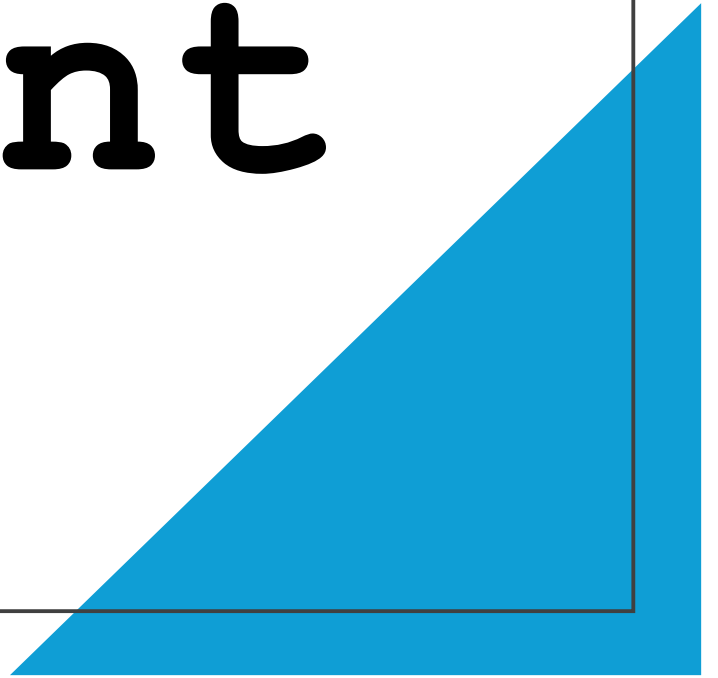
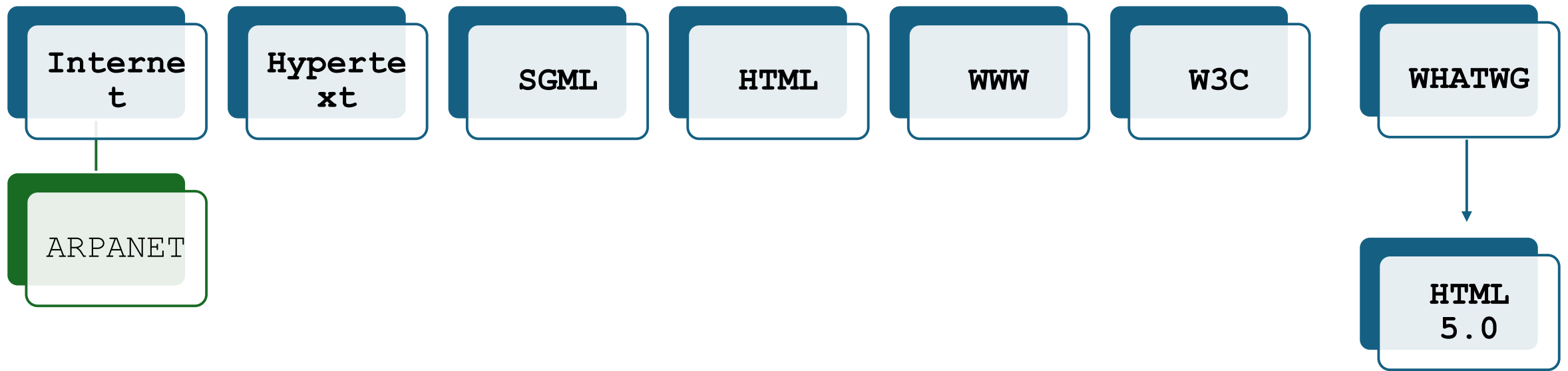


# Full Stack Web Development

by Piyush  
Bagla



# Why HTML 5.0 ?



# Internet

---

- The Internet is a global network of interconnected computer networks that communicate using standardized protocols.

---

- The Internet provides the infrastructure for transmitting data between devices and networks worldwide.

---

- The WWW operates on top of the Internet, leveraging its infrastructure to connect web servers and clients, enabling the exchange of hypertext documents (web pages) between users and servers.

# History of Internet



# History of Internet

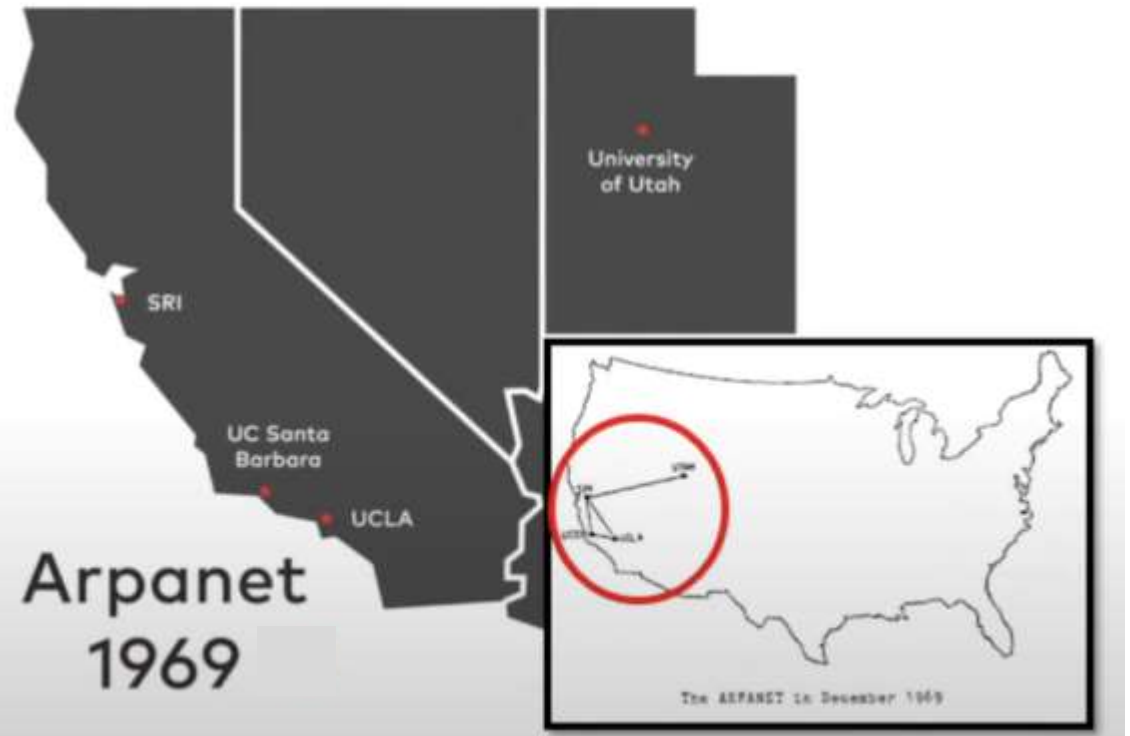
## mid 1960s

The **Advanced Research Projects Agency (ARPA)** in the Department of Defense (DOD) was interested in finding a way to connect computers together.

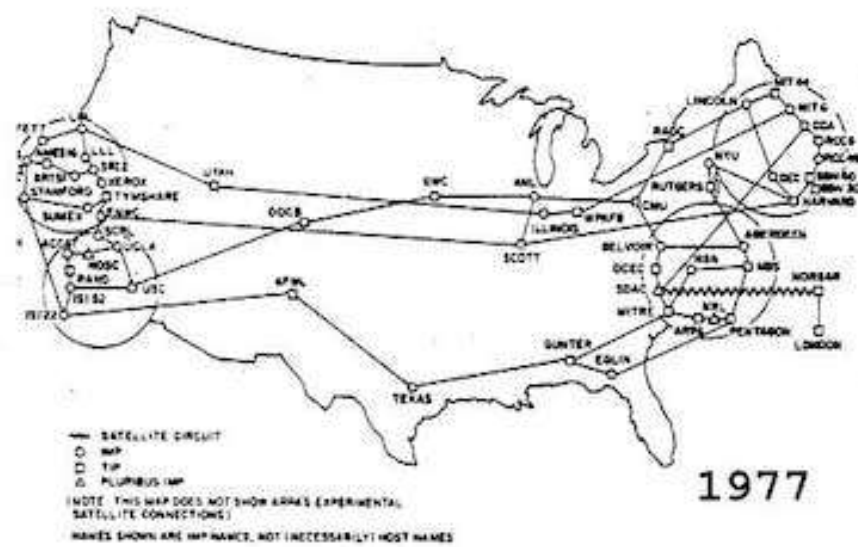
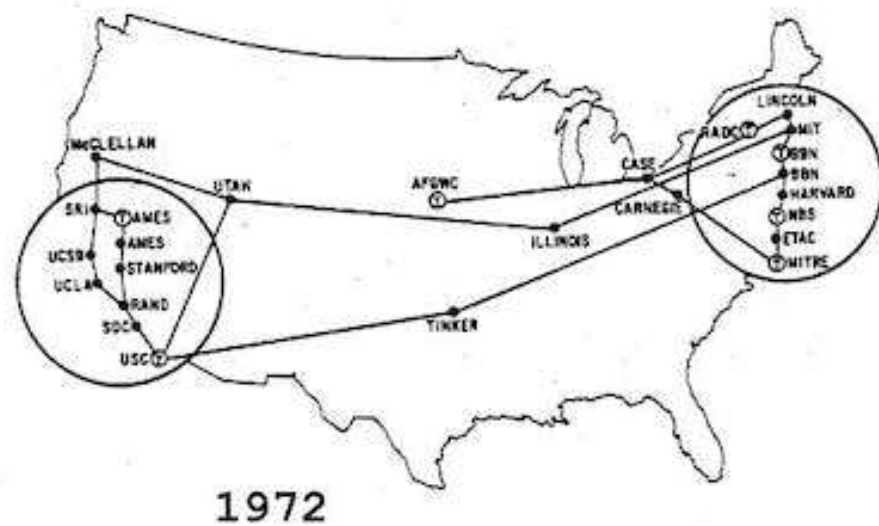
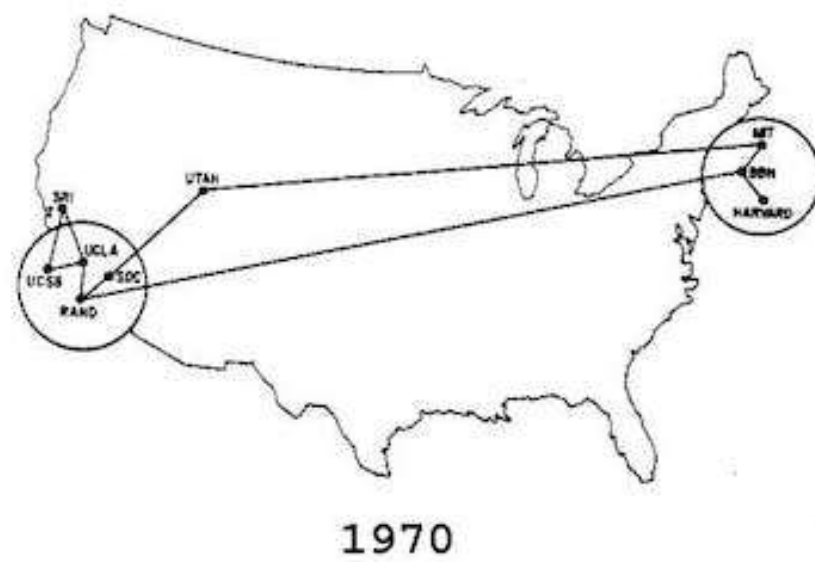
So that the researchers they funded could share their findings, thereby reducing costs and eliminating duplication of effort.

## by 1969

**Four nodes**, at the University of California at Los Angeles (**UCLA**), the University of California at Santa Barbara (**UCSB**), Stanford Research Institute (**SRI**), and the **University of Utah**, were connected.



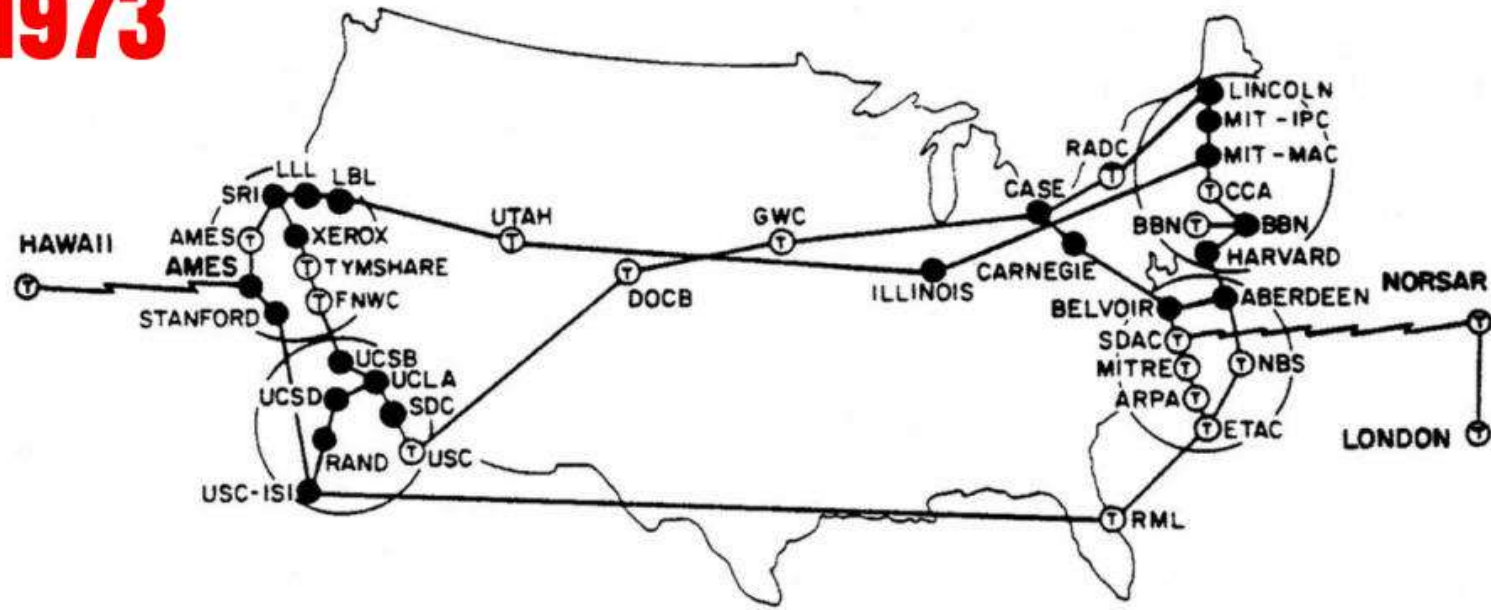
# ARPANET



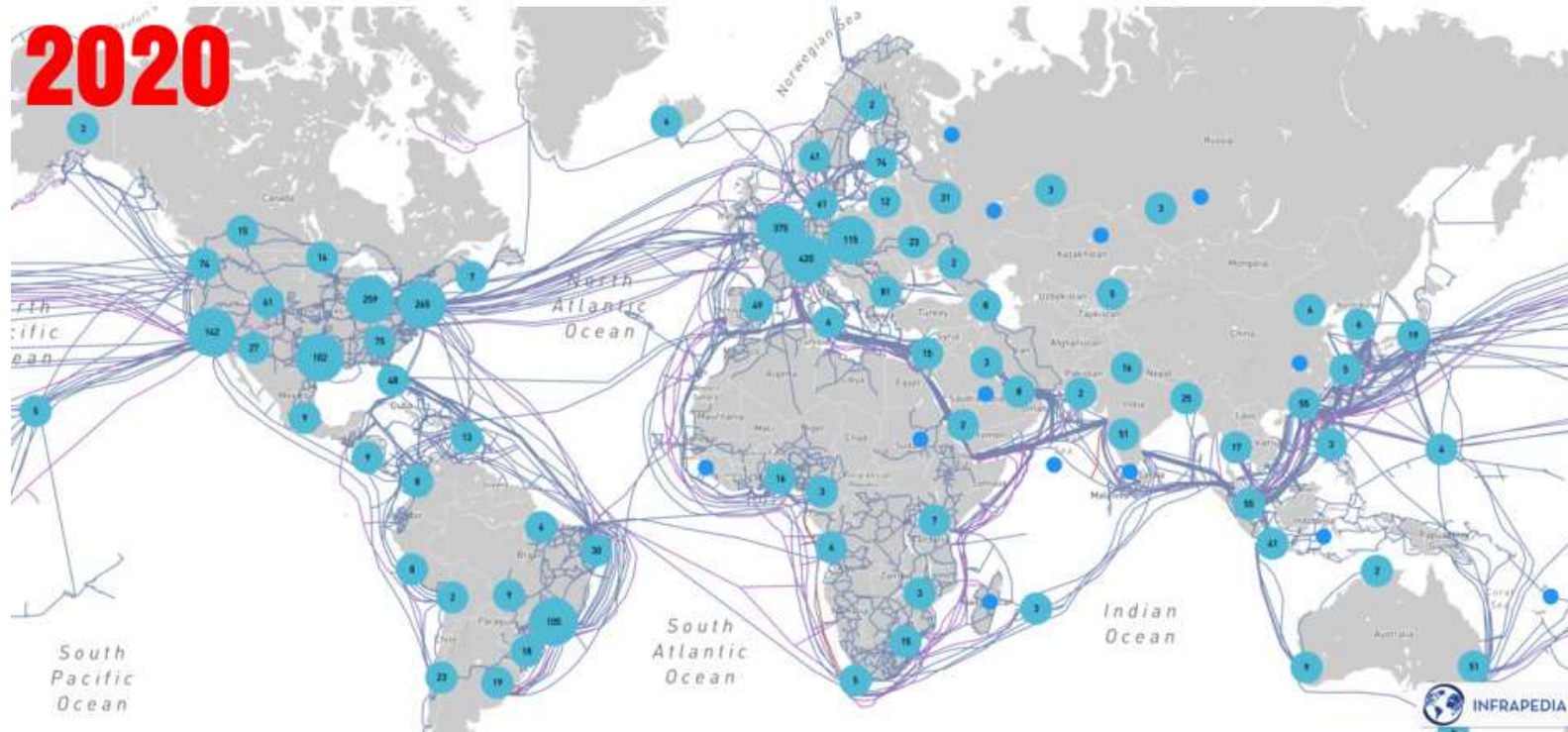




1973



2020





# Hypertext

---

- Hypertext is a concept of organizing and linking text documents electronically, allowing users to navigate between related pieces of information through hyperlinks.

---

- Hypertext enables non-linear navigation, where users can jump from one document to another by clicking on hyperlinks embedded within the text.

# SGML

---

- SGML stands for Standard Generalized Markup Language. It's used for defining the structure and attributes of documents.

---

- SGML serves as a framework for defining other markup languages and has been influential in developing various document formats and standards.

---

- SGML allows users to define the hierarchical structure of documents, specify elements and attributes, and establish relationships between different parts of the document. SGML-aware software can then process these documents to extract, manipulate, or present the information contained within them.

# SGML Application Areas

---



In the publishing industry, SGML has been used to define the structure of technical documentation, such as user manuals for complex machinery or equipment.



The aerospace industry has extensively used SGML for creating structured documentation, such as aircraft maintenance manuals.



In the healthcare industry, SGML could be used to define the structure of electronic medical records (EMRs)

# HTML

---

- HTML provided a standardized markup language specifically designed for creating hypertext documents for the World Wide Web.

---

- HTML provides a standardized way to structure and format hypertext documents for the web, using tags to define elements such as headings, paragraphs, images, and hyperlinks.

---

- HTML tags allow content creators to embed hyperlinks within web pages, enabling users to navigate between different documents on the web.

# World Wide Web

---

- The WWW is a system of interlinked hypertext documents accessed via the Internet.

---

- The WWW was invented by Tim Berners-Lee in 1989 as a decentralized system for sharing and accessing hypertext documents (web pages) globally.

---

- Hyperlinks within web pages allow users to navigate between different documents on the web, creating a network of interconnected information.



# Link Between Them

---

- Hypertext laid the conceptual groundwork for linking text documents electronically, enabling non-linear navigation between related pieces of information.

---

- HTML provided the language and structure for creating hypertext documents within the framework of the WWW, allowing content creators to embed hyperlinks within web pages.

---

- The WWW combined hypertext with standardized protocols like HTTP (Hypertext Transfer Protocol) and URLs (Uniform Resource Locators) to create a decentralized system for sharing and accessing hypertext documents (web pages)

---

via the Internet.

- The Internet provides the underlying infrastructure for transmitting data between devices and networks, enabling the exchange of hypertext documents between users and servers worldwide.

# W3C

---

-The consortium formed in 1994 by Tim and it focused on establishing foundational standards for the web, including HTML, HTTP, and URLs

---

-This included specifications for markup languages (HTML, XML), style sheet languages (CSS), document object models (DOM), and web accessibility guidelines (WCAG) .

---

- The W3C's work helped establish a common framework for web development and ensured the compatibility of web technologies across different platforms and devices.

# WHATWG

---

-The WHATWG was formed in 2004 by individuals from Apple, Mozilla, and Opera in response to perceived slow progress in web standards development at the W3C.

---

-At that time, the W3C was focusing on XHTML 2.0, which was seen by some as diverging from the practical needs of web developers. The WHATWG aimed to address this gap by focusing on developing specifications that reflected the needs of the modern web.

---

-Formed HTML 5.0 in collaboration with W3C

# HTML 5.0 Predecessors

HTML Version Published	Development Body	Year
•   HTML 1.0	Tim Berners-Lee at CERN	1991
•   HTML 2.0	Internet Engineering Task Force	1995
•   HTML 3.2	W3C	1997
•   HTML 4.01	W3C	1999
•   XHTML	W3C	2000
•   HTML5 2008 (Ongoing)	W3C & WHATWG	

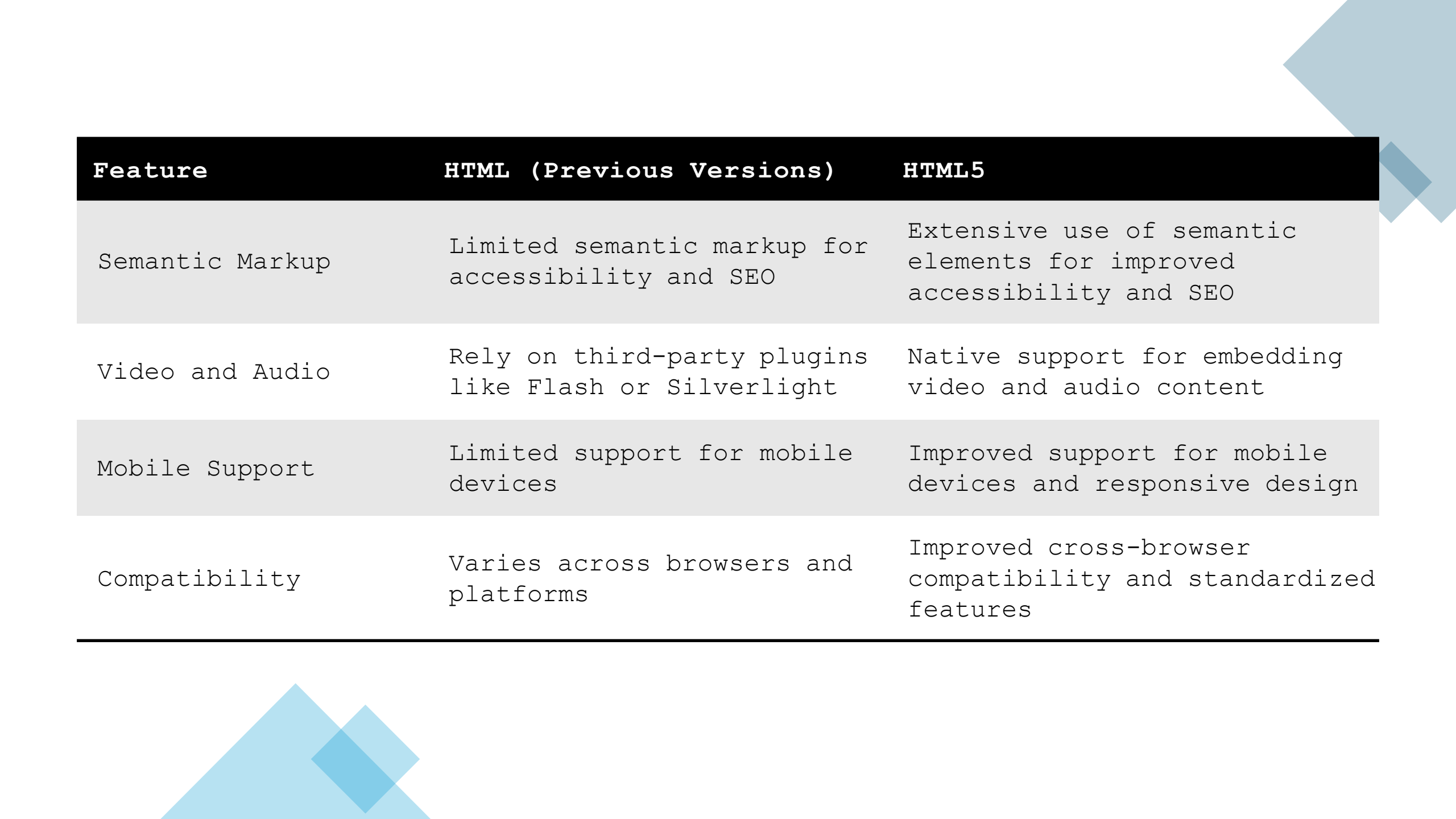
# Comparison of HTML 5 Elements with previous versions

HTML Version	Elements Added
HTML 1.0	It was the initial version that laid the groundwork for subsequent versions to build upon.
HTML 2.0	<code>&lt;img&gt;</code> , <code>&lt;form&gt;</code> , <code>&lt;input&gt;</code> , <code>&lt;textarea&gt;</code> , <code>&lt;select&gt;</code> , <code>&lt;button&gt;</code> , <code>&lt;ol&gt;</code> , <code>&lt;ul&gt;</code> , <code>&lt;li&gt;</code> , <code>&lt;table&gt;</code> , <code>&lt;tr&gt;</code> , <code>&lt;td&gt;</code> , <code>&lt;th&gt;</code>
HTML 3.2	<code>&lt;embed&gt;</code> , <code>&lt;iframe&gt;</code> , <code>&lt;font&gt;</code>
HTML 4.01	<code>&lt;meta&gt;</code> , <code>&lt;link&gt;</code> , <code>&lt;script&gt;</code> , <code>&lt;div&gt;</code> , <code>&lt;span&gt;</code>
XHTML 1.0	Same elements as HTML 4.01 but reformulated in XML syntax
HTML5	<code>&lt;header&gt;</code> , <code>&lt;nav&gt;</code> , <code>&lt;section&gt;</code> , <code>&lt;article&gt;</code> , <code>&lt;aside&gt;</code> , <code>&lt;footer&gt;</code> , <code>&lt;audio&gt;</code> , <code>&lt;video&gt;</code> , <code>&lt;canvas&gt;</code> , <code>&lt;svg&gt;</code> , <code>&lt;details&gt;</code> , <code>&lt;summary&gt;</code> , <code>&lt;progress&gt;</code> , <code>&lt;meter&gt;</code> , <code>&lt;datalist&gt;</code> , <code>&lt;output&gt;</code> , <code>&lt;time&gt;</code> , <code>&lt;mark&gt;</code> , <code>&lt;meter&gt;</code> , <code>&lt;progress&gt;</code> , <code>&lt;ruby&gt;</code> , <code>&lt;rt&gt;</code> , <code>&lt;rp&gt;</code> , and many more



Feature	HTML (Previous Versions)	HTML5
Doctype Declaration	Various doctypes for HTML 4.01, XHTML 1.0, etc.	Simplified to <code>&lt;!DOCTYPE html&gt;</code> for HTML5
Structural Elements	Limited semantic elements ( <code>&lt;div&gt;</code> , <code>&lt;span&gt;</code> , <code>&lt;table&gt;</code> , etc.)	Rich set of semantic elements ( <code>&lt;header&gt;</code> , <code>&lt;nav&gt;</code> , <code>&lt;section&gt;</code> , <code>&lt;footer&gt;</code> , etc.)
Multimedia Support	Rely on third-party plugins like Flash or Silverlight	Native <code>&lt;audio&gt;</code> and <code>&lt;video&gt;</code> elements
Canvas	Not available	<code>&lt;canvas&gt;</code> element for dynamic graphics
Forms Enhancements	Limited input types and attributes, rely on JavaScript for validation	New input types (email, url, date, number, etc.) and form validation attributes (required, pattern, etc.)
Local Storage	Rely on JavaScript cookies (eg. Lang of webpage)	localStorage and sessionStorage APIs for client-side storage

Feature	HTML (Previous Versions)	HTML5
Geolocation	Not available	Geolocation API for retrieving user's location
Drag and Drop	Not available	Native Drag and Drop API for dragging and dropping elements
Offline Application Cache	Not available	Application Cache API for offline web applications (eg calculator, note-taking application)
Web Workers	Not available	Web Workers API for running scripts in background threads (eg. Image processing)



Feature	HTML (Previous Versions)	HTML5
Semantic Markup	Limited semantic markup for accessibility and SEO	Extensive use of semantic elements for improved accessibility and SEO
Video and Audio	Rely on third-party plugins like Flash or Silverlight	Native support for embedding video and audio content
Mobile Support	Limited support for mobile devices	Improved support for mobile devices and responsive design
Compatibility	Varies across browsers and platforms	Improved cross-browser compatibility and standardized features

---

# Semantics in HTML

- A semantic element clearly describes its meaning to both the browser and the developer.
- They are also called structural elements.
- In HTML, there are some semantic elements that can be used to define different parts of a web page.
- Examples of **semantic** elements: `<form>`, `<table>`, and `<article>` - Clearly define its content.

`<section>` is used for grouping related content, while `<article>` is used for stand-alone, independent content items that can be syndicated or shared separately.

---

- `<footer>`
- `<header>`
- `<main>`
- `<mark>`
- `<nav>`
- `<article>`
- `<aside>`
- `<details>`
- `<figcaption>`
- `<figure>`
- `<section>`
- `<summary>`
- `<time>`



# HTML Links

- Use the `<a>` element to define a link
- Use the `href` attribute to define the link address
- Use the `target` attribute to define where to open the linked document
- Use the `<img>` element (inside `<a>`) to use an image as a link
- Use the `mailto:` scheme inside the `href` attribute to create a link that opens the user's email program



# HTML Links

## Bookmarks



- `<p><a href="#C4">Jump to Chapter 4</a></p>`

- 
- 
- 
- 
- 

- `<h2 id="C4">Chapter 4</h2>`

- `<p>This chapter explains ba bla bla</p>`

# HTML Links

## Absolute vs Relative URL

There are two ways to specify the URL in the src attribute:

- **1. Absolute URL** - Links to an external image that is hosted on another website. Example: `src="https://www.w3schools.com/images/img_girl.jpg"`.

**Notes:** External images might be under copyright. If you do not get permission to use it, you may be in violation of copyright laws. In addition, you cannot control external images; it can suddenly be removed or changed.

- **2. Relative URL** - Links to an image that is hosted within the website. Here, the URL does not include the domain name.
  - If the URL begins without a slash, it will be relative to the current page.
    - Example: `src="img_girl.jpg"`.
  - If the URL begins with a slash, it will be relative to the domain.
    - Example: `src="/images/img_girl.jpg"`.

**Tip:** It is almost always best to use relative URLs. They will not break if you change the domain.

# HTML Images and Icon

## Image

- Basic tag and attributes
- Image Map
- The Picture Element

## Icon

Step 1 - Copy the style link from any website (which provides this functionality) inside the

Example - **Font Awesome CDN**

Step 2 - Copy the desired icon link inside body section

Example - **Font Awesome Icon**

# HTML Images and Icon

## Image Map

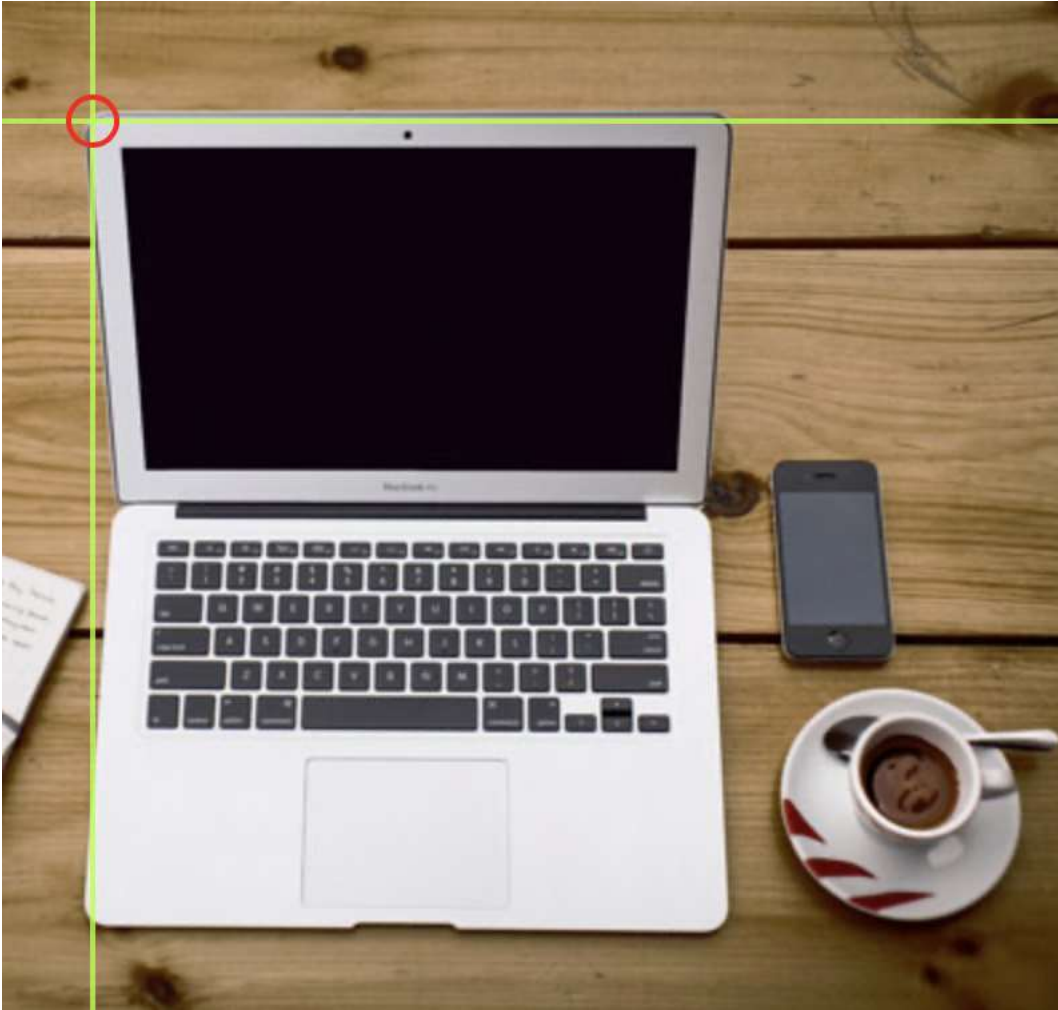
An image map is an image with clickable areas.



```


<map name="workmap">
  <area shape="rect" coords="34,44,270,350" alt="Computer" href="computer.htm">
  <area shape="rect" coords="290,172,333,250" alt="Phone" href="phone.htm">
  <area shape="circle" coords="337,300,44" alt="Coffee" href="coffee.htm">
</map>
```





The coordinates  $34, 44$  is located 34 pixels from the left margin and 44 pixels from the top



The coordinates  $270, 350$  is located 270 pixels from the left margin and 350 pixels from the top



The coordinates 34, 44 is located 34 pixels from the left margin and 44 pixels from the top

The coordinates 270, 350 is located 270 pixels from the left margin and 350 pixels from the top

# HTML Images and Icon

## Image Map

An image map is an image with clickable areas.



# HTML Images and Icon

## The Picture Element

The HTML `<picture>` element allows you to display different pictures for different devices or screen sizes.



```
<picture>
  <source media="(min-width:
650px)" srcset="img_food.jpg">
  <source media="(min-width:
465px)" srcset="img_car.jpg">
  
</picture>
```

# HTML List

- Unordered
  - type
    - circle
    - Square
    - Disc
    - none
- Ordered
  - type
    - 1,2,3
    - a,b,c
    - A,B, C
    - i, ii, iii
    - I, II, III,
- Description
- Nested List

## Unordered

- List item
- List item
- List item
- List item

## Ordered

1. 1st item
2. 2nd item
3. 3rd item
4. 4th item

## Description

List item title

- list item description

List item title

- list item description

List item title

- list item description

# HTML Table

Cell Padding

Cell Spacing

The diagram illustrates an HTML table with two columns and three rows. The first column contains labels: 'name', 'age', and 'job'. The second column contains values: 'Samy', '20', and 'Manager'. A light blue rounded rectangle highlights the entire table structure. A bracket on the left side of the first row, labeled 'Cell Padding', indicates the space between the cell border and the text. A bracket on the right side of the first row, labeled 'Cell Spacing', indicates the space between the two columns.

'name'	Samy
'age'	20
'job'	Manager



# HTML Table

***Simple table***

Sr. No	Roll No.	Name
1	101	ABC
2	102	XYZ

***Table with colspan***

Name	Phone.	
ABC	123	789
XYZ	345	654

***Table with Rowspan***

Name	ABC
Phone No.	123
	345

# HTML Table

## TIME TABLE

Day/Period	I 9:30-10:20	II 10:20-11:10	III 11:10-12:00	12:00-12:40	IV 12:40-1:30	V 1:30-2:20	VI 2:20-3:10	VII 3:10-4:00
Monday	Eng	Mat	Che	L U N C H	LAB			Phy
Tuesday	LAB				Eng	Che	Mat	SPORTS
Wednesday	Mat	phy	Eng		Che	LIBRARY		
Thursday	Phy	Eng	Che		LAB			Mat
Friday	LAB				Mat	Che	Eng	Phy
Saturday	Eng	Che	Mat		SEMINAR			SPORTS

# iFrame in HTML

HTML element used to embed another document within the current HTML document. It allows you to display content from another source, such as a web page, PDF document, video, or interactive application, within the context of your own web page.

## Example of Iframe

### Two iframes

Username:

Password:

Gender:

Message:

☐ Additional Info

[Labnol.org](#)

- [Home](#)
- [Archives](#)
- [About](#)
- [Contact](#)

**Article Title**

Lorem ipsum dolor sit amet consectetur adipisicing elit. Doloremque temporibus consequuntur natus voluptatum, veritatis quis quisquam soluta necessitatibus magnam, suscipit culpa dicta earum. Neque, voluptatem repellat? Soluta temporibus dolor excepturi!

# Block level and Inline elements

- A block-level element always starts on a new line, and the browsers automatically add some space (a margin) before and after the element.
- A block-level element always takes up the full width available (stretches out to the left and right as far as it can)

Eg. `<p>`, `<div>`, `<ul>`, `<form>` etc

**Note:** The `<div>` element is often used as a container for other HTML elements.

- An inline element does not start on a new line.
- An inline element only takes up as much width as necessary.

Eg. `<span>`, `<img>`, `<a>` etc

# HTML Form

Username:  Password:  Email:  Age:

Birthdate:  Subscribe to newsletter: ☒

Gender: ☐ Male ☐ Female ☐ Other

Country:  Message:




# Get vs POST

## **GET:**

- Appends the form data to the URL in name/value pairs
- NEVER use GET to send sensitive data! (the submitted form data is visible in the URL!)
- The length of a URL is limited (2048 characters)
- Useful for form submissions where a user wants to bookmark the result
- GET is good for non-secure data, like query strings in Google

## **POST:**

- Appends the form data inside the body of the HTTP request (the submitted form data is not shown in the URL)
  - POST has no size limitations and can be used to send large amounts of data.
  - Form submissions with POST cannot be bookmarked.
-



# Accessibility in HTML5

Accessibility in HTML5 refers to the practice of designing and coding web content in a way that ensures it is accessible (a good way to navigate and interact) to all users, including those with disabilities.

- Semantic HTML Elements
- Alternative Text
- Keyboard Navigation
- Create Good Link Text
- Form attributes

etc

---





# The W3C Markup Validation Service

A *valid* Web page is not necessarily a good web page, but an *invalid* Web page has little chance of being a good web page.

<https://validator.w3.org>

---



# The W3C CSS Validation Service

**Note:** If you want to validate your CSS style sheet embedded in an (X)HTML document, you should first [check that the \(X\)HTML you use is valid](#).

<https://jigsaw.w3.org/css-validator/>

---



# HTML | CSS | Javascript

1. HTML to define the basic structure of web pages.
  2. CSS to specify the layout of web pages
  3. JavaScript to program the behavior of web pages
-

# CSS (Cascading Style Sheets)

selector

p

declaration

{ color:blue; }

property

value

```
p  
{  
  color: blue;  
}
```



# Types of CSS or Ways to Insert CSS

- External CSS
  - Internal CSS
  - Inline CSS
-

---

# Inline CSS

```
<html>
<head>
<body>
<h1>How to add CSS</h1>
<p style="font-size:20px; color:red">This is paragraph</p>
</body>
</html>
```



Inline CSS for `<p>` tag

Here we apply CSS on that line where html element (`<p>`) are use so this type of css is called Inline css

---

# Internal CSS

```
<html>  
<head>  
<style>
```

```
p  
{  
font-size: 20px;  
color: red;  
}
```

```
</style>
```

```
<html>  
</head>  
</head>  
<body>  
<h1>How to add css</h1>  
<p>This is my first Html code</p>  
</body>  
</html>
```

`<style>` tag are used for Internal css within `<head>....</head>` tag

Inline css for paragraph `<p>` tag this is common for all `<p>` tag

# External CSS

mystyle.css

```
<style>
p
{
margin-left: 20px;
color: yellow;
}
body
{
background-color: #000000;
}
</style>
```

Here we first create two files one for css which is mystyle.css and another for html homepage.html

homepage.html

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
<body>
<h1>How to add css</h1>
<p>This is my first code</p>
</body>
</html>
```

add css pag  
on html pag





# HTML and CSS Comments


HTML

```
<!-- These paragraphs will be red -->
```

CSS

```
/*These paragraphs will be red */
```

---



# CSS Selectors

CSS selectors are used to select the HTML elements you want to style.

CSS selectors are divided into five categories:

- **Simple** selectors (select elements based on name, id, class)
  - **Combinator** selectors (select elements based on a specific relationship between them)
  - **Pseudo-class** selectors (select elements based on a certain state)
  - **Pseudo-elements** selectors (select and style a part of an element)
  - **Attribute** selectors (select elements based on an attribute or attribute value)
-

# Simple selectors

```
p {  
  text-align: center;  
  color: red;  
}
```

Single Selector

```
p,h1 {  
  text-align: center;  
  color: red;  
}
```

Group Selector



# ID selector

- The id attribute is used to specify a unique id for an HTML element
  - The value of the id attribute must be unique within the HTML document
  - The id attribute is used by CSS and JavaScript to style/select a specific element
  - The value of the id attribute is case-sensitive
  - The id attribute is also used to create HTML bookmarks
  - JavaScript can access an element with a specific id with the `getElementById()` method
-



# Class selector

- The HTML class attribute specifies one or more class names for an element
  - Classes are used by CSS and JavaScript to select and access specific elements
  - The class attribute can be used on any HTML element
  - The class name is case-sensitive
  - Different HTML elements can point to the same class name
  - JavaScript can access elements with a specific class name with the `getElementsByClassName()` method
-



# Combinator selectors

There are four different combinators in CSS:

- descendant selector (space)
  - child selector (>)
  - adjacent sibling selector (+)
  - general sibling selector (~)
-

# Pseudo-class selectors

**A pseudo-class is used to define a special state of an element.**

Syntax

```
selector:pseudo-class {  
  property: value;  
}
```

```
/* unvisited link */  
a:link {  
  color: #FF0000;  
}  
  
/* visited link */  
a:visited {  
  color: #00FF00;  
}
```

```
div:hover p {  
  display: block;  
}
```

# Pseudo-element selectors

**A CSS pseudo-element is used to style specified parts of an element.**

Syntax

```
selector::pseudo-element {  
  property: value;  
}
```

```
p::first-line {  
  color: #ff0000;  
  font-variant: small-  
caps;  
}
```



# Attribute selectors

**It is possible to style HTML elements that have specific attributes or attribute values.**

```
a[target] {  
  background-color: yellow;  
}
```

```
a[target="_blank"] {  
  background-color: yellow;  
}
```

# CSS Colors

There are four common ways to apply colors

1. Direct Color Name

```
<h1 style="color:Red;">Hello World</h1>
```

2. RGB (Red, Green, Blue)

```
rgb(255,255,255)
```

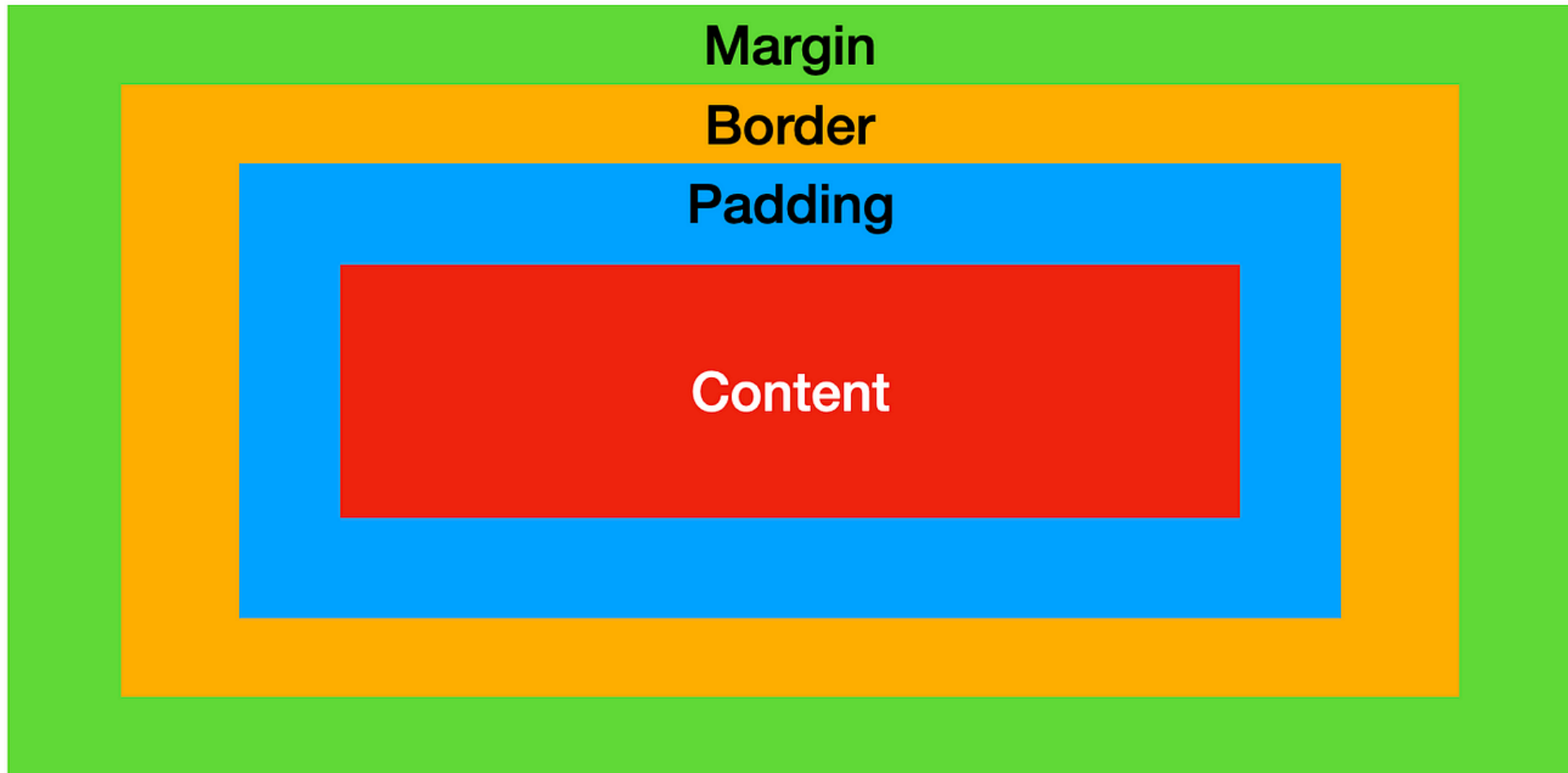
3. HEX

```
#RRGGBB, #ff6678
```

4. HSL (hue, saturation, and lightness) **hsl(0, 100%, 50%)**

- Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.
- Saturation is a percentage value. 0% means a shade of gray, and 100% is the full color.
- Lightness is also a percentage. 0% is black, 50% is neither light or dark, 100% is white

# CSS Box Model



# CSS Box Model

All HTML elements can be considered as boxes.

## Width and Height of an Element

**Note:** When you set the width and height properties of an element with CSS, you just set the width and height of the **content area**.

- To calculate the total width and height of an element, you must also include the padding and borders.

**Note:** The margin property also affects the total space that the box will take up on the page, but the margin is not included in the actual size of the box. The box's total width and height stops at the border.



# CSS Fonts

The shorthand `font` property, combines several font-related properties into one. The order of the values in the shorthand `font` property is as follows:

- 1.font-style:** Specifies the font style (e.g., italic).
- 2.font-variant:** Specifies the font variant (e.g., small-caps).
- 3.font-weight:** Specifies the font weight (e.g., bold).
- 4.font-size/line-height:** Specifies the font size and optional line height (e.g., 12px/30px).
- 5.font-family:** Specifies the font family (e.g., Georgia, serif).

**Note:** The `font-size` and `font-family` values are required. If one of the other values is missing, their default value are used.

---




# Bootstrap

- Bootstrap is a free front-end framework for faster and easier web development.
  - Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins.
  - Bootstrap also gives you the ability to easily create responsive designs.
-



# Bootstrap Versions

- **Bootstrap 5** (released 2021) is the newest version with new components, faster stylesheet and more responsiveness. However, Internet Explorer 11 and down is not supported.
  - Previous versions are Bootstrap 3 & 4.
  - Bootstrap 5 has switched to vanilla JavaScript instead of jQuery.
-



# Where to Get Bootstrap 5?

There are two ways to start using Bootstrap 5 on your own website.

- Include Bootstrap 5 from a CDN.
  - Download Bootstrap 5 from [www.getbootstrap.com](https://www.getbootstrap.com)
-





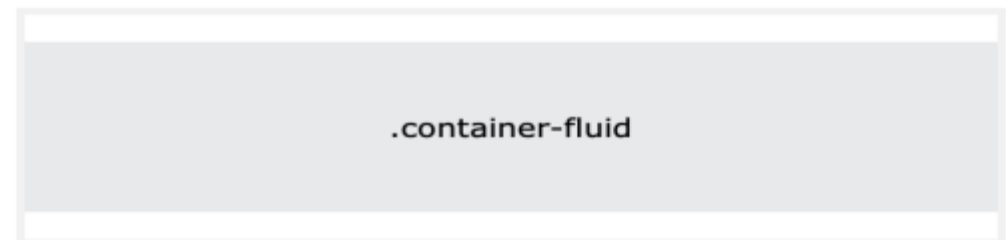
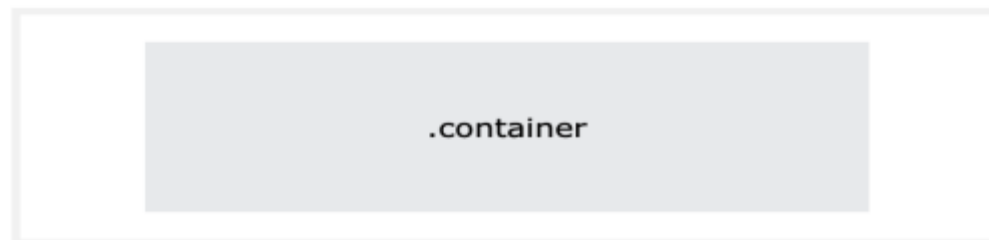
# Advantage of Bootstrap 5 from CDN

- Many users already have downloaded Bootstrap 5 from jsDelivr when visiting another site. As a result, it will be loaded from cache when they visit your site, which leads to faster loading time.
  - Also, most CDN's will make sure that once a user requests a file from it, it will be served from the server closest to them, which also leads to faster loading time.
-

# Bootstrap 5 Containers

Containers are used to pad the content inside of them, and there are two container classes available:

1. The **.container** class provides a responsive **fixed-width container**.
2. The **.container-fluid** class provides a **full-width container**, spanning the entire width of the viewport.





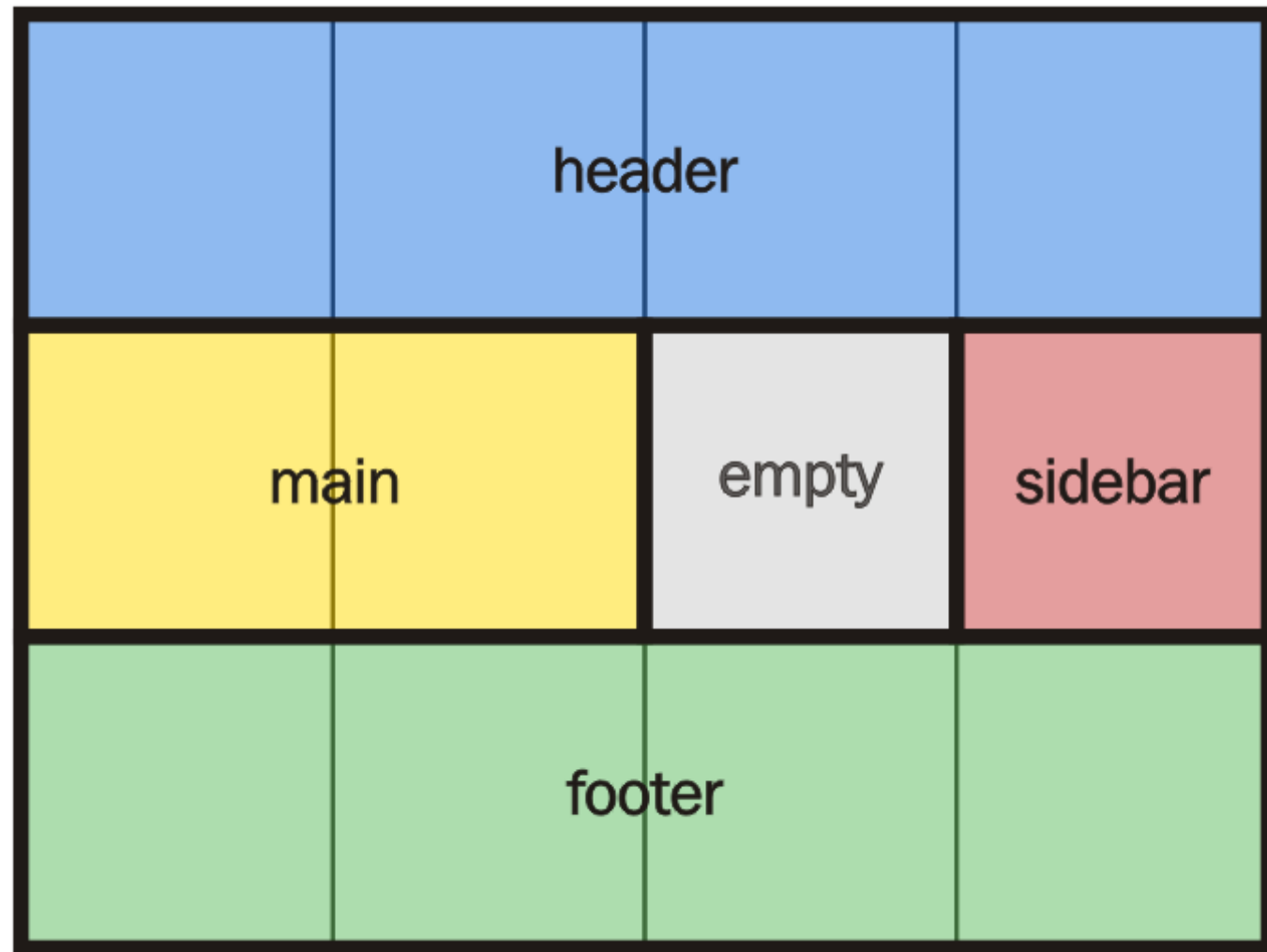
# Grid System

It consists of a series of rows and columns organized into a grid, which provides a structured way to arrange content on a web page.

Here are the key components of the grid system in Bootstrap:

- **Container:** The outermost element that wraps all the rows and columns. It provides a fixed-width container for your content. Bootstrap offers two types of containers: **.container** for fixed-width containers and **.container-fluid** for full-width containers.
  - **Row:**
  - **Column:** Columns are the building blocks of the grid system. They are placed inside rows and are used to divide the horizontal space within a row. Bootstrap provides a set of predefined column classes, such as **.col-**, **.col-sm-**, **.col-md-**, **.col-lg-**, and **.col-xl-**, which define the width of the column at different breakpoints.
-

# Grid System



# Basic functionality of JavaScript

- JavaScript can "display" data in different ways.
- Writing into an HTML element, using **innerHTML**.
  - The **innerHTML** property defines the HTML content
- Writing into the HTML output using **document.write()**.
  - Using **document.write()** after an HTML document is loaded, will **delete all existing HTML**.
  - The **document.write()** method should only be used for testing.
- Writing into an alert box, using **window.alert()**.
- Writing into the browser console, using **console.log()**.
  - For debugging purposes, you can call the **console.log()** method in the browser to display data.

# JavaScript can be included in an HTML document in several different ways; the most common are :

- **Inline JavaScript**

```
<button onclick="alert('Hello,  
world!')">Click me</button>
```

- **Internal JavaScript**

- **Inside <body>**

```
<script>  
    // JavaScript  
    code here  
</script>
```

- **Inside <head>**

```
<script>  
    // JavaScript  
    code here  
</script>
```

- **External JavaScript**

```
<script src="script.js"></script>
```

# Common ways to select HTML element

- `getElementById()`
- `getElementsByClassName()`
- `getElementsByName()`
- `getElementsByTagName()`

# JavaScript Variables

Variables  
can be  
declared  
in 4 ways:

Automatica  
lly

```
a = 5;
```

Using **var**

```
var a = 5;
```

Using **let**

```
let a = 5;
```

Using **const**  
**t**

```
const a = 5;
```





# Block Scope

- Before ES6 (2015), JavaScript did not have **Block Scope**.
  - JavaScript had **Global Scope** and **Function Scope**.
  - ES6 introduced the two new JavaScript keywords: **let** and **const**.
  - These two keywords provided **Block Scope** in JavaScript.
  - Variables declared with the **var** always have **Global Scope or Function Scope**.
  - Variables declared with the **var** keyword can NOT have block scope.
  - Variables declared with **var** inside a { } block can be accessed from outside the block.
-



# Introduced

**var** - pre ES2015

**let** - ES2015 (ES6)

**const** – ES2015 (ES6)

---

# Declaration

```
var a; // allowed
```

```
let b; // allowed
```

```
const c; // not allowed
```

SyntaxError: Missing initializer in const  
declaration

```
const c = 10 ; // allowed
```

# Initialization

## *Same line Initialization*

```
.  
var a = 10,  
let b = 20;  
const c = 30;
```

## *Later Initialization :*

```
var a;  
a = 10; // allowed  
  
let b;  
b = 20; // allowed  
  
const c;  
c = 30; // not allowed  
SyntaxError: Missing initializer in const  
declaration
```

# Redeclaration

```
var a = 10;  
var a = 20; //its possible to with var
```

```
let b = 10;  
let b = 20; //its not possible to with let  
SyntaxError: Identifier 'b' has already been declared
```

```
const c = 10;  
const c = 20; //its not possible with const  
SyntaxError: Identifier 'c' has already been declared
```

# Redeclaring **Var**

- Variables defined with **var** **can** be redeclared.
- Redeclaring a variable using the **var** keyword can impose problems.

```
var x = 10;  
// Here x is 10  
{  
var x = 2;  
// Here x is 2  
}  
// Here x is 2
```

- If you re-declare a JavaScript variable declared with **var**, it will not lose its value.

```
var a = 5;  
var a; // var is still 5
```

# Reinitialization

```
var a = 10; //declared once  
a = 20; //reintialized again --> its possible with var
```

```
let b = 10; //declared once  
b = 20; //reintialized again --> its possible with let
```

```
const c = 10; //declared once  
c = 20; //reintialized again --> its NOT possible with let,  
TypeError: Assignment to constant variable.
```

# Scope

## a) **Functional Scope:** *declare greeting variable without **var***

When we don't declare variables without any var, let and const, variables get hoisted globally.

```
function wishFoofi() {  
  greeting = "Hello, foofi!"; // hoisted globally  
  console.log(greeting);  
}  
wishFoofi();  
console.log(greeting); // Hello, foofi!
```

### **Output:**

Hello, foofi!

Hello, foofi!



# Scope

**a) Functional Scope:** *declare greeting variable with **var** now*

```
function wishFoofi() {  
  var greeting = "Hello, foofi!"; //greeting remained functional scoped  
  console.log(greeting);  
}
```

**Output:**

Hello, foofi!

ReferenceError: greeting is not defined

# Scope

## b) Block Scope

```
{  
  var x = 10;  
}  
console.log(x); // output 10  
  
{  
  let y = 20;  
}  
console.log(y); // output : ReferenceError: y is not defined  
  
{  
  const z = 30;  
}  
console.log(z); // output : ReferenceError: z is not defined
```

# Hoisted

```
console.log(x); // Outputs 'undefined'  
var x = 10; // Assignment remains in its original position
```

***let and const*** : Not Hoisted\*

\* variables with **let** and **const** are also **hoisted** but **at the top of block scope** and they are not assigned with **undefined**.

Keywords	var	let	const
Eample	var a = 10	let b = 10	const c = 10
Initialization	Can be declared without an initial value	Can be declared without an initial value	Must be assigned an initial value when declared
Re-declaration	Can be redeclared within the same scope	Cannot be redeclared within the same block scope	Cannot be redeclared within the same block scope
Re-initialization	Can be reassigned	Can be reassigned	Cannot be reassigned
Scope	Function-scoped	Block-scoped	Block-scoped
Hoisted	Hoisted to the function/global scope, initialized with undefined	Hoisted to the block scope, not initialized	Hoisted to the block scope, not initialized
Introduced	Available in JavaScript since the beginning-1995	Introduced in ECMAScript 6 (ES6), also known as ECMAScript 2015	Introduced in ECMAScript 6 (ES6), also known as ECMAScript 2015

# JavaScript Datatypes

In JavaScript:

- **var**, **let**, and **const** are not data types themselves; they are keywords used for variable declaration.
- Data types in JavaScript are the types of values that variables can hold, such as **numbers**, **strings**, **booleans**, **objects**, **arrays**, etc.
- When you write something like **var a = 10;**, a can hold a number value (10 in this case), so we say that a is a variable holding a numeric literal.
- In summary, while in **C** and **C++**, we **explicitly specify the data type of a variable when declaring it**, in **JavaScript**, we declare variables using keywords like **var**, **let**, or **const**, and the data type of a variable is determined by the value it holds.

# JavaScript Data Types

## JavaScript has 8 Datatypes

1. String
2. Number
3. BigInt
4. Boolean
5. Undefined
6. Null
7. Symbol
8. Object

## The Object Datatype

The object data type can contain:

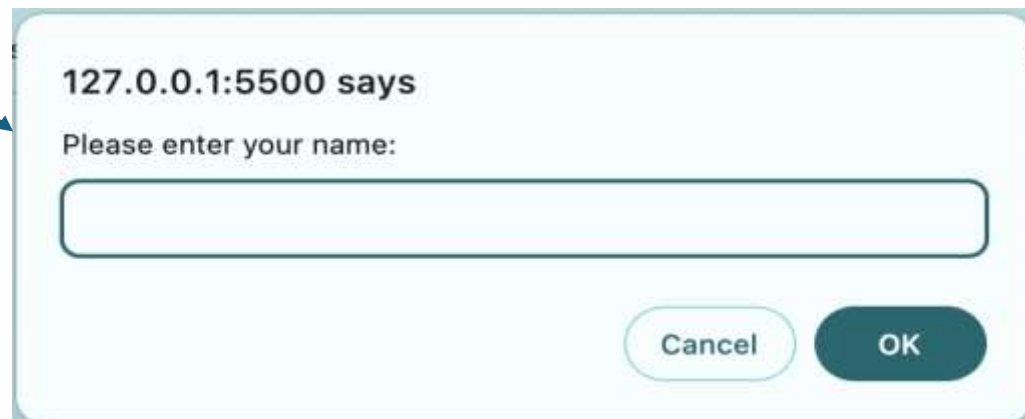
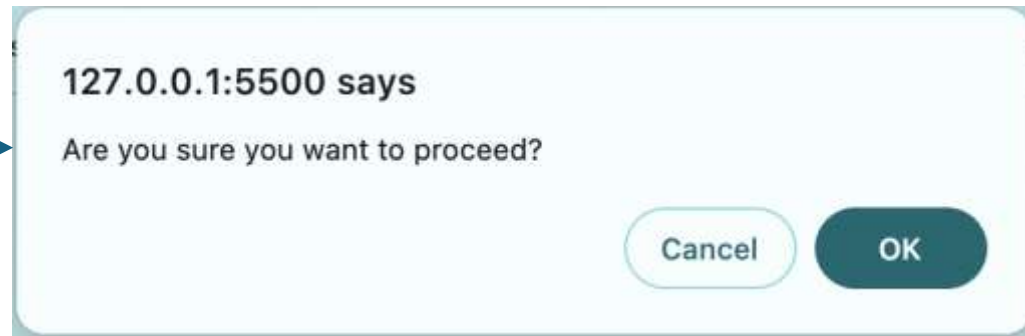
1. Primitive values
2. Other objects
3. Arrays
4. Functions
5. Dates
6. Symbols and more

## Pop-up boxes in JavaScript

1.Alert Box

2.Confirm Box

3.Prompt Box



# JavaScript Form Validation

Forms can be validated using HTML, and JavaScript.

**1.HTML Validation:** HTML5 introduced several built-in validation attributes for form inputs such as required, min, max, pattern, etc.

**2.JavaScript Validation:** You can use JavaScript to implement custom validation logic beyond what HTML provides. This allows you to perform complex validation tasks, such as validating the format of input data, checking for uniqueness, or interacting with external data sources for validation.

**3.CSS Role:** While CSS itself cannot perform form validation, it can be used to enhance the visual feedback of validation errors. For example, you can style invalid form inputs using the **:invalid** and **:valid** pseudo-classes to provide visual cues to users.



# Regular Expression

Regular expressions in JavaScript, also known as **regex** or **regexp**, are a sequence of characters that form a search pattern.

## 1. Creating Regular Expression

Regular expressions in JavaScript can be created using the **RegExp** constructor or by using a regex literal enclosed in forward slashes (/).

```
// Using RegExp constructor
```

```
const regex1 = new RegExp('pattern');
```

```
// Using regex literal
```

```
const regex2 = /pattern/;
```

## 2. Matching Patterns

Regular expressions are used with string methods like **test()** and **match()** to check if a pattern matches a string or to extract substrings that match the pattern.

**Example:**

```
const str = 'Hello, world!';  
const pattern = /hello/i; // Case-insensitive match  
console.log(pattern.test(str)); // Output: true  
console.log(str.match(pattern)); // Output: ["Hello"]
```

### 3. Modifiers

Regular expressions support modifiers that affect how a pattern is matched, such as:

1. i: Case-insensitive match
2. g: Global match (find all matches, not just the first)
3. m: Multiline match

#### Example:

```
const str = 'Apple, apple, APPLE';  
const pattern = /apple/ig; // Case-insensitive, global match  
console.log(str.match(pattern)); // Output: ["Apple", "apple", "APPLE"]
```

#### Replace

```
str.replace(/apple/i, "GEHU");
```

# Regular Expression Patterns

**Brackets** are used to find a range of characters:

Expression	Description
[abc]	Find any of the characters between the brackets
[0-9]	Find any of the digits between the brackets
(x y)	Find any of the alternatives separated with

**Metacharacters** are characters with a special meaning:

Metacharacter	Description
\d	Find a digit
\s	Find a whitespace character
\b	Find a match at the beginning of a word like this: \bWORD, or at the end of a word like this: WORD\b
\uxxxx	Find the Unicode character specified by the hexadecimal number xxxx

# Regular Expression Patterns

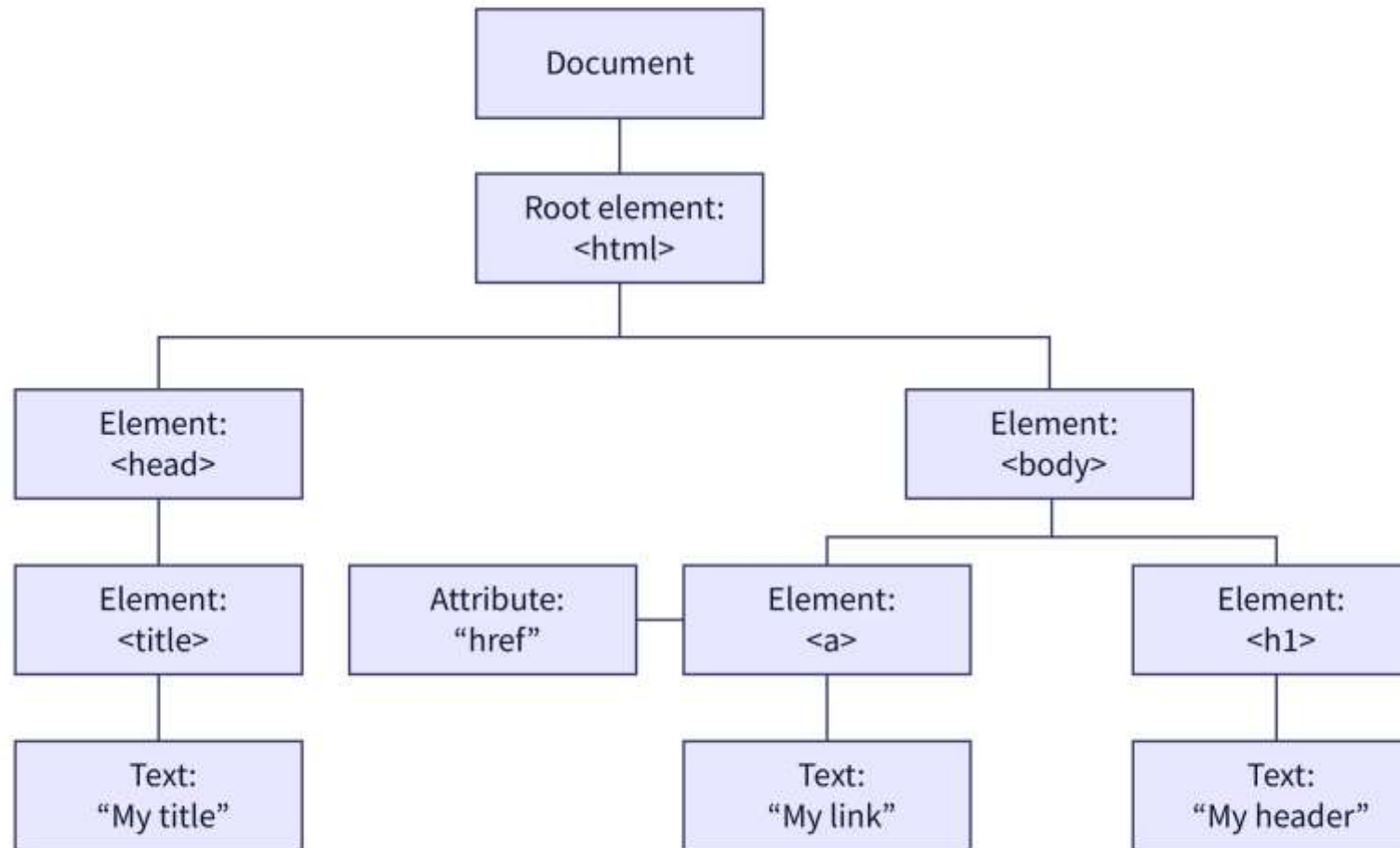
**Quantifiers** define quantities:

Quantifier	Description
n+	Matches any string that contains at least one n
n*	Matches any string that contains zero or more occurrences of n
n?	Matches any string that contains zero or one occurrences of n

# DOM – Document Object Model

- The DOM is a W3C (World Wide Web Consortium) standard.
- The DOM (Document Object Model) is a programming interface provided by web browsers that represents HTML, XML, and XHTML documents as a structured tree of objects.
- It defines the logical structure of documents and how a document is accessed and manipulated.
- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.

# HTML DOM



## Note:

**window** object represents the browser window and provides browser-related functionality such as navigating to URLs (**window.location**), managing browser history (**window.history**), setting timeouts (**window.setTimeout**), and more., the **document** object represents the HTML document loaded in that window and allows manipulation of its content

# Manipulation Using DOM

Below are some of the functionality that JavaScript can perform

- JavaScript Can Change HTML Content
- JavaScript Can Change HTML Attribute Values
- JavaScript Can Change HTML Styles (CSS)
- JavaScript Can Hide HTML Elements
- JavaScript Can Show HTML Elements
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page



# JQuery (\$)

**What is**

**jQuery?**

- Lightweight JavaScript Library.
- Simplifies JavaScript programming.
- **Write Less, Do More.**
- Browser independent.
- Used by Google, Microsoft, IBM, Netflix

**Features:**

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX etc



# Where to get JQuery?

There are two ways to include JQuery on your own website.

- [Download](#) the JQuery library.
    - Production Version – minify and compressed
    - Development Version – uncompressed and human readable
  - Include JQuery from a [CDN](#) say Google.
-

# Syntax

**\$**(Selector)**.**action();

**\$ Sign denotes  
jQuery function**

**Select the  
HTML element**

**. separator**

**Perform action on  
selected element**

## JavaScript

```
document.getElementById("id");
```

## jQuery

```
$("#id");
```

# Syntax

```
$(document).ready(function(){  
    // jQuery methods go here...  
});
```

OR

```
$(function(){  
    // jQuery methods go here...  
});
```

# Types of Selectors

Followings are the three major types of selectors in JQuery

- Element Selector
  - ID selector
  - Class Selector
-



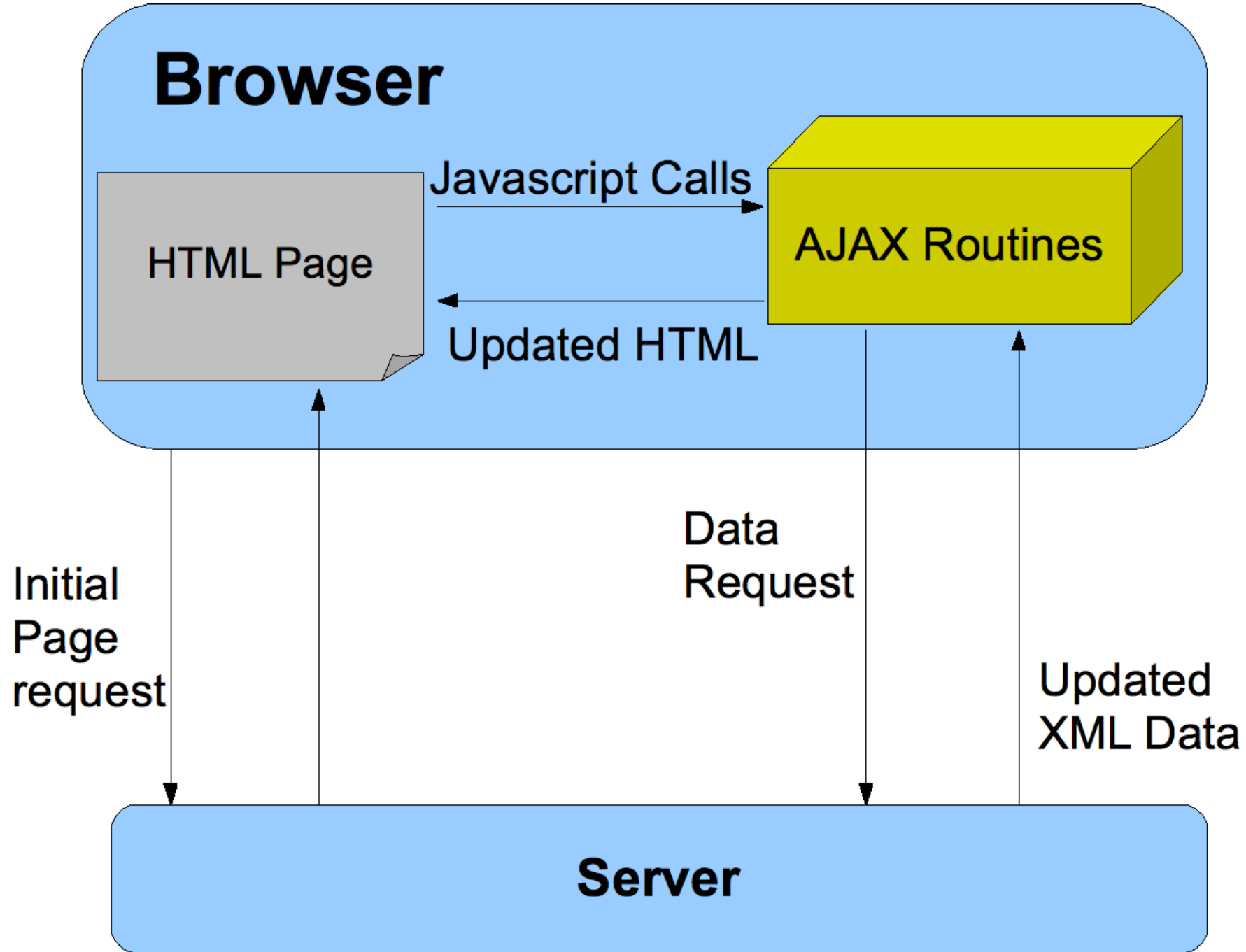
# AJAX

- **A**synchronous **J**avaScript **A**nd **X**ML.
  - AJAX is the art of exchanging data with a server and updating parts of a web page - without reloading the whole page.
  - Examples of applications using AJAX are Gmail, Google Maps, YouTube, and Facebook tabs.
  - With the jQuery AJAX methods, you can request text, HTML, XML, or JSON from a remote server using both HTTP Get and HTTP Post - And you can load the external data directly into the selected HTML elements of your web page!
-

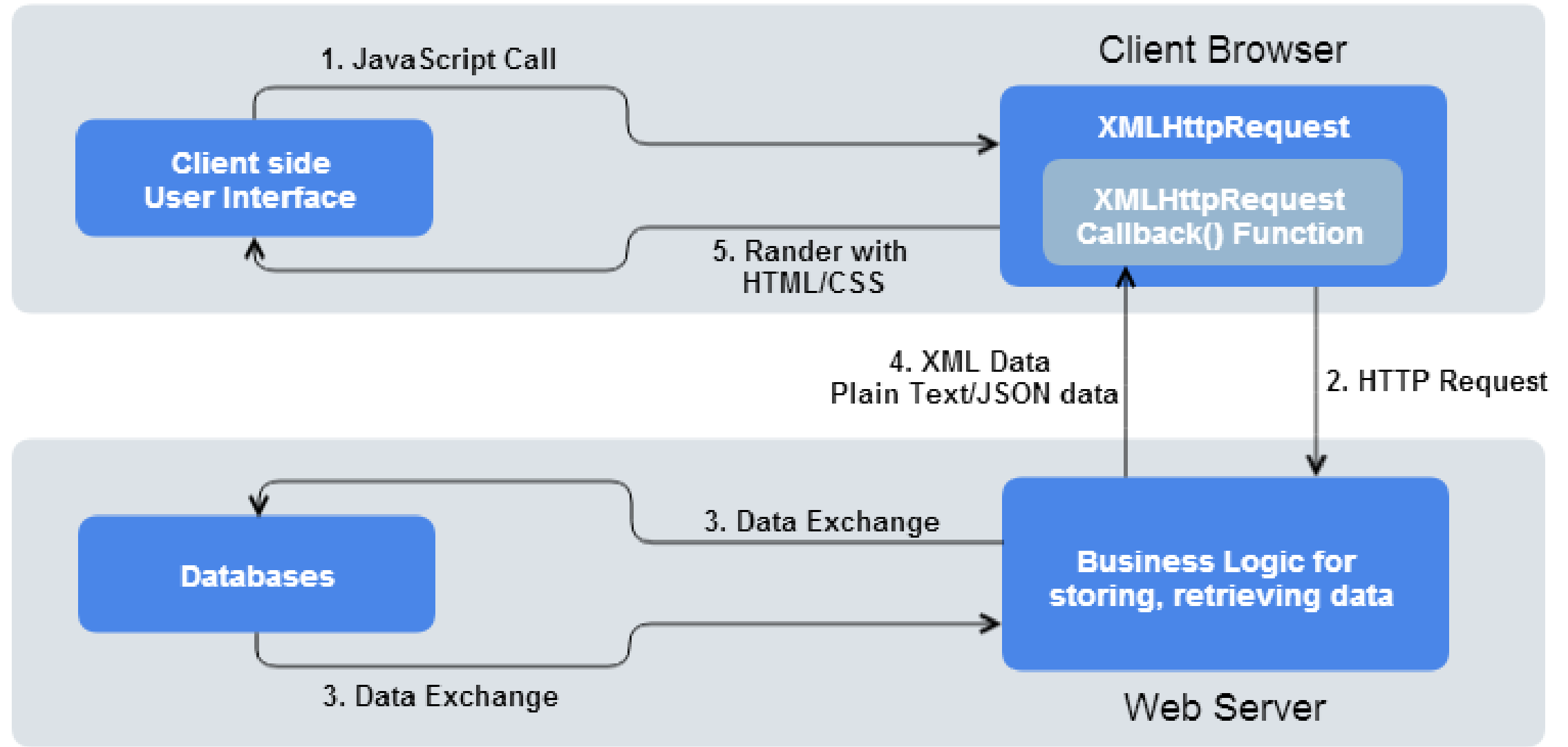


### **Note:**

Writing regular AJAX code can be complex due to variations in syntax across different browsers. This requires additional code to handle browser compatibility issues. However, jQuery simplifies AJAX implementation by providing a unified syntax, allowing developers to write AJAX functionality with just one line of code.







# load()

```
$(selector).load(URL, data, callback);
```

***Data** - send additional data to the server along with the request.*

The callback function can have different parameters:

- **responseTxt** - contains the resulting content if the call succeeds
- **statusTxt** - contains the status of the call
- **xhr** - contains the XMLHttpRequest object

# get()

```
$.get(URL, callback);
```

The callback function can have different parameters:

- **data** - holds the content of the page requested,
- **status** - holds the status of the request.

# post()

```
$.post(URL, data, callback);
```

*Data* - send additional data to the server along with the request.

The callback function can have different parameters:

- **data** - holds the content of the page requested,
- **status** - holds the status of the request.

# XML



XML stands for  
**eX**tensible  
**M**arkup  
**L**anguage



XML is a  
markup  
language much  
like HTML



XML was  
designed to  
store and  
transport data



XML was  
designed to be  
self-  
descriptive



XML is a W3C  
Recommendation

## The Difference Between XML and HTML

- XML was designed to carry data - with focus on what data is
- HTML was designed to display data - with focus on how data looks
- XML tags are not predefined like HTML tags.

# XML is Extensible

- We can add our own tags
- Most XML applications will work as expected even if new data is added (or removed).

```
<note>  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend!</body>  
</note>
```

```
<note>  
  <date>2015-09-01</date>  
  <hour>08:30</hour>  
  <to>Tove</to>  
  <from>Jani</from>  
  <body>Don't forget me this weekend!</body>  
</note>
```

# XML Syntax Rules

- XML Documents Must Have a Root Element
- The XML Prolog

```
<?xml version="1.0" encoding="UTF-8"?>
```

The XML prolog is optional. If it exists, it must come first in the document.

- All XML Elements Must Have a Closing Tag
- XML Tags are Case Sensitive
- XML Elements Must be Properly Nested
- XML Attribute Values Must Always be Quoted
- White-space is Preserved in XML

# XML Syntax Rules

## Entity References

Some characters have a special meaning in XML.

- If you place a character like "<" inside an XML element, it will generate an error because the parser interprets it as the start of a new element.

This will generate an XML error:

```
<message>salary < 1000</message>
```

To avoid this error, replace the "<" character with an **entity reference**:

```
<message>salary &lt; 1000</message>
```



# Well Formed vs Valid XML Documents

- ❖ An XML document with correct syntax is called "Well Formed".
- ❖ A "well formed" XML document is not the same as a "valid" XML document.
- ❖ A "valid" XML document must be well formed. In addition, it must conform to a document type definition.

There are two different document type definitions that can be used with XML:

- DTD - The original Document Type Definition
- XML Schema - An XML-based alternative to DTD

# DTD

A DTD defines the structure and the legal elements and attributes of an XML document.

Note.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

```
<!DOCTYPE note
[
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

# XML Schema

An XML Schema describes the structure of an XML document, just like a DTD.

```
<xs:element name="note">

  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

</xs:element>
```

# Why XML Schemas are More Powerful than DTD?

- XML Schemas are written in XML
- XML Schemas are extensible to additions
- XML Schemas support data types
- It is easier to convert data between different data types
- XML Schemas support namespaces
- You don't have to learn a new language
- You can use your XML editor to edit your Schema files
- You can use your XML parser to parse your Schema files
- You can manipulate your Schemas with the XML DOM
- You can transform your Schemas with XSLT