

Web Technology S2 2021/22

Week 01 -Introduction



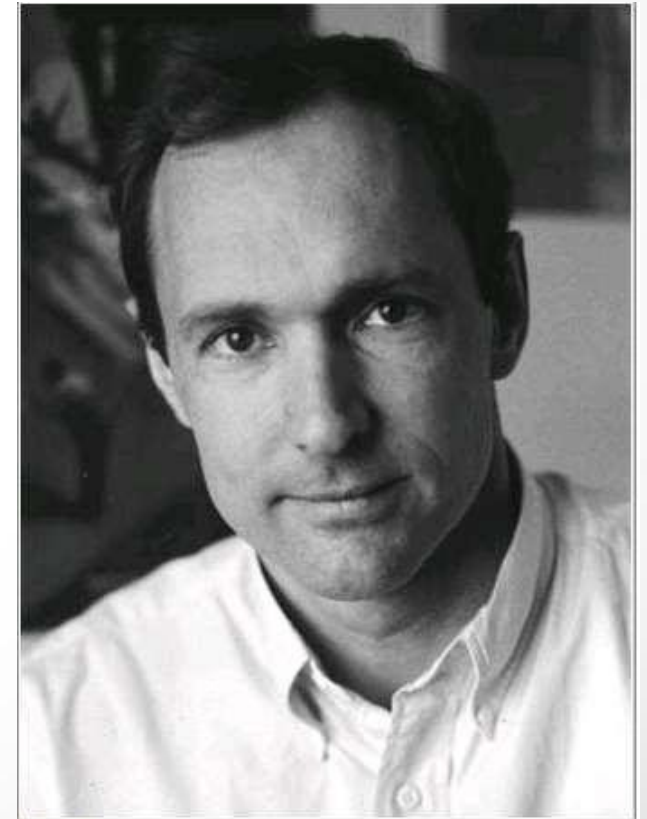


WWW Short History:

Year	Achievements
1980	Tim Berners-Lee invents the WWW, at CERN (the world famous nuclear research lab at Switzerland).
1990	Concepts like - HTTP, Web browser, Uniform Resource Identifier (URI) and HTML
1993	Launch of the first graphical web browser, named - MOSAIC at USA.
1994	Hosting of the first International WWW Conference, formation of World Wide Web Consortium (W3C)
1996	Commercialization of the Web
1998	Google was founded by Larry Page and Sergey Brin
1999	Concept of Dot-com, its boom and bust.
2002	Launch of Web 2.0
2004	Launch of Facebook and use of Internet for social networking

World Wide Web (WWW):

- ❖ Tim Berners-Lee has developed WWW, HTML, URLs, and HTTP.
- ❖ In 1980: Tim Berners-Lee invents the WWW, at CERN (the world famous nuclear research lab at Switzerland).
- ❖ In 1989: Tim Berners-Lee invents the Web with **HTML** as its publishing language.
- ❖ In June 1994, the IETF published Berners-Lee's RFC-1630, the first Request for Comments that acknowledged the existence of URLs and URNs. Most importantly, it defined a formal syntax for Universal Resource Identifiers (URI).
- ❖ Vint Cerf is considered to be one of the fathers of the **Internet**, having been the co-inventor of **TCP/IP**.

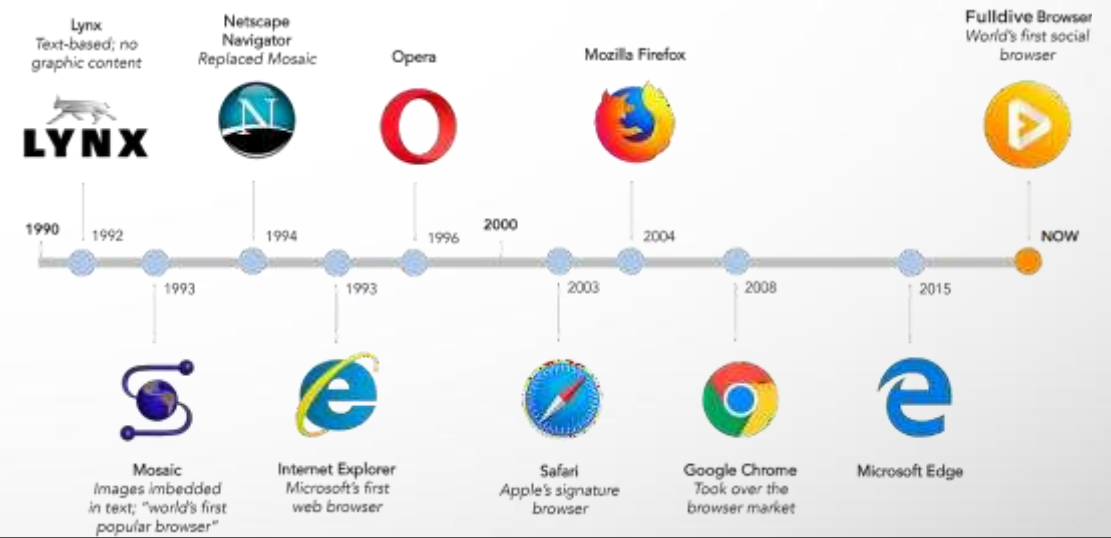


[107] Tim Berners-Lee, the father of the World Wide Web.

Web Browser:



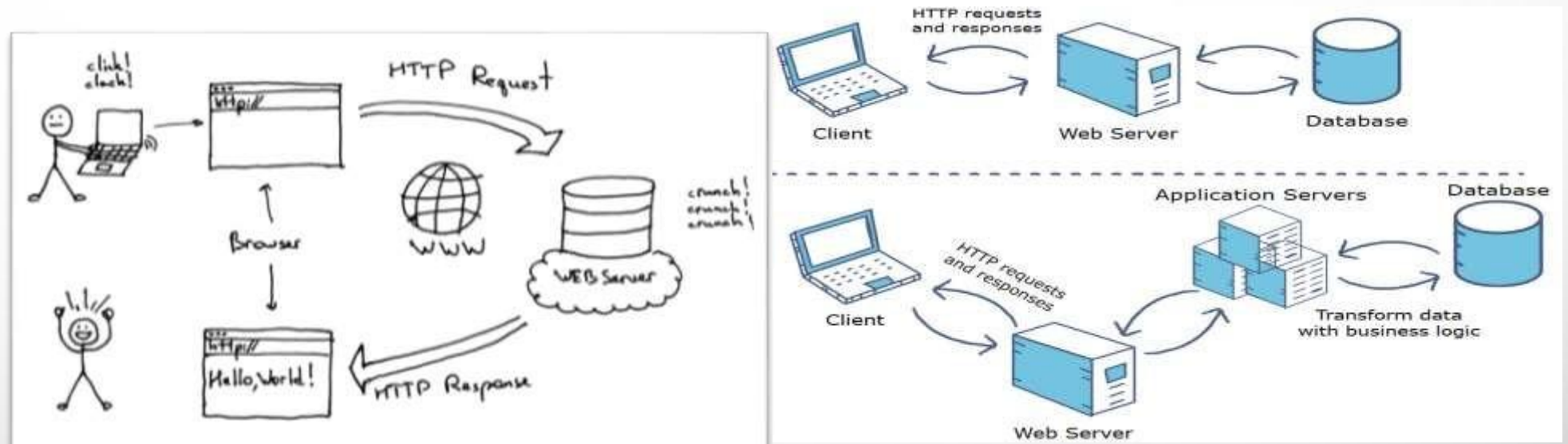
- ❖ A web browser is a software application for retrieving, presenting, and traversing information resources on the World Wide Web.
- ❖ The primary purpose of a web browser is to bring information resources to the user.
- ❖ The major web browsers are Windows Internet Explorer, Mozilla Firefox, Apple Safari, Google Chrome and Opera.



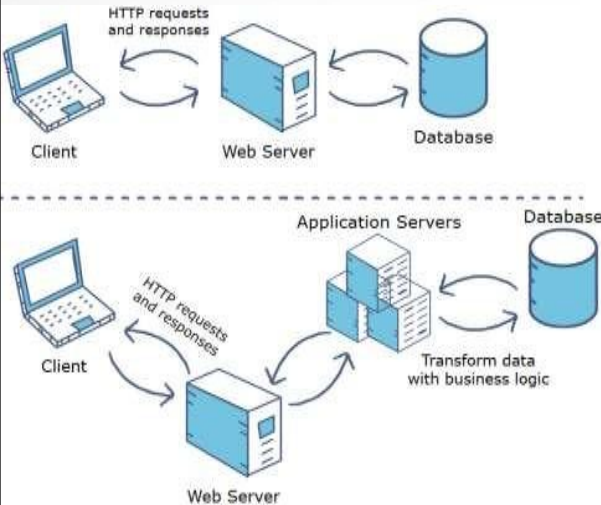
Web Server:



- ❖ The term **web server** or **webserver** can mean one of two things:
- ❖ A computer program that accepts HTTP requests and return HTTP responses with optional data content.
- ❖ A computer that runs a computer program as described above.

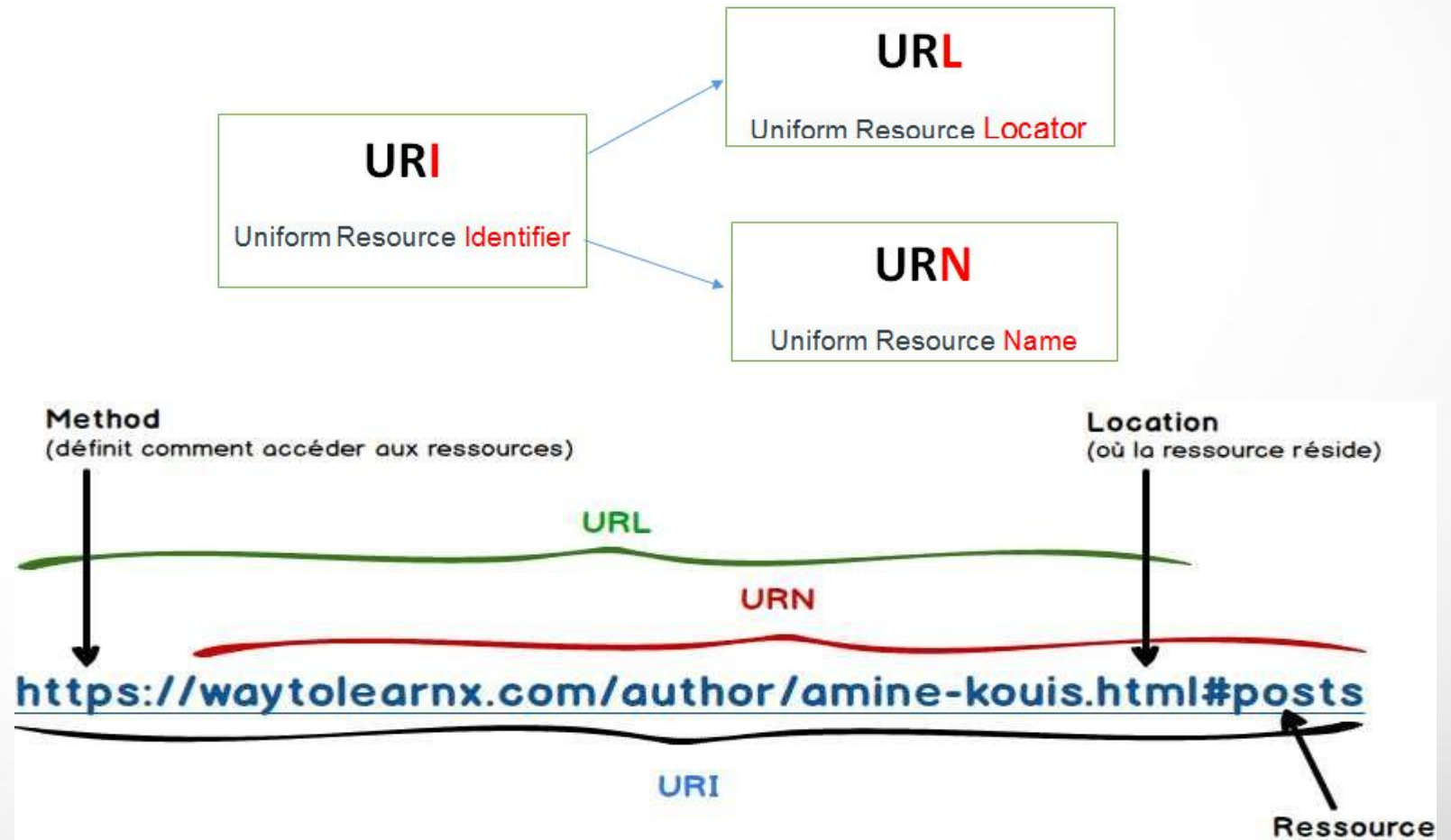


Web Server vs Application Server:



Web Server	Application Server
Deliver static content.	Delivers dynamic content.
Content is delivered using the HTTP protocol only.	Provides business logic to application programs using several protocols (including HTTP).
Serves only web-based applications.	Can serve web and enterprise-based applications.
No support for multi-threading.	Uses multi-threading to support multiple requests in parallel.
Facilitates web traffic that is not very resource intensive.	Facilitates longer running processes that are very resource-intensive.

URI:

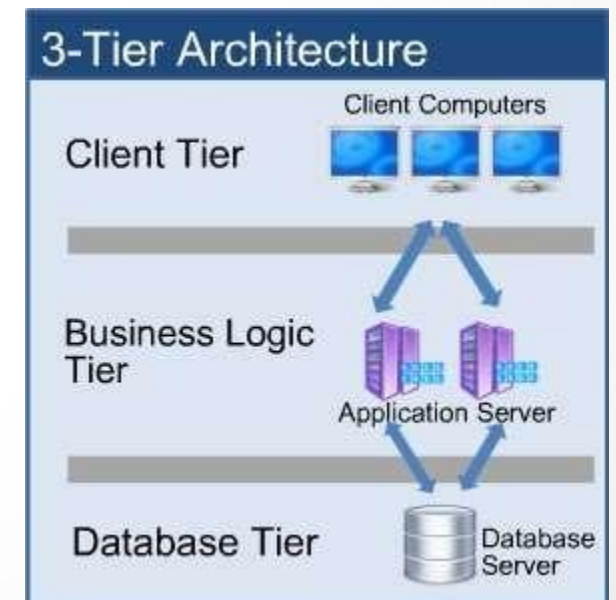
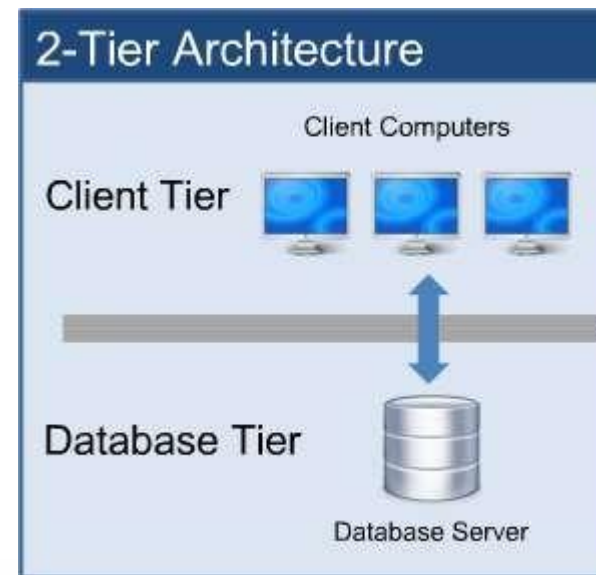




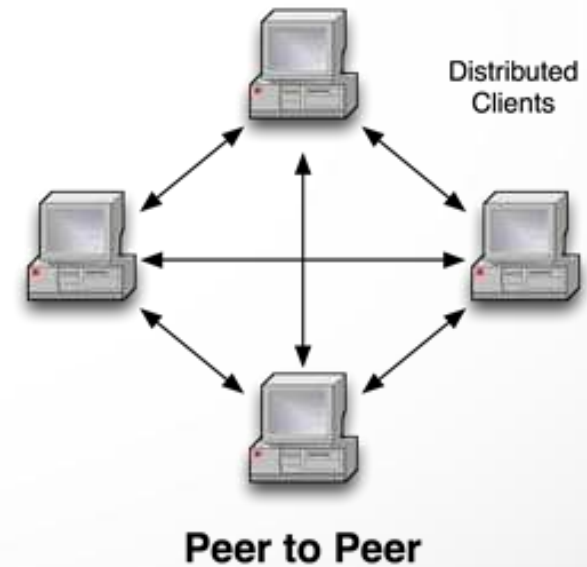
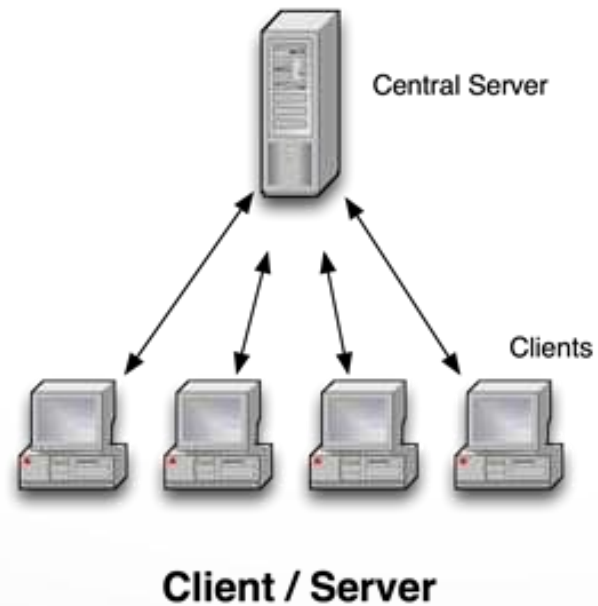
HTTP:

- ❖ HTTP is an application-level protocol for distributed, collaborative, hypermedia information systems.
- ❖ HTTP is a request/response standard of a client and a server.
- ❖ Typically, an HTTP client initiates a request.
- ❖ Resources to be accessed by HTTP are identified using Uniform Resource Identifiers (URIs).
- ❖ Port Id is 80.

Web Architecture:



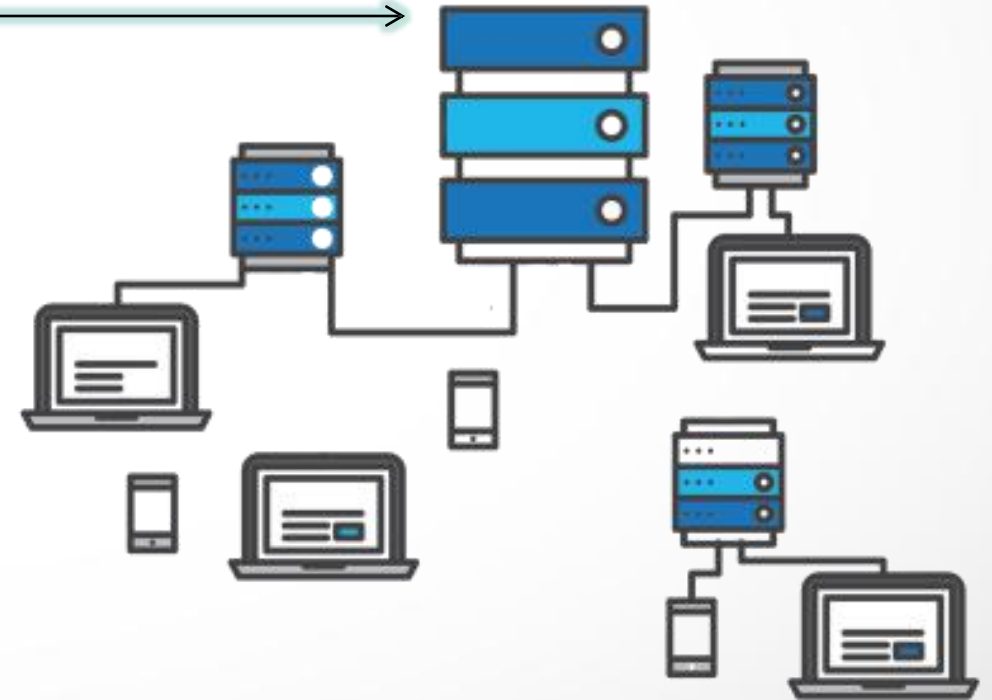
Different types of Networks:



Website Access:



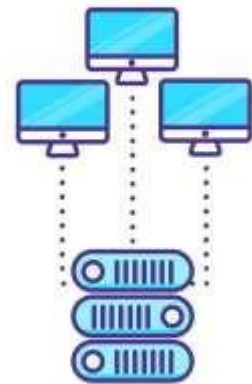
Server Side Programming



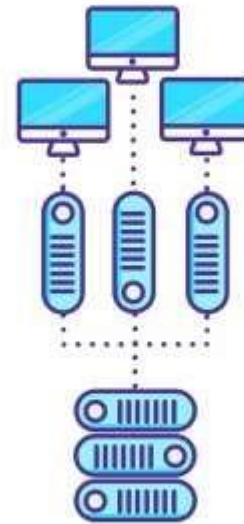
Access Website / Server Side Program

Different types of Webhosting:

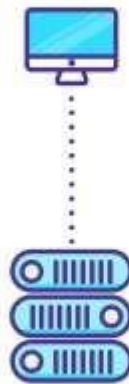
The Three Main Types of Web Hosting



Shared Hosting

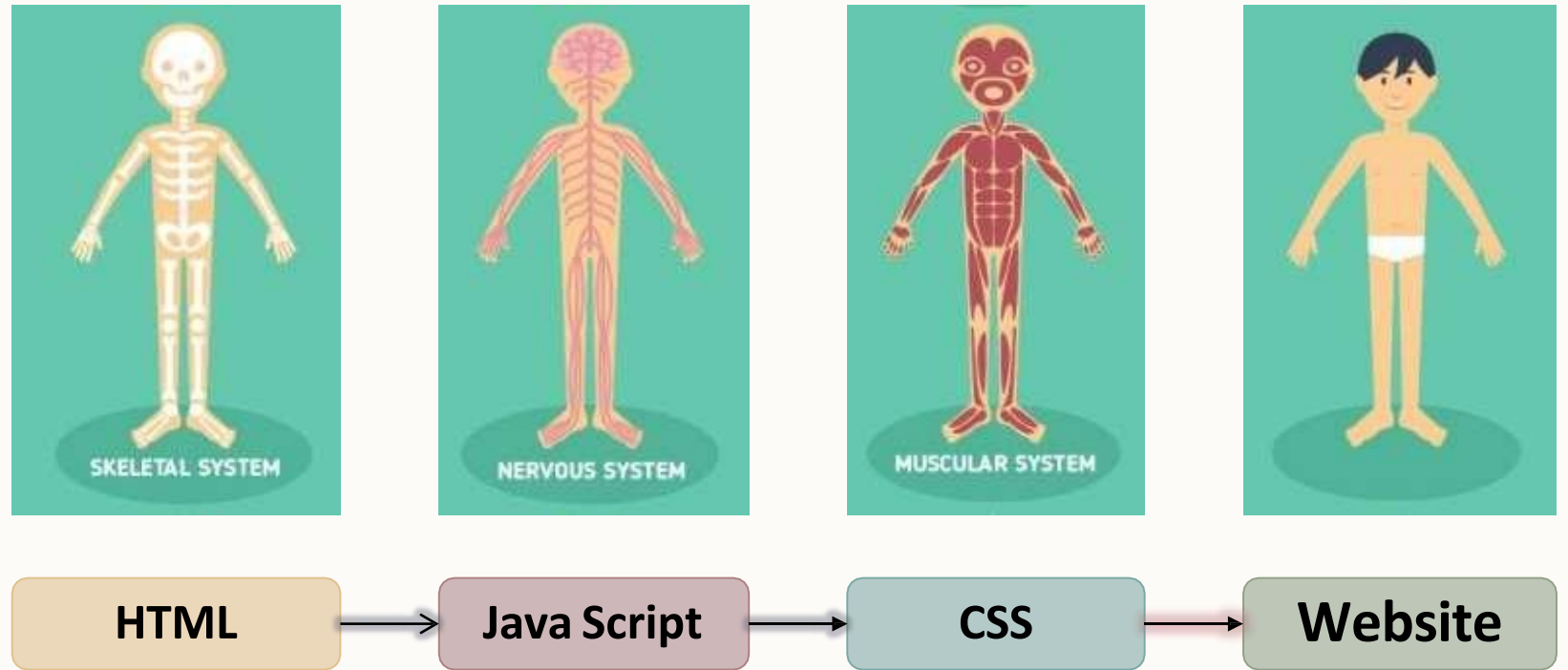


VPS Hosting

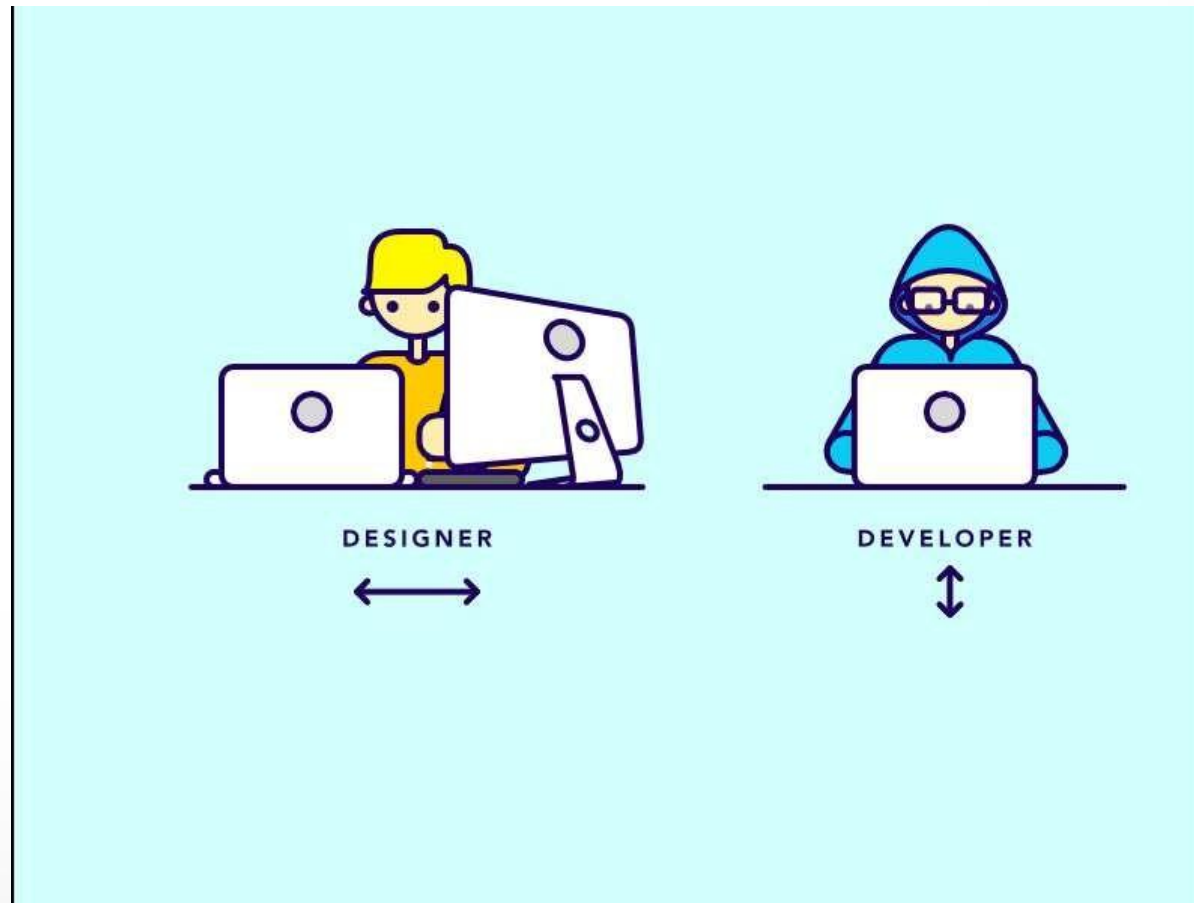


Dedicated Hosting

Components of Website:



Designer vs Developer:



UX vs UI:



User Research

- Primary Research
- Desktop Research
- Competitor Research

Research Analysis

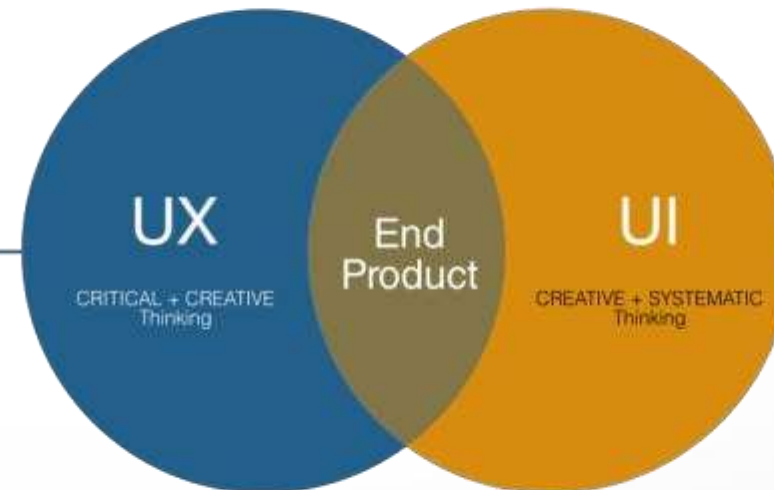
- Insights and Findings
- Customer Journey Maps
- Personas
- Empathy Maps
- Site Map
- Task flows
- User Flows

Design Ideation

- Idea sketches
- Wireframes
- Prototypes
- User Testing
- Design Iterations

Execution

- Co-ordination with stakeholders
- Transition to UI design
- Collabrate with UI designers



Interface Design

- Knowledge transfer on wireframes
- Look and Feel
- Branding and guidelines
- Moodboards
- Style Tiles
- Layout and Responsiveness
- User Testing

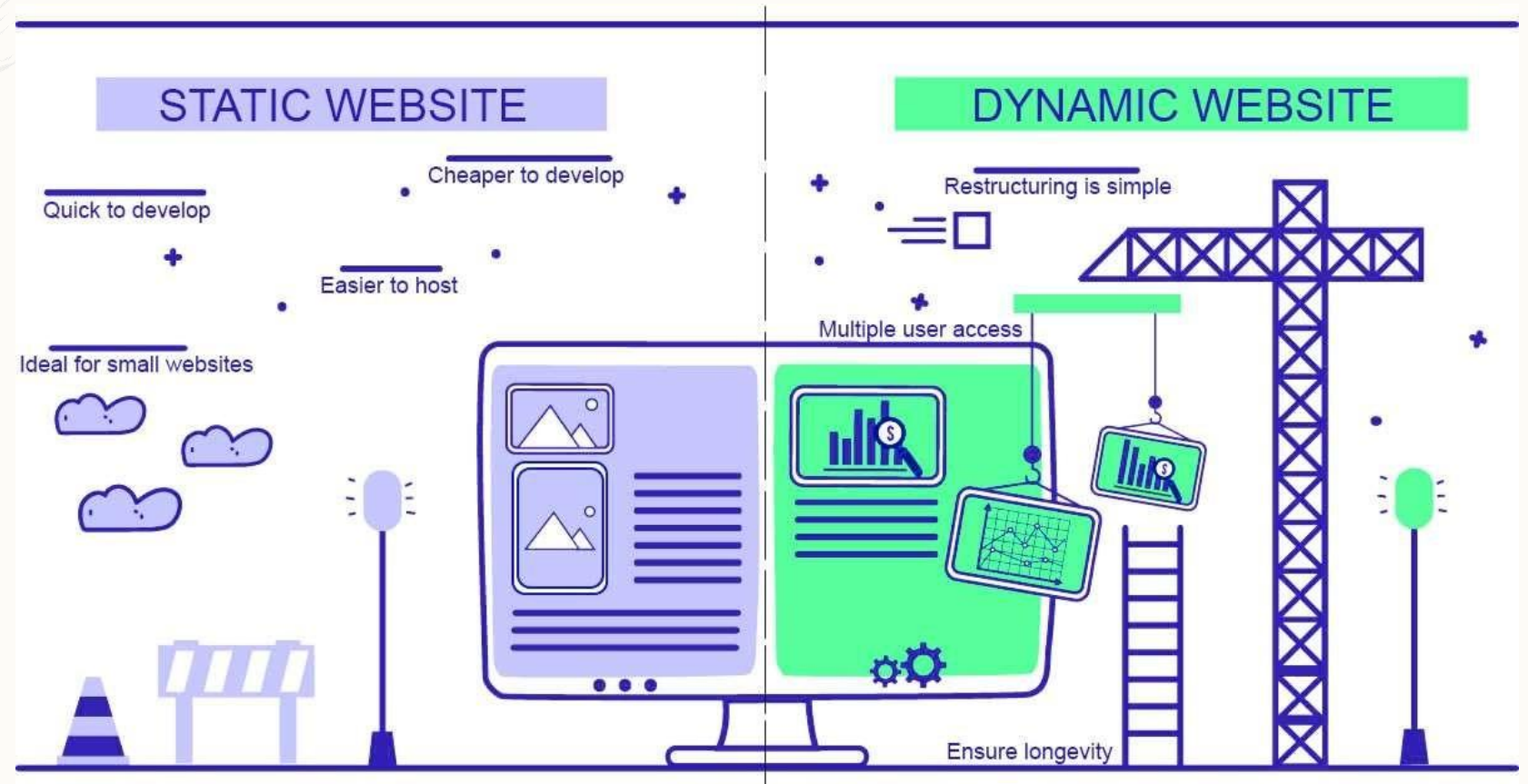
Design Specs

- UI Prototype
- Visual Design Documentation
- Developer's handout
- Icon and Illustration set
- Adaption to form factors and resolutions

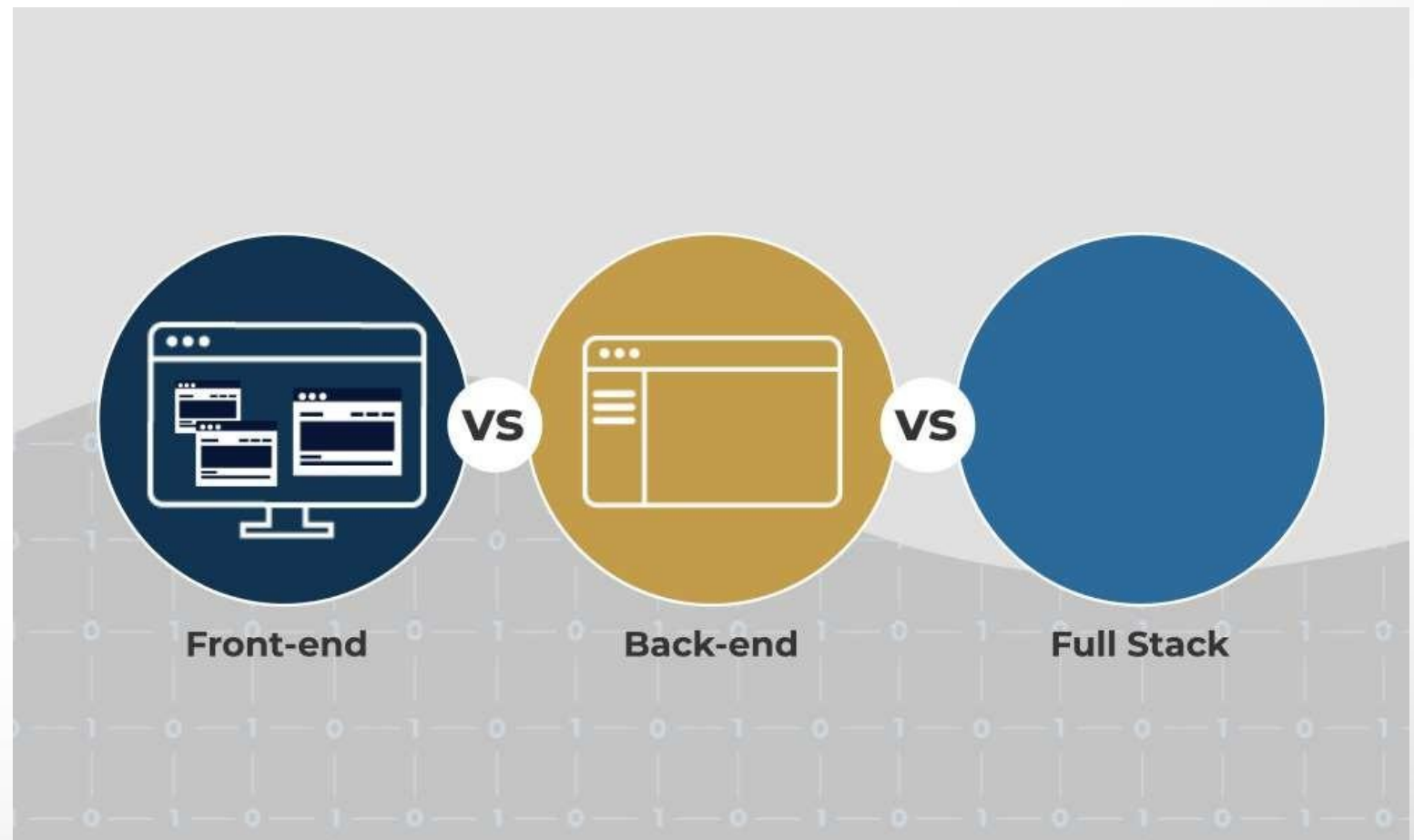
Execution

- Assist developers
- Implementation Reviews
- Co-ordination with stakeholders
- Transition to final design

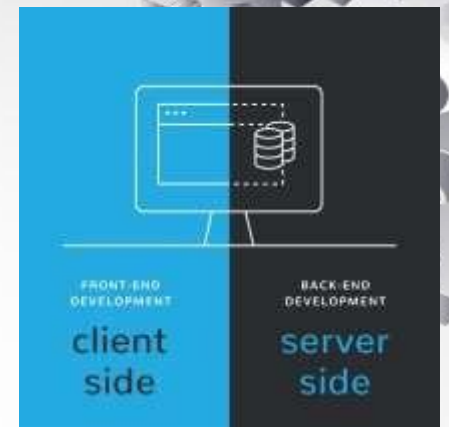
Types of Website:



Types of languages:



Front-End vs Back-End:



Front End

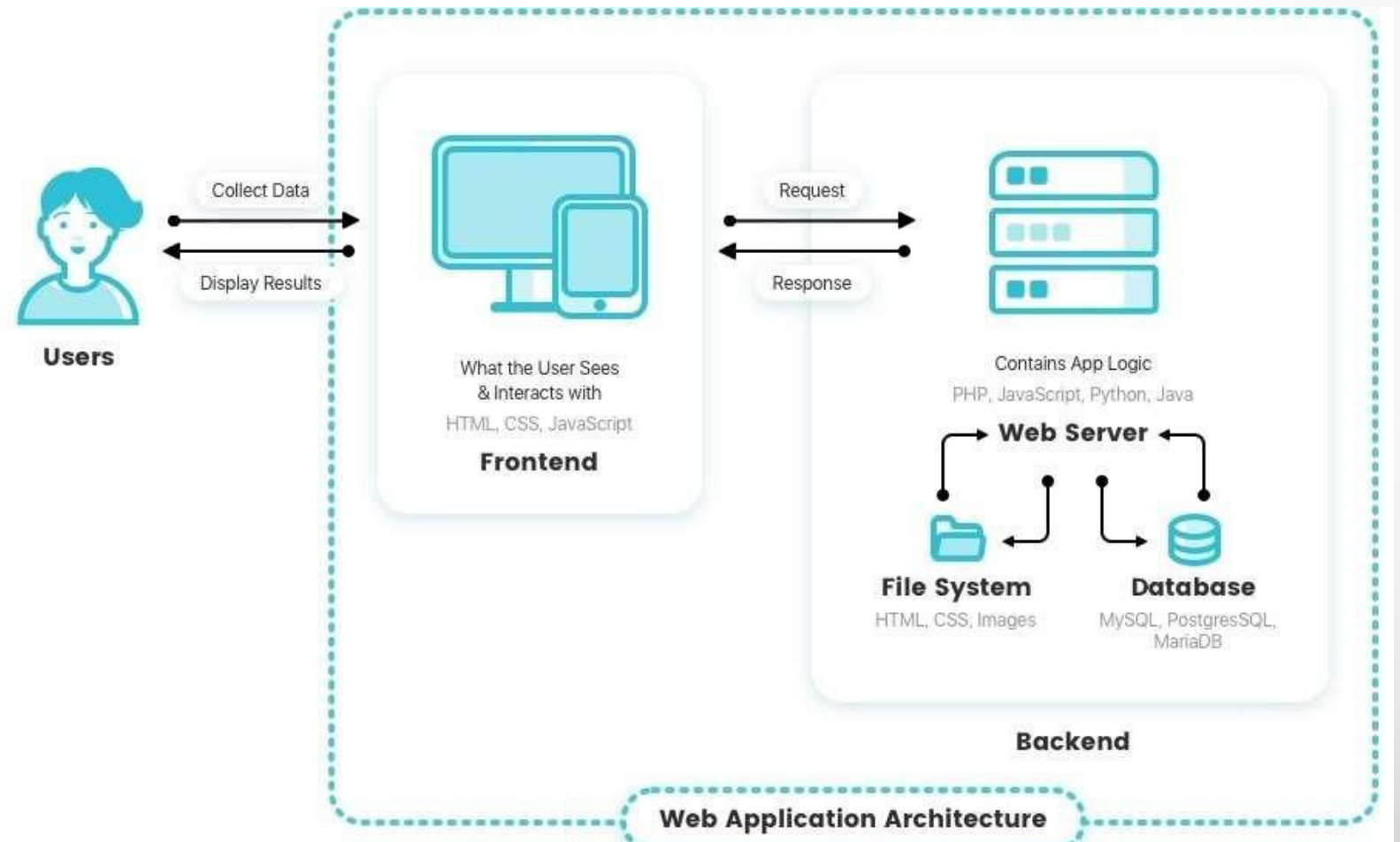
- Markup and web languages such as HTML, CSS and Javascript
- Asynchronous requests and Ajax
- Specialized web editing software
- Image editing
- Accessibility
- Cross-browser issues
- Search engine optimisation



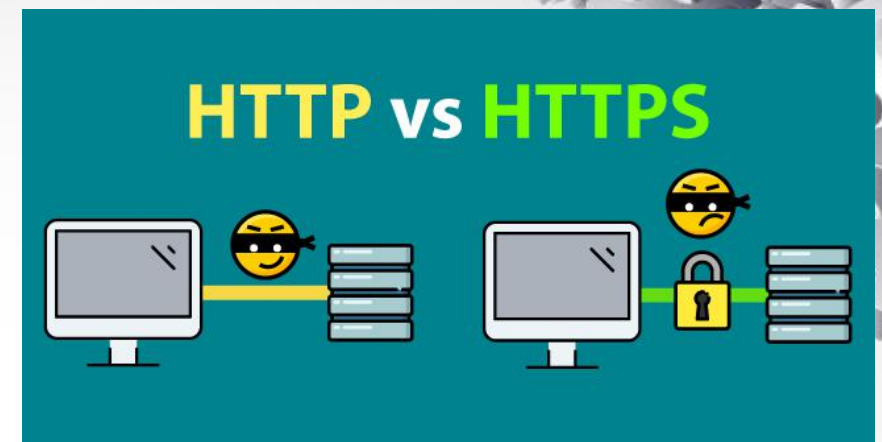
Back End

- Programming and scripting such as Python, Ruby and/or Perl
- Server architecture
- Database administration
- Scalability
- Security
- Data transformation
- Backup

Web Page Access:

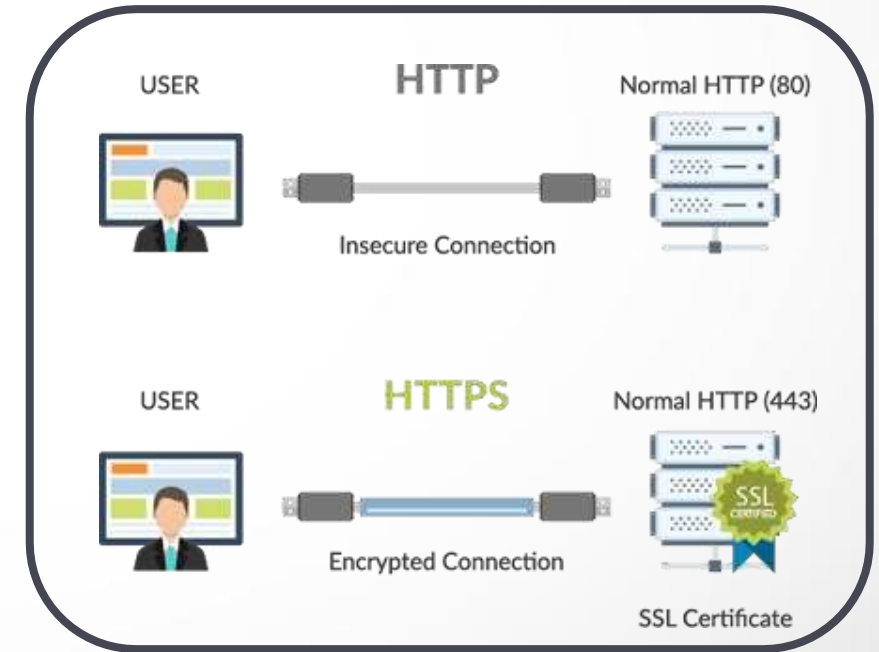
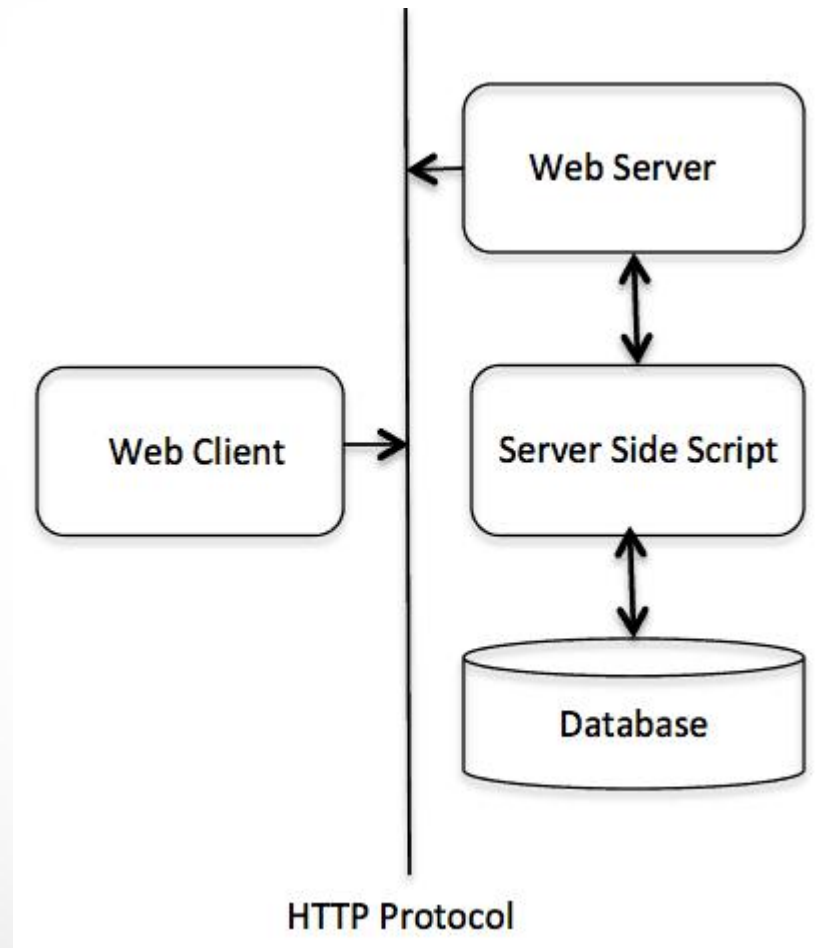


HTTP:

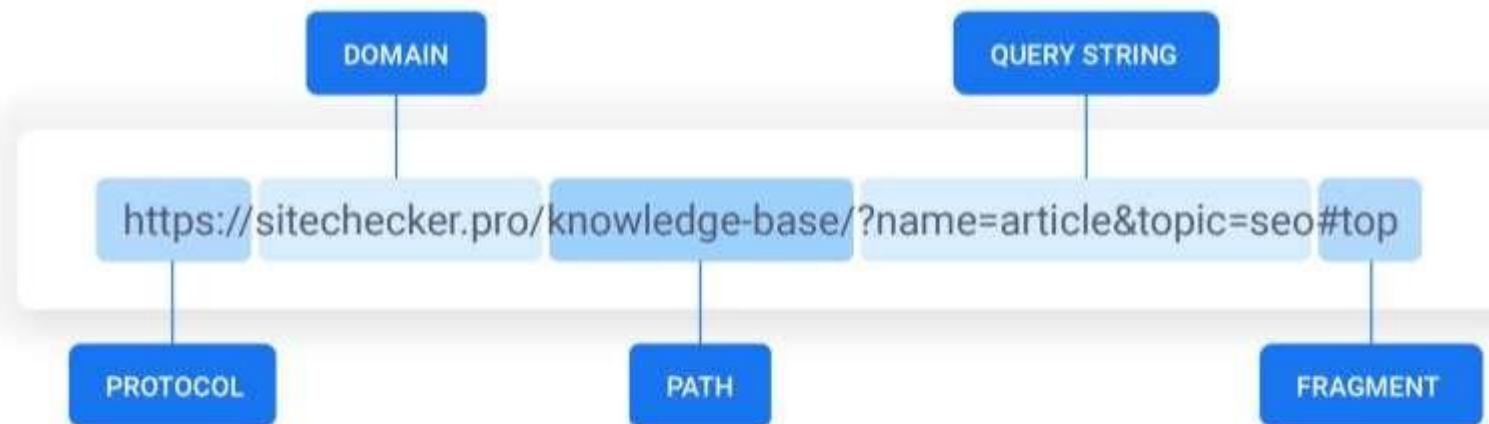


- ❖ HTTP is an application-level protocol for distributed, collaborative, hypermedia information systems.
- ❖ HTTP is a request/response standard of a client and a server.
- ❖ Typically, an HTTP client initiates a request.
- ❖ Resources to be accessed by HTTP are identified using Uniform Resource Identifiers (URIs).
- ❖ This tutorial is based on RFC-2616 specification, which defines the protocol referred to as **HTTP/1.1**.
- ❖ **HTTP/1.0** uses a new connection for each request/response exchange, where as **HTTP/1.1** connection may be used for one or more request/response exchanges.
- ❖ Port Id of HTTP is **80**.

HTTP Architecture:



URI:



HTTP Responses:



S.N.	Code and Description
1	1xx: Informational It means the request was received and the process is continuing.
2	2xx: Success It means the action was successfully received, understood, and accepted.
3	3xx: Redirection It means further action must be taken in order to complete the request.
4	4xx: Client Error It means the request contains incorrect syntax or cannot be fulfilled.
5	5xx: Server Error It means the server failed to fulfill an apparently valid request.

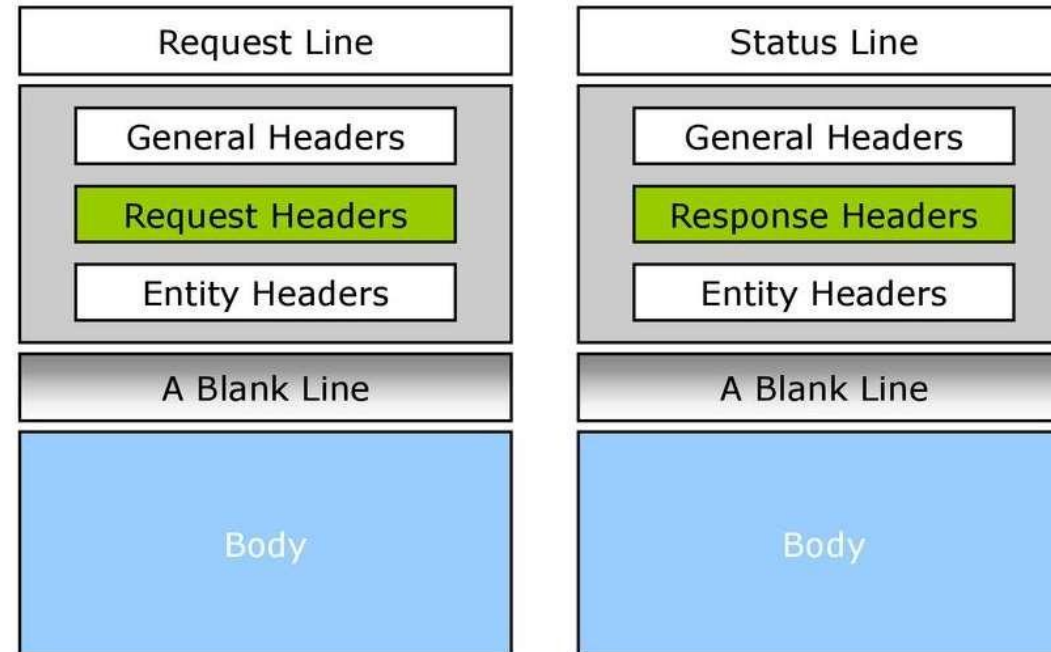


HTTP Methods:

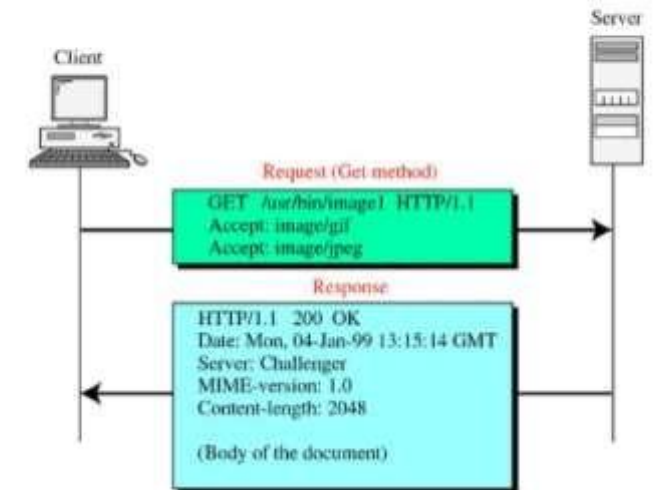
S.N.	Code and Description
1	GET : The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data.
2	HEAD : Same as GET, but transfers the status line and header section only.
3	POST : A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms.
4	PUT : Replaces all current representations of the target resource with the uploaded content.
5	DELETE : Removes all current representations of the target resource given by a URI.
6	CONNECT : Establishes a tunnel to the server identified by a given URI.
7	OPTIONS : Describes the communication options for the target resource.
8	TRACE : Performs a message loop-back test along the path to the target resource.

HTTP Header:

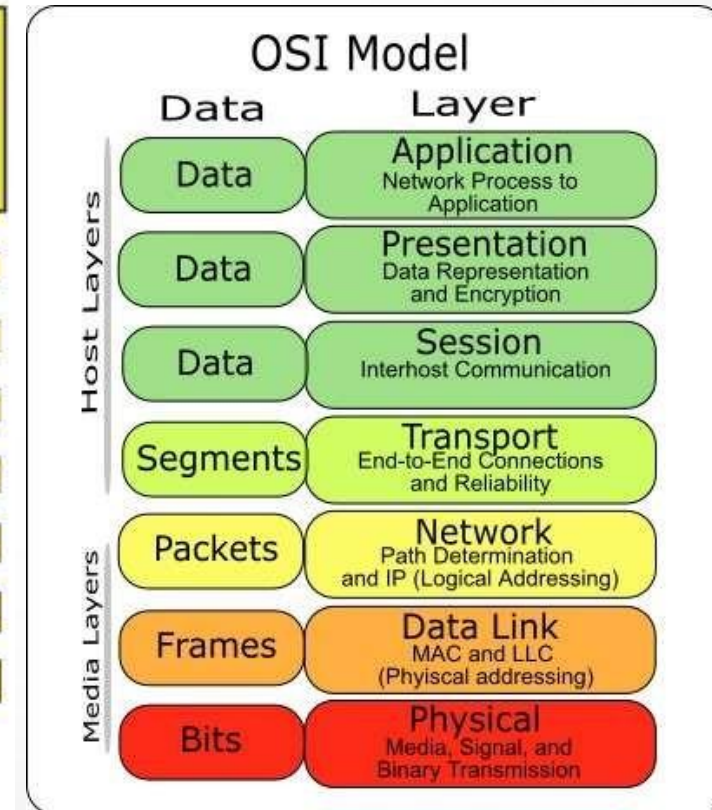
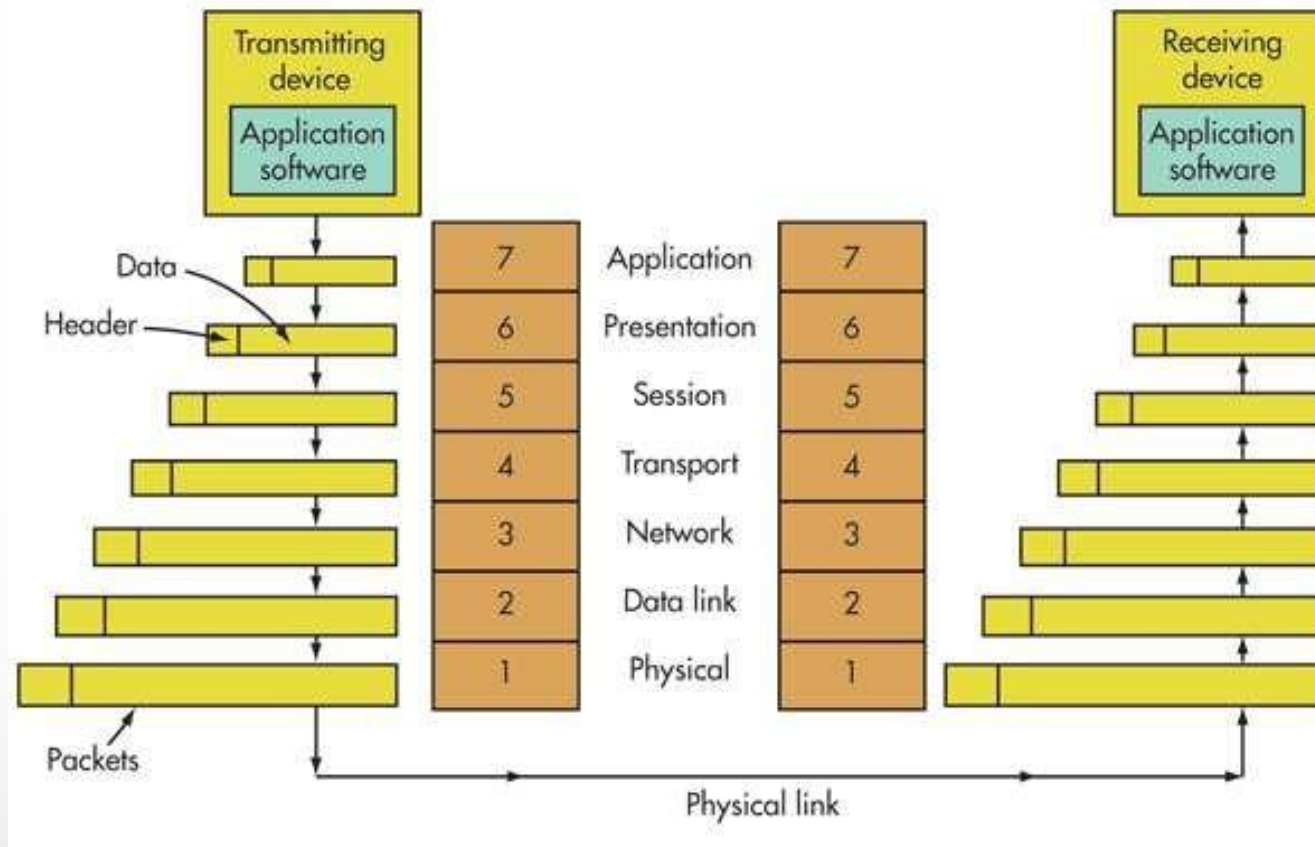
Headers



Example Of Request/Response

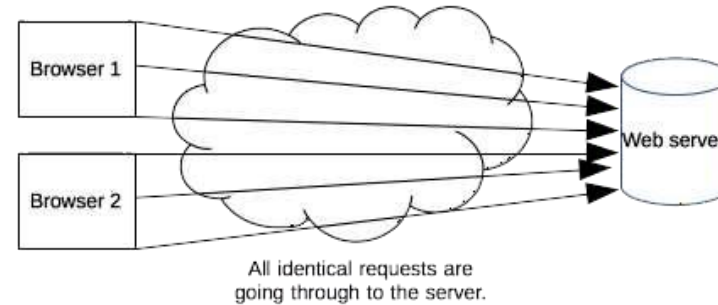


OSI Layer:

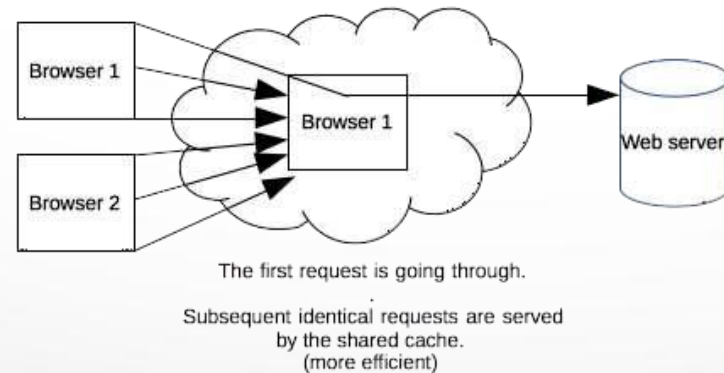


HTTP Cache:

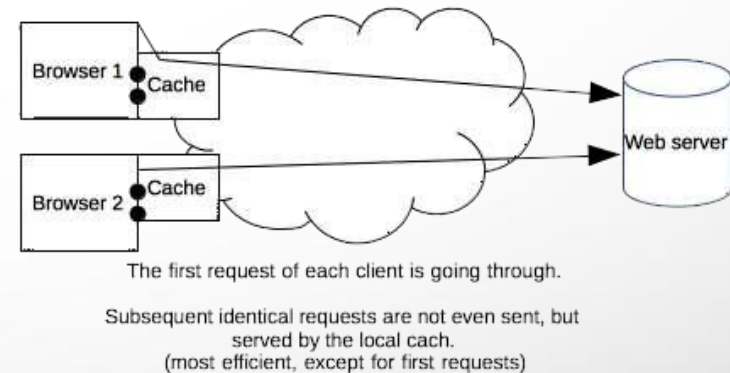
No cache



Shared cache



Local (private) cache



Web Caching:



More about Web caching

- ❑ cache acts as both client and server
- ❑ typically cache is installed by ISP (university, company, residential ISP)

Why Web caching?

- ❑ reduce response time for client request
- ❑ reduce traffic on an institution's access link.
- ❑ Internet dense with caches: enables "poor" content providers to effectively deliver content (but so does P2P file sharing)



HTTP Connection:

Nonpersistent HTTP

- At most one object is sent over a TCP connection.
- HTTP/1.0 uses nonpersistent HTTP

Persistent HTTP

- Multiple objects can be sent over single TCP connection between client and server.
- HTTP/1.1 uses persistent connections in default mode

Web Technology S2 2021/22

HTML and CSS (Revision)





HTML (Hyper Text Markup Language) :



<html>

Start with fresh.

HTML:

- ❖ HTML stands for Hyper Text Markup Language.
- ❖ HTML was released in 1993
- ❖ HTML is the standard markup language for creating Web pages.
 - ❖ HTML consists of a series of elements, which inform the browser how to display the content .
 - ❖ The latest version of HTML 5 was published in 2012. World Wide Web Consortium (W3C) defines the specifications of HTML..
- ❖ HTML 5.2 was released on 27 May 2019.



HTML Tag List:

Tag	Description
<html> ... </html>	Declares the Web page to be written in HTML
<head> ... </head>	Delimits the page's head
<title> ... </title>	Defines the title (not displayed on the page)
<body> ... </body>	Delimits the page's body
<h <i>n</i> > ... </h <i>n</i> >	Delimits a level <i>n</i> heading
 ... 	Set ... in boldface
<i> ... </i>	Set ... in italics
<center> ... </center>	Center ... on the page horizontally
 ... 	Brackets an unordered (bulleted) list
 ... 	Brackets a numbered list
 ... 	Brackets an item in an ordered or numbered list
 	Forces a line break here
<p>	Starts a paragraph
<hr>	Inserts a horizontal rule
	Displays an image here
 ... 	Defines a hyperlink



HTML Table:

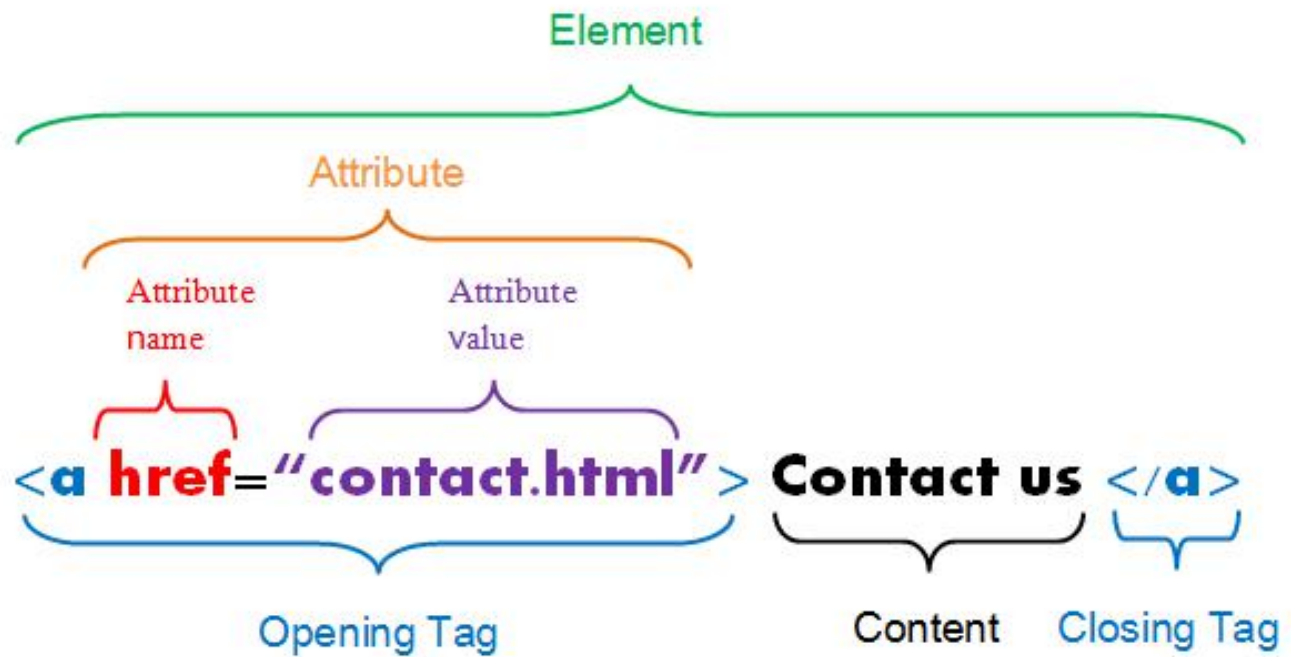
- ❖ **<table>** element to define a table
- ❖ **<tr>** element to define a table row
- ❖ **<td>** element to define a table data
- ❖ **<th>** element to define a table heading
- ❖ **<caption>** element to define a table caption
- ❖ **border** property to define a border
- ❖ **border-collapse** property to collapse cell borders
- ❖ **text-align** property to align cell text
- ❖ **border-spacing** property to set the spacing between cells
- ❖ Use the **colspan** attribute to make a cell span many columns
- ❖ Use the **rowspan** attribute to make a cell span many rows



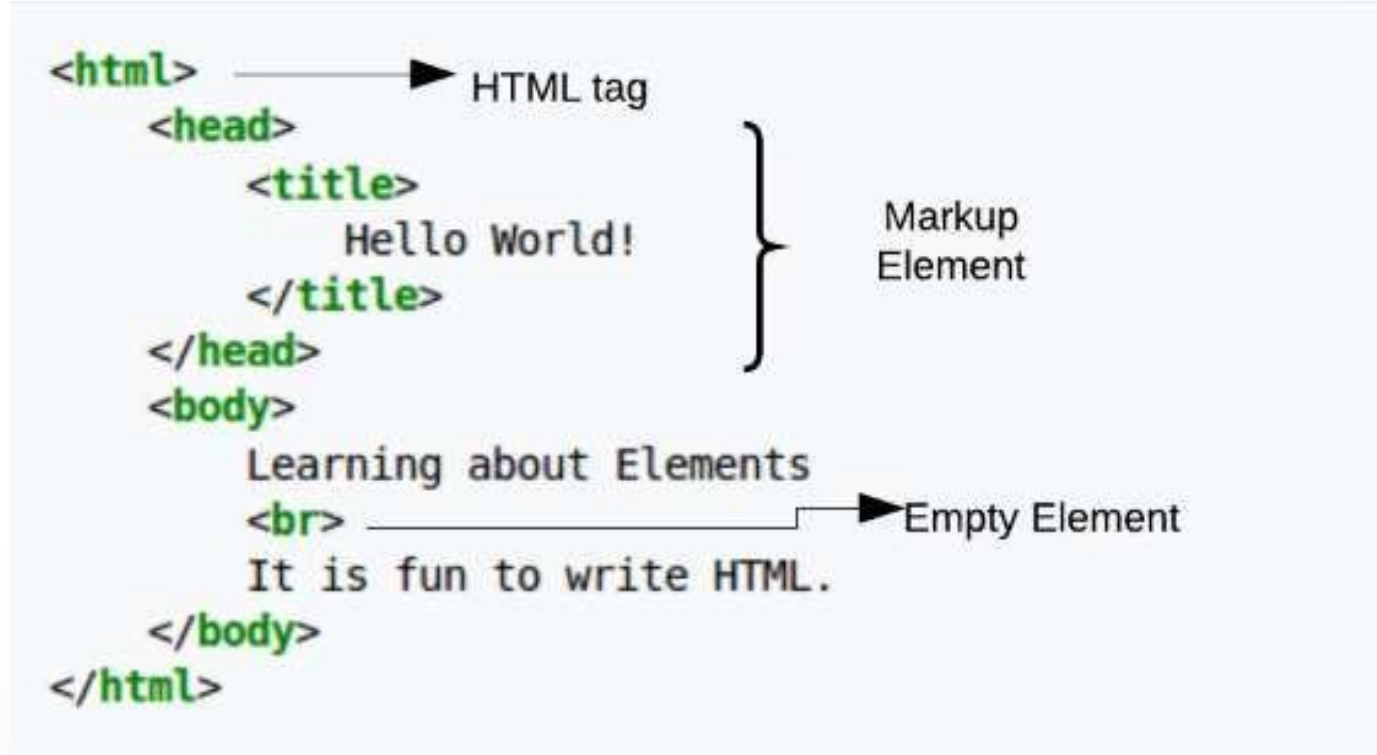
HTML Form:


TAG	Definition
<form>	Defines an HTML form for user input
<input>	Defines an input control
<textarea>	Defines a multiline input control (text area)
<label>	Defines a label for an <input> element
<fieldset>	Groups related elements in a form
<legend>	Defines a caption for a <fieldset> element
<select>	Defines a drop-down list
<optgroup>	Defines a group of related options in a drop-down list
<option>	Defines an option in a drop-down list
<button>	Defines a clickable button
<datalist>	Specifies a list of pre-defined options for input controls

HTML Tag Details:



HTML Document Structure :





HTML Tags:

<h1>Heading 1</h1>

<h2>Heading 2</h2>

<h3>Heading 3</h3>

<h4>Heading 4</h4>

<h5>Heading 5</h5>

<h6>Heading 6</h6>

<!--Comment-->

<!--

HTML code or content

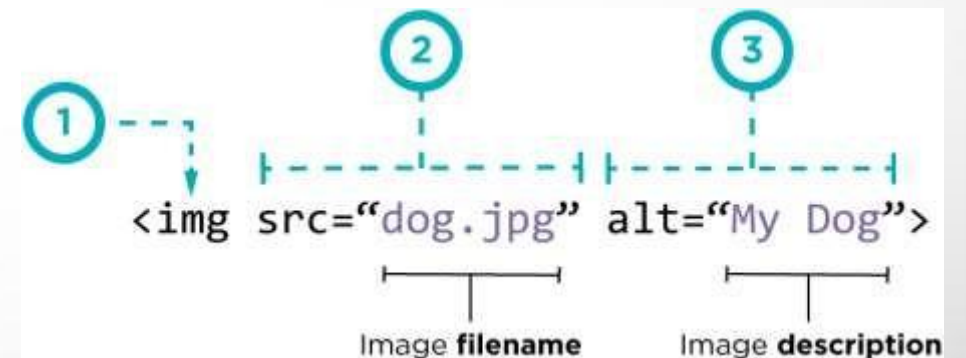
-->

Comment Start Code or content Comment End

HTML Tags (Cont.):

```
<html>
  <head>
    <title> Video Test </title>
  </head>
  <body>
    <center>
      <video src="VideoTest.mp4" controls autoplay>
        This browser does not support HTML5 video.
      </video>
    </center>
  </body>
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
    .
  </head>
  <body>
    <img src="" width="" height="" alt="">
  </body>
</html>
```



HTML Attributes:

Attribute	Value	Description
align	Right, left, centre	Horizontally aligns tags
bgcolor	Numeric, hexadecimal, RGB values	Places a background colour behind an element
background	URL	Places a background image behind an element
class	User defined	Classifies an element for use with Cascading Style Sheets
height	Numeric value	Specifies the height of tables, images or table cells

Attribute	Value	Description
id	User defined	Names an element for use with Cascading Style Sheets
title	User defined	Pop-up title of the elements
valign	Top, middle, bottom	Vertically aligns tags within an HTML element
width	Numeric value	Specifies the width of tables, images or table cells

HTML Character Entities & Formatting:

Character	Character Entity	Description
	 	Non-breaking space
<	<	Less than
>	>	Greater than
&	&	Ampersand
"	"	Quotation mark
'	'	Apostrophe

Tag	Description
	Bold text
<big>	Larger text
	Deleted text
<div>	Grouping content
<i>	Italic text
<ins>	Inserted text
<pre>	Preformatted text
<small>	Smaller text
	Grouping content

Tag	Description
<strike>	Strike text
<sub>	Subscript text
<sup>	Superscript text
<tt>	Teletype (monospaced) text
<u>	Underlined text

CSS:



Håkon Wium Lie

- CSS stands for Cascaded Style Sheet.
- Håkon Wium Lie has proposed the concept of CSS in 1994
- The latest version of CSS 3 was published in 1999. World Wide Web
- Consortium (W3C) defines the specifications of CSS.

CSS Syntax:

Selector

h1

Declaration

{color:blue; font-size:12px;}

Declaration

Property

Value

Property

Value

Types of CSS:

1) External CSS

2) Internal CSS

3) Inline CSS





External CSS:

- ❖ With an external style sheet, you can change the look of an entire website by changing just one file.
- ❖ Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

```
body {
  background-color: lightblue;
}

h1 {
  color: navy;
  margin-left: 20px;
}
```



Internal CSS:

- ❖ An inline style may be used to apply a unique style for a single element.
- ❖ To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      body {background-color: linen;}
      h1 {color: maroon;
        margin-left: 40px; }
    </style>
  </head>
```

```
    <body>

      <h1>This is a heading</h1>
      <p>This is a paragraph.</p>

    </body>

  </html>
```

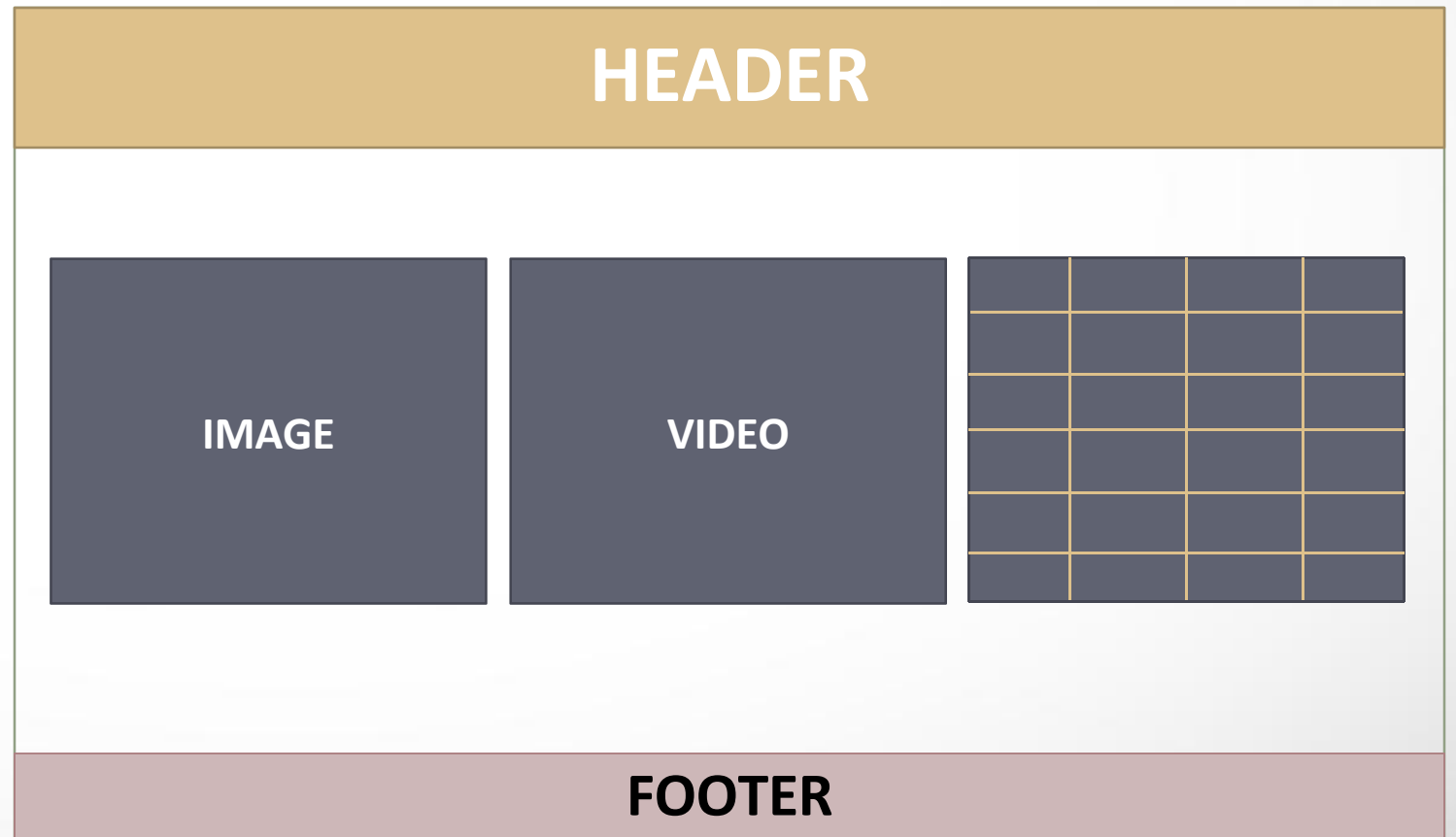


Inline CSS:

- ❖ An inline style may be used to apply a unique style for a single element.
- ❖ To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

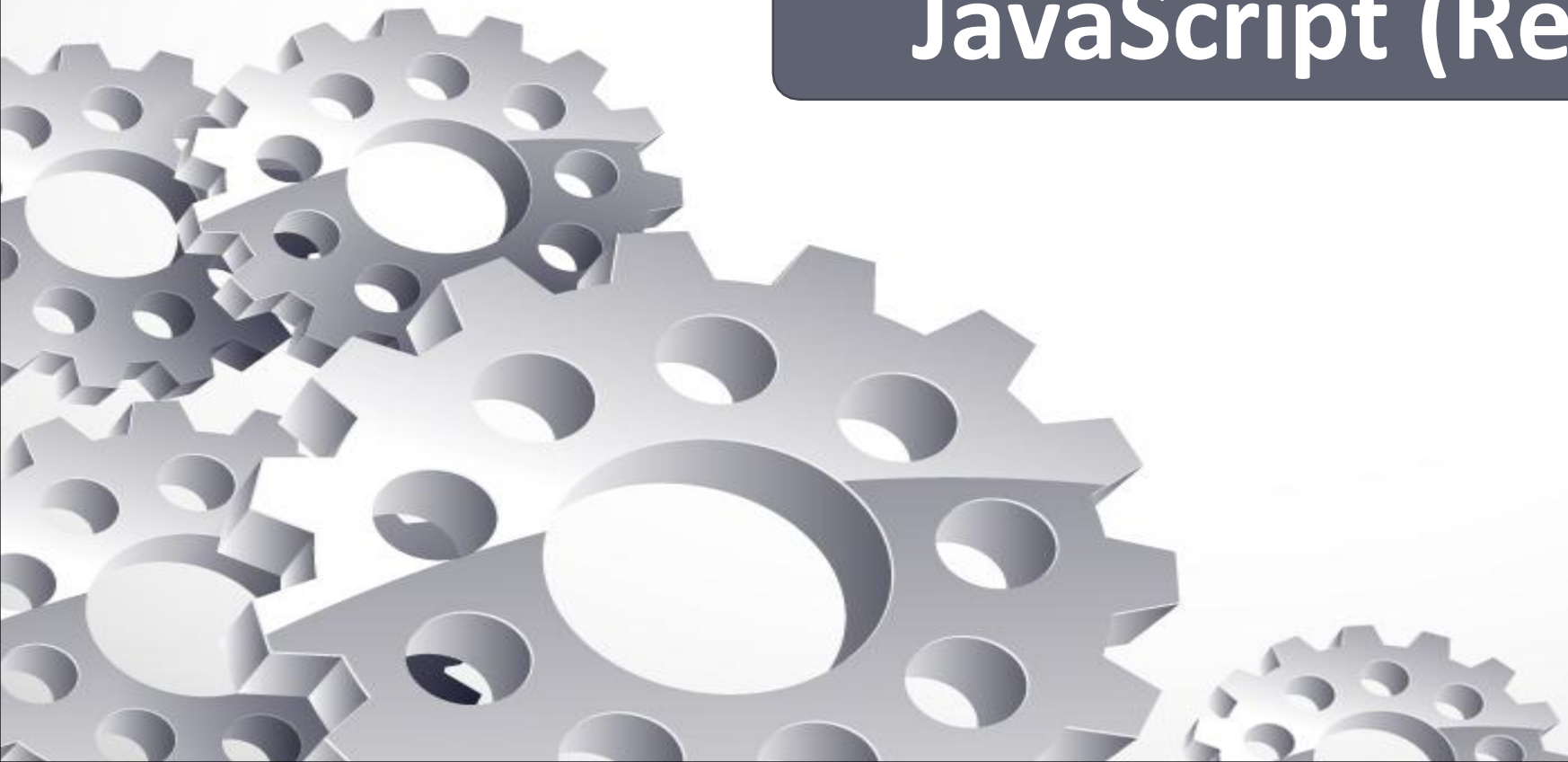
```
<!DOCTYPE html>
<html>
  <body>
    <h1 style="color:blue;text-align:center;">
      This is a heading
    </h1>
    <p style="color:red;">
      This is a paragraph.
    </p>
  </body>
</html>
```

Home Work



Web Technology S2 2021/22

JavaScript (Revision)



JavaScript:



Brendan Eich

- ❖ **JavaScript** enables interactive web pages and is an essential part of web applications.
- ❖ First released in **December 4, 1995**
- ❖ **JavaScript** is the dominant client-side scripting language of the Web, with **95%** of websites using it for this purpose.
- ❖ All major web browsers have a built-in JavaScript engine that executes the code on the user's device.
- ❖ "**JavaScript**" is a trademark of Oracle Corporation in the United States.



JavaScript Syntax:

```
var x, y, z;  
x = 5; y = 6;  
z = x + y;
```

```
// How to declare variables  
// How to assign values  
// How to compute values
```

```
<!DOCTYPE html>  
<html>  
  <body>  
  
    <h2>JavaScript Strings</h2>  
  
    <p>Strings can be written with double or single quotes.</p>  
  
    <p id="demo"></p>  
  
    <script>  
      document.getElementById("demo").innerHTML = 'John Doe';  
    </script>  
  
  </body>  
</html>
```



JavaScript Function:

```
<script>
function myFunction()
{ document.getElementById("demo").innerHTML =
"Paragraph changed.";
}
function myFunction1()
{ document.getElementById("demo").innerHTML =
"Paragraph changed1.";
}
</script>
```

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript in Body</h2>

<p id="demo">A Paragraph.</p>

<button type="button" onclick="myFunction()">Try it</button>

<button type="button" onclick="myFunction1()">Try it1</button>

</body>
</html>
```



JavaScript Random Function:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Math.random()</h2>

<p>Math.random() returns a random number between 0 (included) and 1
(excluded):</p>
<button onclick="check()"> Random </button>
<p id="demo"></p>

<script>
var a=10;
function check(){
document.getElementById("demo").innerHTML = a*Math.random();
}
</script>

</body>
</html>
```



JavaScript Array:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Arrays</h2>

<p>JavaScript array elements are accessed using numeric indexes
(starting from 0).</p>

<p id="demo"></p>

<script>
var cars = ["Saab", "Volvo", "BMW"];
document.getElementById("demo").innerHTML = cars[0];
</script>

</body>
</html>
```



JavaScript Regular Expression:

- ❖ A regular expression is a sequence of characters that forms a search pattern.
- ❖ The search pattern can be used for text search and text replace operations.

What Is a Regular Expression?

- ❖ A regular expression is a sequence of characters that forms a search pattern.
- ❖ When you search for data in a text, you can use this search pattern to describe what you are searching for.
- ❖ A regular expression can be a single character, or a more complicated pattern.
- ❖ Regular expressions can be used to perform all types of text search and text replace operations.

JavaScript replace() & search():



replace()

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Regular Expressions</h2>

<p>Replace "microsoft" with "W3Schools" in the paragraph below:</p>

<button onclick="myFunction()">Try it</button>

<p id="demo">Please visit Microsoft and Microsoft!</p>

<script>
function myFunction() {
  var str = document.getElementById("demo").innerHTML;
  var txt = str.replace(/microsoft/i,"W3Schools");
  document.getElementById("demo").innerHTML = txt;
}
</script>

</body>
</html>
```

search()

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Regular Expressions</h2>

<p>Search a string for "w3Schools", and display the position of the
match:</p>

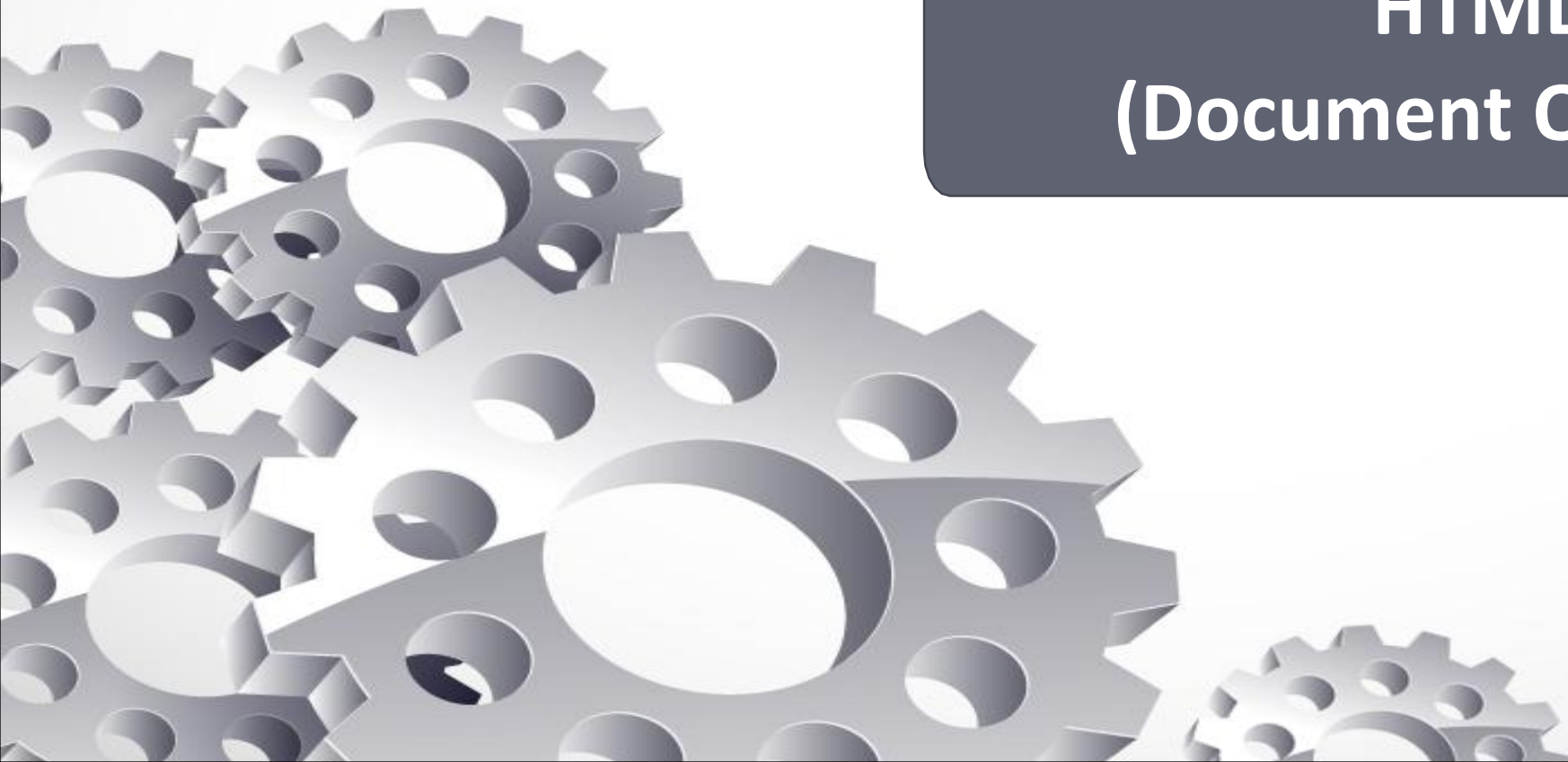
<p id="demo"></p>

<script>
var str = "Visit W3Schools!";
var n = str.search(/w3Schools/i);
document.getElementById("demo").innerHTML = n;
</script>

</body>
</html>
```


Web Technology S2 2021/22

HTML DOM
(Document Object Model)

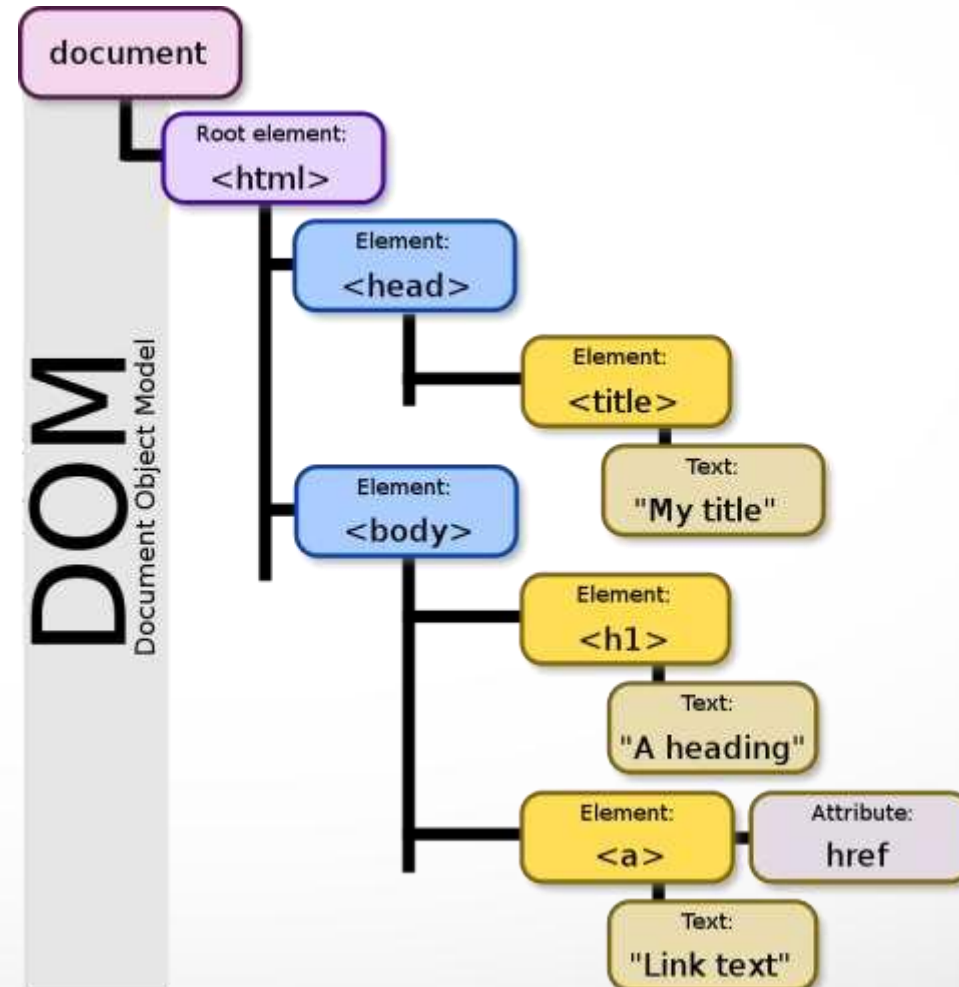




HTML DOM:

- ❖ The Document Object Model (DOM) is a ***programming interface*** for **HTML** and **XML**(Extensible markup language) documents.
- ❖ It defines the **logical structure** of documents and the way a document is accessed and manipulated.
- ❖ It is called as a Logical structure because DOM doesn't specify any relationship between objects.

HTML DOM:





HTML DOM:

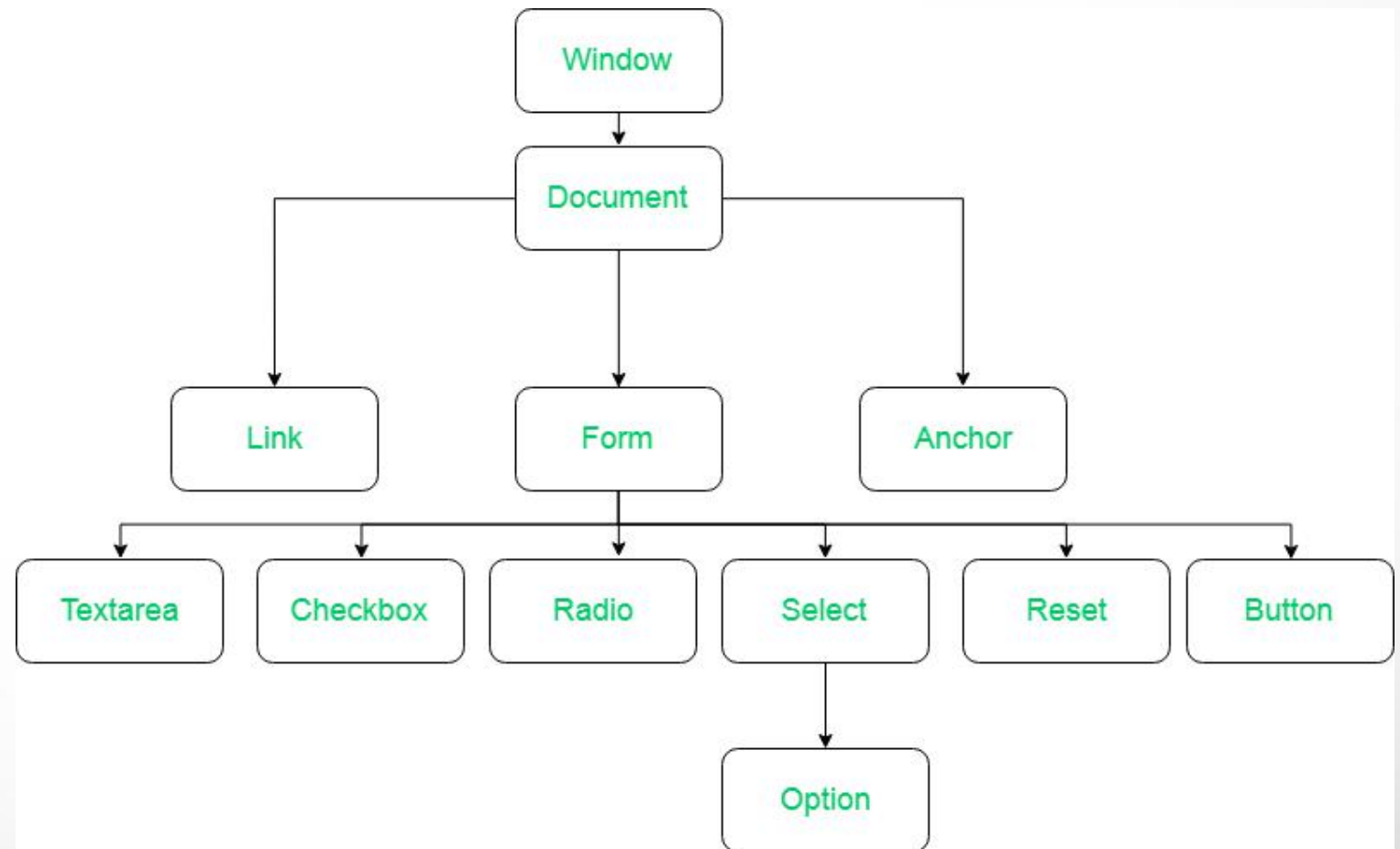
- ❖ It is a World Wide Web Consortium (W3C) standard, used to access documents such as HTML and XML.
- ❖ It is a programming API.
- ❖ It defines the objects, properties and methods for accessing documents. It has three parts:

1. Core DOM

2. XML DOM

3. HTML DOM.

HTML DOM:





HTML DOM:

1. Window Object:

Window Object is always at top of hierarchy.

2. Document object:

When HTML document is loaded into a window, it becomes a document object.

3. **Form Object:** It is represented by ***form*** tags.

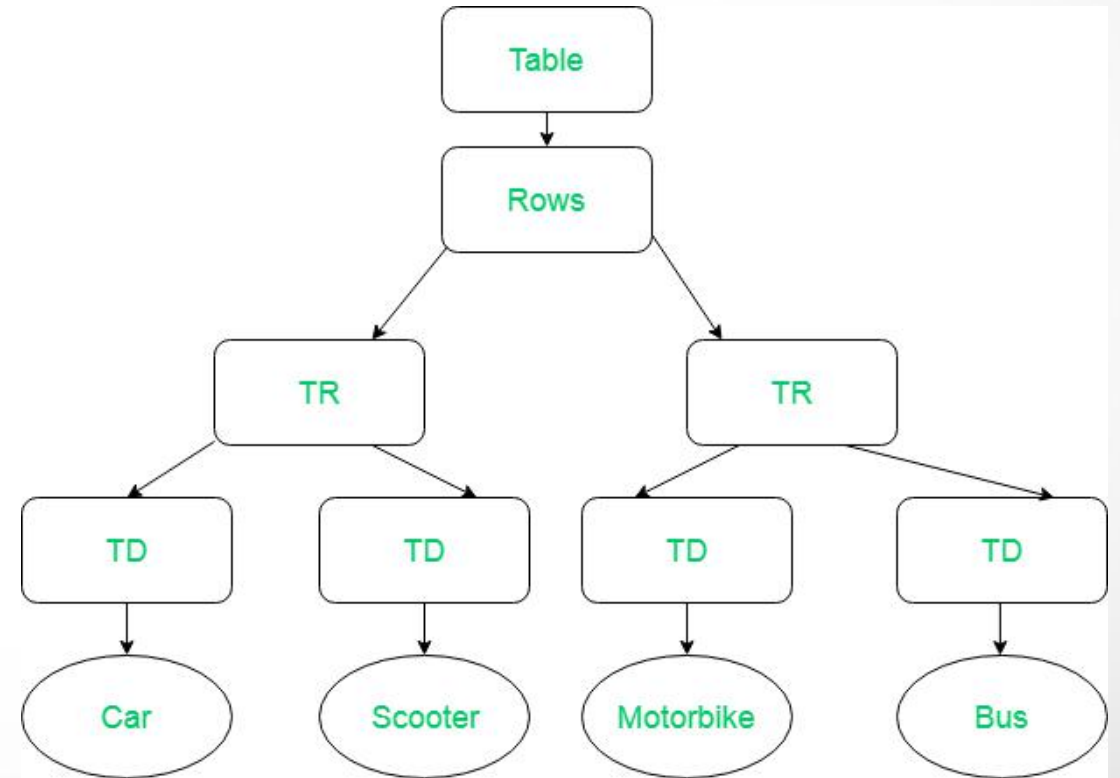
4. **Link Objects:** It is represented by ***link*** tags.

5. **Anchor Objects:** It is represented by ***a href*** tags.

6. **Form Control Elements::** Form can have many control elements such as text fields, buttons, radio buttons, and checkboxes, etc.

HTML TREE DOM:

```
<Table>  
<ROWS>  
  <TR>  
    <TD>Car</TD>  
    <TD>Scooter</TD>  
  </TR>  
  <TR>  
    <TD>MotorBike</TD>  
    <TD>Bus</TD>  
  </TR>  
</ROWS>  
</Table>
```



Web Technology S2 2021/22

BOM
(Browser Object Model)





BROWSER OBJECT MODEL:

- ❖ The **Browser Object Model** (BOM) is used to interact with the browser.
- ❖ The default object of browser is window means you can call all the functions of window by specifying window or directly.
- ❖ **window.alert("hello javatpoint");** is same as **alert("hello javatpoint");** .

BROWSER OBJECT MODEL:

navigator object

window object

frame object

frame object

document object

<p> paragraph object

<h2> heading object

The Document Object Model

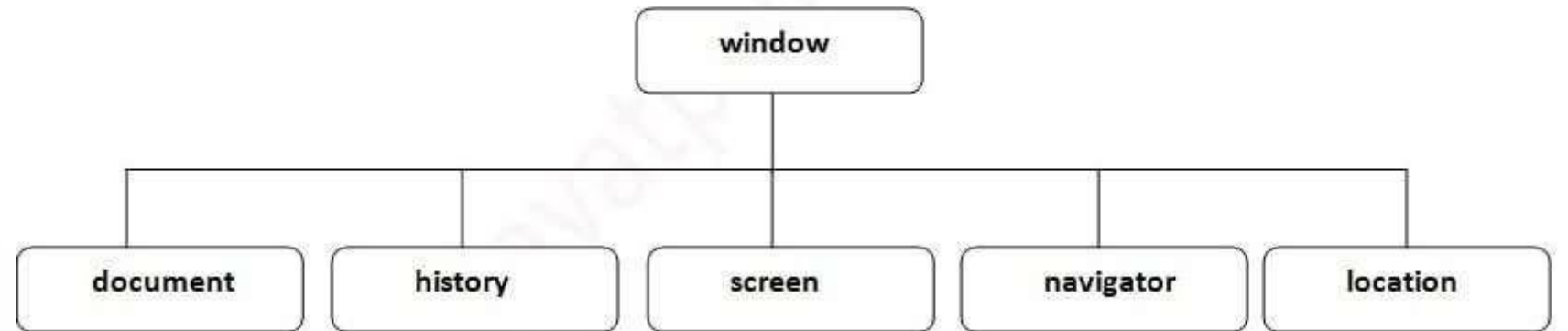
A Web page applying JavaScript processing to the XHTML elements on that page. We have considered XHTML tags simply as markup codes providing structure to page content and supplying mechanisms through which styling is applied to that content. Importantly, though, XHTML tags are also **software objects**. That is, all XHTML tags have **properties** and **methods** that can be programmed. As is the case with all software objects, properties refer to characteristics of the element; methods refer to actions the object can perform. XHTML tags, then, are programmable through JavaScript processing routines that set their properties and activate their methods in order to make Web pages dynamic.

The programming interface to the DOM is the **Document Object Model (DOM)**. The DOM is a hierarchy of browser components, and it provides the means for identifying and manipulating these components to produce dynamic changes.

The DOM Hierarchy

Basically, the DOM is a hierarchy of browser components. At the top-most level is the browser (**navigator**) object. At the next level down the hierarchy is the **window** object, the main browser window within which Web pages appear. Within the window are optional **frame** objects (if the window is divided into frames), and these window and frame objects contain the **document** objects representing Web pages. The page itself contains other objects, including

BOM Hierarchy:



BOM:

1. The top level object in the BOM is the window object. The **window object** represents the browser window. All other browser objects are contained within the window object. The window object includes a number of properties and methods that can be used to control the Web browser. The window object along with its properties and methods are discussed in more detail in a later section.
2. The **document object** represents the Web page displayed in the browser. All elements on a Web page including HTML tags are contained within the document object. Since the document object is often considered the most important part of the BOM, it is represented by its own object model called the **Document Object Model or DOM**. The DOM will be discussed in more detailed in later tutorials.
3. Other objects of the browser object model include the **navigator object**, the **screen object**, that contains information about the visitor's screen, the **history object**, that is part of the window object and contains the URLs that have been visited by the user, and the **location object** that contains information about the current URL. Within the window object are **document objects** representing elements within the Web pages.



BOM Objects:

Reference	Object
window	The main browser window
window.navigator	Information about the browser itself
window.screen	The user's screen
window.history	URLs visited by a user
window.location	The current URL
window.document (document)	The document appearing in the main browser window
document.getElementById("id")	An HTML element appearing in a document and identified by its assigned <i>id</i> value.



Window Object:

The **window object** represents a window in browser. An object of window is created automatically by the browser.

Window is the object of browser, **it is not the object of javascript**. The javascript objects are string, array, date etc.

```
<script type="text/javascript">
function
msg(){ alert("Hello
Alert Box");
}
</script>
<input type="button" value="click" onclick="msg()"/>
```

```
<script type="text/javascript">
function msg(){
var v= prompt("Who are you?");
alert("I am "+v);

}
</script>

<input type="button" value="click" onclick="msg()"
/>
```



Window Object:

```
<script type="text/javascript">  
function msg(){  
var v= prompt("Who are you?");  
alert("I am "+v);  
}  
</script>
```

```
<input  
type="button" value="click"  
onclick="msg()"/>
```

```
1.<script type="text/javascript">  
2.function msg(){  
3.var v= confirm("Are u sure?");  
4.if(v==true){  
5.alert("ok");  
6.}  
7.else  
8.{  
9.alert("cancel"); }  
9.}  
10.<input type="button" value="delete record" onclick="msg()"/>
```



Navigator Object:

The **navigator object** is used for browser detection. It can be used to get browser information such as appName, appCodeName, userAgent etc.

```
<html>
<body>
<h2>JavaScript Navigator Object</h2>
<script>
document.writeln("<br/>navigator.appCodeName: "+navigator.appCodeName);
document.writeln("<br/>navigator.appName: "+navigator.appName);
document.writeln("<br/>navigator.appVersion: "+navigator.appVersion);
document.writeln("<br/>navigator.cookieEnabled: "+navigator.cookieEnabled);
document.writeln("<br/>navigator.language: "+navigator.language);
document.writeln("<br/>navigator.userAgent: "+navigator.userAgent);
document.writeln("<br/>navigator.platform: "+navigator.platform);
document.writeln("<br/>navigator.onLine: "+navigator.onLine);

</script>
</body>
</html>
```



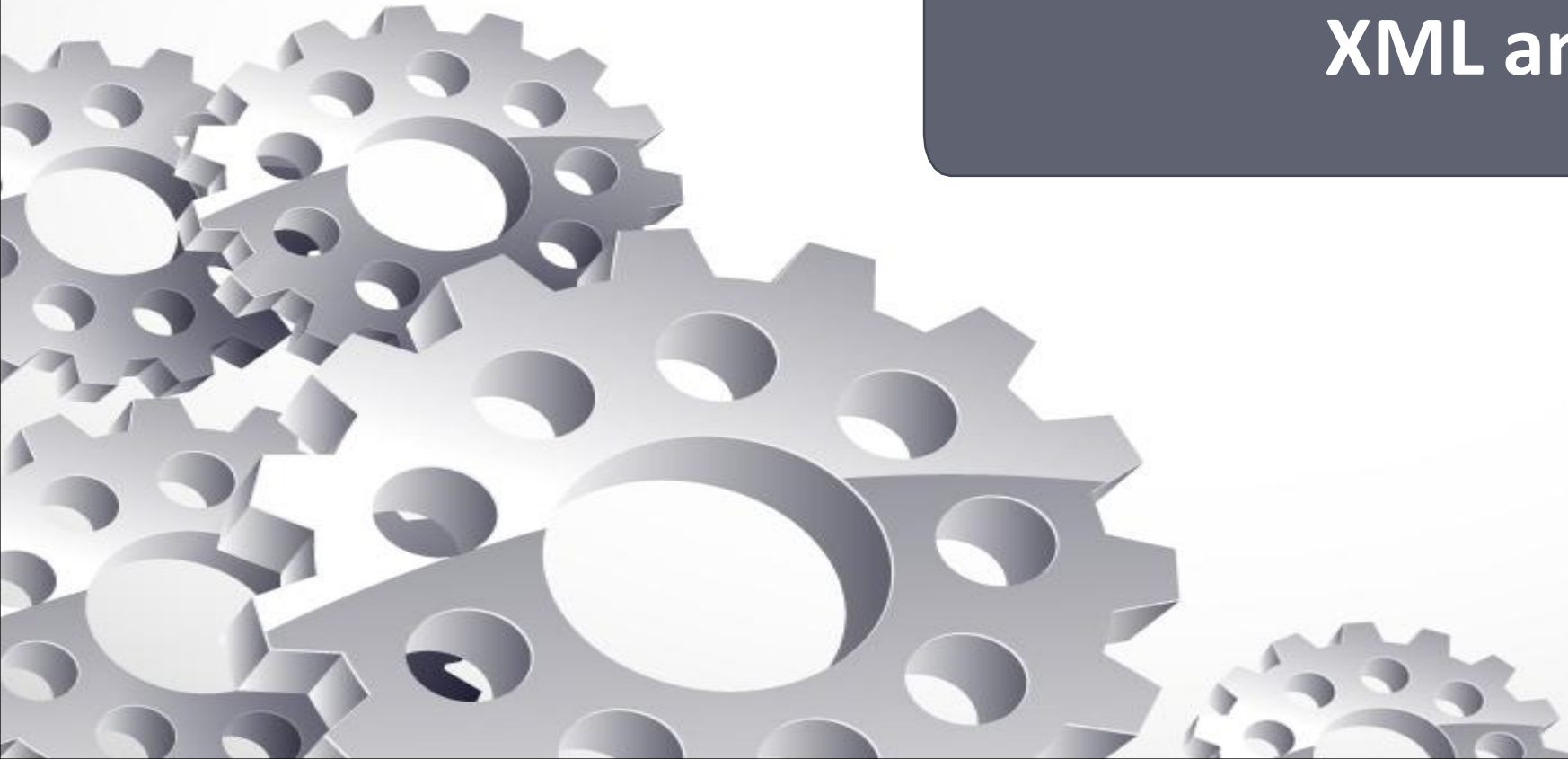
Screen Object :

The **JavaScript screen object** holds information of browser screen. It can be used to display screen width, height, colorDepth, pixelDepth etc.

```
<html>
<body>
<script>
document.writeln("<br/>screen.width: "+screen.width);
document.writeln("<br/>screen.height: "+screen.height);
document.writeln("<br/>screen.availWidth: "+screen.availWidth);
document.writeln("<br/>screen.availHeight: "+screen.availHeight);
document.writeln("<br/>screen.colorDepth: "+screen.colorDepth);
document.writeln("<br/>screen.pixelDepth: "+screen.pixelDepth);
</script>
</body>
</html>
```

Web Technology S2 2021/22

XML and AJAX





HTML vs XML:

No.	HTML	XML
1	HTML is used to display data and focuses on how data looks.	XML is a software and hardware independent tool used to transport and store data. It focuses on what data is.
2	HTML is a markup language itself.	XML provides a framework to define markup languages.
3	HTML is not case sensitive.	XML is case sensitive.
4	HTML is a presentation language.	XML is neither a presentation language nor a programming language.
5	HTML has its own predefined tags.	You can define tags according to your need.
6	In HTML, it is not necessary to use a closing tag.	XML makes it mandatory to use a closing tag.
7	HTML is static because it is used to display data.	XML is dynamic because it is used to transport data.
8	HTML does not preserve whitespaces.	XML preserve whitespaces.



XML:

- ❖ **XML** (eXtensible Markup Language) is a mark up language.
- ❖ XML is designed to store and transport data.
- ❖ Xml was released in late 90's. it was created to provide an easy to use and store self describing data.
- ❖ XML became a **W3C** Recommendation on **February 10, 1998**.
- ❖ XML is not a replacement for HTML.
- ❖ XML is designed to be self-descriptive.
- ❖ XML is designed to carry data, not to display data.
- ❖ XML tags are not predefined. You must define your own tags.
- ❖ XML is platform independent and language independent.



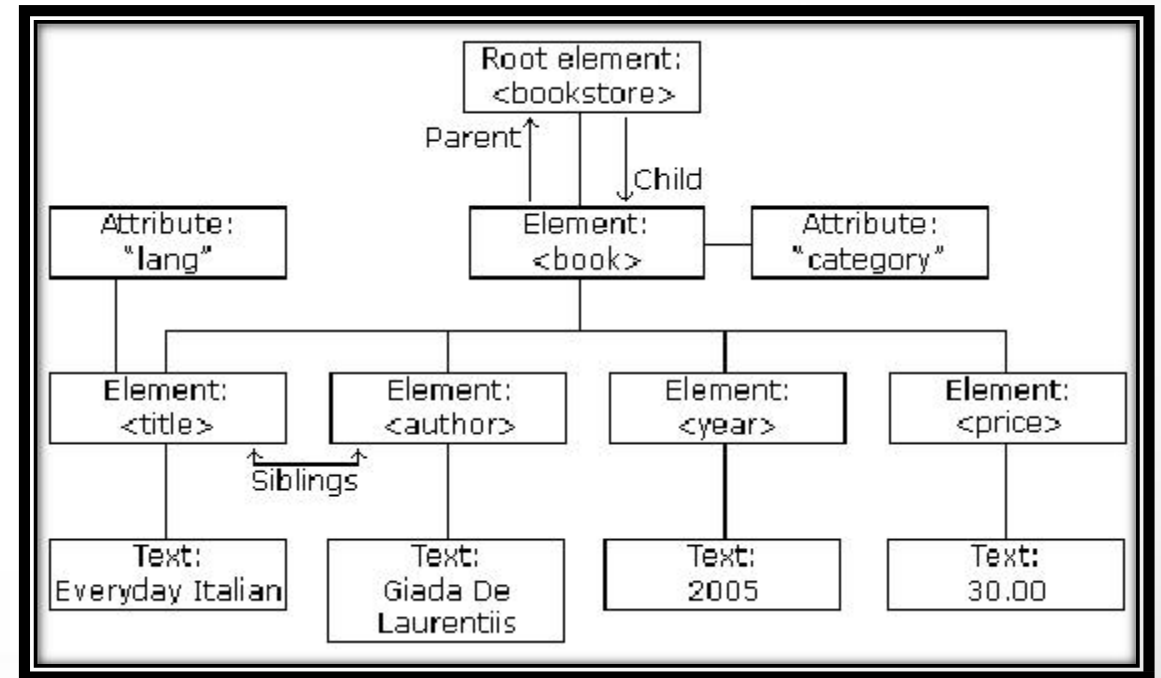
Advantage XML:

1. Defining your own elements
2. Better organized documents
3. Sorting of database

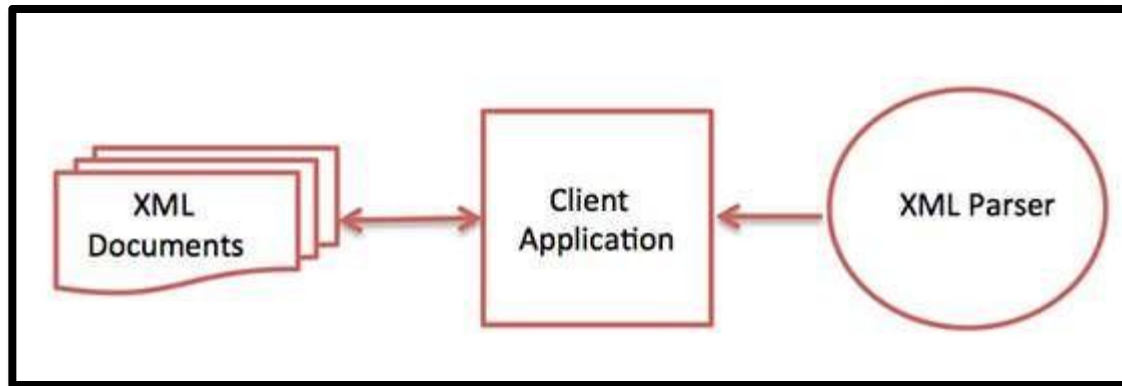
XML DOM:

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```



XML - Parsers:



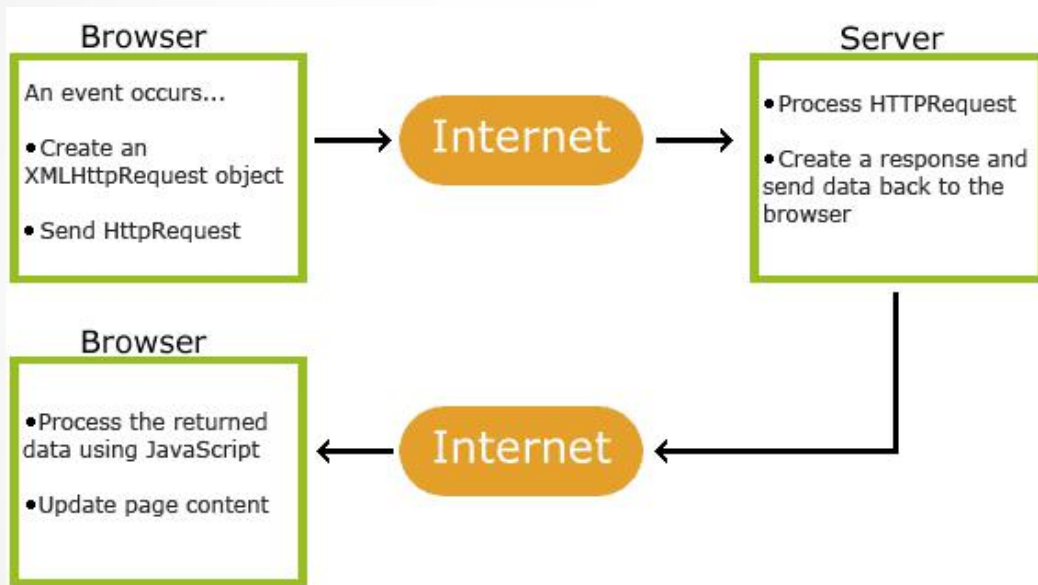
- ❖ **XML parser** is a software library or a package that provides interface for client applications to work with XML documents.
- ❖ It checks for proper format of the XML document and may also validate the XML documents.
- ❖ Modern day browsers have built-in XML parsers.



AJAX:

- ❖ Full Form of AJAX is **A**synchronous **J**avaScript **A**nd **X**ML.
- ❖ AJAX is not a programming language.
- ❖ AJAX just uses a combination of:
 - a) A browser built-in XMLHttpRequest object (to request data from a web server)
 - b) JavaScript and HTML DOM (to display or use the data)

How AJAX works ?



1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript



XMLHttpRequest Object:

```
<!DOCTYPE html>
<html>
<body>

<h1>The XMLHttpRequest Object</h1>

<p id="demo">Let AJAX change this text.</p>

<button type="button"
onclick="loadDoc()">Change Content</button>
```

```
<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200)
      { document.getElementById("demo").innerHTML =
        this.responseText;
      }
  };
  xhttp.open("GET", "test.txt", true);
  xhttp.send();
}
</script>

</body>
</html>
```

Send a Request To a Server:



Method	Description
open(method, url, async)	Specifies the type of request method: the type of request: GET or POST url: the server (file) location async: true (asynchronous) or false (synchronous)
send()	Sends the request to the server (used for GET)
send(string)	Sends the request to the server (used for POST)

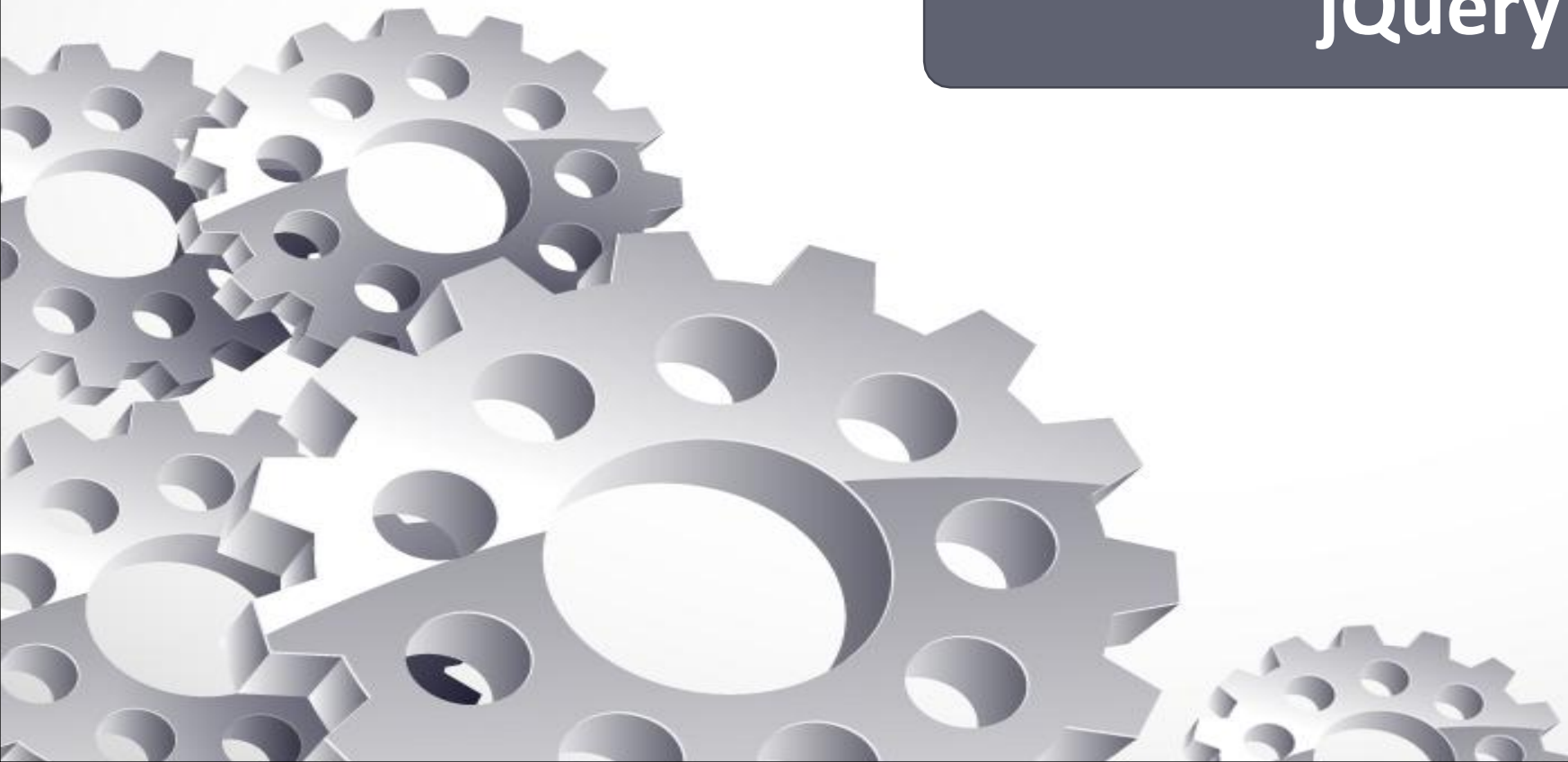


Server Response:

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 403: "Forbidden" 404: "Page not found"
statusText	Returns the status-text (e.g. "OK" or "Not Found")

Web Technology S2 2021/22

jQuery & JSON



jQuery:



- ❖ jQuery is a small and lightweight JavaScript library.
- ❖ jQuery is cross-platform.
- ❖ jQuery means "write less do more".
- ❖ jQuery simplifies AJAX call and DOM manipulation.



jQuery HTML:

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.1
1.3/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("p").html("Hello <b>Javatpoint.com</b>");
    });
});
</script>
</head>
```

```
<body>
<button>Click here to change the
content of all p elements</button>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
</body>
</html>
```



jQuery CSS:

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery
/1.11.3/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("p").css({"background-color": "yellow",
"font-size": "100%"});
    });
});
</script>
</head>
```

```
<body>
<h2>This is a heading</h2>
<p style="background-color:#ff0000">The
background-color of this paragraph is red.</p>
<p style="background-color:#00ff00">The
background-color of this paragraph is green.</p>
<p style="background-color:#0000ff">The
background-color of this paragraph is blue.</p>
<p>This paragraph has no background-color.</p>
<button>Click here to set multiple styles for all
selected elements.</button>
</body>
</html>
```

jQuery DOM:



```
<html>
<head>
  <title>The jQuery Example</title>
  <script type = "text/javascript" src =
"https://ajax.googleapis.com/ajax/libs/jquery/2.1.3
/jquery.min.js">      </script>
  <script type = "text/javascript" language
= "javascript">
    $(document).ready(function() {
      $("div").click(function () {
        $(this).before('<div class="div"></div>' );
      });    });
  </script>
```

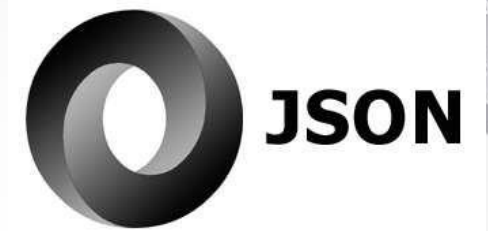
```
    <style>
      .div{ margin:10px;padding:12px;
border:2px solid #666; width:60px;}
    </style>
  </head>
  <body>
    <p>Click on any square below:</p>
    <span id = "result"> </span>
    <div class = "div" style = "background-
color:blue;"></div>
    <div class = "div" style = "background-
color:green;"></div>
    <div class = "div" style = "background-
color:red;"></div>
  </body>
</html>
```



jQuery Animate:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("div").animate({
      height: 'toggle'
    });
  });
});
</script>
</head>
<body>
<button>Start Animation</button>
<div style="background:#98bf21;height:50%;width:50%;position:absolute;"></div>
</body>
</html>
```


JSON:



- ❖ JSON: **J**ava**S**cript **O**bject **N**otation.
- ❖ JSON is a syntax for storing and exchanging data.
- ❖ JSON is text, written with JavaScript object notation.
- ❖ JSON is a lightweight data-interchange format
- ❖ JSON is "self-describing" and easy to understand



JSON Syntax:

```
<!DOCTYPE html>
<html>
<body>

<h2>Create Object from JSON String</h2>

<p id="demo"></p>

<script>
var txt = '{"name":"John", "age":30, "city":"New York"}'
var obj = JSON.parse(txt);
document.getElementById("demo").innerHTML = obj.name + ", " + obj.age;
</script>

</body>
</html>
```



JSON vs XML:

JSON vs XML

- ❖ JSON doesn't use end tag
- ❖ JSON is shorter
- ❖ JSON is quicker to read and write
- ❖ JSON can use arrays

JSON is Like XML

- ❖ Both JSON and XML are "self describing" (human readable).
- ❖ Both JSON and XML are hierarchical (values within values).
- ❖ Both JSON and XML can be parsed and used by lots of programming languages.
- ❖ Both JSON and XML can be fetched with an XMLHttpRequest.

```
{  
  "employees": [  
    { "firstName": "John", "lastName": "Doe" },  
    { "firstName": "Anna", "lastName": "Smith" },  
    { "firstName": "Peter", "lastName": "Jones" }  
  ]  
}
```

```
<employees>  
  <employee>  
    <firstName>John</firstName>  
    <lastName>Doe</lastName>  
  </employee>  
  <employee>  
    <firstName>Anna</firstName>  
    <lastName>Smith</lastName>  
  </employee>  
  <employee>  
    <firstName>Peter</firstName>  
    <lastName>Jones</lastName>  
  </employee>  
</employees>
```



JSON.stringify():

- ❖ A common use of JSON is to exchange data to/from a web server.
- ❖ When sending data to a web server, the data has to be a string.
- ❖ Convert a JavaScript object into a string with JSON.stringify()

```
<!DOCTYPE html>
<html>
<body>

<h2>Create JSON string from a JavaScript object.</h2>


<p id="demo"></p>

<script>
var obj = { name: "John", age: 30, city: "New York" };
var myJSON = JSON.stringify(obj);
document.getElementById("demo").innerHTML = myJSON;
</script>

</body>
</html>
```

JSON Array:

```
<!DOCTYPE html>
<html>
<body>
<p>Access an array value of a JSON object.</p>
<p id="demo"></p>
<script>
var myObj, x;
myObj =
{ "name":"John",
  "age":30,
  "cars":["Ford", "BMW", "Fiat" ]
};
x = myObj.cars[0];
document.getElementById("demo").innerHTML = x;
</script>
</body>
</html>
```



```
<!DOCTYPE html>
<html>
<body>
<p>Looping through an array using a for in loop:</p>
<p id="demo"></p>
<script>
var myObj, i, x = "";
myObj =
{ "name":"John",
  "age":30,
  "cars":["Ford", "BMW", "Fiat" ]
};
for (i in myObj.cars) {
  x += myObj.cars[i] + "<br>";
}
document.getElementById("demo").innerHTML = x;
</script>
</body>
</html>
```

End of Slides:

*Thank
you*

