The Browser Environment

Full Stack Web Development

Host Environment

DOM

DOM Nodes

BOM

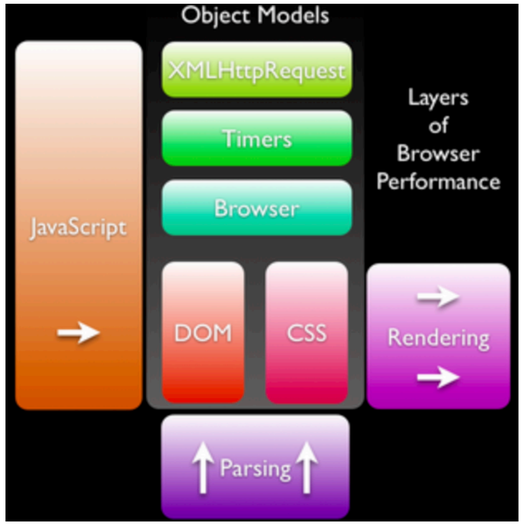
Javascript Host Environment

- The JavaScript language was initially created for web browsers. Since then it has evolved and become a language with many uses and platforms.

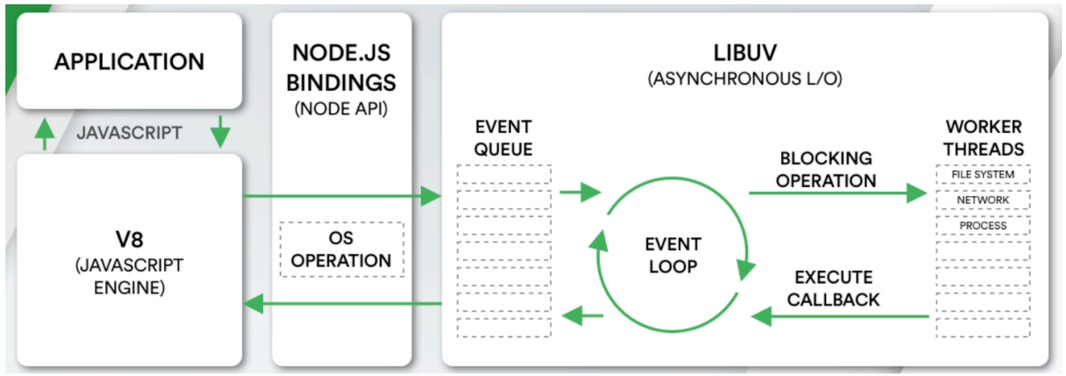
- A platform may be a browser, or a web-server or an embedded device. Each of provides platform-specific functionality. called the *host environment*.

- A host environment provides objects and functions additional to the language core. Web browsers give a means to control web pages. Node.js provides server-side features etc…

Browser

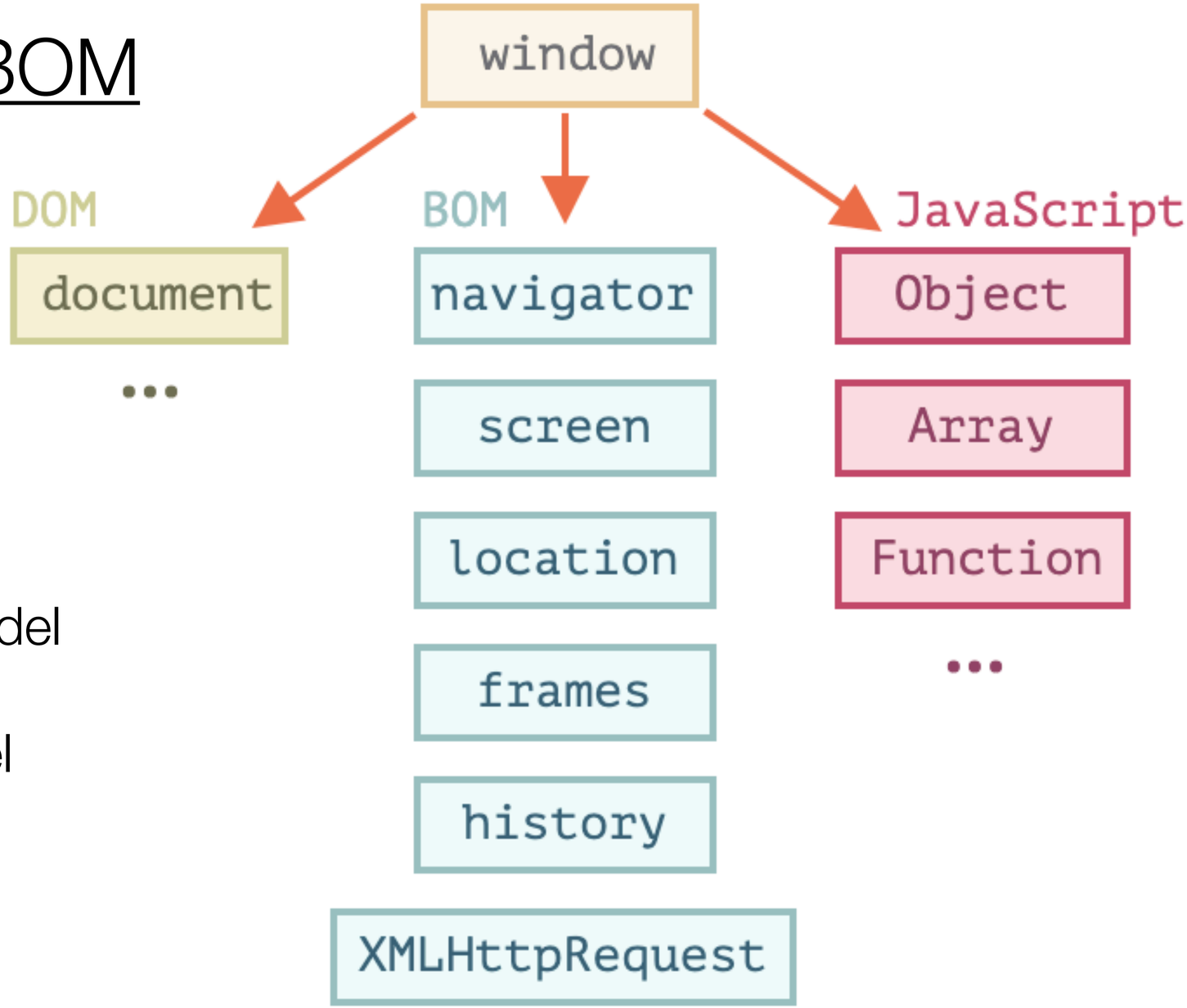


Node



Embedded

Browser Host: DOM & BOM



- DOM - Document Object Model
- BOM - Browser Object Model

DOM

- Document Object Model, or DOM for short, represents all page content as objects that can be modified.

- The document object is the main “entry point” to the page. We can change or create anything on the page using it

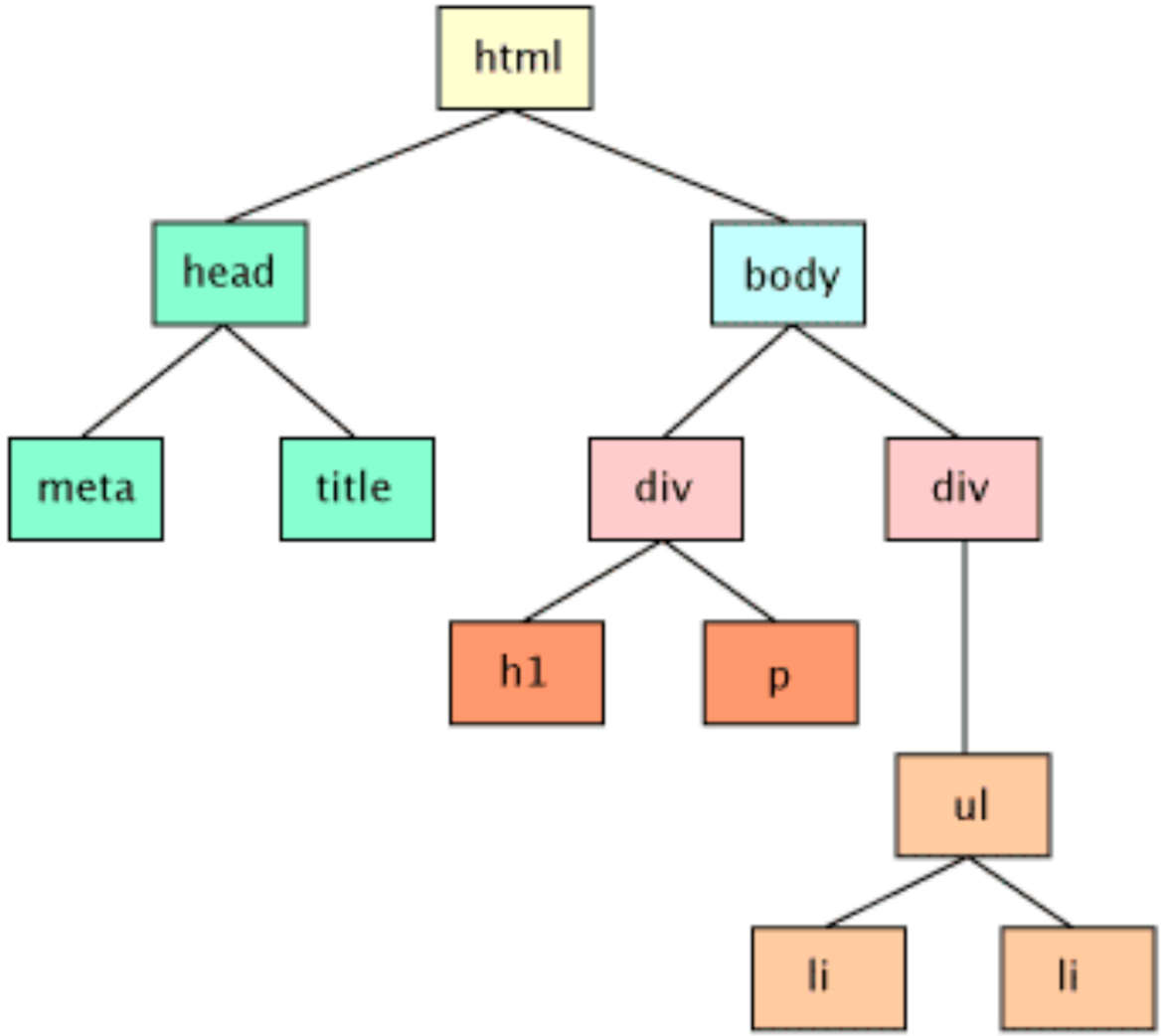
```
<html>
  <head>
    <title>JavaScript DOM</title>
  </head>
  <body>
    <p>Hello DOM!</p>
  </body>
</html>
```

DOM tree

- The backbone of an HTML document is tags.

- Every HTML tag is an object. Nested tags are “children” of the enclosing one. The text inside a tag is also an object

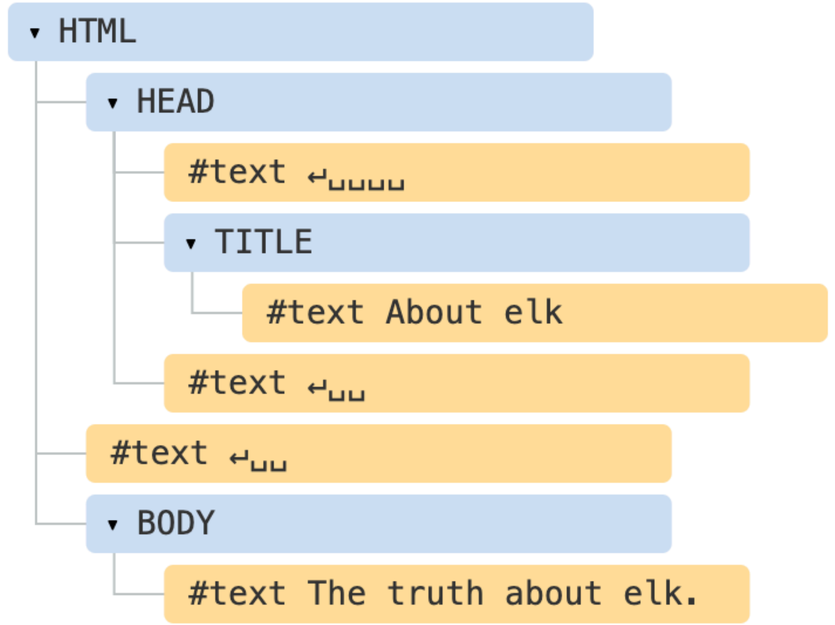
- All these objects are accessible using JavaScript.

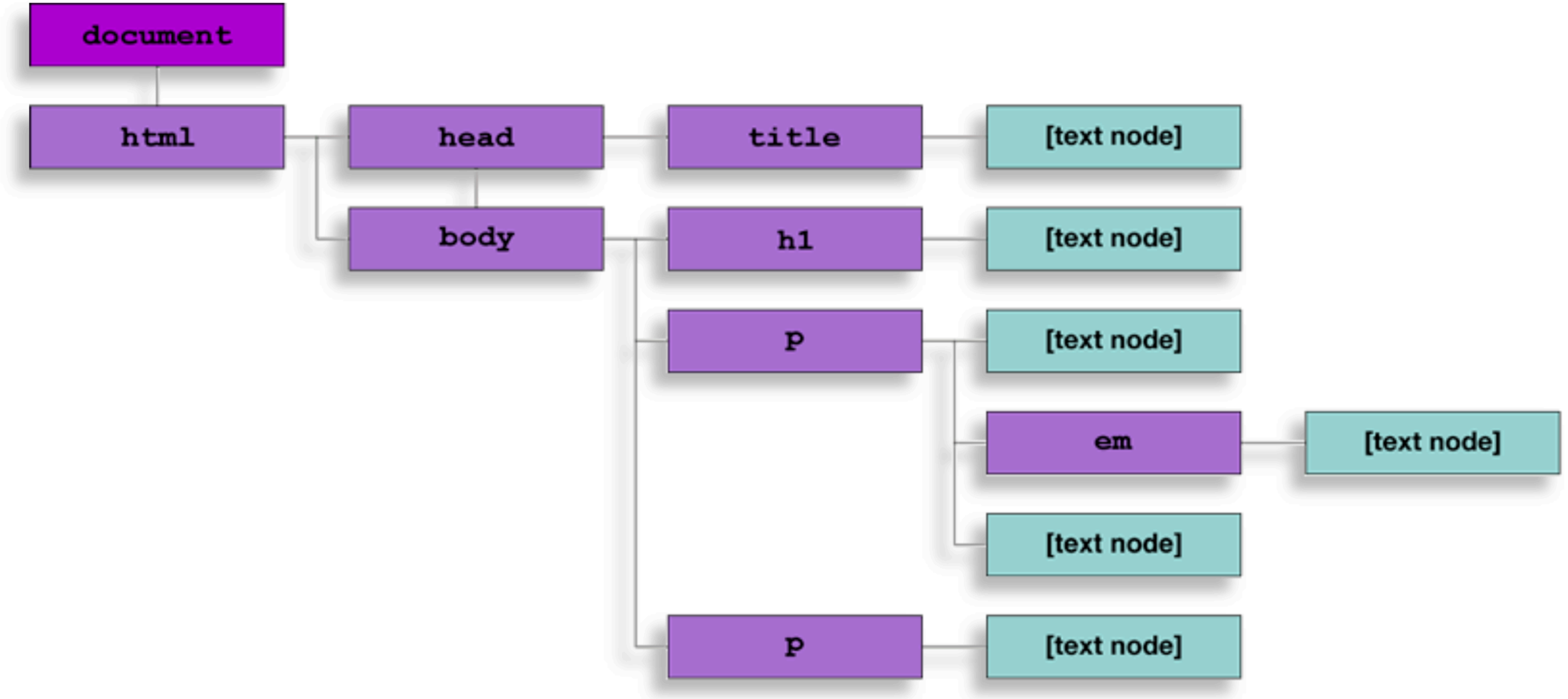


DOM tree

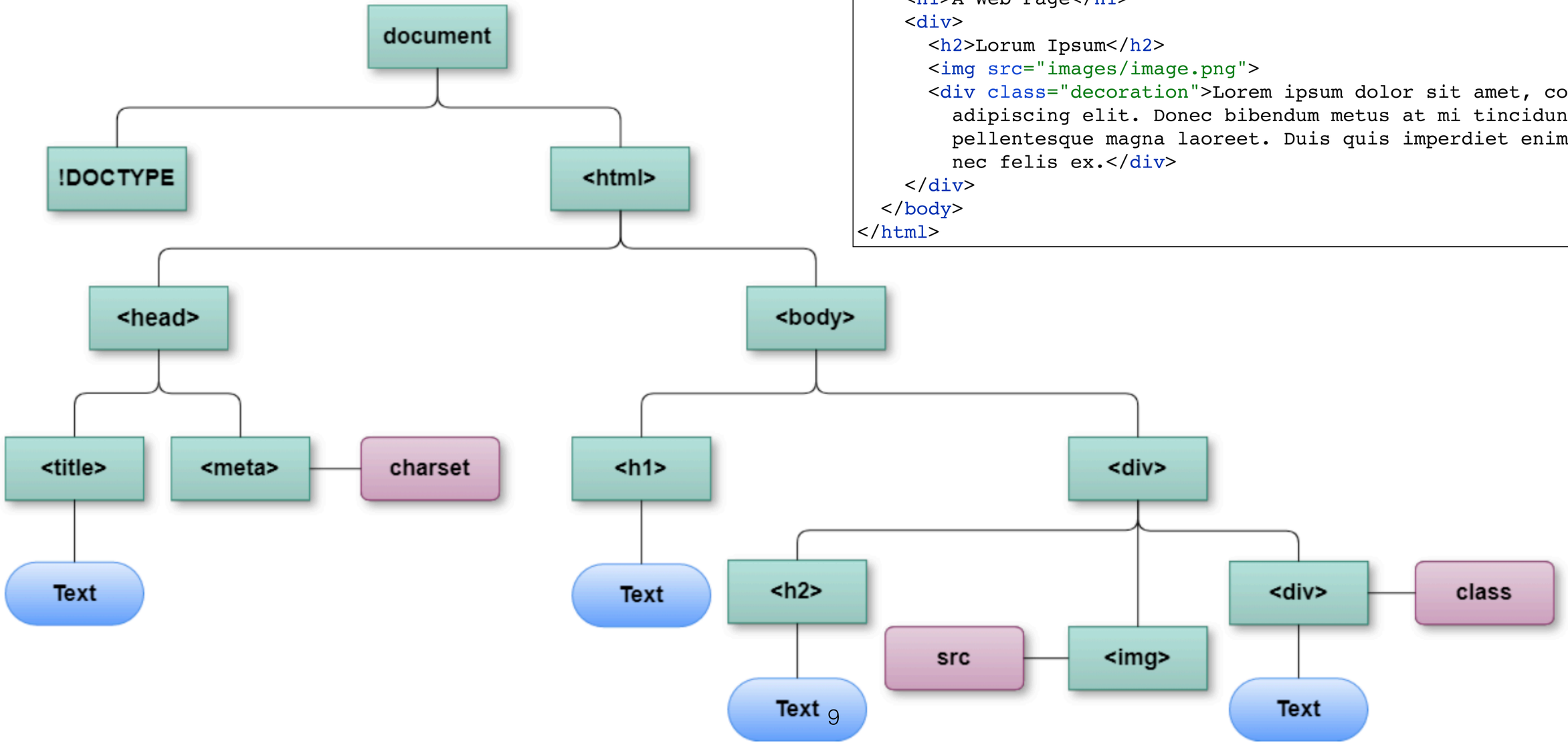
- Every tree node is an object.
- Tags are element nodes (or just elements) and form the tree structure: <html> is at the root, then <head> and <body> are its children, etc.
- The text inside elements forms text nodes. A text node contains only a string. It may not have children and is always a leaf of the tree.
- For instance, the <title> tag has the text "About elk".

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>About elk</title>
  </head>
  <body>
    The truth about elk.
  </body>
</html>
```

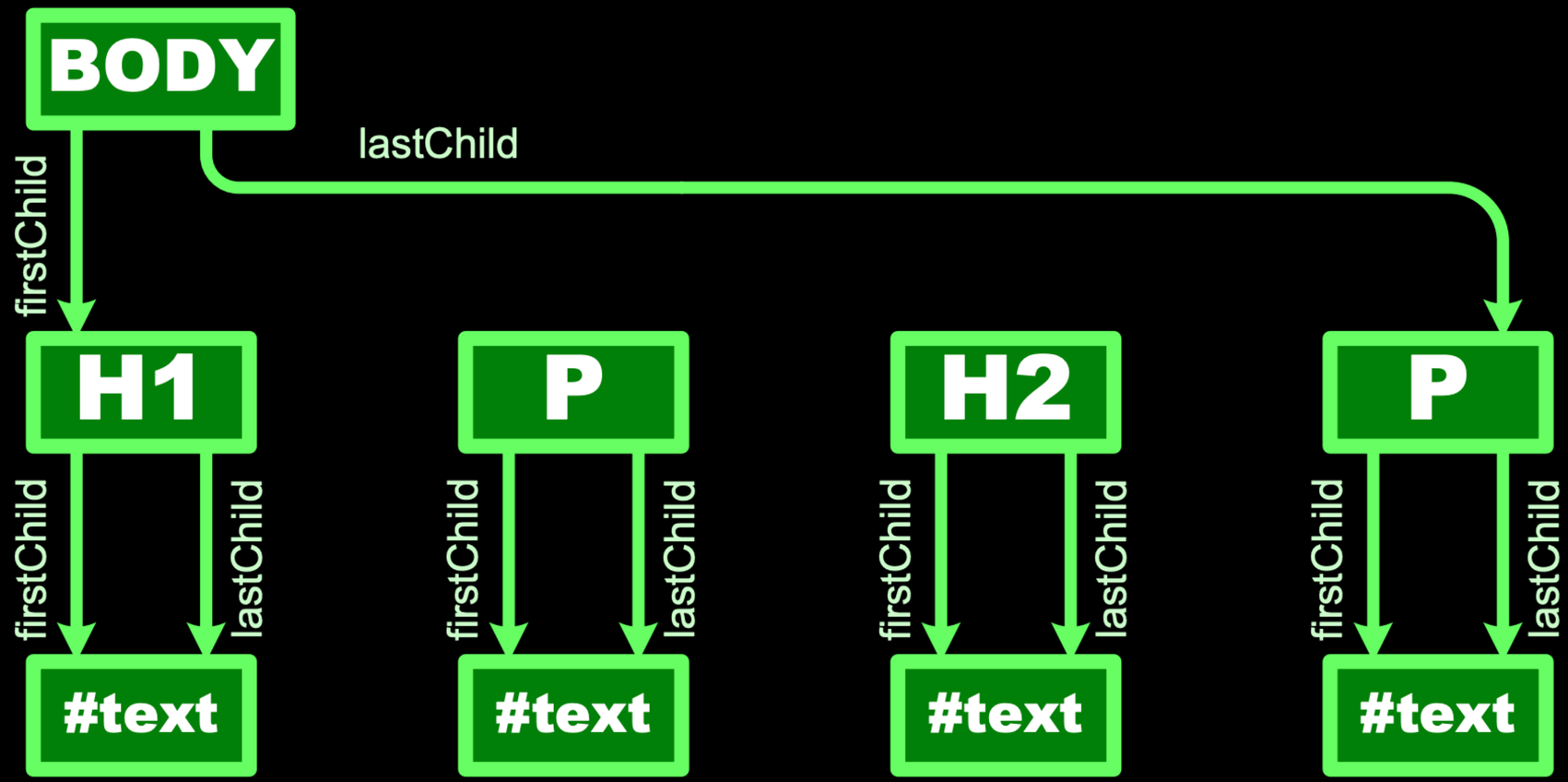



document
html
head
title
[text node]
body
h1
[text node]
p
[text node]
em
[text node]
[text node]
p
[text node]

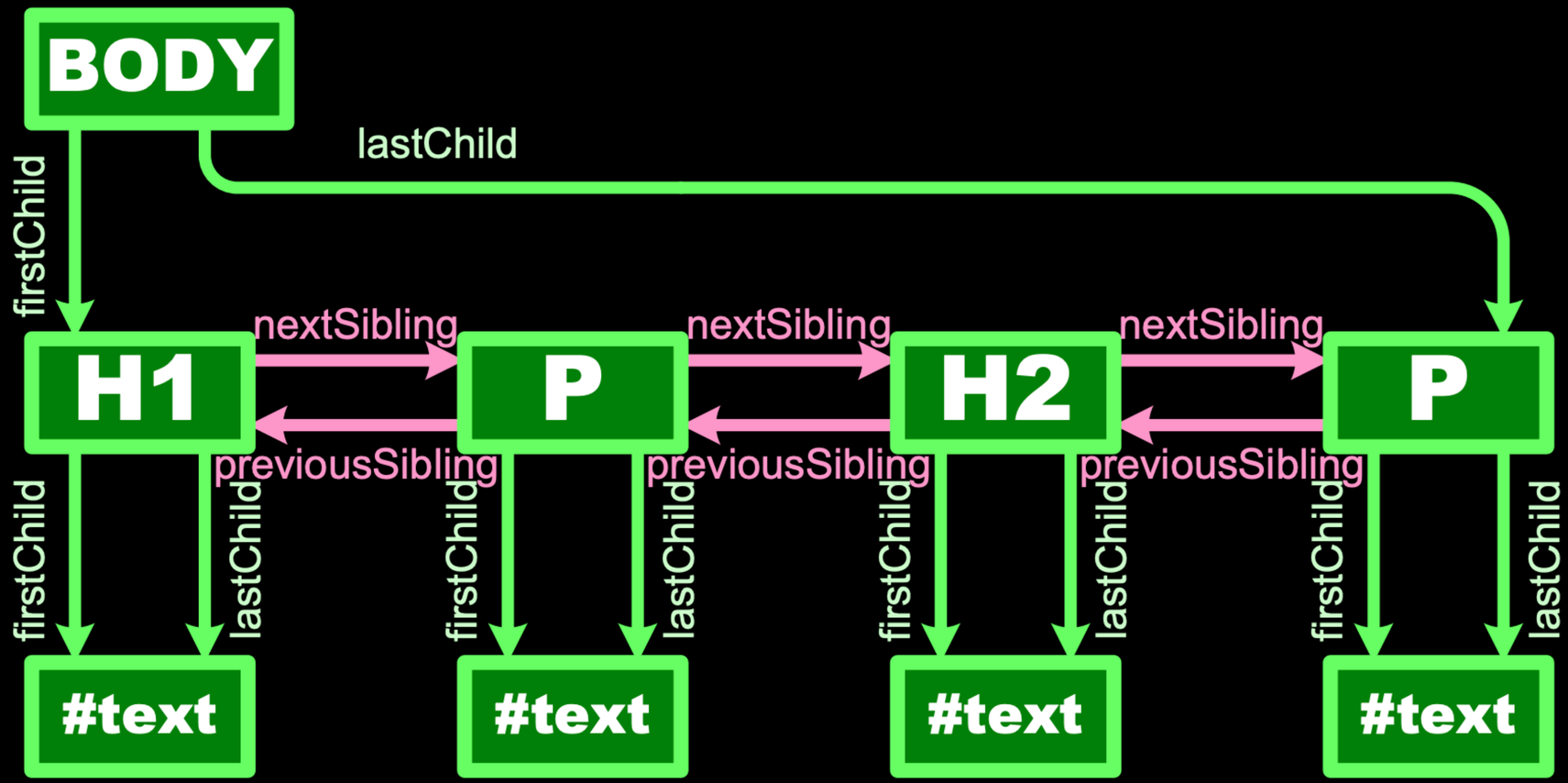
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Lorum Ipsum</title>
  </head>
  <body>
    <h1>A Web Page</h1>
    <div>
      <h2>Lorum Ipsum</h2>
      <img src="images/image.png">
      <div class="decoration">Lorem ipsum dolor sit amet, consectetur
        adipiscing elit. Donec bibendum metus at mi tincidunt, a
        pellentesque magna laoreet. Duis quis imperdiet enim. Sed
        nec felis ex.</div>
    </div>
  </body>
</html>
```



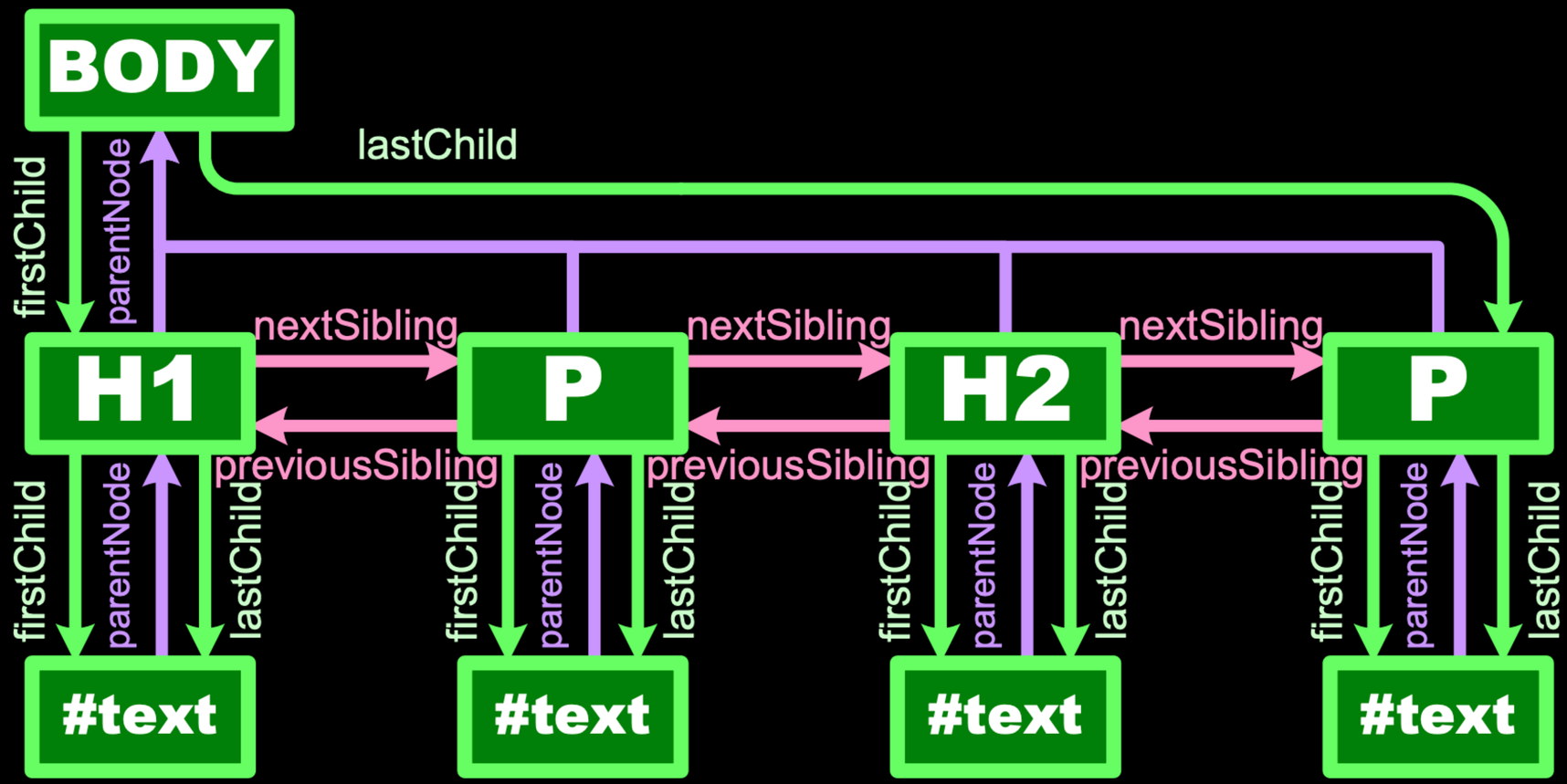
child, sibling, parent



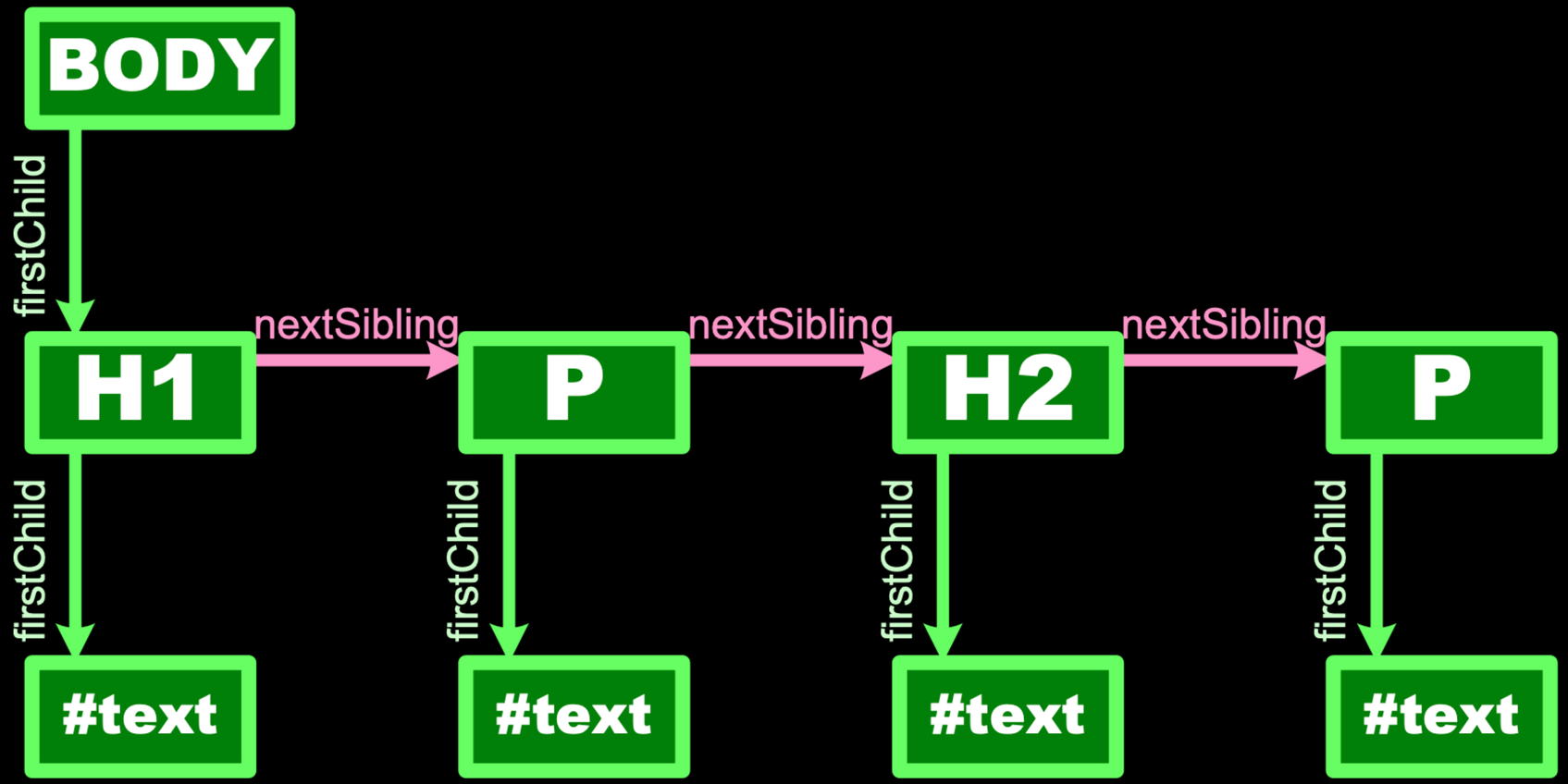
child sibling



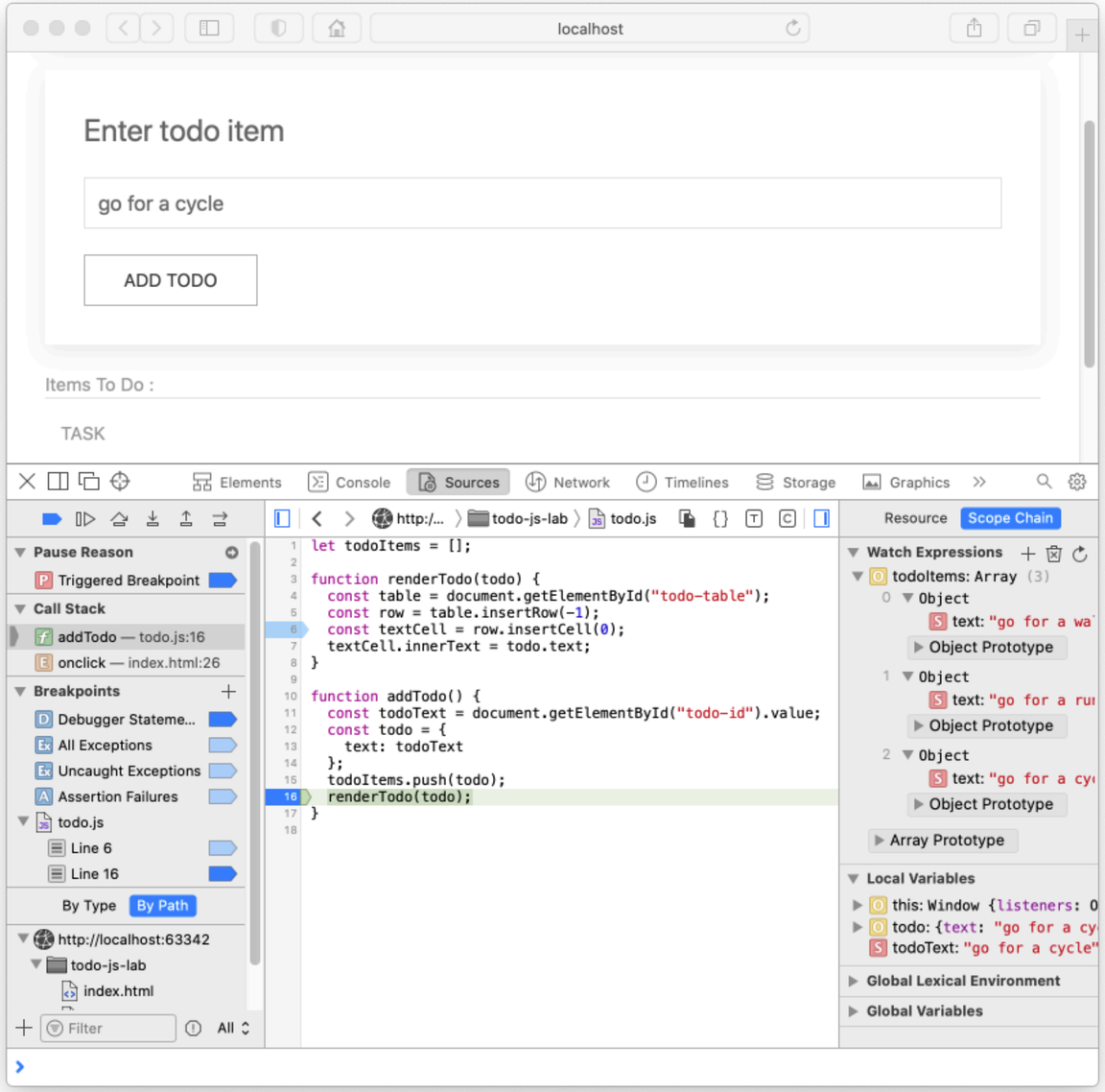
child sibling parent



child sibling parent



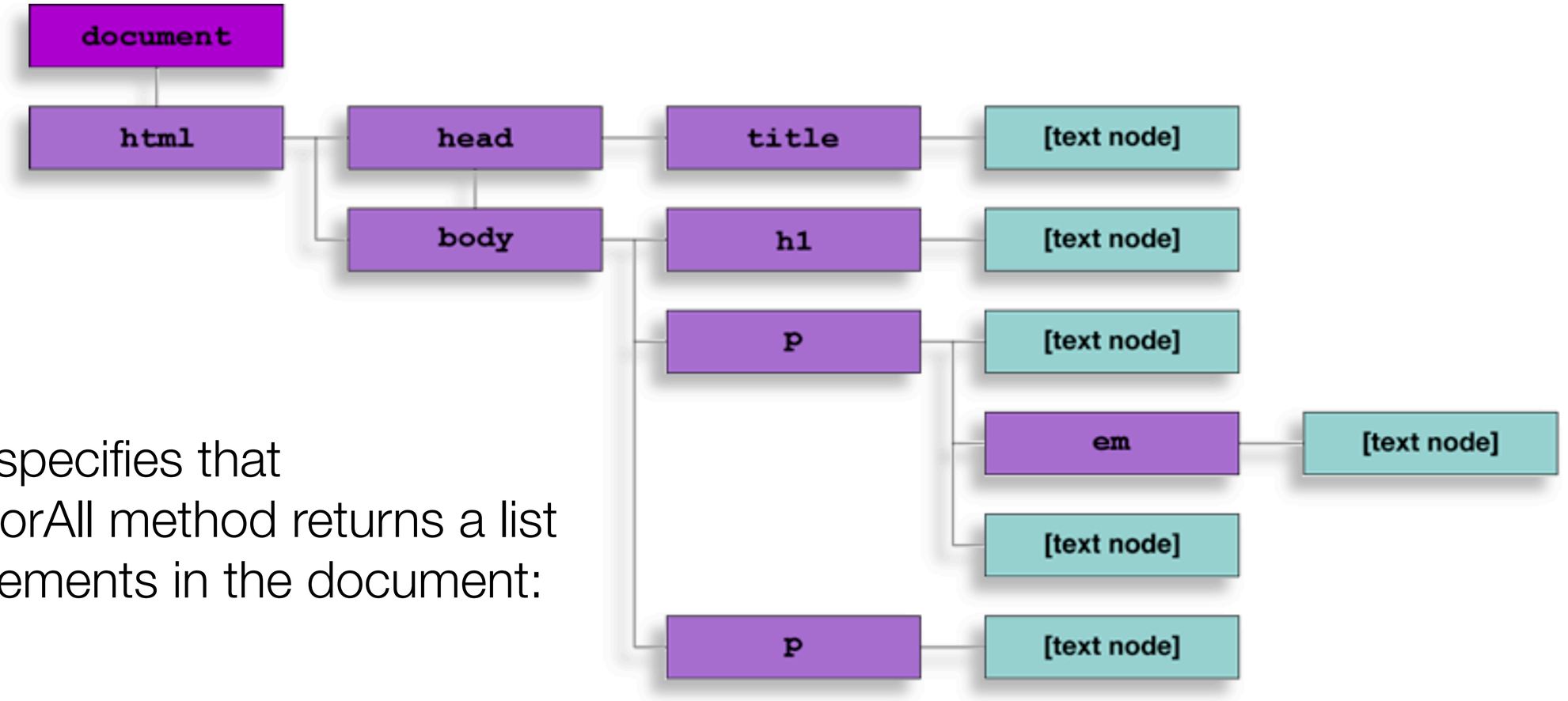
DOM API



- The Document Object Model (DOM) is an application programming interface (API) for manipulating HTML documents.

- The DOM represents a document as a tree of nodes. It provides API that allows you to add, remove, and modify parts of the document effectively.

- Note that the DOM is cross-platform and language-independent ways of manipulating HTML documents.



- standard DOM specifies that
  the querySelectorAll method returns a list
  of all the <p> elements in the document:

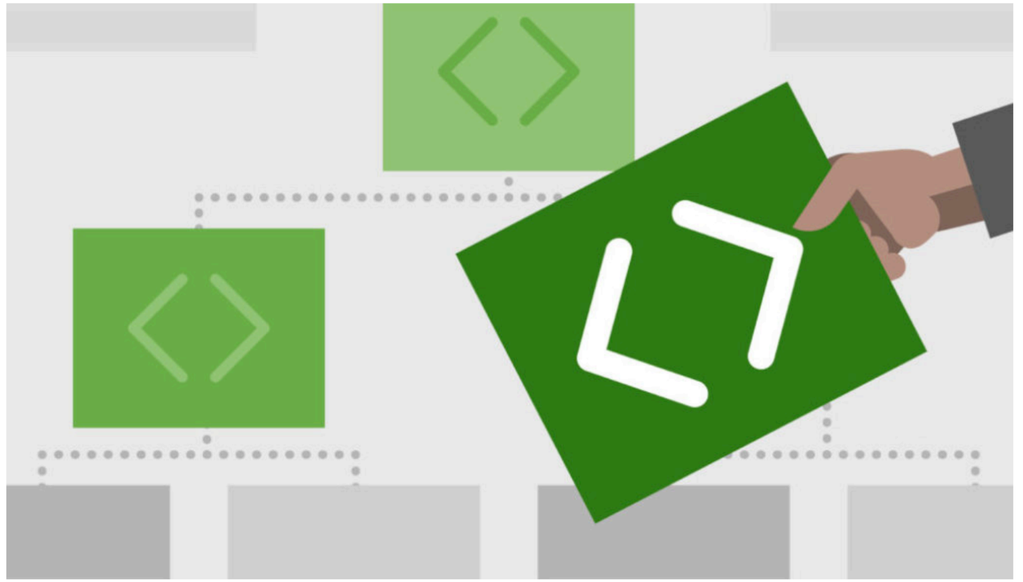
```
<script>
  const paragraphs = document.querySelectorAll("p");
  // paragraphs[0] is the first <p> element
  // paragraphs[1] is the second <p> element, etc.
  alert(paragraphs[0].nodeName);
<script>
```

- All of the properties, methods, and events available for manipulating and creating web pages are organized into objects

  - for example, the document object that represents the document itself,
    the table object that implements the special HTMLTableElement DOM interface for accessing HTML tables.

- The modern DOM is built using multiple APIs that work together.

  - The core DOM defines the objects that fundamentally describe a document and the objects within it.

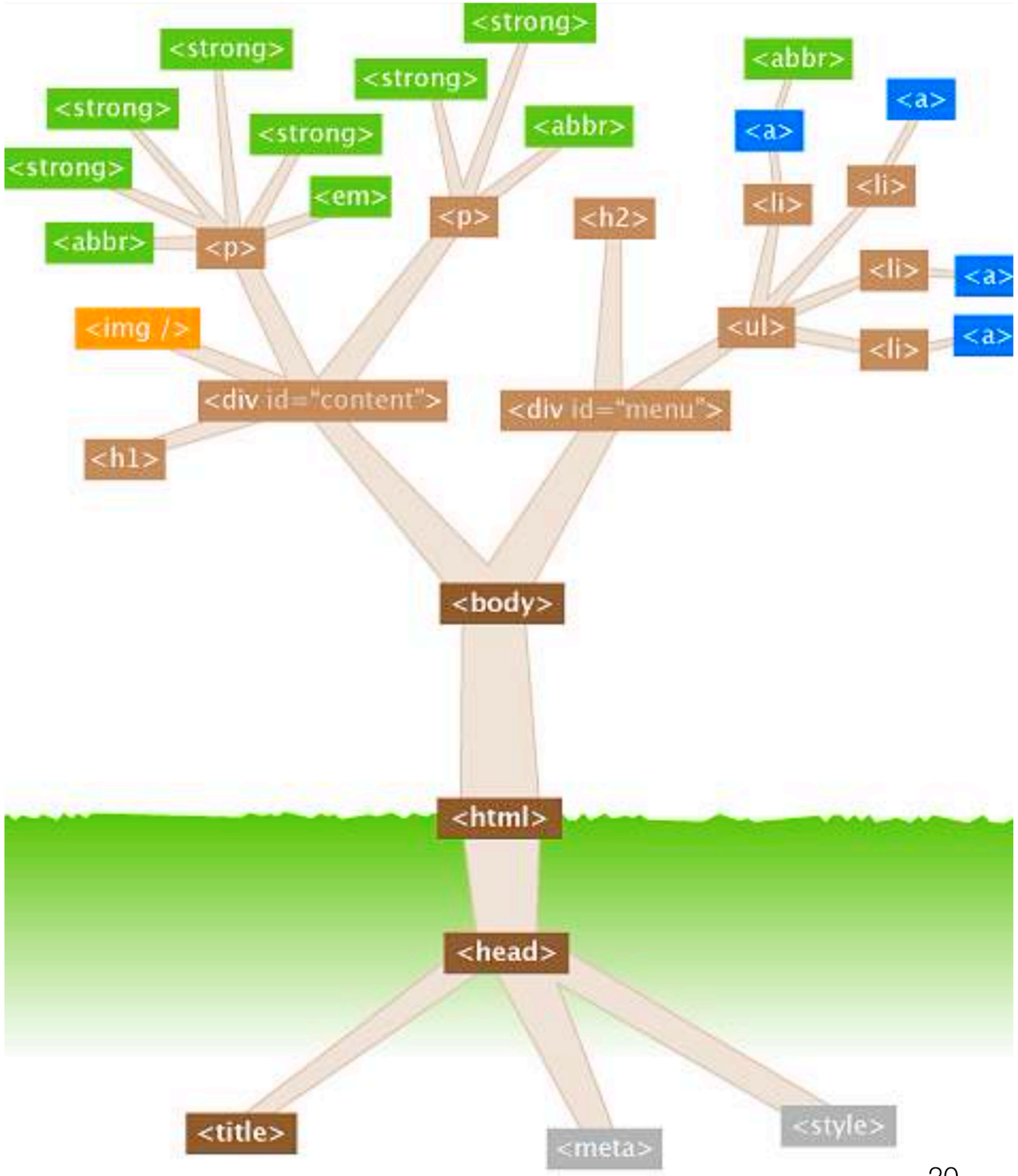


```
const paragraphs = document.querySelectorAll("p");
// paragraphs[0] is the first <p> element
// paragraphs[1] is the second <p> element, etc.
alert(paragraphs[0].nodeName);
```

- This example is written in JavaScript, but it uses the DOM to access the document and its elements.

- The DOM is not a programming language, but without it, the JavaScript language wouldn't have any model or notion of web pages, HTML documents and their component parts.

DOM Nodes



Node Types

- Each node in the DOM tree is identified by a node type. JavaScript uses integer numbers to determine the node types.

Table 1

Document	When a member returns an object of type document (e.g., the ownerDocument property of an element returns the document to which it belongs), this object is the root document object itself. The DOM document Reference chapter describes the document object.
Node	Every object located within a document is a node of some kind. In an HTML document, an object can be an element node but also a text node or attribute node.
Element	The element type is based on node. It refers to an element or a node of type element returned by a member of the DOM API. Rather than saying, for example, that the document.createElement() method returns an object reference to a node, we just say that this method returns the element that has just been created in the DOM. element objects implement the DOM Element interface and also the more basic Node interface, both of which are included together in this reference. In an HTML document, elements are further enhanced by the HTML DOM API's HTMLElement interface as well as other interfaces describing capabilities of specific kinds of elements (for instance, HTMLTableElement for <table> elements).
NodeList	A nodeList is an array of elements, like the kind that is returned by the method document.querySelectorAll(). Items in a nodeList are accessed by index in either of two ways: list.item(1) list[1] These two are equivalent. In the first, item() is the single method on the nodeList object. The latter uses the typical array syntax to fetch the second item in the list.
Attribute	When an attribute is returned by a member (e.g., by the createAttribute() method), it is an object reference that exposes a special (albeit small) interface for attributes. Attributes are nodes in the DOM just like elements are, though you may rarely use them as such.
NamedNodeMap	A namedNodeMap is like an array, but the items are accessed by name or index, though this latter case is merely a convenience for enumeration, as they are in no particular order in the list. A namedNodeMap has an item() method for this purpose, and you can also add and remove items from a namedNodeMap.

Node Types

Constant	Value	Description
`Node.ELEMENT_NODE`	1	An `Element` node like `<p>` or `<div>`.
`Node.TEXT_NODE`	3	The actual `Text` inside an `Element` or `Attr`.
`Node.PROCESSING_INSTRUCTION_NODE`	7	A `ProcessingInstruction` of an XML document, such as `<?xml-stylesheet … ?>`.
`Node.COMMENT_NODE`	8	A `Comment` node, such as `<!-- … -->`.
`Node.DOCUMENT_NODE`	9	A `Document` node.
`Node.DOCUMENT_TYPE_NODE`	10	A `DocumentType` node, such as `<!DOCTYPE html>`.
`Node.DOCUMENT_FRAGMENT_NODE`	11	A `DocumentFragment` node.

nodeType Property

- To get the type of a node, you use the nodeType property

- You can compare the nodeType property with the constants (previous slide) to determine the node type.

```
if (node.nodeType == Node.ELEMENT_NODE) {
    // node is the element node
}
```

nodeName and nodeValue properties

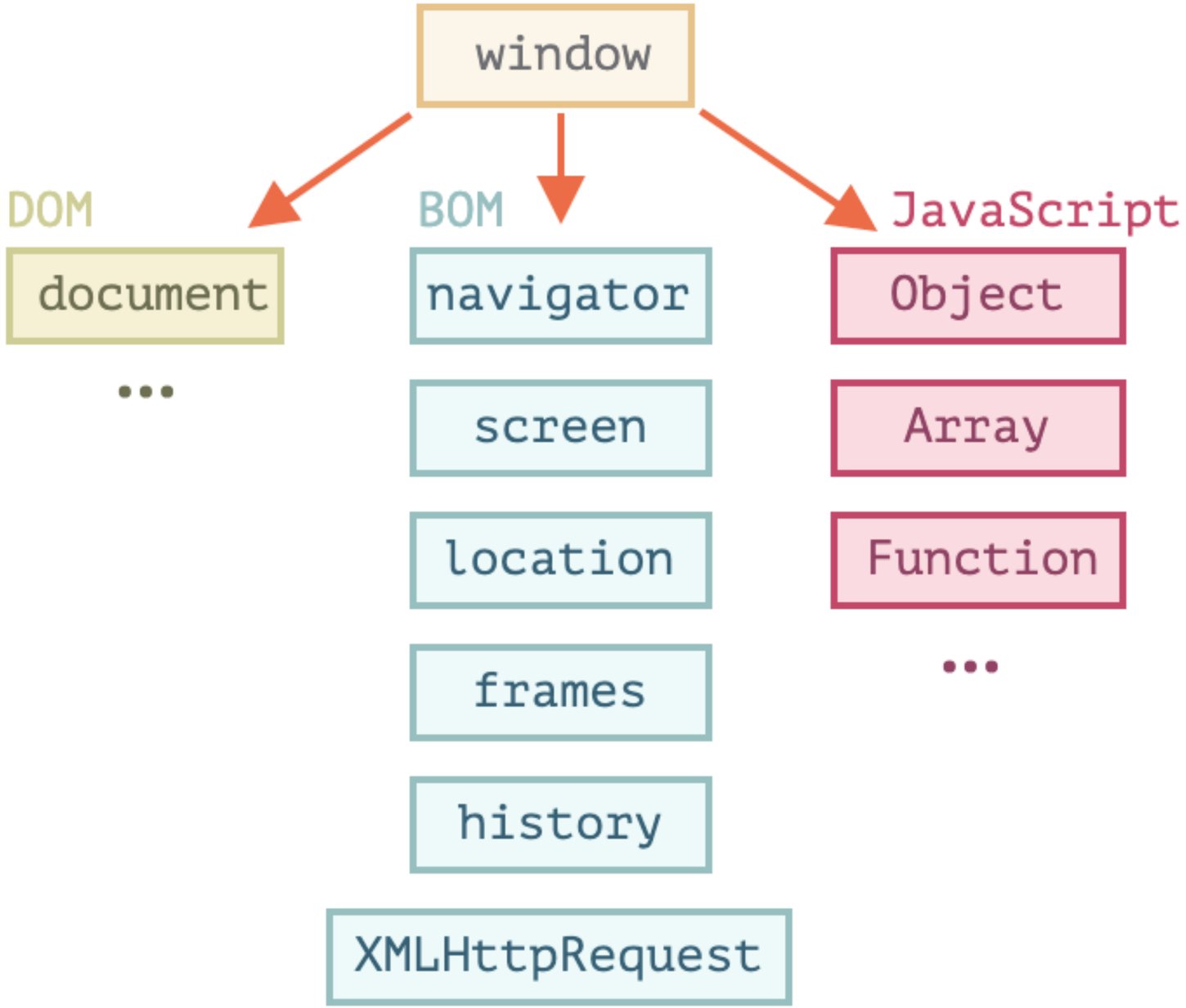
- A node has two important properties: nodeName and nodeValue.

- The values of these properties depends on the node type.

  - e.g. if the node type is the element node, the nodeName is always the same as element's tag name and nodeValue is always null.

```
if (node.nodeType == Node.ELEMENT_NODE) {
    let name = node.nodeName; // tag name like <p>
}
```

BOM


window
DOM
document
...
BOM
navigator
screen
location
frames
history
XMLHttpRequest
JavaScript
Object
Array
Function
...

Browser Object Model - BOM

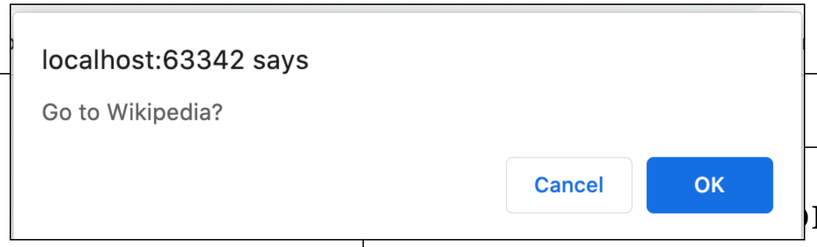
- The Browser Object Model (BOM) represents additional objects provided by the browser (host environment) for working with everything except the document.

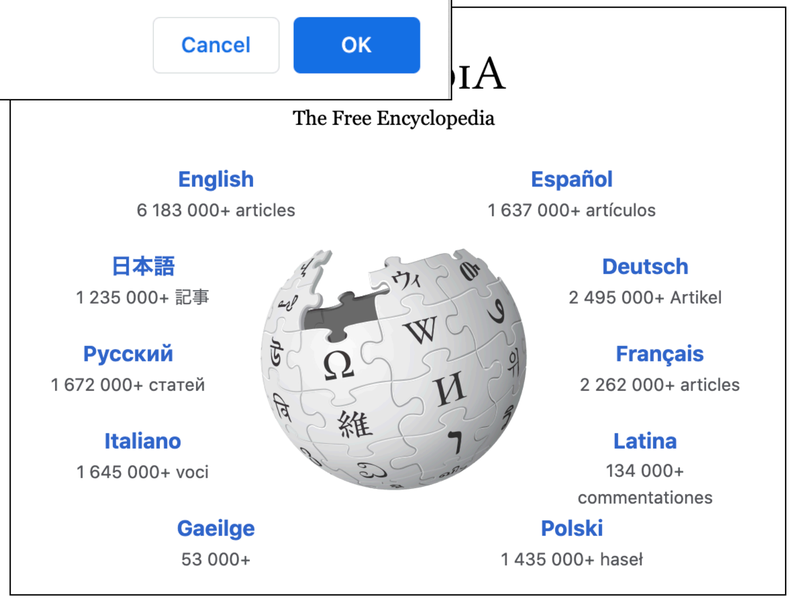
- *de facto* spec supported by browsers eg:

  - The location object retrieves the browser window current URL

  - The alert() function displays a dialog window

```
<html>
  <head>
    <title>JavaScript DOM</title>
  </head>
  <body>
    <p>Hello DOM!</p>
    <script>
      alert(location.href); // shows current URL
      if (confirm("Go to Wikipedia?")) {
        location.href = "https://wikipedia.org"; // redirect the browser to another URL
      }
    </script>
  </body>
</html>
```





Window Object

- The window object is supported by all browsers. It represents the browser's window.
- All global JavaScript objects, functions, and variables automatically become members of the window object.
- Global variables are properties of the window object.
- Global functions are methods of the window object.

```
window.document.getElementById("header");
```

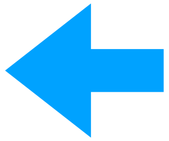
… equivalent to

```
document.getElementById("header");
```

```
var w = window.innerWidth
var h = window.innerHeight
```

BOM Objects

- Windows Object
- Navigator Object
- Location Object
- Screen Object
- Storage Object



These objects have reasonably simple API allowing Javascript access to various aspects of the browser

The Browser Environment

Full Stack Web Development