to modify the DOM tree and still make it valid.

DOM Architecture

DOM XPath

- The DOM XPath defines a set of convenient, easy to use functions to query a DOM tree using an XPath 1.0 expression, such as evaluate.

Levels of Document Object Model (DOM)

Levels of DOM

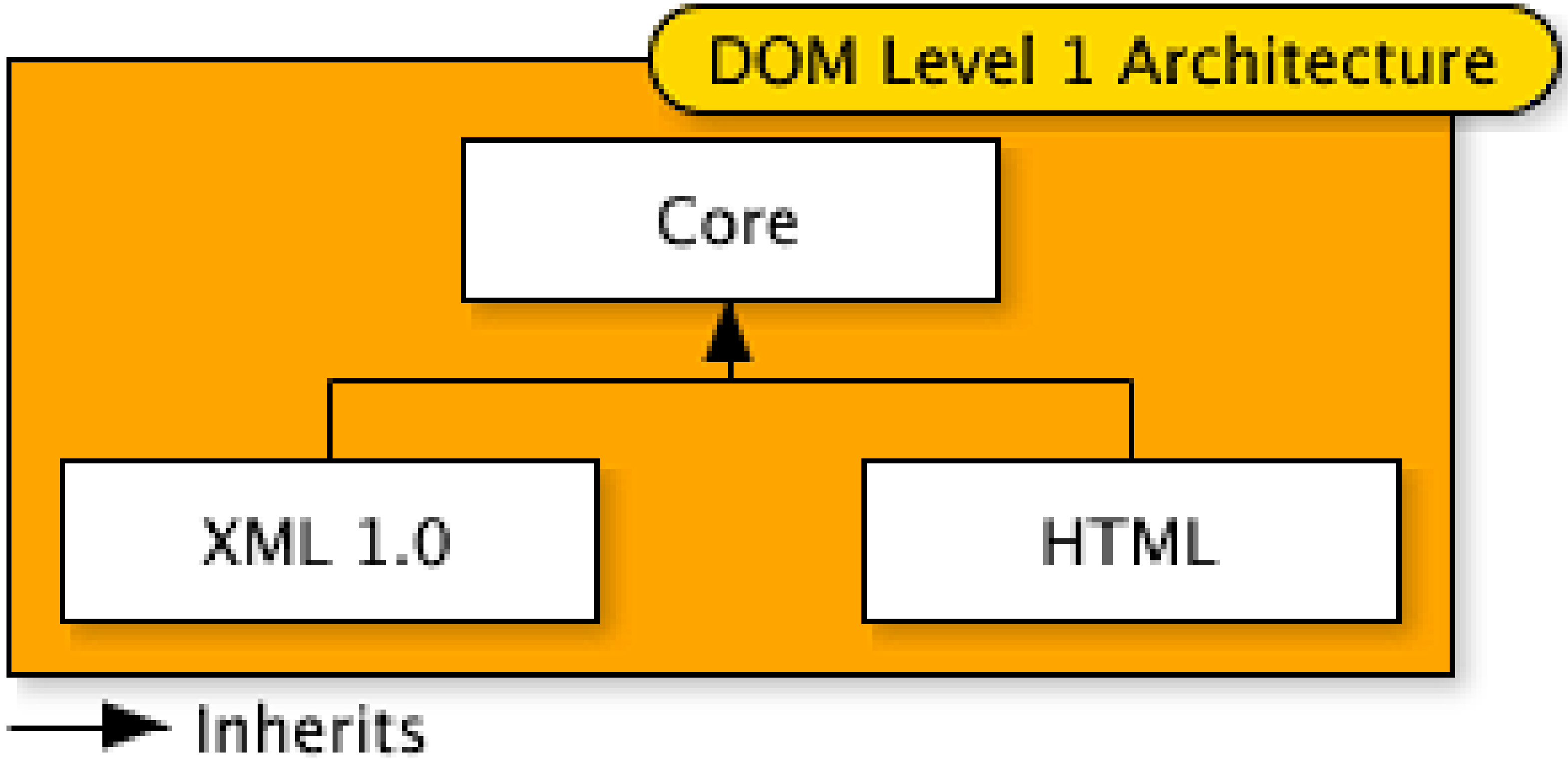
DOM Level 0

- ➢ Functionalities equivalent to the ones exposed in Netscape Navigator 3.0 and Microsoft Internet Explorer 3.0 are informally referred to as "Level 0". There is no W3C specification for this Level.
- ➢ Provides a low-level set of interfaces.

Levels of DOM

DOM Level 1

- DOM Level 1 was completed in October 1998 and provides support for XML 1.0 and HTML 4.0.



Levels of DOM

DOM Level 1

- CORE provides low-level interfaces that can be used to represent any structured document.
- HTML provides high-level interfaces that can be used to represent HTML documents.
- XML provides high-level interfaces that can be used to represent XML documents.

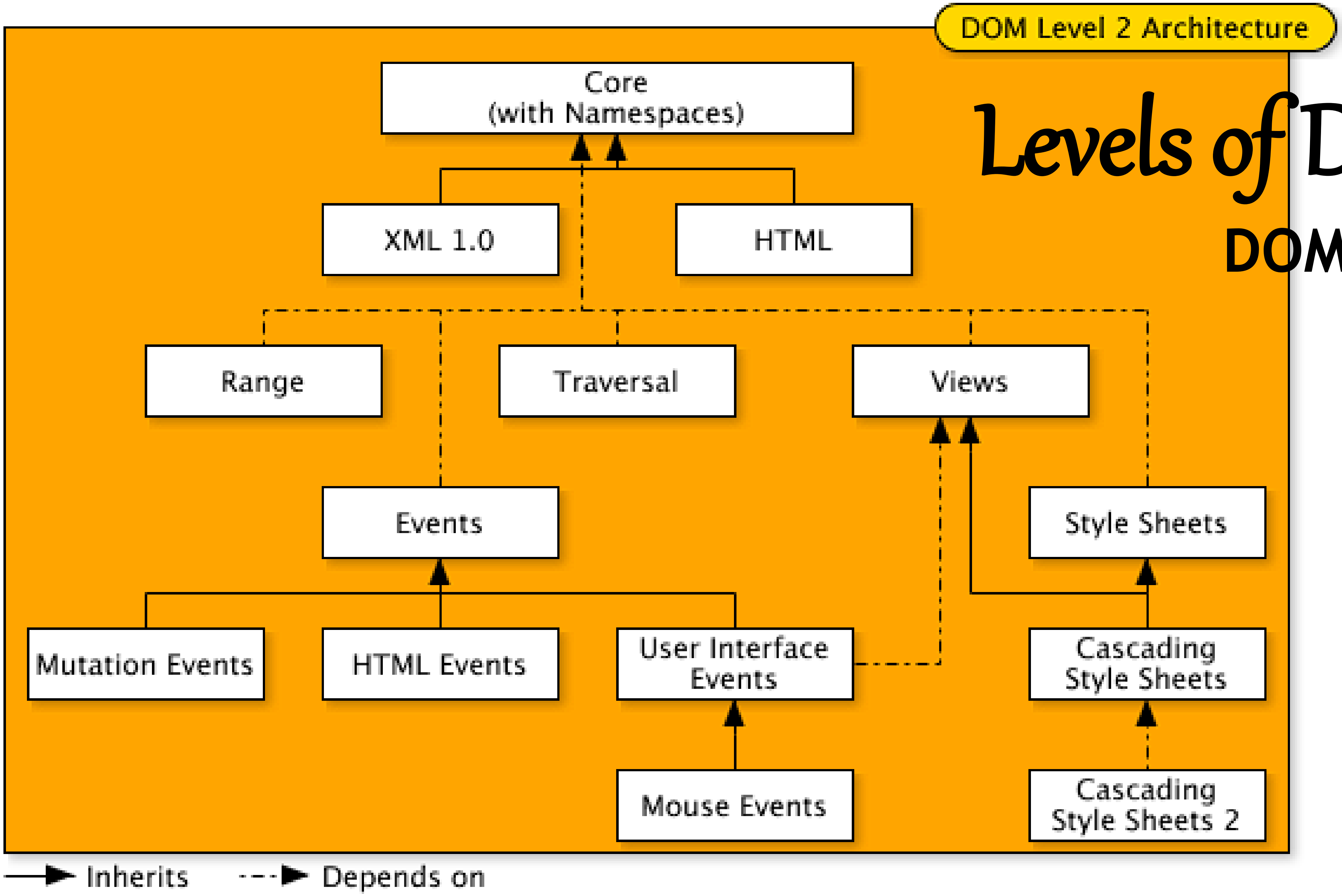
Levels of DOM

DOM Level 2

- DOM Level 2 was completed in November 2000 extending Level 1 with support for XML 1.0 with *namespaces*, adding supports for *Cascading Style Sheets (CSS)*, events such as user interface events and tree manipulation events, and enhancing tree manipulation methods (tree ranges and traversal mechanisms). Level 2 HTML is a W3C Recommendation since January 2003.

Levels of DOM

DOM Level 2



Levels of DOM

DOM Level 2

- CORE2: extends the functionality of CORE specified by DOM level 1.
- VIEWS: views allows programs to dynamically access and manipulate the content of the document.
- EVENTS: Events are scripts that are either executed by the browser when the user reacts to the web page.

Levels of DOM

DOM Level 2

- STYLE: allows programs to dynamically access and manipulate the content of style sheets.
- TRAVERSAL: This allows programs to dynamically traverse the document.
- RANGE: This allows programs to dynamically identify a range of content in the document.

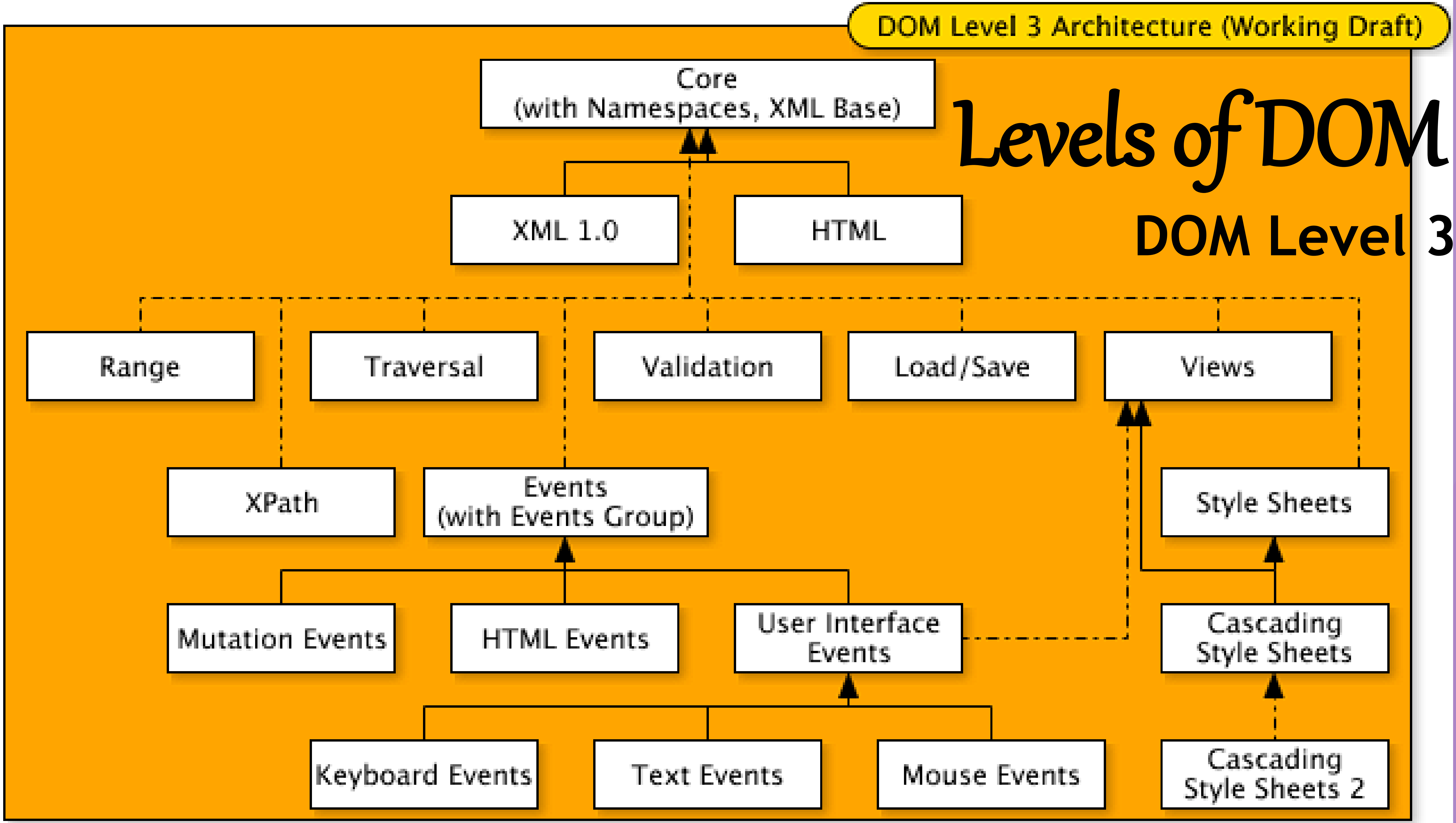
Levels of DOM

DOM Level 3

- DOM Level 3 is currently under development. Level 3 will extend Level 2 by finishing support for XML 1.0 with namespaces aligning the DOM Core with the *XML Infoset*, adding support for *XML Base*, and extending the user interface events (keyboard). Level 3 will also add support for validation, the ability to load and save a document, explore further mixed markup vocabularies and their implications on the DOM API ("Embedded DOM"), and will support *XPath*.

Levels of DOM

DOM Level 3



Levels of DOM

DOM Level 3

- CORE3: extends the functionality of CORE specified by DOM level 2.
- LOAD and SAVE: This allows the program to dynamically load the content of the XML document into the DOM document and save the DOM Document into an XML document by serialization.

Levels of DOM

DOM Level 3

- VALIDATION: This allows the program to dynamically update the content and structure of the document while ensuring the document remains valid.
- EVENTS: extends the functionality of Events specified by DOM Level 2.
- XPATH: XPATH is a path language that can be used to access the DOM tree.

Document Object Model (DOM) Nodes

DOM Nodes

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 (deprecated)
- All comments are comment nodes

With the HTML DOM, all nodes in the node tree can be accessed by JavaScript.
New nodes can be created, and all nodes can be modified or deleted.

Node Relationships

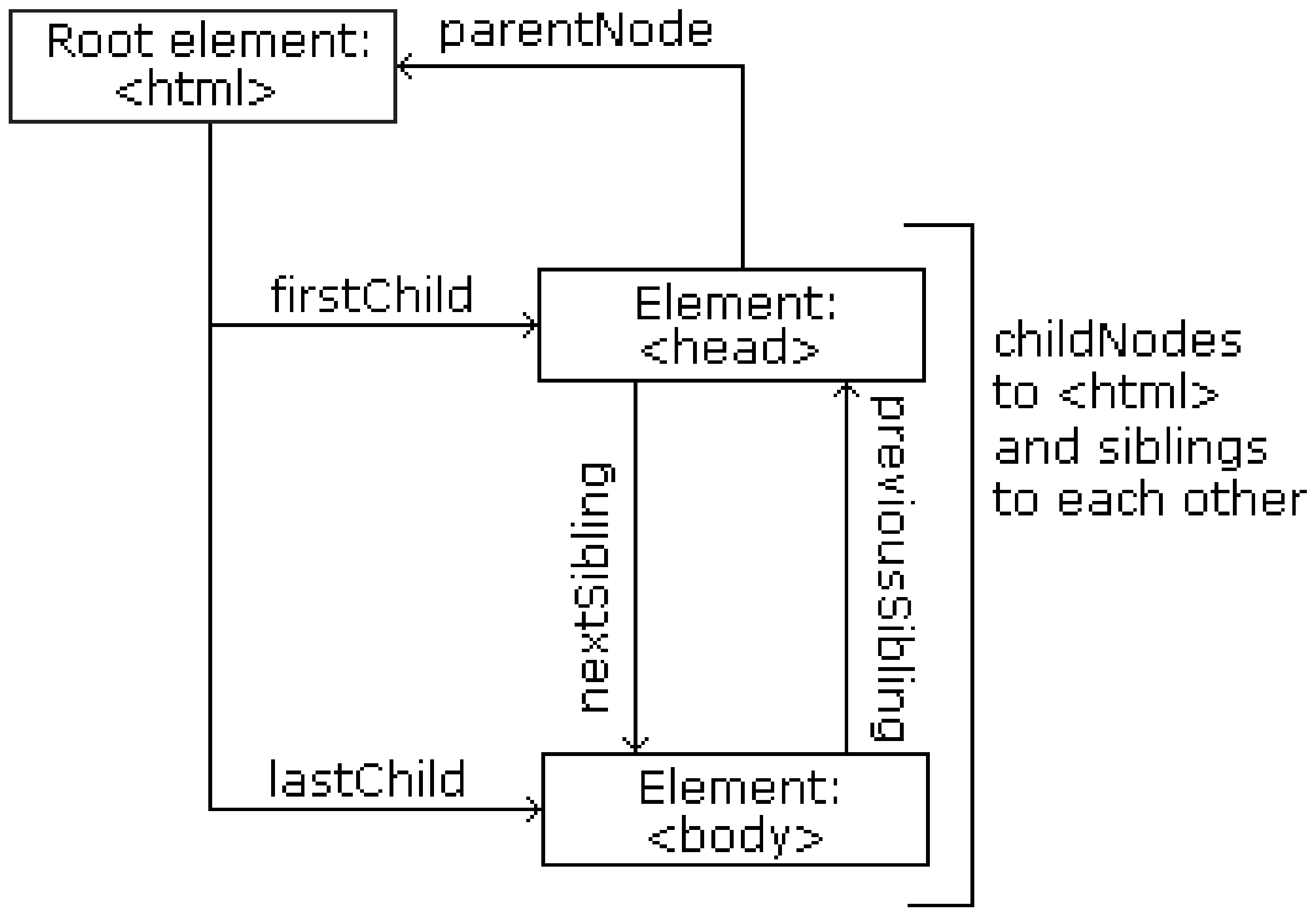
Node Relationships

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.

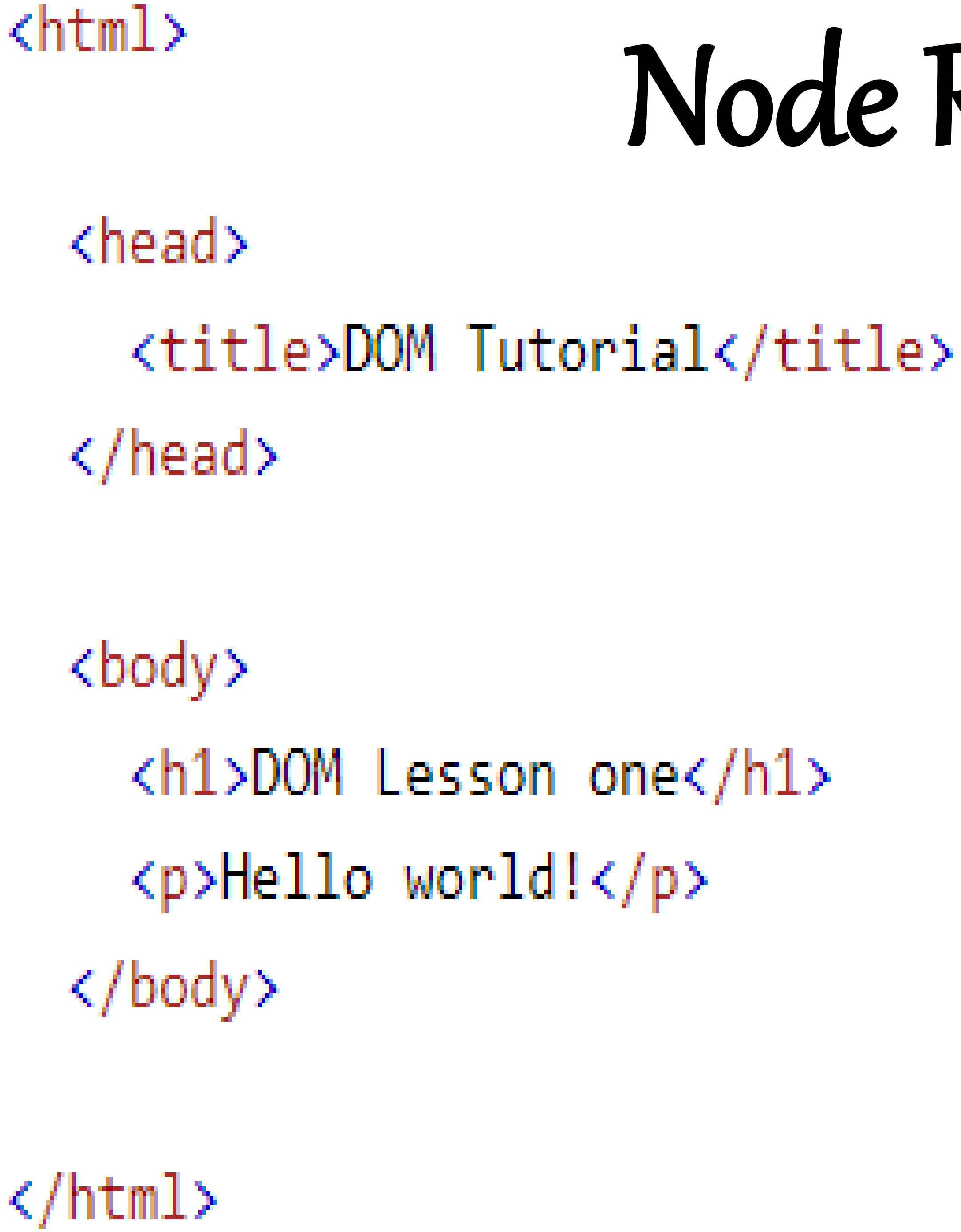
- In a node tree, the top node is called the root (or root node)
- Every node has exactly one parent, except the root (which has no parent)
- A node can have a number of children
- Siblings (brothers or sisters) are nodes with the same parent

Node Relationships

```
<html>

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

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

</html>
```



Node Relationships



```
<html>

  <head>

    <title>DOM Tutorial</title>

  </head>


  <body>

    <h1>DOM Lesson one</h1>

    <p>Hello world!</p>

  </body>


</html>
```

From the HTML you can read:

- `<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

Applications???

Any questions???

REFERENCES:

- Document Object Model (DOM). (n.d.). http://imigtds.med.uni-giessen.de/xml/DOM/standards/Activity.html
- DOM (Document Object Model). (2021). https://www.geeksforgeeks.org/dom-document-object-model/
- JavaScript Tutorial. (n.d.). https://www.javatpoint.com/javascript-tutorial
- JavaScript Tutorial. (n.d.). https://www.tutorialrepublic.com/javascript-tutorial/
- JavaScript Tutorial. (n.d.). https://www.w3schools.com/js/default.asp