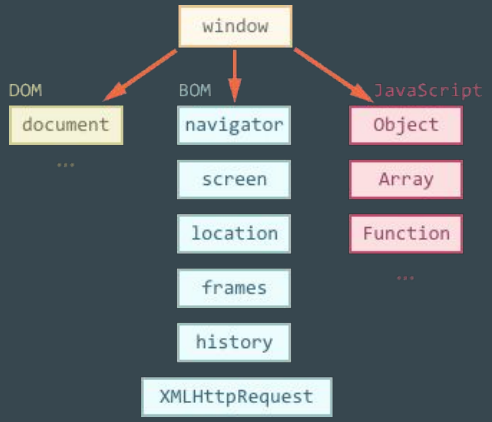
M03 - Document Object Model

•••

Introduction to DOM

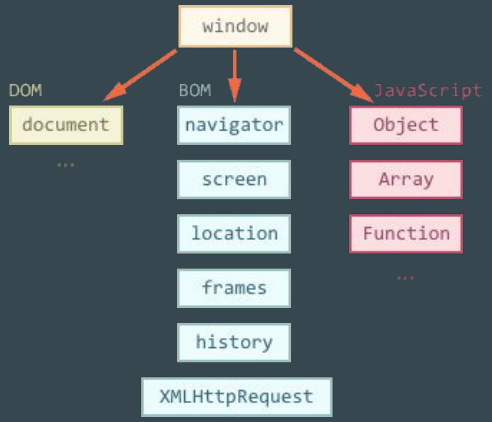
Introduction to DOM

- JavaScript was initially created for web browsers
- Since then, it has evolved into many platforms:
  - Web (web browsers and servers)
  - Desktop
  - Mobile
  - Other platforms: washing machines, ...
- Each provides platform-specific functionality
- When JS runs in a browser there is a root object called window:
  - it is a global object for JS code
  - represents the browser window and provides methods for controlling it



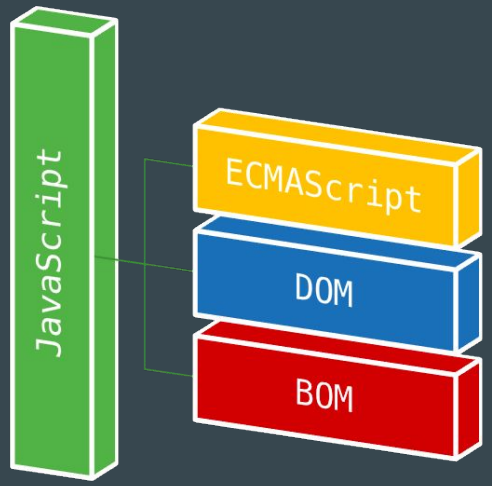
Introduction to DOM

- The global window object gives access to 3 object models:
  - Document Object Model (DOM)
    - has a document object that gives access to the content of the page
    - we can create/change anything on the HTML page
  - Browser Object Model (BOM)
    - additional objects provided by the browser to work with everything except the document
  - JavaScript
    - native JavaScript language objects



Introduction to DOM

1. Document Object Model (DOM)
2. Browser Object Model (BOM)

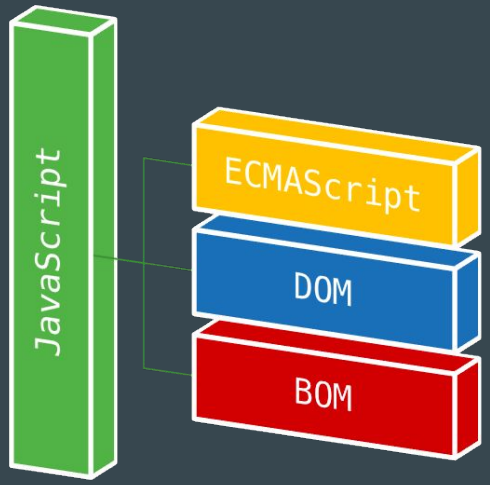


Introduction to DOM

1. Document Object Model

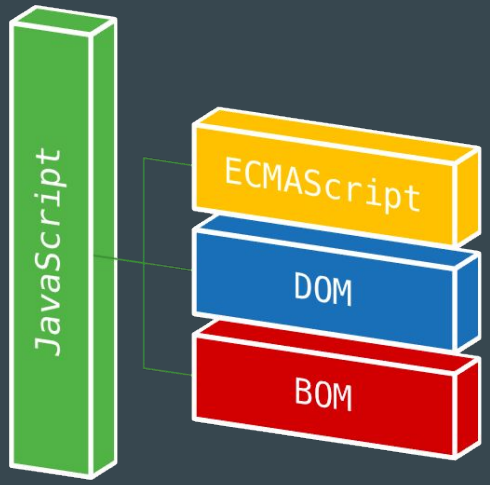
Introduction to DOM

1. Document Object Model
   - DOM Tree
   - Search
   - Edit
   - Navigation
   - Node management



Introduction to DOM

1. Document Object Model
   - DOM Tree
   - Search
   - Edit
   - Navigation
   - Node management

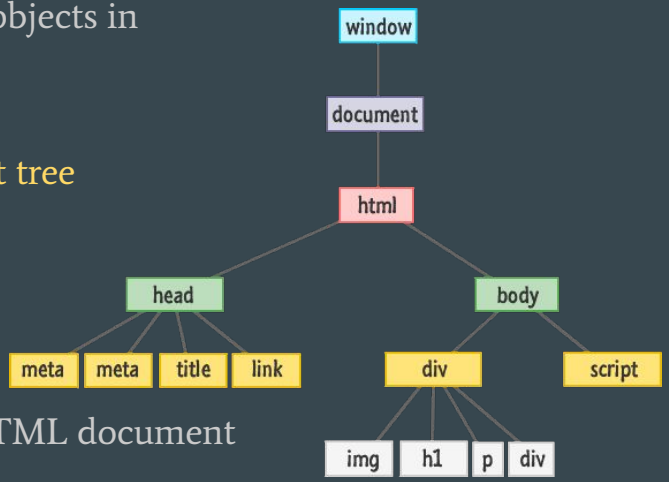


Introduction to DOM

1. Document Object Model

   DOM Tree

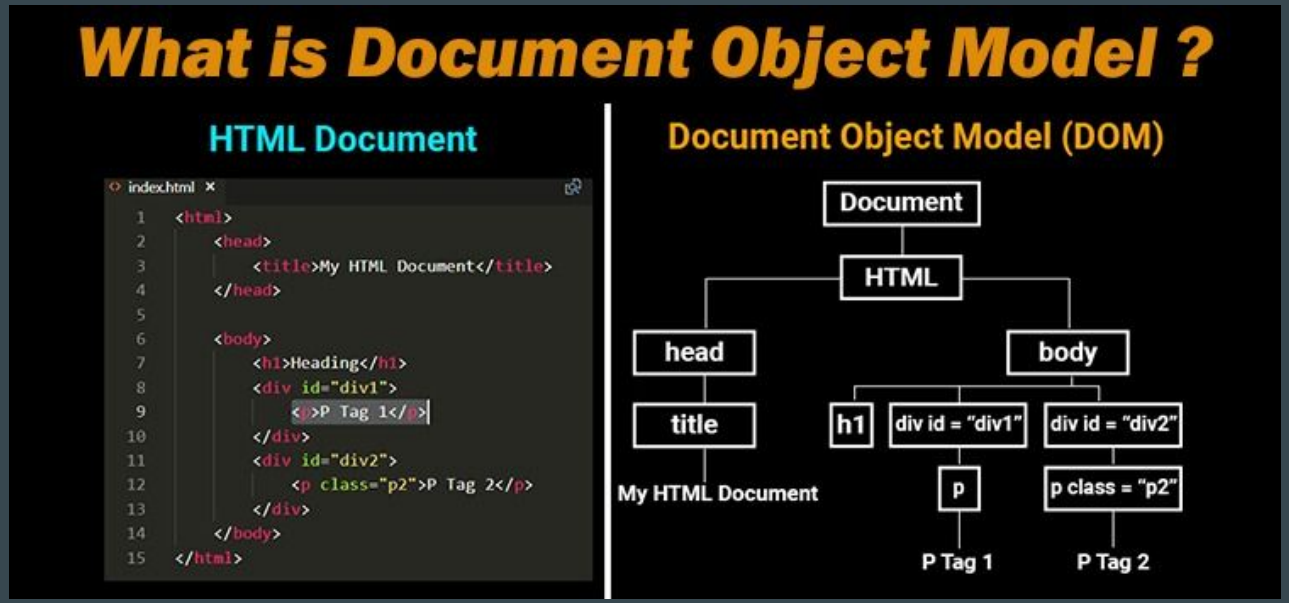
   - When an HTML page is loaded, the browser creates a tree of objects in memory representing the entire HTML document
   - HTML DOM is
     - an interface that will allow standard access to this object tree
     - it's a W3C standard: https://www.w3.org/TR/domcore/
   - With this interface, JavaScript can create dynamic HTML:
     - add/change/remove HTML elements and attributes
     - create/react to existing HTML events on the page
   - The document object is the starting point for accessing the HTML document



Introduction to DOM

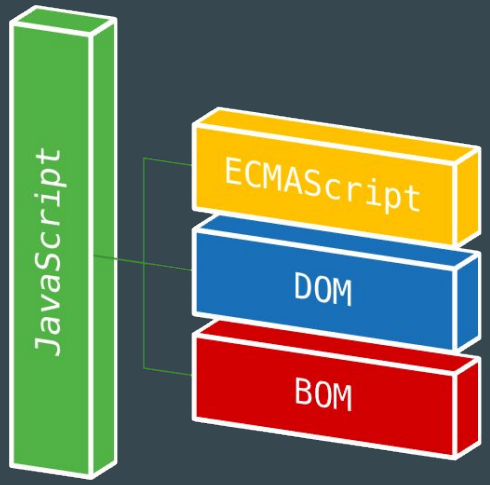
1. Document Object Model

   DOM Tree



Introduction to DOM

1. Document Object Model
   - DOM Tree
   - Search
   - Edit
   - Navigation
   - Node management



Introduction to DOM

1. Document Object Model

   Search elements

   - Often, with JavaScript, it is necessary to manipulate HTML elements
   - To do this, you need to first find the elements
   - The DOM allows you to search for elements by
     - id
     - tag name
     - class name
     - CSS selectors

Introduction to DOM

1. Document Object Model

   Search elements by id

   - Simplest way to search for an HTML element
   - Use of the method getElementById(identifier)
   - Search for an HTML element that contains an id attribute with a value equal to the identifier

```
<p id='intro'>Hi</p>
<p id='intro2'>John</p>

<script>
    const myRef = document.getElementById('intro');
    console.log(myRef) // <p id='intro'>Hi</p>
</script>
```

getElementById(id) returns an object Element

   - If the element to be searched for
     - is found, the method returns the corresponding Element object
     - is not found, returns null value

```
<p id="intro">Hi</p>
```

Introduction to DOM

1. Document Object Model

   Search elements by tag name

   - Use of the method getElementsByTagName(tag)
   - Searches for HTML elements that are defined by a tag equal to tag (method parameter)

```
<p id='intro'>Hi</p>
<p id='intro2'>John</p>

<script>
    const myRef = document.getElementsByTagName('p');
    console.log(myRef.length) // 2
</script>
```

getElementsByTagName(tag)
returns an HTMLCollection object

   - The method always returns an object HTMLCollection
   - The object HTMLCollection is an array (collection) of HTML elements

```
▼ HTMLCollection(2) i
  ▶ 0: p#intro
  ▶ 1: p#intro2
    length: 2
```

Introduction to DOM

1. Document Object Model

   Search elements by class name

   - Use of the method getElementsByClassName(class)
   - Search for HTML elements that contain a class attribute with a value equal to class

```
<p id='intro' class='red'>Hi</p>
<p id='intro2' class='blue'>John</p>

<script>
    const myRef = document.getElementsByClassName('red');
    console.log(myRef.length) // 1
</script>
```

getElementsByClassName(class) returns an um HTMLCollection object

   - The method returns an HTMLCollection object

```
▼HTMLCollection [p#intro.red, intro: p#intro.red]
  ▶0: p#intro.red
   length: 1
```

Introduction to DOM

1. Document Object Model

   Search elements by CSS selectors

   - If you want to find all the HTML elements that match a specified CSS selector (id, class name, types, attributes, attribute values, etc.)
   - Use of the method querySelectorAll(selector)

```
<p class='intro'>Hi</p>
<p class='intro'>Peter</p>
<p class='intro2'>John</p>

<script>
    const myRef = document.querySelectorAll('p.intro');
    console.log(myRef.length) // 2
</script>
```

querySelectorAll(selector) returns a NodeList object

```
▼ NodeList(2) [p.intro, p.intro]
  ▶ 0: p.intro
  ▶ 1: p.intro
    length: 2
```

   - Method returns a NodeList object - a set of nodes
   - You can use the method querySelector if you want to find just one element

Introduction to DOM

1. Document Object Model

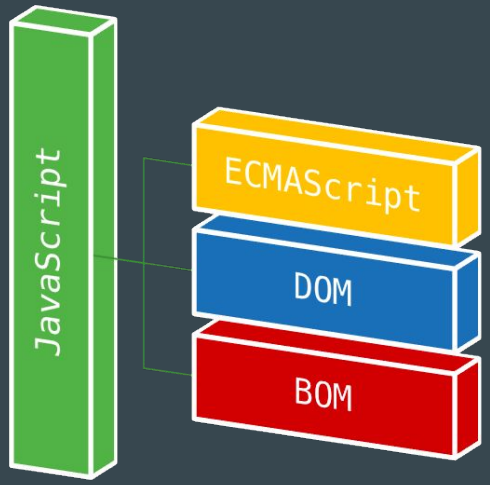
   Search elements

   - Summary table

Method	Searches by...	Can call on an element?	Live?
`querySelector`	CSS-selector	✓	-
`querySelectorAll`	CSS-selector	✓	-
`getElementById`	`id`	-	-
`getElementsByName`	`name`	-	✓
`getElementsByTagName`	tag or `'*'`	✓	✓
`getElementsByClassName`	class	✓	✓

Introduction to DOM

1. Document Object Model
   - DOM Tree
   - Search
   - Edit
   - Navigation
   - Node management



Introduction to DOM

1. Document Object Model

   Edit elements

   - The DOM allows you to change elements, more specifically:
     - edit element content
     - edit attribute values
     - edit element styles

Introduction to DOM

1. Document Object Model

   Edit element content

   - The easiest way to modify the content of an HTML element is to use the innerHTML property
   - Syntax:

     document.getElementById(id).innerHTML = new HTML

   - Example:

```
<p id='p1'>Hello World!</p>
<p id='p2'>Another Text</p>

<script>
    document.getElementById('p1').innerHTML = 'New text!';
</script>
```

New text!

Another Text

Introduction to DOM

1. Document Object Model

   Edit attribute values

   - To change the value of an HTML attribute, use this syntax:

     document.getElementById(id).attribute = new value

   - Example:

```
<img id='myImage' src='smiley.gif'>

<script>
    document.getElementById('myImage').src = 'landscape.jpg';
</script>
```

Introduction to DOM

1. Document Object Model

   Edit element styles

   - To change the style of an HTML element, use this syntax:

     document.getElementById(id).style.property = new style

   - Example:

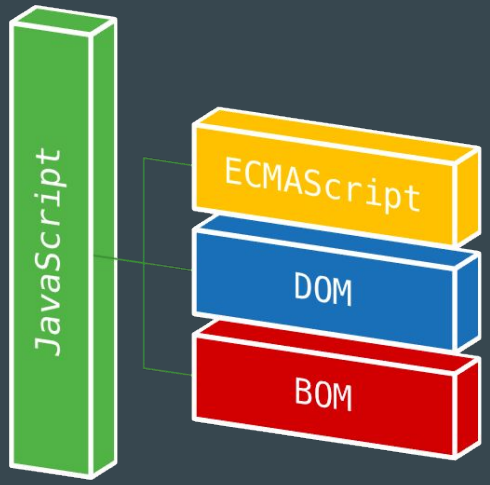
```
<p id='p1'>Hello World!</p>

<script>
    document.getElementById('p1').style.color = 'blue';
</script>
```

Hello World!

Introduction to DOM

1. Document Object Model
   - DOM Tree
   - Search
   - Edit
   - Navigation
   - Node management

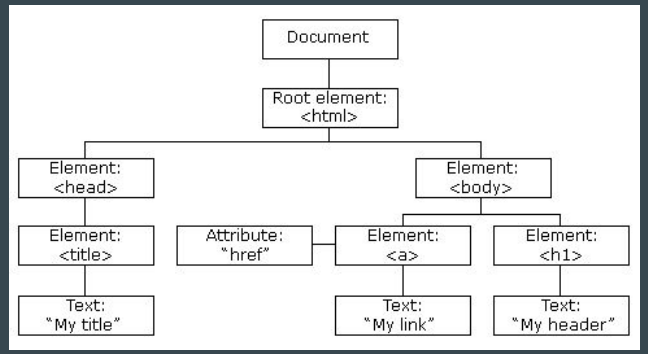


Introduction to DOM

1. Document Object Model

   Navigation

   - With the HTML DOM, you can navigate the node tree using node relationships
   - The W3C HTML DOM standard defines that everything in an HTML document is a node:
     - The entire document is a document node
     - Every HTML element is an element node
     - Text inside HTML elements are text nodes
     - HTML attribute is an attribute node (deprecated)
     - All comments are comment nodes

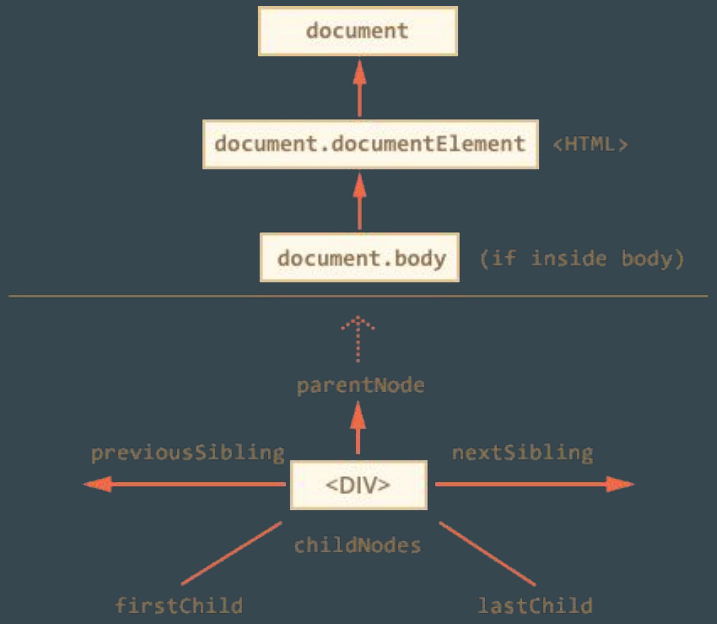


Introduction to DOM

1. Document Object Model

   Navigation

   - The nodes in the HTML tree have a hierarchical relationship with each other
   - The terms parent, child and sibling describe relationships
     - In the node tree, the top node is called the root node
     - Each node has a parent, except the root (which has no parent)
     - A node can have a number of children
     - Siblings are nodes with the same father

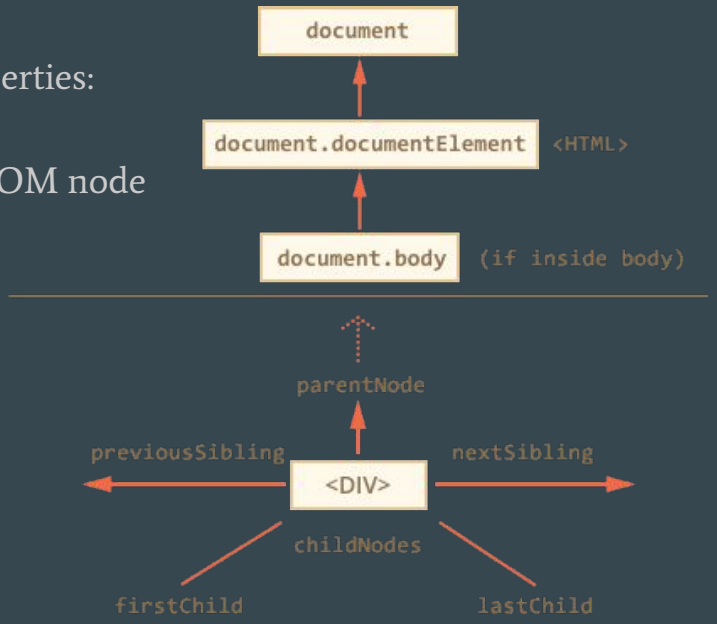


Introduction to DOM

1. Document Object Model

   Navigation

   - The top tree nodes are available directly as document properties:
     - document.documentElement
       - The 1st node of the document. It is the tag's DOM node <html>
     - document.body
       - Node that references the element <body>
     - document.head
       - Node that references the element <head>



Introduction to DOM

1. Document Object Model

   Navigation

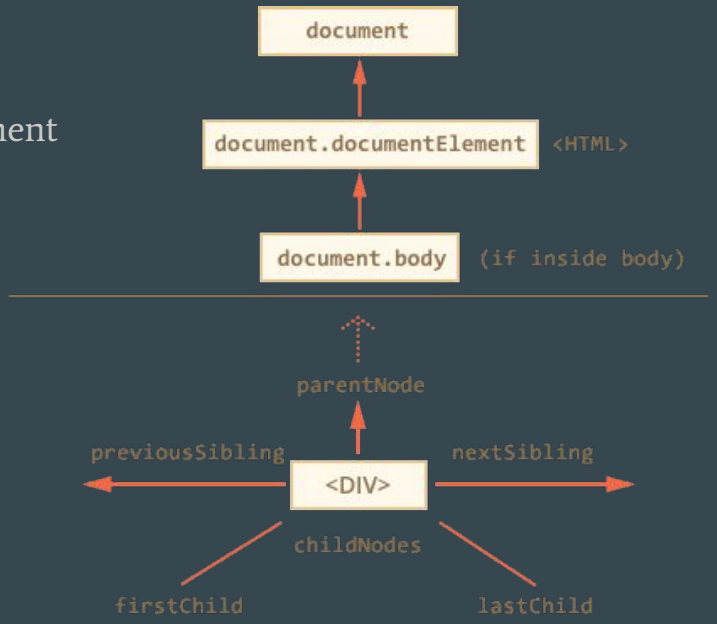
   - Collections are usually represented by objects NodeList:
     - is a list (collection) of nodes extracted from a document
     - it's almost the same as an object HTMLCollection
   - Example: paint all <p> in red

```
<h2>I'm an header!</h2>

<p>I'm a paragraph!</p>

<p>I'm a second paragraph!</p>

<script>
    const myNodelist = document.querySelectorAll('p');
    for (let node of myNodelist) {
        node.style.color = 'red';
    }
</script>
```



Introduction to DOM

1. Document Object Model

   Navigation

   - Important properties of a node
     - nodetype
     - nodename
     - nodevalue

Introduction to DOM

1. Document Object Model

   Navigation

   - Important properties of a node
     - nodetype - specifies the type of node (read only)
     - nodename
     - nodevalue

```
<title id='demo'>Introduction to DOM</title>

<script>
    console.log(document.getElementById('demo').nodeType) // 1
</script>
```

Node	Type	Example
ELEMENT_NODE	1	<h1 class="heading">W3Schools</h1>
ATTRIBUTE_NODE	2	class = "heading" (deprecated)
TEXT_NODE	3	W3Schools
COMMENT_NODE	8	<!-- This is a comment -->
DOCUMENT_NODE	9	The HTML document itself (the parent of <html>)
DOCUMENT_TYPE_NODE	10	<!Doctype html>

Introduction to DOM

1. Document Object Model

   Navigation

   - Important properties of a node
     - nodetype
     - nodename - specifies the name of a node (read only)
       - Obtaining different values in a
         - element node is the same as the tag name
         - attribute node is the name of the attribute
         - text node is always #text
         - document node is always #document
     - nodevalue

```
<title id='demo'>Introduction to DOM</title>

<script>
    console.log(document.getElementById('demo').nodeName) // TITLE
</script>
```

Introduction to DOM

1. Document Object Model

   Navigation

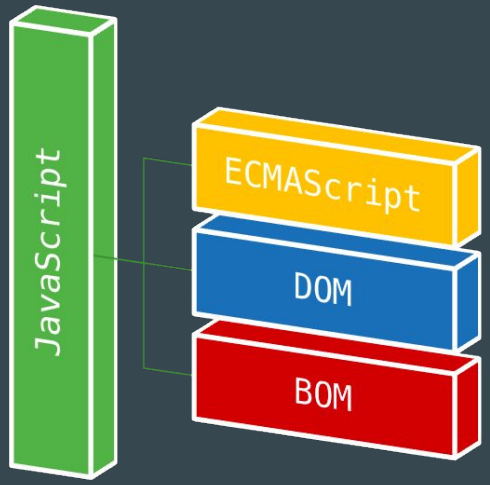
   - Important properties of a node
     - nodetype
     - nodename
     - nodevalue - specifies the value of a node
       - Obtaining different values in a
         - element node is null
         - text node is the text itself
         - attribute node is the attribute value

```
<title id='demo'>Introduction to DOM</title>

<script>
    console.log(document.getElementById('demo').nodeValue) // null
</script>
```

Introduction to DOM

1. Document Object Model
   - DOM Tree
   - Search
   - Edit
   - Navigation
   - Node management



Introduction to DOM

1. Document Object Model

   Node management

   - Involves
     - Creation of new nodes
     - Replacement of existing nodes
     - Removal of existing nodes

Introduction to DOM

1. Document Object Model

   Node management

   - Creation of new nodes
     - To add a new element to the HTML DOM:
       - create the element (element node) with the methods createElement/CreateTextNode
       - attach it to an existing element with the method appendChild

```
<div id='div1'>
    <p id='p1'>A paragraph</p>
    <p id='p2'>Another paragraph</p>
</div>

<script>
    const para = document.createElement('p');
    const node = document.createTextNode('A new paragraph');
    para.appendChild(node);

    const element = document.getElementById('div1');
    element.appendChild(para);
</script>
```

A paragraph

Another paragraph

A new paragraph

Introduction to DOM

1. Document Object Model

   Node management

   - Replacement of existing nodes
     - To replace an element in the HTML DOM, use the method replaceChild

```
<div id='div1'>
    <p id='p1'>A paragraph</p>
    <p id='p2'>Another paragraph</p>
</div>

<script>
    const para = document.createElement('p');
    const node = document.createTextNode('A new paragraph');
    para.appendChild(node);

    const parent = document.getElementById('div1');
    const child = document.getElementById('p1');
    parent.replaceChild(para, child);
</script>
```

A new paragraph

Another paragraph

Introduction to DOM

1. Document Object Model

   Node management

   - Removal of existing nodes
     - To remove an HTML element, you must
       - know the element's parent
       - use the method removeChild

```
<div id='div1'>
    <p id='p1'>A paragraph</p>
    <p id='p2'>Another paragraph</p>
</div>

<script>
    const parent = document.getElementById('div1');
    const child = document.getElementById('p1');
    parent.removeChild(child);
</script>
```

Another paragraph

Introduction to DOM

1. Document Object Model

   Node management

   - Node management should be used sparingly given verbosity
   - For example, when adding elements
   - It is preferable to use template strings in combination with the innerHTML property

```
<table id='table1'>
    <tr>
        <th>Name</th>
        <th>Age</th>
    </tr>
    <tr>
        <td>John</td>
        <td>43</td>
    </tr>
</table>

<script>
    const myTable = document.getElementById('table1');
    const otherName = 'Peter';
    const otherAge = 8;

    myTable.innerHTML += `<tr><td>${otherName}</td><td>${otherAge}</td></tr>`;
</script>
```

Name	Age
John	43
Peter	8

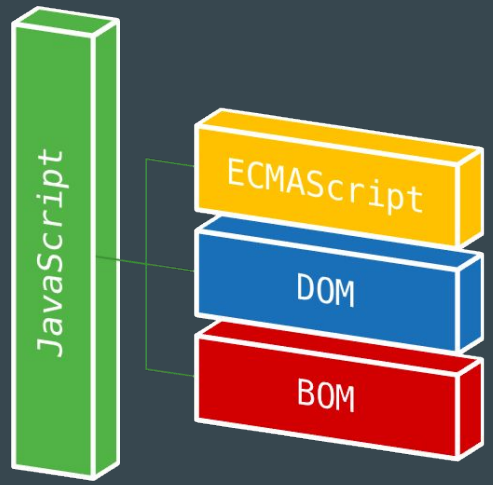
Introduction to DOM

2. Browser Object Model

Introduction to DOM

2. Browser Object Model

- The Browser Object Model (BOM) allows JavaScript to "talk to" the browser
- Main objects:
  - Window
  - Screen
  - Location
  - History
  - Navigator
  - Modals
  - Timing



Introduction to DOM

1. Browser Object Model

- Object window
  - Represents the browser window
  - Supported by all browsers
  - All JS objects, global functions and variables automatically become members
    - Global variables are properties of the window object.
    - Global functions are methods of the window object.
    - Even the document object (from the HTML DOM) is a property of the window object.
  - Main properties and methods:
    - window.innerHeight - the internal height of the browser window (in pixels)
    - window.innerWidth - the internal width of the browser window (in pixels)
    - window.open() - opens a new window
    - window.close() - closes the current window
    - window.moveTo() - move the current window
    - window.resizeTo() - resizes the current window

Introduction to DOM

1. Browser Object Model

- Object screen
  - Contains information about the user's screen
  - Can be written without the prefix window
  - Main properties and methods:
    - screen.width/screen.height - returns the width/height of the user's screen in pixels
    - screen.availWidth/screen.availWidth - returns the width/height of the user's screen, in pixels, minus the interface features, such as the Windows Taskbar.
    - screen.colorDepth - returns the number of bits used to display a color.
      - All modern computers use 24- or 32-bit hardware for color resolution:
      - 24 bits = 16.777.216 different "true colors"
      - 32 bits = 4.294.967.296 different "deep colors"

Introduction to DOM

1. Browser Object Model

- Object location
  - Can be used to obtain the current page's address (URL) and redirect the browser to a new page
  - Main properties:
    - window.location.href - returns the href (URL) of the current page
    - window.location.hostname - returns the domain name of the host
    - window.location.pathname - returns the path and file name of the current page
    - window.location.protocol - returns the web protocol used (http or https)
    - window.location.assign - uploads a new document

```
alert(`The URL of this page is ${window.location.href}`);
```

Introduction to DOM

1. Browser Object Model

- Object history
  - Contains the browser history
  - Main methods:
    - history.back() - the same as clicking back on the browser
    - history.forward() - the same as clicking forward in the browser

```
<input type='button' value='Back'>

<script>
    const myButton = document.querySelector('input');
    myButton.addEventListener('click', function () {
        window.history.back();
    })
</script>
```

Introduction to DOM

1. Browser Object Model

- Object navigator
  - Contains information about the visitor's browser
  - Some examples:
    - navigator.appName
    - navigator.appCodeName
    - navigator.platform

  - Information from the navigator object can often be misleading and should not be used to detect browser versions

Introduction to DOM

1. Browser Object Model
   - Sync events
     - JavaScript events can be executed at time intervals
     - This is called event synchronization
     - The two main methods for using with JavaScript are:
       - setTimeout(function, milliseconds)
         - Executes a function, after waiting a specified number of milliseconds
       - setInterval(function, milliseconds)
         - Same as setTimeout(), but repeatedly executes the function

```
<button onclick='setTimeout(myFunction, 3000)'>Greeting after 3 seconds</button>
<button onclick='setInterval(myFunction, 3000)'>Greeting every 3 seconds</button>

<script>
    function myFunction() {
        console.log('Hello World!');
    }
</script>
```