WEB TECHNOLOGIES

Document Object Model

Prof.Vinay Joshi and Dr.Sarasvathi V
Department of
Computer Science and Engineering

DOM – Document Object Model

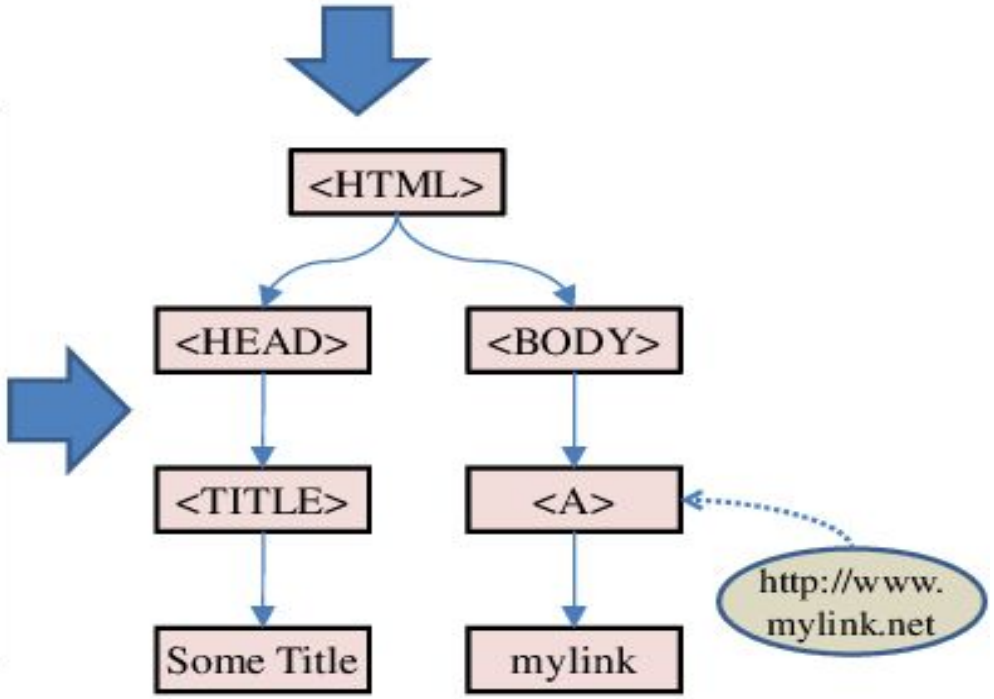
- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
- The **HTML DOM** model is constructed as a tree of **Objects.**
- The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

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

DOM – Document Object Model

```
<html>
<head> <title>Some Title</title> <!-- some text --> </head>
<body> <a href=http://www.mylink.net>mylink</a></body>
</html>
```

```
<HTML>
<HEAD>
<TITLE>Some Title
</TITLE>
<!-- other text -->
</HEAD>
<BODY>
<A HREF=http://www.mylink.net>
mylink</A>
</BODY>
</HTML>
```



DOM – Document Object Model

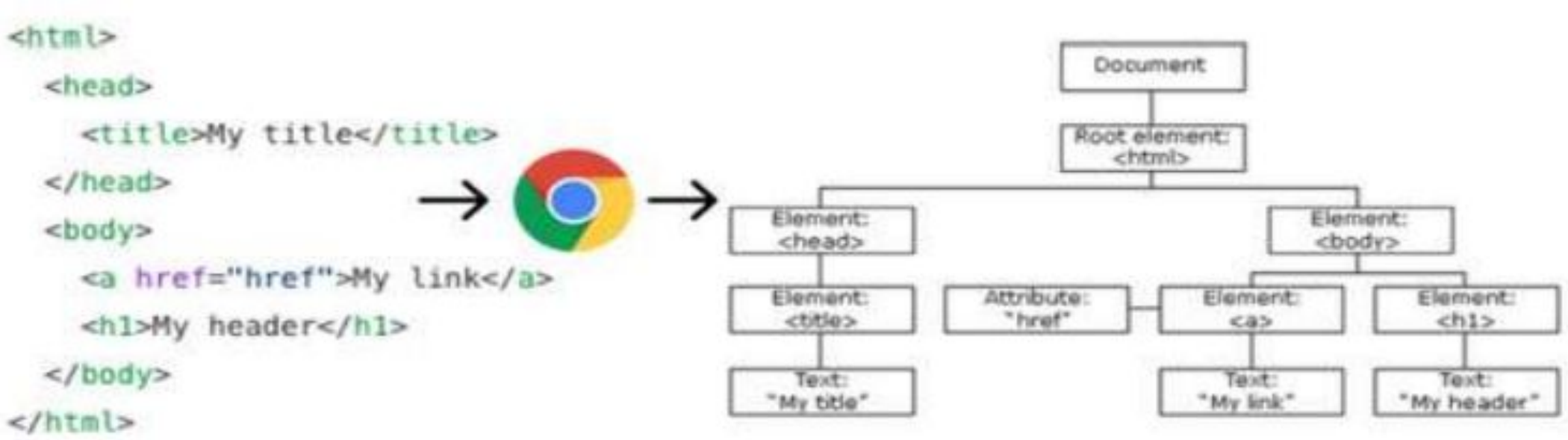
- The Document Object Model (DOM) represents that same document so it can be manipulated. The DOM is an object-oriented representation of the web page, which can be modified with a scripting language such as JavaScript.
- The DOM is not:

- part of the JavaScript language

- The DOM is:

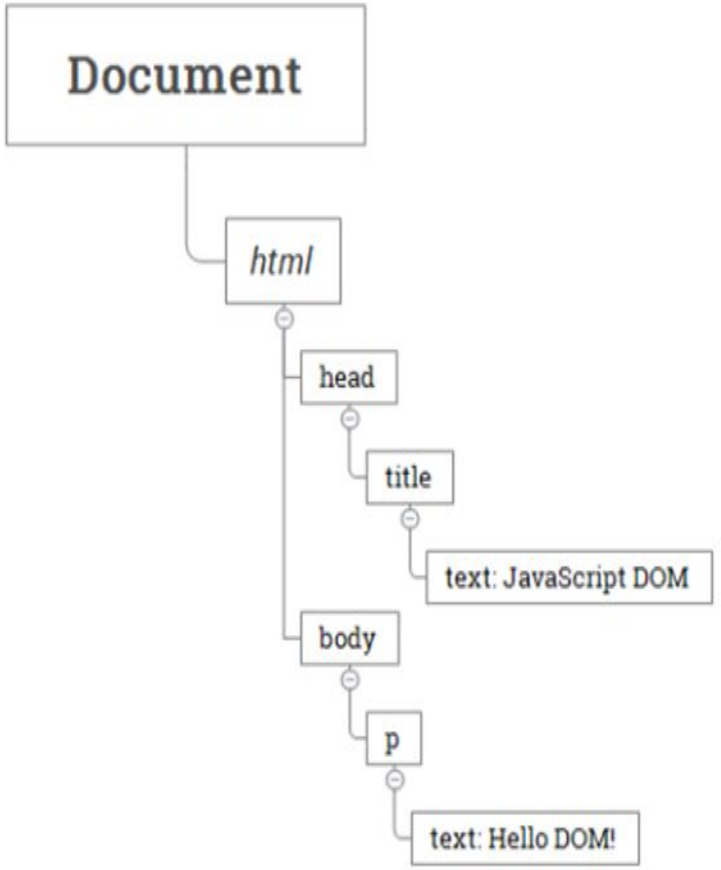
- constructed from the browser

- is globally accessible by JavaScript code using the document object



DOM – Tree Representation

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



In this DOM tree, the document is the root node. The root node has one child which is the <html> element. The <html> element is called the document element.

Document Object Model

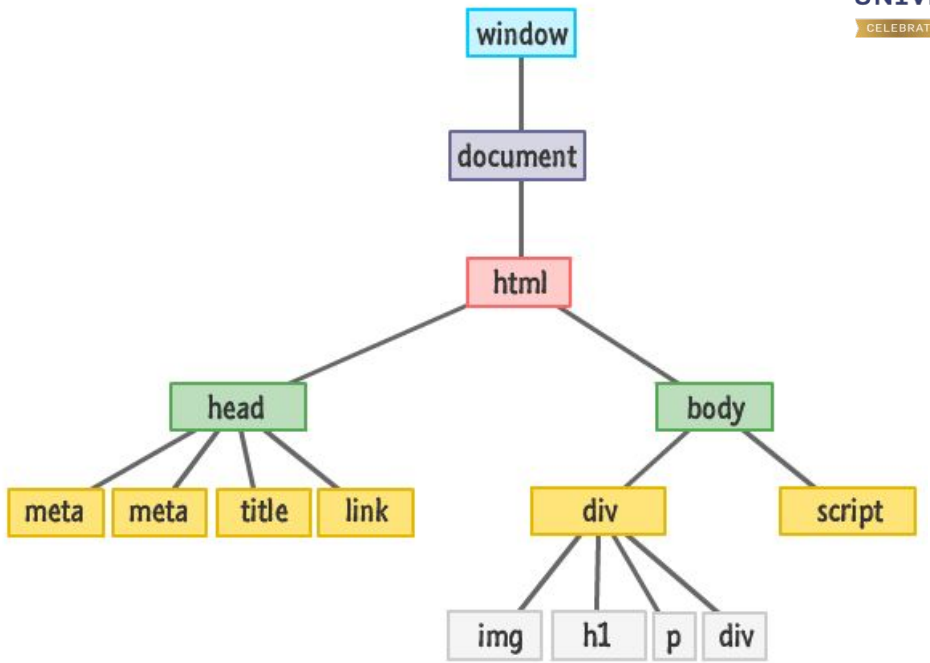
Drawbacks of using document.write()

- *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document.“*
- Document.write executed after the page has finished loading will overwrite the page, or write a new page, or not work
- Document.write practically only appending to the page

Document Object Model

Introduction to DOM

- A Web page is a document. This document can be either displayed in the browser window or as the HTML source. But it is the same document in both cases.
- The DOM is an object-oriented representation of the web page, which can be modified with a scripting language such as JavaScript.



DOM

- Objects have properties and methods, and respond to events.
  - Properties – specify attributes or characteristic of object .
  - Methods – specify functions object can perform.
  - Events – methods corresponding to user actions.

Document Object Model

The document Object

- The document object is the gateway to all the HTML elements and their styling properties that make up what gets shown.
- We can dynamically add elements, remove them, move them around, modify attributes on them.
- Any text, graphics or any information displayed on a web page is part of the document object.

DOM

DOM Elements Are Objects

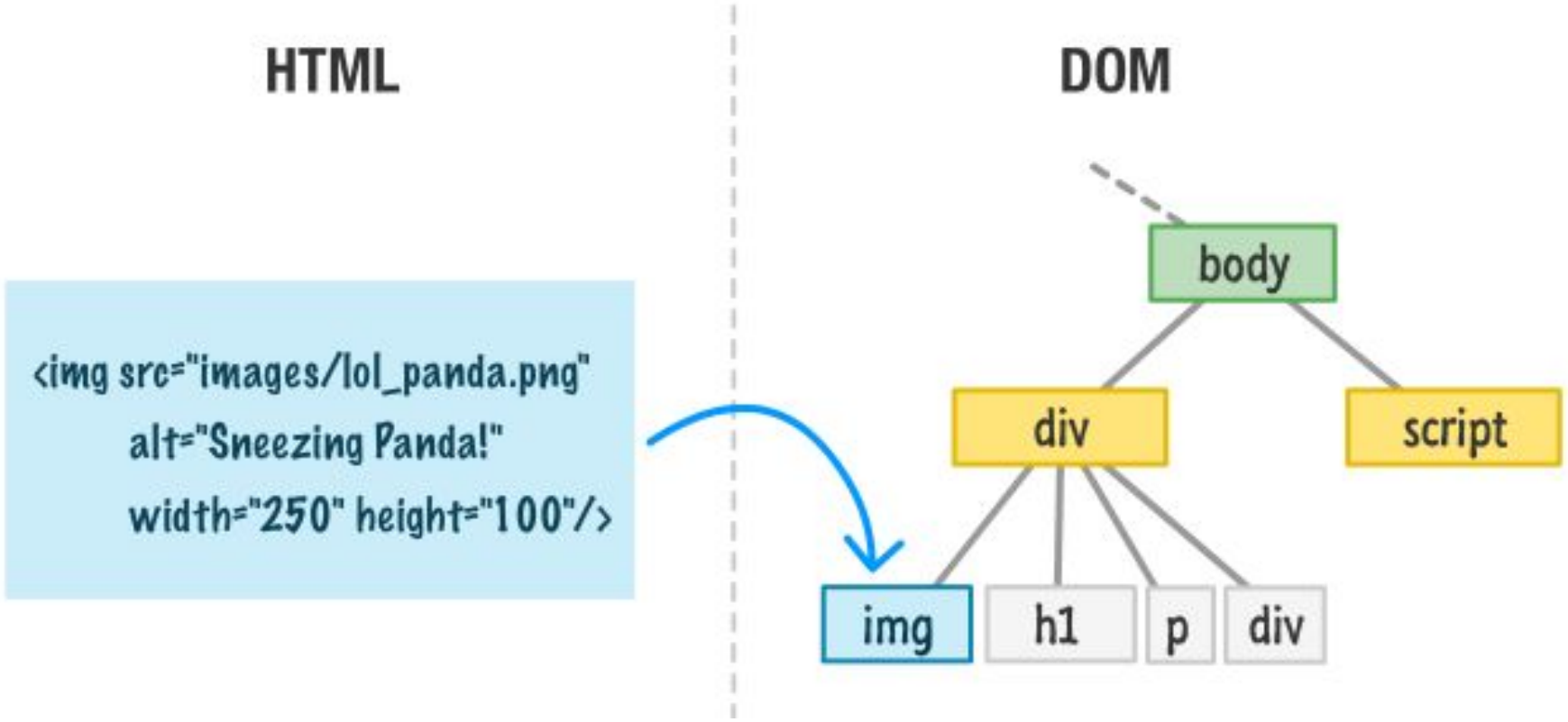
- Every HTML tag, style rule, and other things that go into your page has some sort of a representation in the DOM.

- An image element defined in markup:

```
<img src="images/lol_panda.png" alt="Sneezing Panda!"
width="250" height="100"/>
```

Document Object Model

DOM Elements are Objects



DOM
Accessing 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.

Document Object Model

Accessing Elements in DOM

PES UNIVERSITY
CELEBRATING 50 YEARS

Access Element By	Equivalent Selector	Method
ID	#demo	getElementByID(“demo”)
Class	.demo	getElementsByClassName(“demo”)
Tag	<tag name> like p	getElementsByTagName(“p”)
Selector (single)	Any CSS Selector	querySelector(“selector”)
Selector (all)		querySelectorAll(“selector”)

getElementById()

- The **document.getElementById()** method returns the element of specified id.
- The parameter of *getElementById* can be any expression that evaluates to a string.

syntax-

document.getElementById(“#id");

getElementsByTagName ()

- *getElementsByTagName* is used to access elements and attributes using tag name.
- This method will return an array of all the items with the same tag name as a NodeList object.

syntax-

document.getElementsByTagName(tagname)

getElementsByName()

- The *getElementsByName*() method returns a collection of all elements in the document with the specified name (the **value** of the name attribute), as a NodeList object.

Syntax-

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

Document.querySelector()

- The Document method **querySelector()** returns the first Element within the document that matches the specified selector, or group of selectors.
- If no matches are found, null is returned.

Syntax-

element = document.querySelector(*selectors*);

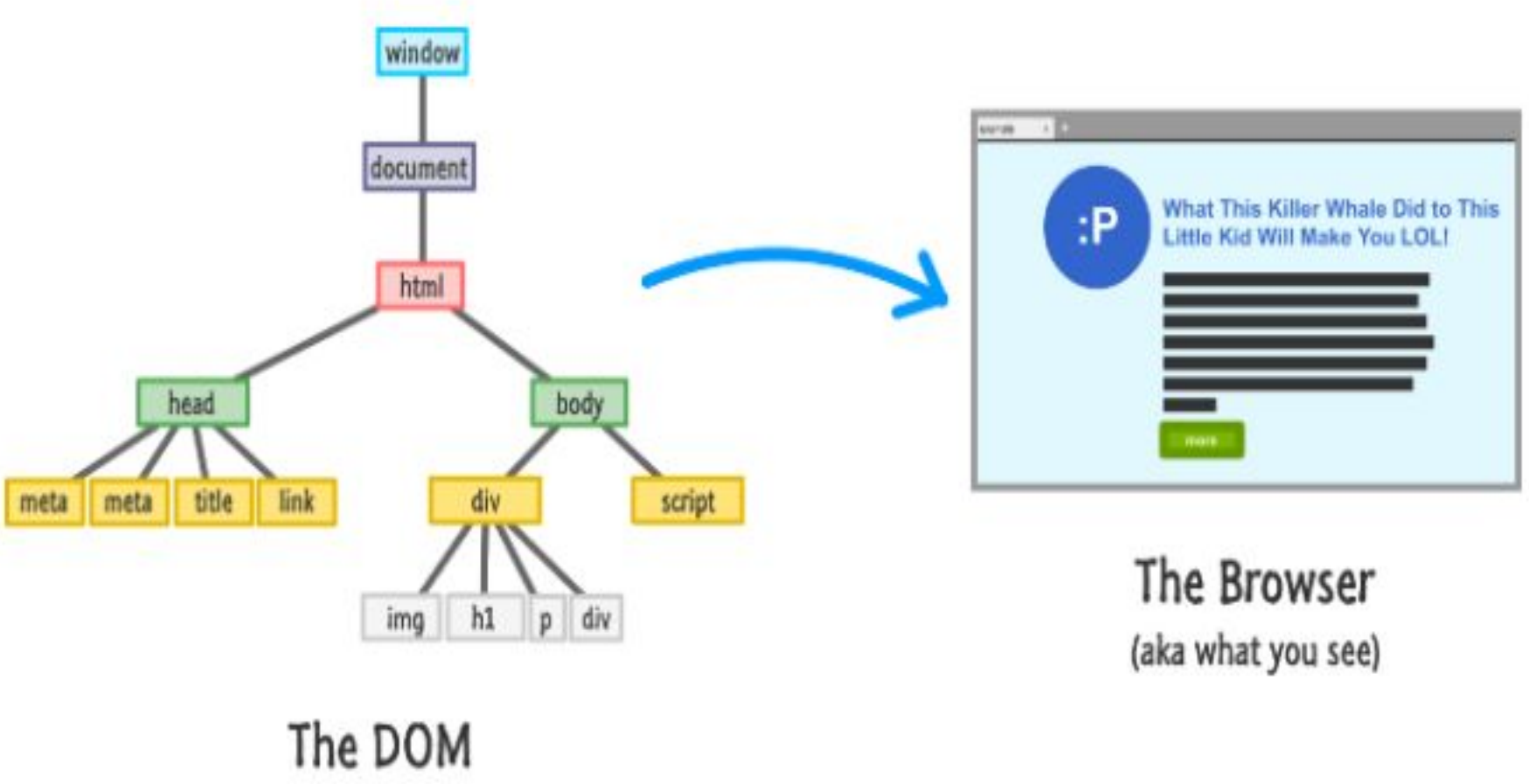
Document.querySelectorAll()

- The Document method **querySelectorAll()** returns a static (not live) NodeList representing a list of the document's elements that match the specified group of selectors.
- If no matches are found, null is returned.

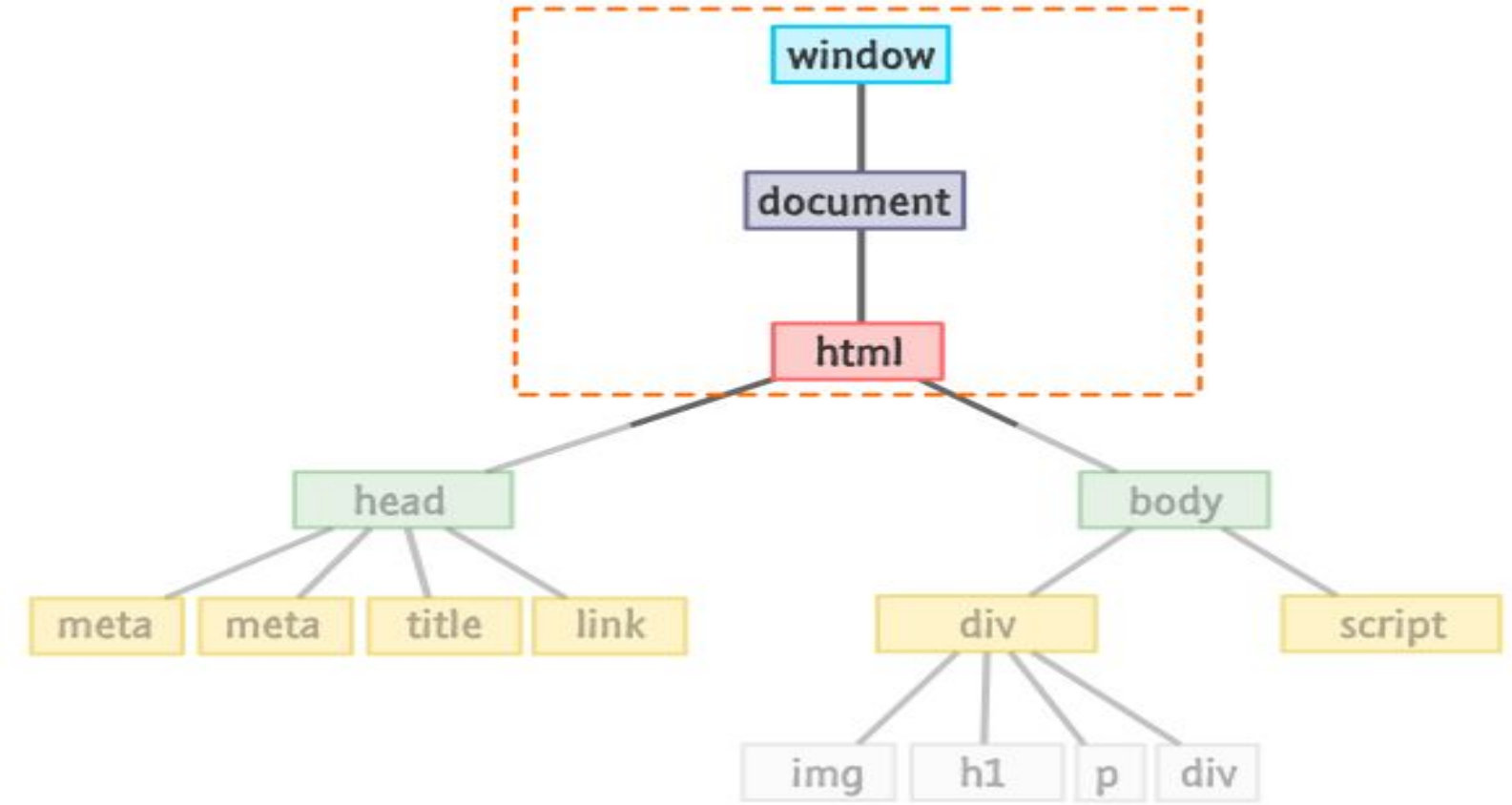
Syntax-

elementList= *parentNode*.querySelectorAll(*selectors*);

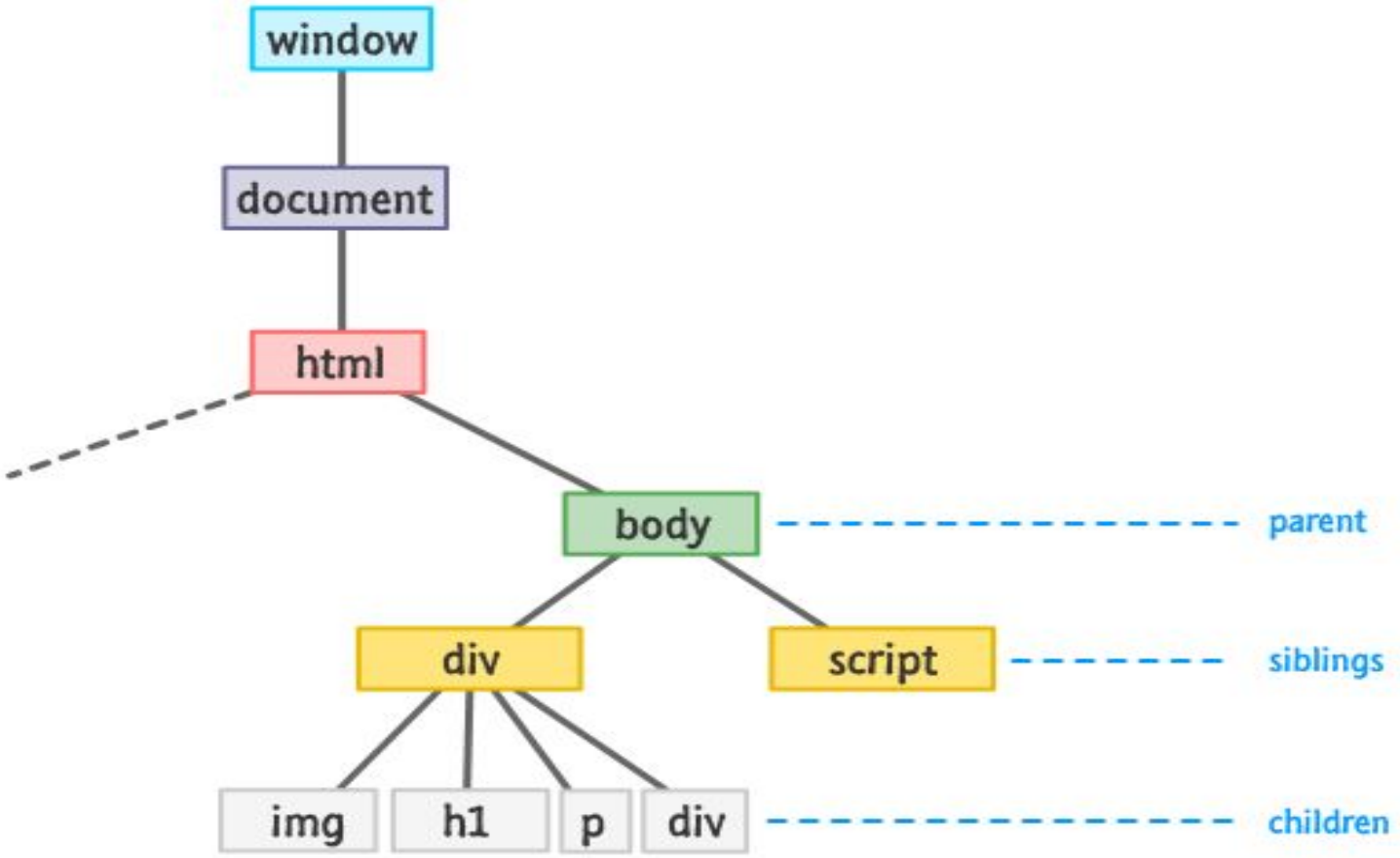
Traversing the DOM



Traversing the DOM

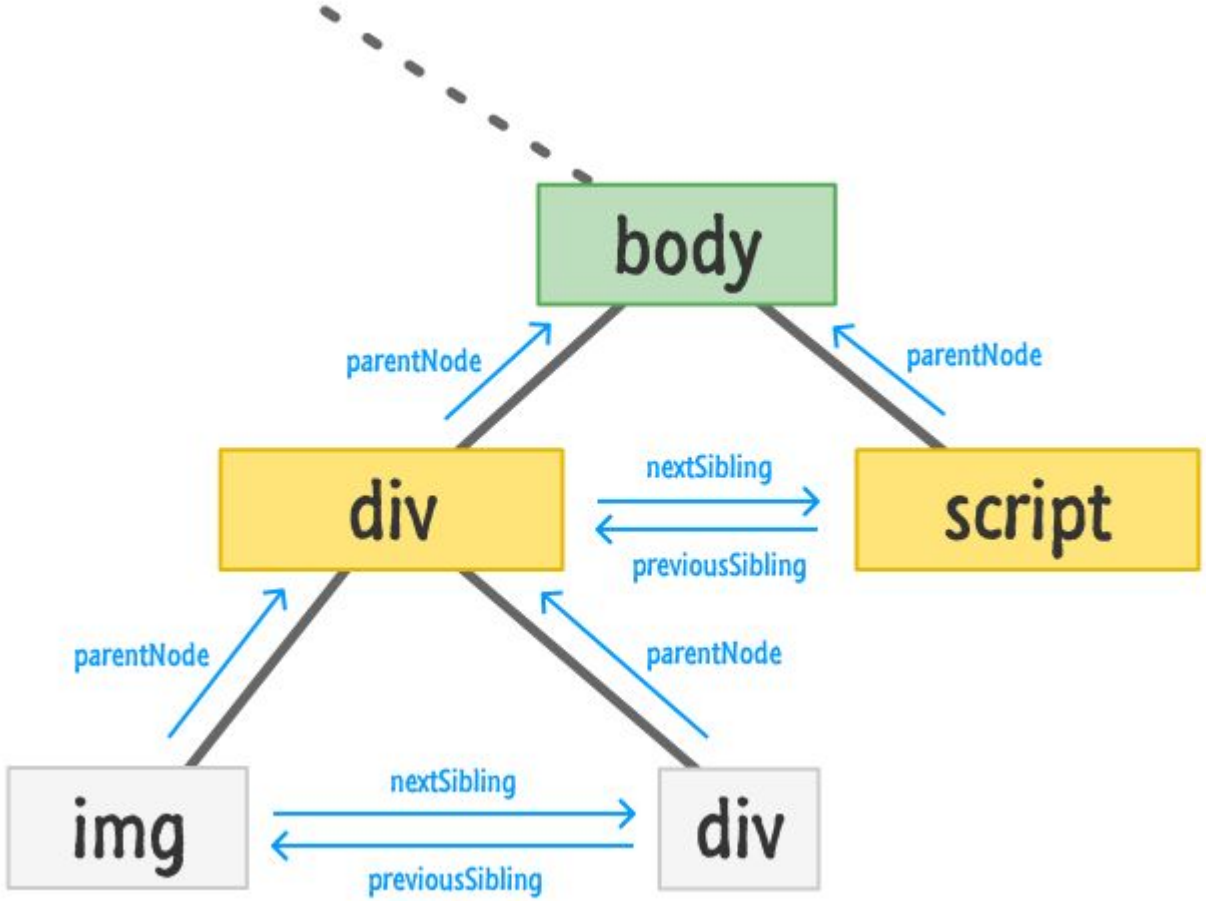


Traversing the DOM



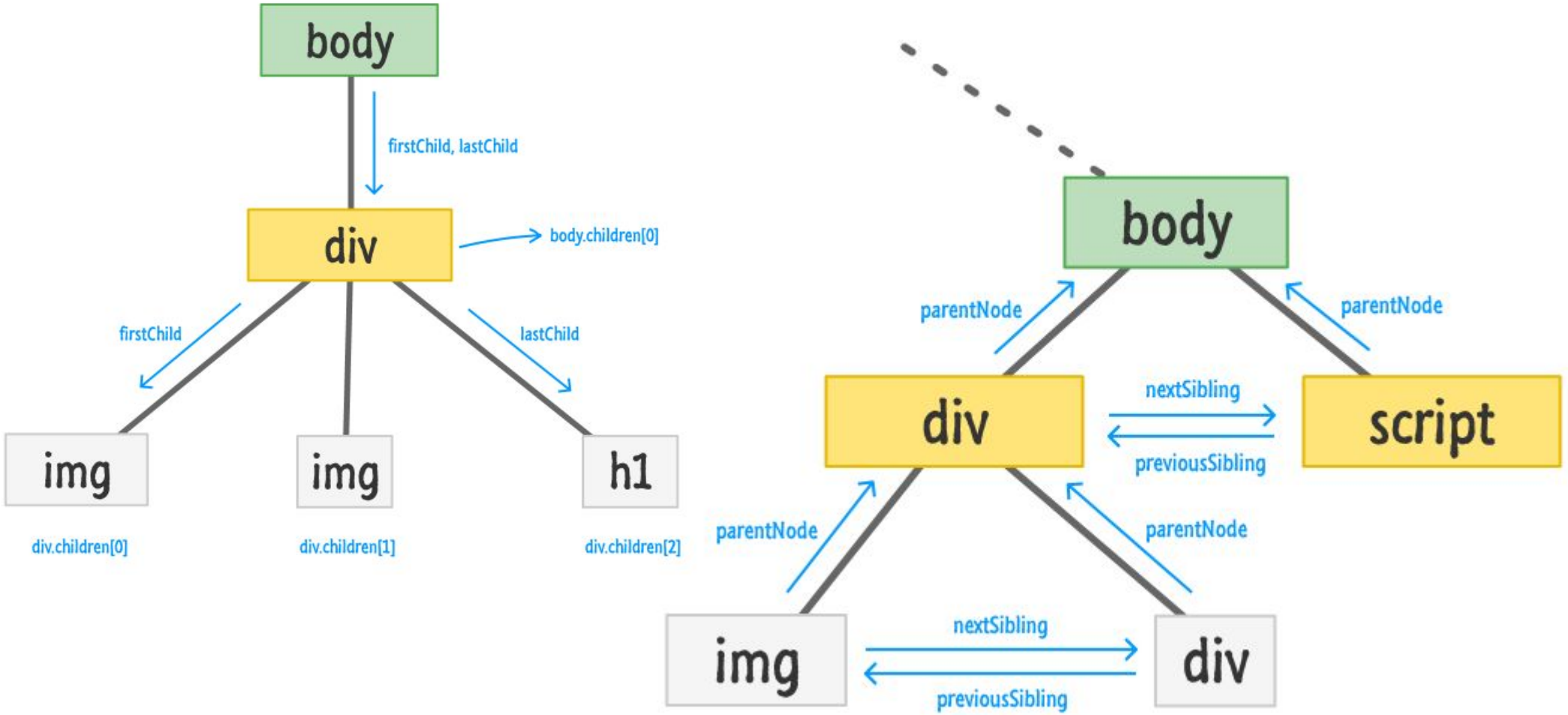
Document Object Model

Traversing the DOM



Document Object Model

Traversing the DOM



Document Object Model

Creating Element Objects

Method	Description
document.createElement()	Create a new element node using tag
document.createTextNode()	Create a new text node

Property	Description
node.textContent or node.innerText	Get or set the text content of an element node (without HTML tags)
node.innerHTML	Get or set the HTML content enclosed in the element tag

Document Object Model

Manipulating Nodes in the DOM

Method	Description
node.appendChild()	Add a node as the last child of the parent element.
node.insertBefore()	Insert a node into the parent before a specific sibling node
node.replaceChild()	Replace an existing node with a new node
node.removeChild()	Removes child node
node.remove()	Removes a node

* **node** here can be document.body or any existing element in the DOM

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu

+91 80 2672 6622