Lesson 10
Document Object Model (DOM)

Maharishi University of Management -Fairfield, Iowa © 2024

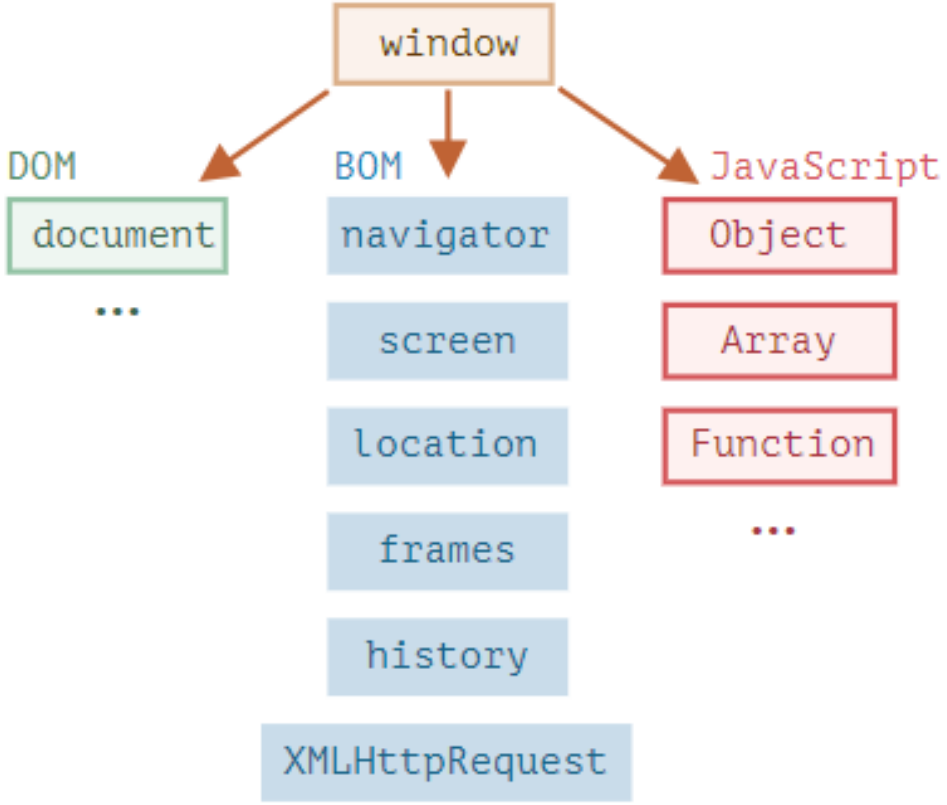
Browser environment

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

when JavaScript runs in a web browser:

The **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.



What is the DOM?

The DOM is a W3C (World Wide Web Consortium) standard.

The DOM defines a standard for accessing documents.

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

What is the HTML DOM?

The HTML DOM is a standard object model and programming interface for HTML.

It defines:

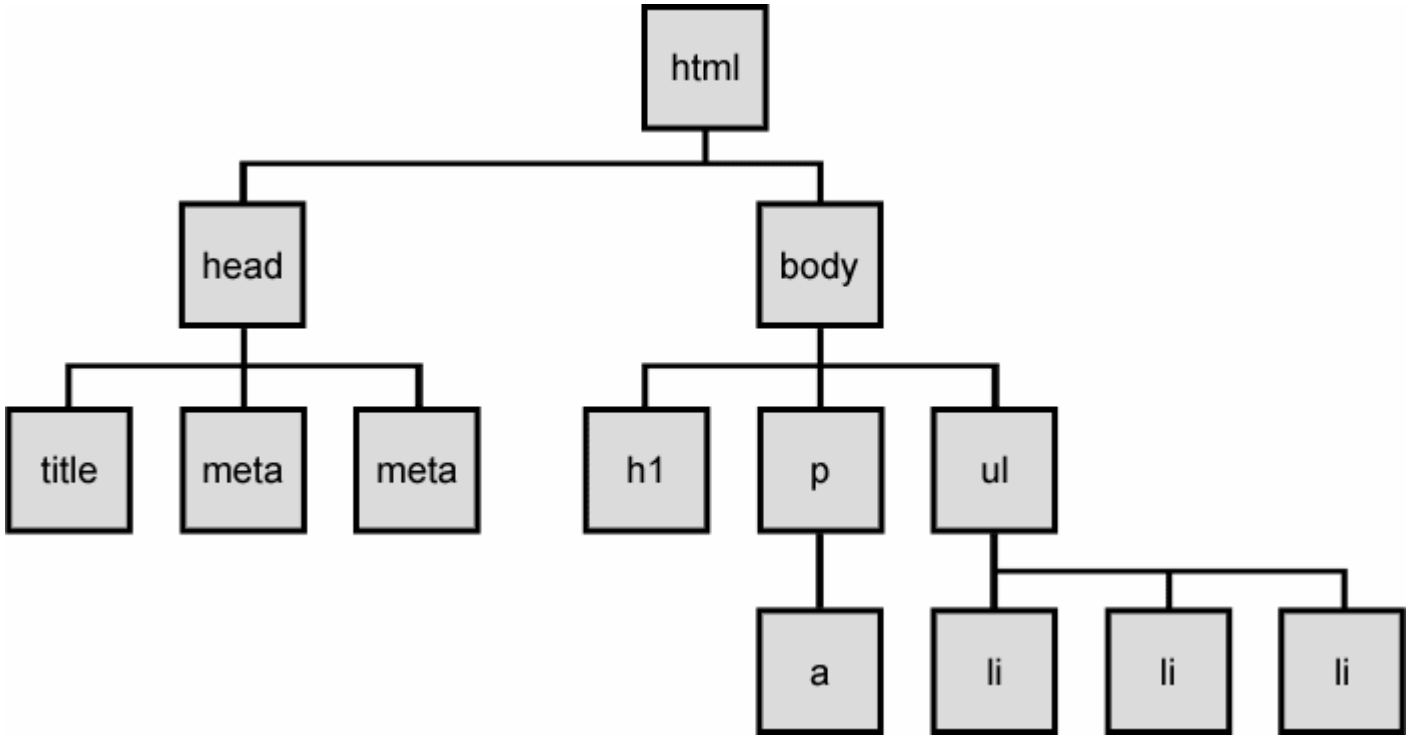
- The HTML elements as objects
- The properties of all HTML elements
- The methods to access all HTML elements
- The events for all HTML elements

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

With the HTML DOM, JavaScript can access and change all the elements of an HTML document.

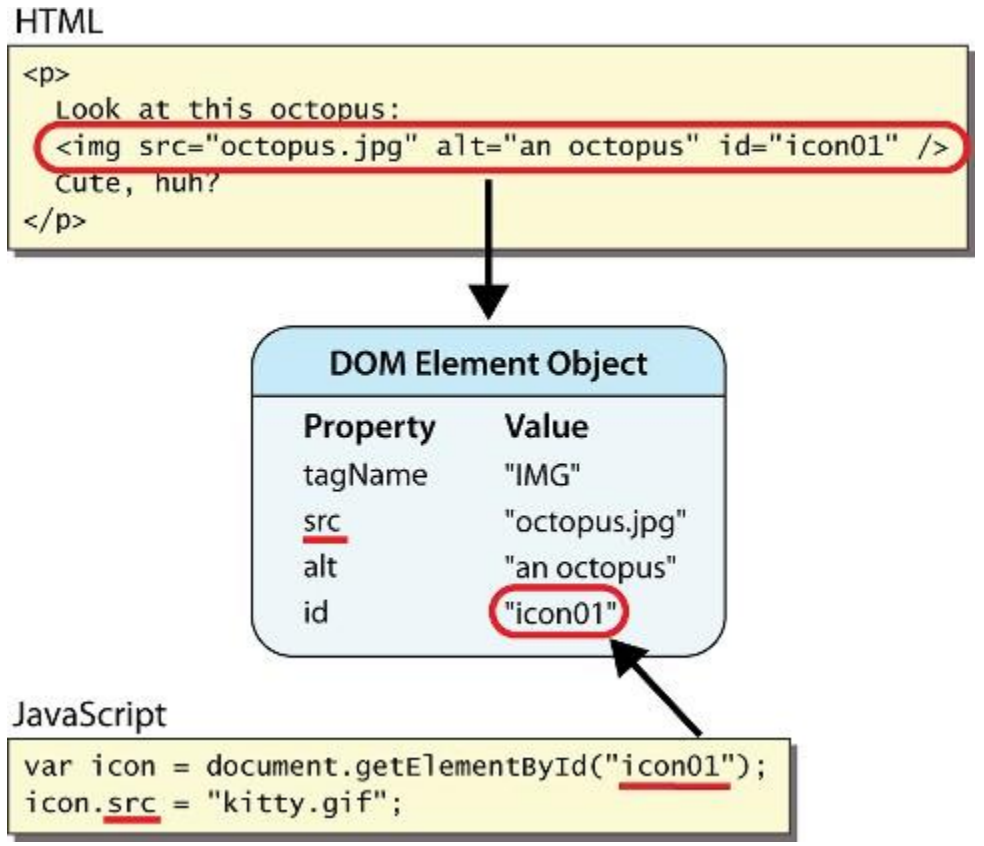
The DOM tree

The elements of a page are nested into a tree-like structure of objects



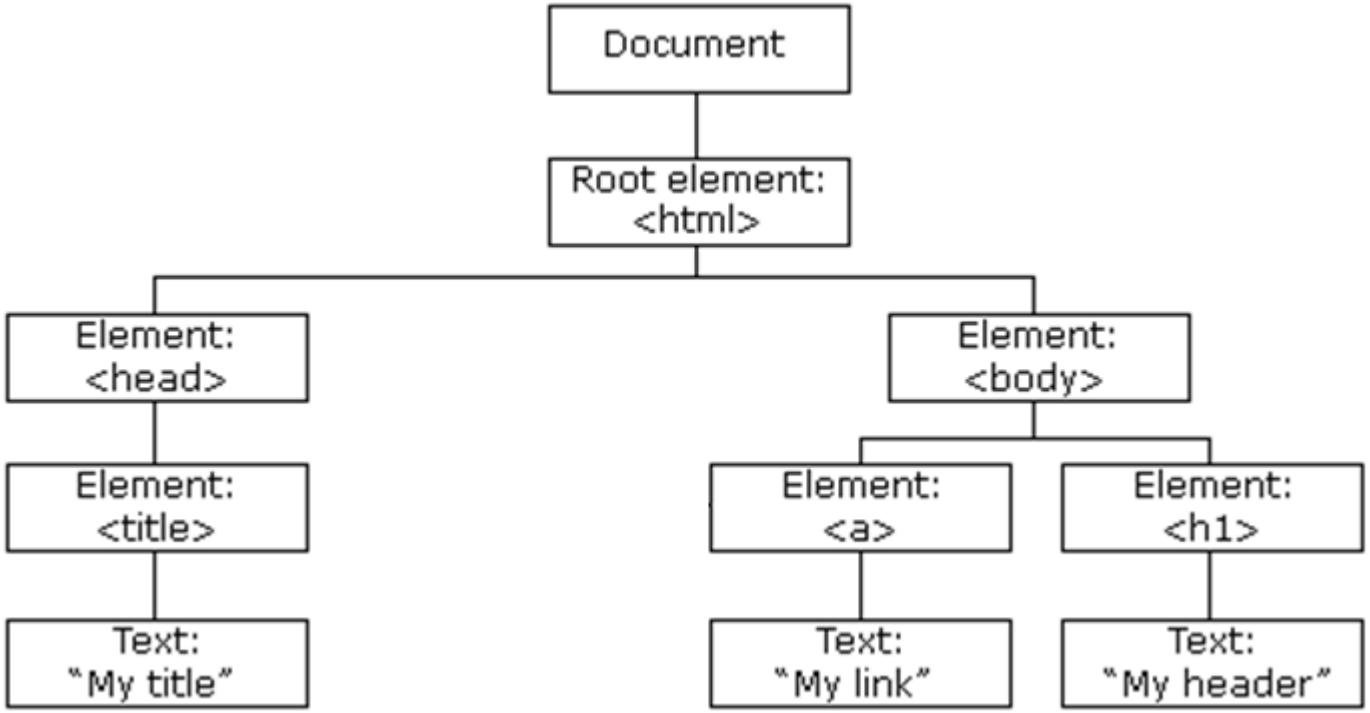
DOM element objects

- Every element on the page has a corresponding DOM object
- We can simply read/modify the attributes of the DOM object with **`objectName.propertyName`**



The DOM tree

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



There are 12 node types.
In practice we usually work with 4 of them:

- **document** – the "entry point" into DOM.
- **element nodes** – HTML-tags, the tree building blocks.
- **text nodes** – contain text.
- **comments** – sometimes we can put information there, it won't be shown, but JS can read it from the DOM.

- The elements of a page are nested into a tree-like structure of objects
- Every tree node is an object.

Finding HTML Elements

name	description
getElementById	returns the first element with the specified id.
getElementsByTagName	returns collection of all elements with the given tag, such as "div"
getElementsByName	returns collection of all elements with the given name attribute (mostly useful for accessing form controls)
getElementsByClassName	returns a collection of elements with a specified class name(s).
querySelector *	returns the first element that would be matched by the given CSS selector string
querySelectorAll *	returns a collection of all elements that would be matched by the given CSS selector string

Changing HTML Elements

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

Adding and Deleting Elements

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

DOM Navigation

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.

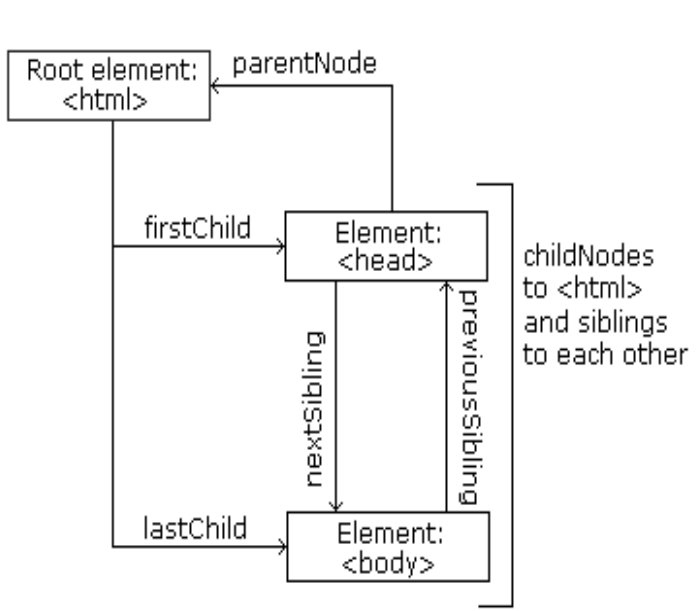
- In a node tree, the top node is called the **root** (or root node)
- Every node has exactly **one parent**, except the root (which has no parent)
- A node can have a number of **children**
- **Siblings** (brothers or sisters) are nodes with the same parent

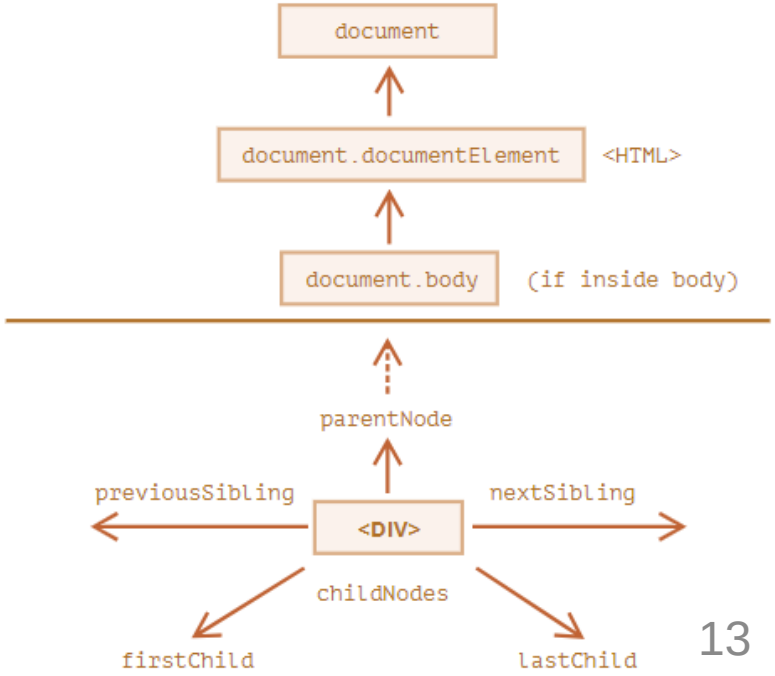
```
<html>

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

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

</html>
```





Traversing the DOM tree

- every element/node's DOM object has the following properties:

name(s) HTML element	name(s) Node	description
firstElementChild, lastElementChild	firstChild, lastChild	start/end of this element/node's list of children
children	childNodes	collection of all this element/node's children
nextElementSibling, previousElementSibling	nextSibling, previousSibling	neighboring elements/nodes with the same parent
parentElement	parentNode	the element that contains this element/node

- complete list of DOM node properties and methods

Modifying DOM nodes

- DOM nodes have fields that correspond to the attributes in HTML tags.

HTML attributes	DOM fields
title	.title
id	.id
class	.className
style="prop: value"	.style.prop = value

Getting/setting CSS classes in DOM

```
function highlightField() {
 // turn text yellow and make it bigger
 var elem = document.getElementById("id");

 if (!elem.className) {
       elem.className = "highlight";
 } else if (elem.className.indexOf("invalid") < 0) {
       elem.className += " highlight";
 }

}
```

- JS DOM's className property corresponds to HTML `class` attribute
- somewhat clunky when dealing with multiple space-separated classes as one big string
- className value is a string, not an array like we would want

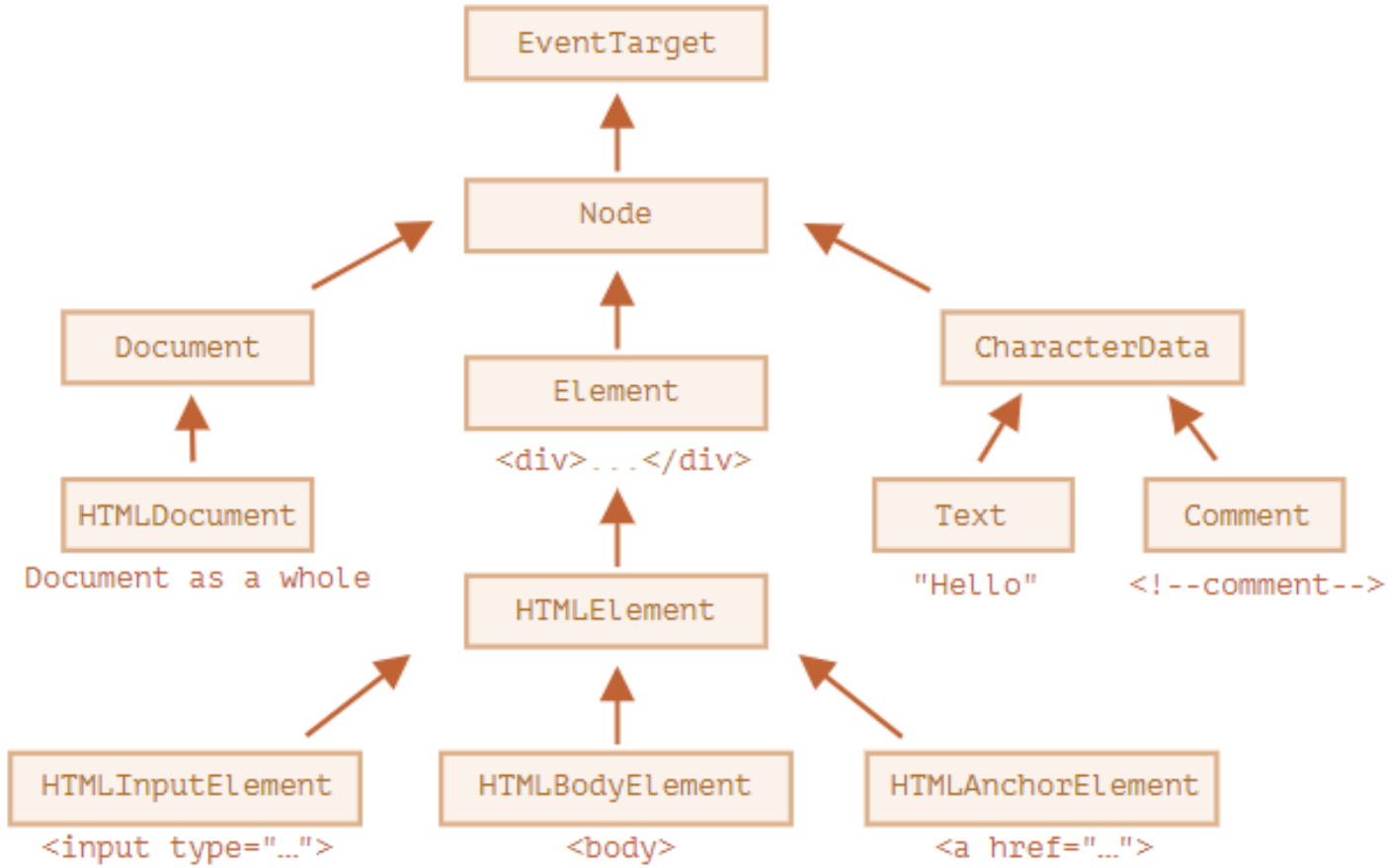
Adjusting styles with the DOM

```
<button id="clickme">Color Me</button>
window.onload = function() {
    document.getElementById("clickme").onclick = changeColor;
};
function changeColor() {
 const clickMe = document.getElementById("clickme");
 clickMe.style.color = "red";
}
```

Property	Description
style	lets you set any CSS style property for an element

- contains same properties as in CSS, but with **camelCasedNames**
  - examples: backgroundColor, borderLeftWidth, fontFamily

DOM node classes




```
console.log(document.body instanceof HTMLBodyElement); // true
console.log(document.body instanceof HTMLElement); // true
console.log(document.body instanceof Element); // true
console.log(document.body instanceof Node); // true
console.log(document.body instanceof EventTarget); // true
```

DOM Collections

The getElementsByTagName() method returns an HTMLCollection object.

An HTMLCollection object is an array-like list (collection) of HTML elements.

The elements in the collection can be accessed by an index number.

```
const myCollection = document.getElementsByTagName("p");
for (let i = 0; i < myCollection.length; i++) {
  myCollection[i].style.color = "red";
}
```

DOM Node Lists

A NodeList object is a list (collection) of nodes extracted from a document.

A NodeList object is almost the same as an HTMLCollection object.

All browsers return a NodeList object for the property childNodes.

Most browsers return a NodeList object for the method querySelectorAll().

```
const myNodelist = document.querySelectorAll("p");
for (let i = 0; i < myNodelist.length; i++) {
  myNodelist[i].style.color = "red";
}
```