

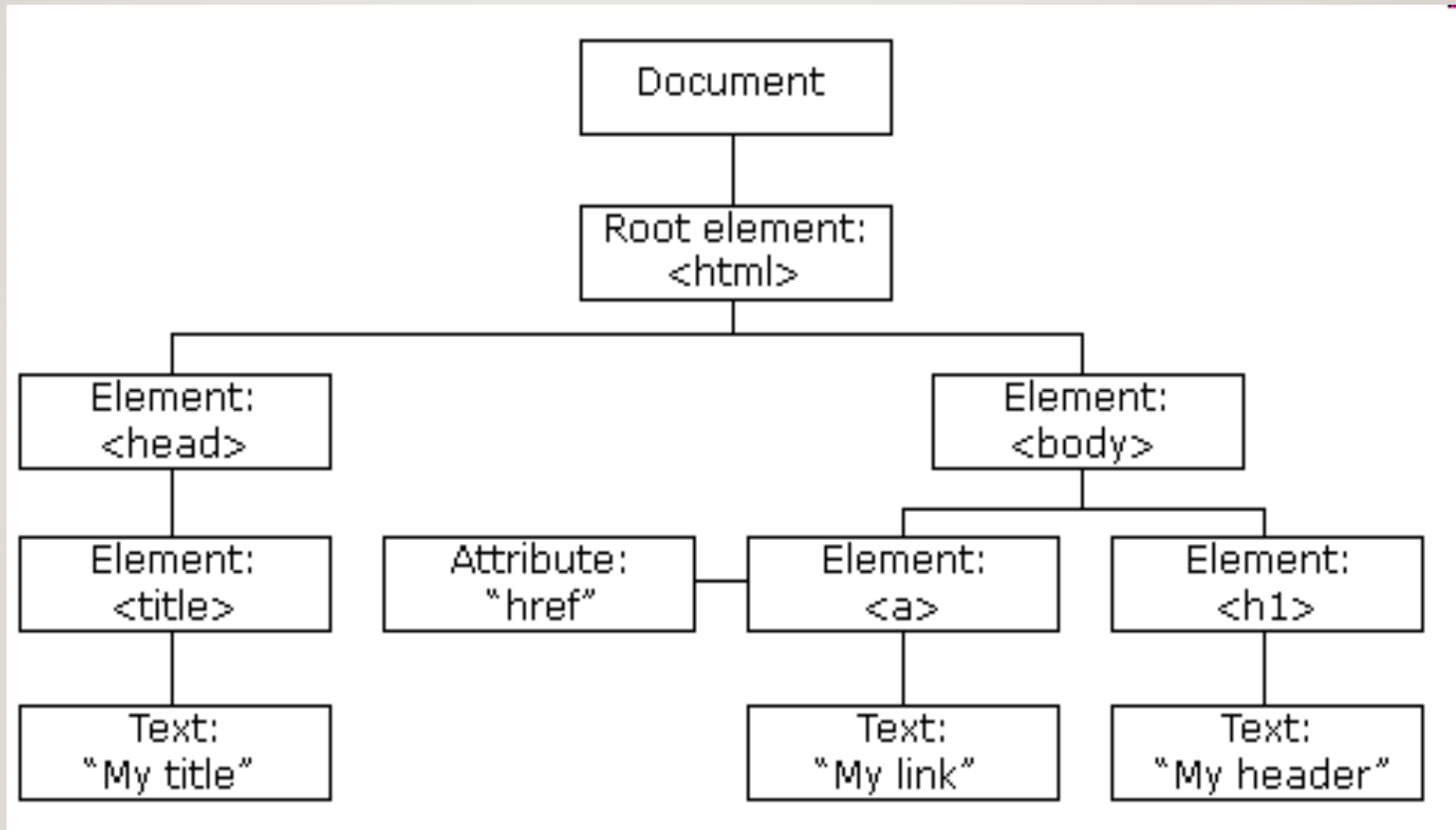
LECTURE – 6

- HTML DOM
- AJAX

INTRODUCTION TO HTML DOM

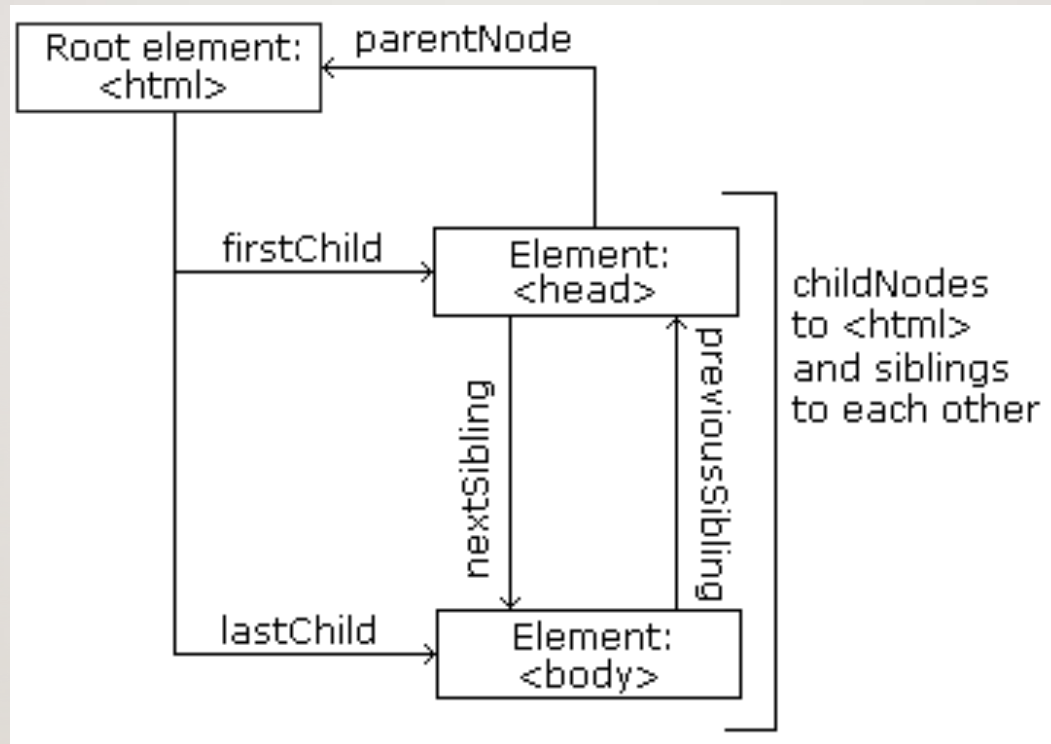
- DOM – Document Object Model
- HTML DOM
 - Defines **objects** and properties of HTML elements.
 - HTML data is modeled as a “tree of objects”.
 - A standard object model for HTML
 - A standard programming interface for HTML
 - Platform and language-independent
- Using Javascript, one can
 - Add new elements (e.g., image, text, video, etc.) to the HTML content
 - Delete existing elements
 - Modify the content, style, background, design, etc. of existing elements.

HTML DOM NODE TREE



Source: W3 schools web page

HTML DOM



Source: W3 schools web page

HTML DOM TREE – EXAMPLE

```
<html>
  <head>
    <title>DOM Tutorial
  </title>
</head>
<body>
  <h1>DOM Lesson one </h1>
  <p> Hello world! </p>
</body>
</html>
```

- `<html>` node is the root node
 - Has no parent node
- Parent node of the `<head>` and `<body>` nodes is the `<html>` node.
- Parent node of the "Hello world!" text node is the `<p>` node
- `<html>` node has two child nodes
 - `<head>` and `<body>`
- `<head>` node has one child node
 - `<title>` node
- `<title>` node has one child node
 - text node "DOM Tutorial"
- `<h1>` and `<p>` nodes are siblings
 - Both child nodes of `<body>`

TREE STRUCTURE ... CONTD.

- Follow the standard “tree” nomenclature
- Top node is called the root
- Every node, except the root, has exactly one parent node.
 - Root has none.
- A node can have any number of children
- Leaf is a node with no children
- Siblings are nodes with the same parent

BACK TO THE EXAMPLE ...

- `document` - the current HTML document
- `getElementById("intro")` - the element with the id "intro"
- `childNodes[0]` - the first child of the element
- `nodeValue` - the value of the node (e.g., text)

HTML DOM

- Everything in HTML is a node.
- Entire document – a document node.
- Every HTML tag – an element node.
- Text in the HTML elements – text nodes.
- Every HTML attribute is an attribute node.
- Comments are comment nodes.

HTML DOM – OBJECT MODEL

- Each node is an **object**.
- Objects have methods
- Can use methods to retrieve or change HTML content **dynamically**.

⇒ Basis for Dynamic HTML (**DHTML**)

ACCESSING NODES

- `getElementById (<id>)`
- `getElementsByTagName(<tag>)`
- A combination of the above
 - Using the DOM tree and parent/child relationship.

HTML METHODS

- For any HTML element (node) x
 - x.getElementById(*id*)
 - get the element with a specified id
 - x.getElementsByTagName(*name*)
 - get all elements with a specified tag name. Tag = “body”, for example.
 - x.appendChild(*node*)
 - insert a child node to x
 - x.removeChild(*node*)
- Details can be found at
https://www.w3schools.com/js/js_htmlDOM_document.asp

HTML DOM PROPERTIES

- For any HTML element (node) `x`,
 - `x.innerHTML` - the inner text value of `x`
 - `x.nodeName` - the name of `x`
 - `x.nodeValue` - the value of `x`
 - `x.parentNode` - the parent node of `x`
 - `x.childNodes` - the child nodes of `x`
 - `x.attributes` - the attributes nodes of `x`

ADDING EVENT LISTENERS

- Event listeners can be added/removed
- E.g., a callback function can be added/removed
 - For any event – mouse click, mouse over, etc.
- Adding event listeners
 - `addEventListener(event, function, useCapture)`
 - E.g. `addEventListener(click, myFunction1, false)`
 - Call `myFunction1` for the event `onClick`
 - `addEventListener(mouseover, myFunction2, false)`
 - Call `myFunction2` for the event `onmouseover`
- Removing event listeners
 - `removeEventListener(event, function)`
 - E.g. `removeEventListener(click, myFunction1)`
 - `removeEventListner(mouseover, myFunction2)`

HTML METHODS ... CONTD.

- `x.firstChild`
 - Returns the first child node
- `x.lastChild`
 - Returns the last child node
- `document.documentElement`
 - Returns the root node
- `document.body`
 - Gives access to the body tag.

HTML NODES

- DOM node properties
 - nodename – read only
 - Element node – same as the tag name
 - Text node – text
 - Document node – document
 - nodevalue – can be changed
 - Elements – undefined
 - Text nodes – text itself
 - Attributes – attribute value
 - nodetype – read only
 - Element – 1; Attribute – 2; Text – 3;
 - Comment -8; Document - 9

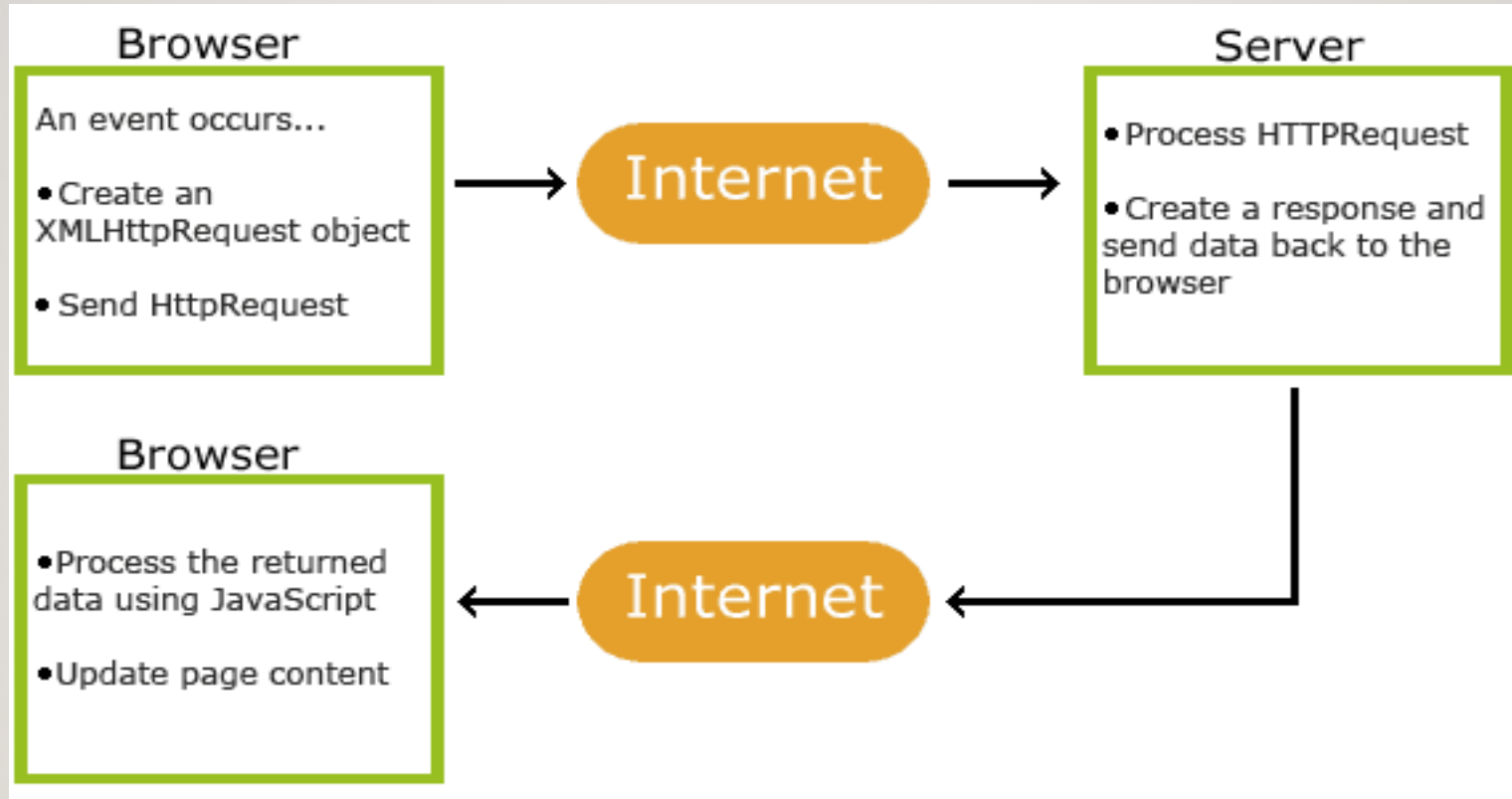
PUTTING IT ALL TOGETHER

- Nodes can be accessed.
 - HTML content can be changed
 - Background color, image sources
 - Text, font, color of an element
 - Add Javascript events to the mix
- ⇒ Can create a very dynamic web content, animation, etc.
- Basis for today's web pages.

AJAX – ASYNCHRONOUS JAVA AND XML

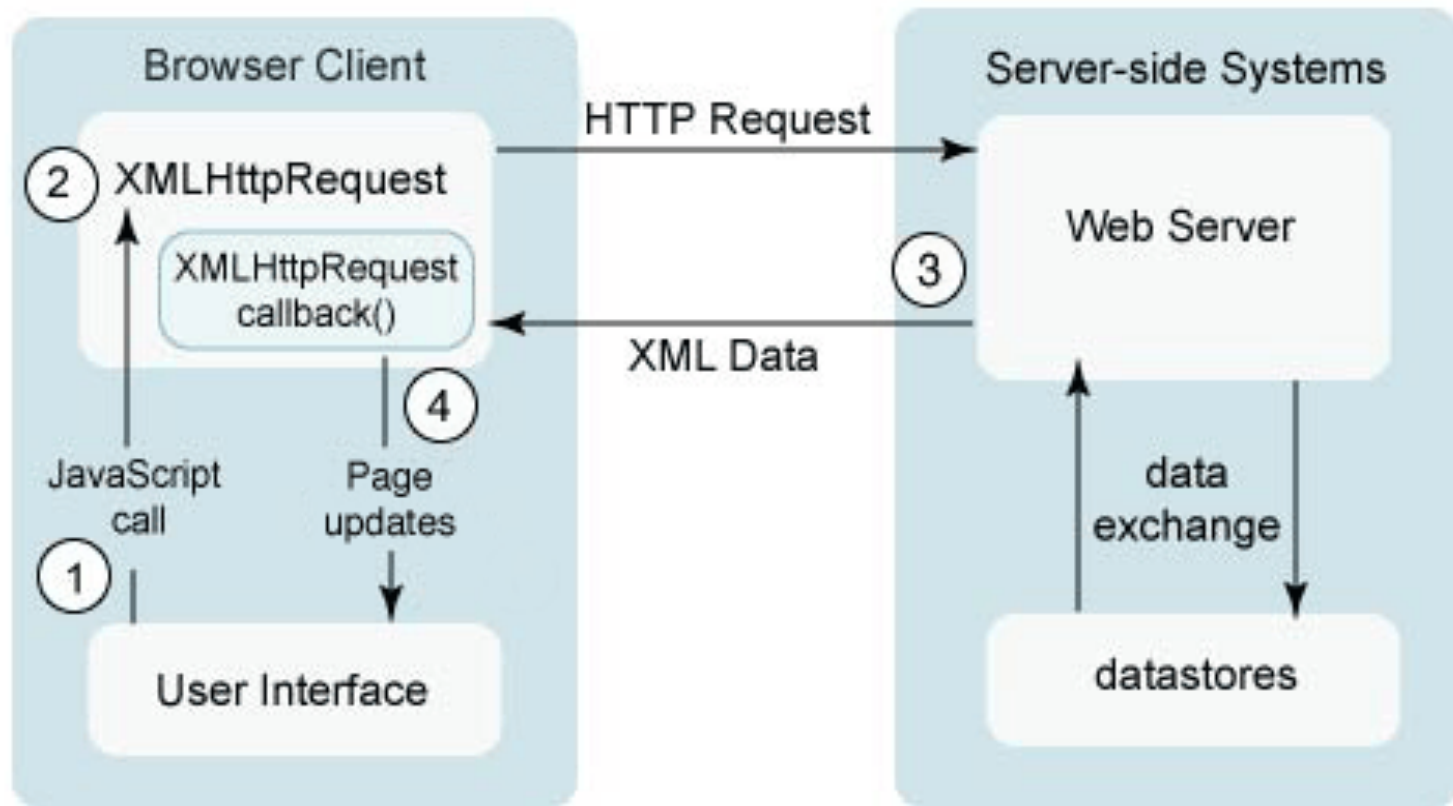
- Made popular by Google (with Google Suggest).
- NOT a new programming language
 - A new way to use existing standards.
- Based on JavaScript and HTTP requests.
- With AJAX, JavaScript communicates
 - Directly with the (web) server
 - using XMLHttpRequest object
 - To retrieve data as needed
 - Using Javascript events (e.g., keyPressed)
 - **WITHOUT** refreshing the page.

HOW DOES AJAX WORK ... CONTD.



Source: W3Schools

HOW DOES AJAX WORK



Source: SUN's JAVA web page

AJAX ... CONTD.

- Note: Data is typically stored in XML format
- XMLHttpRequest
 - The basic data structure interfacing the client with server.
 - Sends a request to a server (e.g., Google suggest server) on any events
 - Like “onKeyUp(..)” when the user types any character search key.
 - Receives data from the server
 - Updates the required fields with data received from server.

REQUESTS AND RESPONSES

- XMLHttpRequest
 - open(..) // open a request to ...
 - setRequestHeader(..) // Set the request header
 - send(..) // Send the request
 - status // response 200 OK, or other values
 - readState // Store the state information
 - onreadystatechange // callback function for status change
 - responseText // response in Text
 - responseXML // response in XML

XMLHTTPREQUEST FUNCTIONS

- `open(method, url, async, user, psw)`
 - method: Get/Post
 - url: address of the server
 - sync – true or false (asynchronous call or not)
 - user – username (optional)
 - psw – Password (optional)
- `setRequestHeader( )` // Sets label/value pairs in the header
- `send( )` or `send(string)` // Sends the request to the server
- `abort( )` // Cancel the current request
- `getResponseHeader( )` // Get specific header information
- `getAllResponseHeaders( )` // Get all headers' information

`XMLHttpRequest` OBJECT PROPERTIES

- `readyState` – Holds the status of `XMLHttpRequest` object
 - 0: Request Not initialized
 - 1: Server connection established
 - 2: Request received
 - 3: Processing request
 - 4: Request finished, response is ready
- `Status` – Returns the status number of a request
 - 200: OK
 - 403: Forbidden
 - 404: Not Found
 - ..Others
- `statusText` – Status text of a response (e.g., “OK”, “Not Found”, etc.)
- `responseText` – response data as a string
- `responseXML` – response as XML data