

# Unit-1

# Introduction-Web World

Web Technologies

# What is World Wide Web?

- World Wide Web, which is also known as a Web, is a collection of websites or web pages stored in web servers and connected to local computers through the internet.
- These websites contain text pages, digital images, audios, videos, etc.
- Users can access the content of these sites from any part of the world over the internet using their devices such as computers, laptops, cell phones, etc.
- The WWW, along with internet, enables the retrieval and display of text and media to your device.

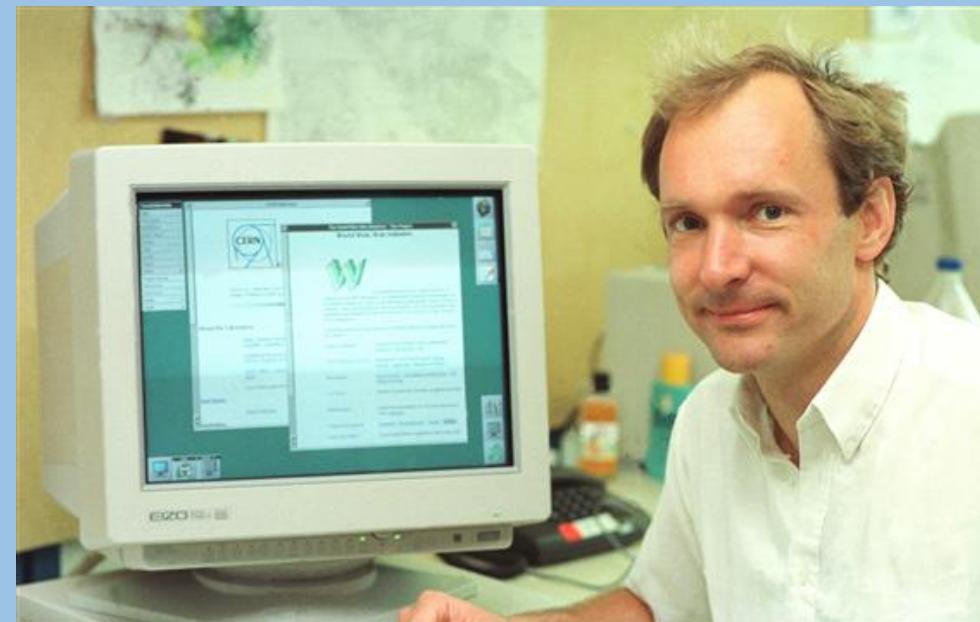
# WWW

- The World Wide Web (abbreviated WWW or the Web) is an information space where documents and other web resources are identified by Uniform Resource Locators (URLs), interlinked by hypertext links, and can be accessed via the Internet.



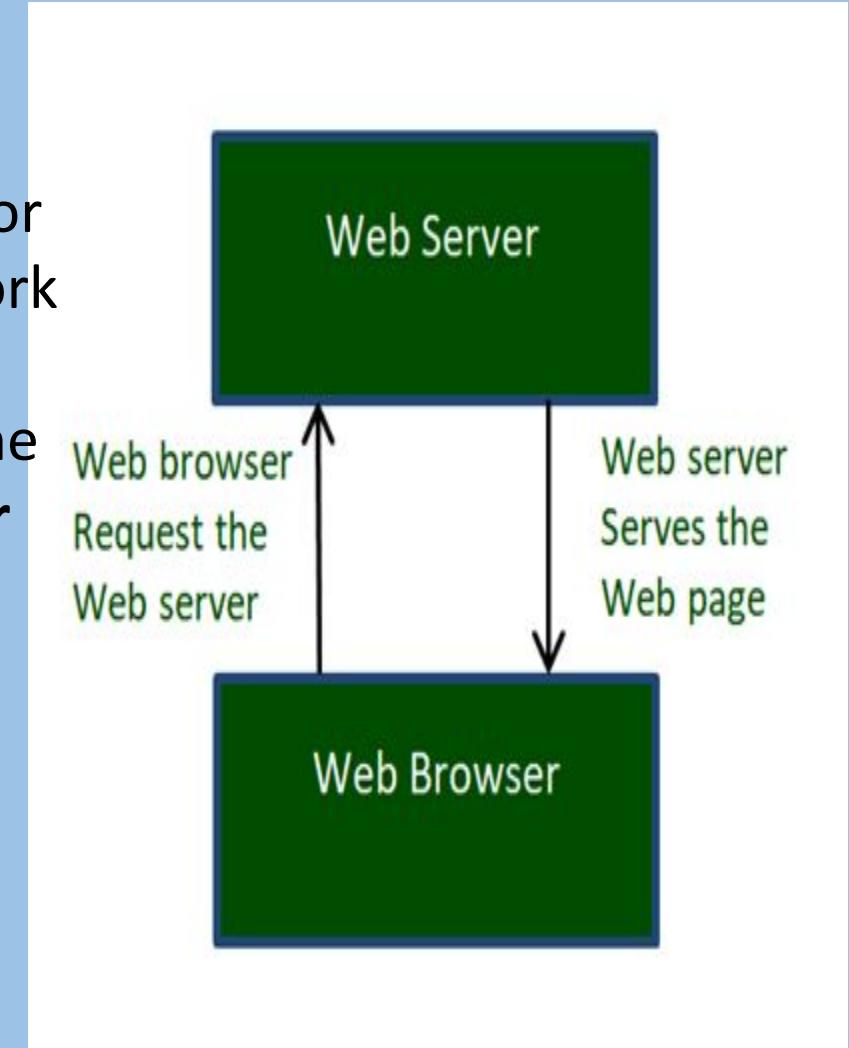
# Where the Web was born

- Tim Berners-Lee, a British scientist, invented the World Wide Web (WWW) in 1989, while working at CERN.
- The Web was originally conceived and developed to meet the demand for automated information-sharing between scientists in universities and institutes around the world.
- Tim focused on three main technologies that could make computers understand each other, HTML, URL, and HTTP.

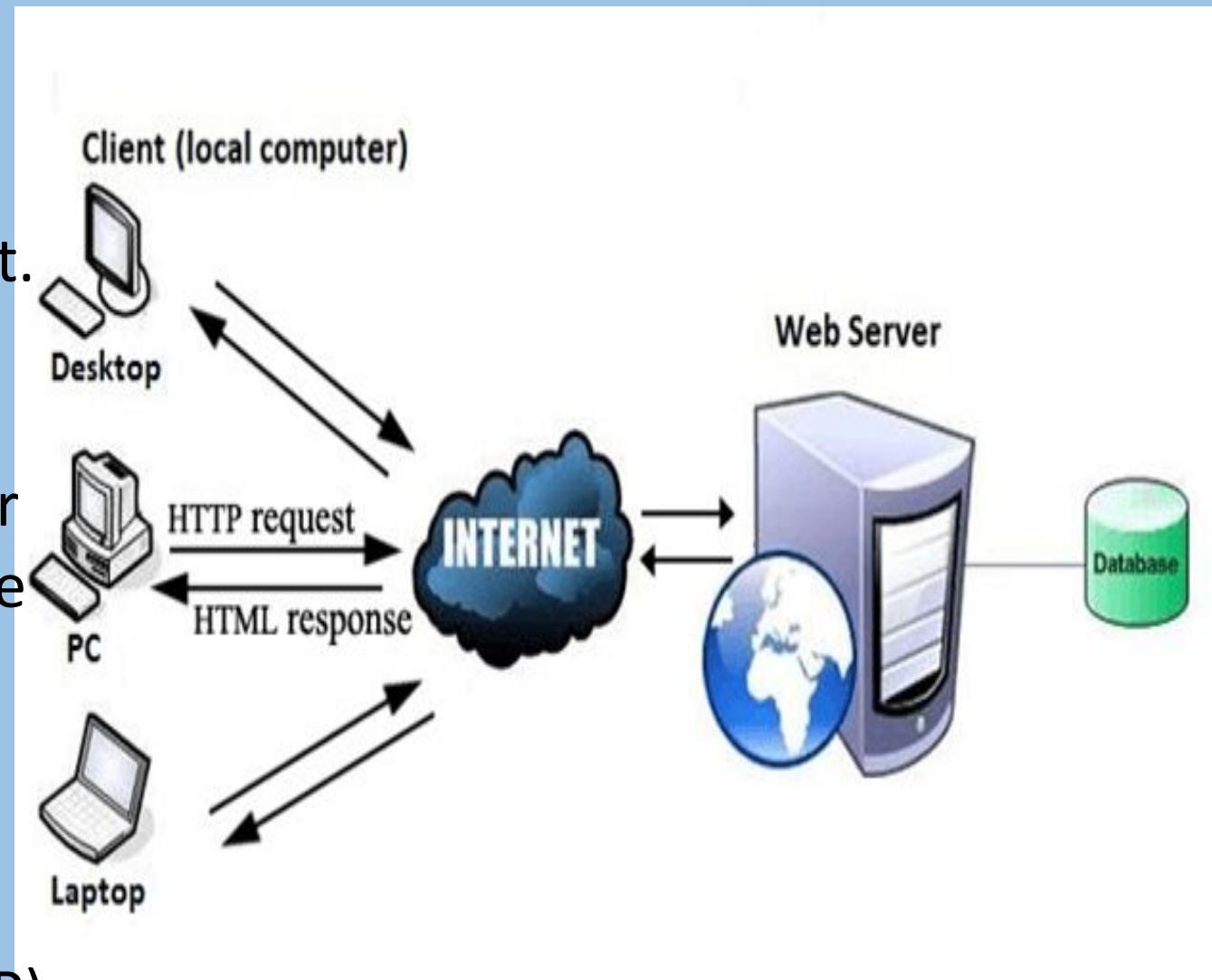


# How the World Wide Web Works?

- The **servers** store and transfer web pages or information to user's computers on the network when requested by the users
- A web server is a software program which serves the web pages requested by web users using a **browser**



- All the websites are stored in web servers.
- Just as someone lives on rent in a house, a website occupies a space in a server and remains stored in it.
- The server hosts the website whenever a user requests its WebPages, and the website owner has to pay the hosting price for the same.
- These technologies include Hypertext Markup Language (HTML)
- Hypertext Transfer Protocol (HTTP)
- Web browsers.



# Browsers:

- WWW Clients, or "Browser": The program you use to access the WWW is known as a browser because it "browses" the WWW and requests these hypertext documents.
- Browsers can be graphical, allows to see and hear the graphics and audio;
- All of these programs understand http and other Internet protocols such as FTP, gopher, mail, and news.
- Google Chrome, Microsoft Edge ,Mozilla Firefox , Opera, Opera Neon



Chrome



Safari



UC Browser



Firefox



IE



Microsoft edge

# Web browsers...

- Web browsers open the door to the display of various kinds of multimedia.
- Web browsers are now able to display many different kinds of information including text, graphics, video, audio and even virtual reality.
- Web browsers are able to support different multimedia formats through the use of plug-ins and add-ons that tell the browser how to display the different kinds of standard files that may be sent from Web servers.
- **Graphics** : graphic images that are used on the Web are GIF (Graphics Interchange Format) and JPEG (Joint Photographic Experts Group).
- **Audio**: Web audio is based on 8 bit sampling which is good for voice and low quality music. Common audio file formats are **.wav** and **.au**
- **Video**: Web video is often in MPEG (Motion Pictures Expert Group) format

# Uniform Resource Locators, or URLs

- Browser interprets the information in the URL in order to connect to the proper Internet server and to retrieve your desired document.
- Each time a click on a hyperlink in a WWW document instructs browser to find the URL that's embedded within the hyperlink.
- URLs are based on the following general format:
- **protocol://hostComputer/path:port**
- The elements in a URL: *Protocol://server's address/filename*
- **Hypertext protocol:** http://www.aucegypt.edu File Transfer
- **Protocol:** ftp://ftp.dartmouth.edu Telnet
- **Protocol:** telnet://pac.carl.org
- **News Protocol:** [news:alt.rock-n-roll.stones](#)
- **Path :** Any text after the host computer name is interpreted as the path name of the document to be retrieved. If no path is specified, the HTTP default is to search for a directory named **public\_html** and a file called **index.html**.
- **Optional Internet Port :** When you connect to a Web server from your Web browser you can perform several Web activities at once. For example, you can download a Web page, transfer a file using ftp and check Email, all from the same server. To keep things straight, the server uses different ports for different activities.
- a Port numbers appear as part of the host computer name and are preceded by a colon as in:
- **http://www.w3.org:80**

# What are Domains?

- Domains divide World Wide Web sites into categories based on the nature of their owner, and they form part of a site's address, or uniform resource locator (URL).
- Common top-level domains are:

.com—commercial enterprises	.mil—military site
org—organization site (non-profits, etc.)	int—organizations established by international treaty
.net—network	.biz—commercial and personal
.edu—educational site (universities, schools, etc.)	.info—commercial and personal
.gov—government organizations	.name—personal sites

- **MIME (Multi-Purpose Internet Mail Extensions)**:- MIME is an extension of the original Internet e-mail protocol.
- that lets people use the protocol to exchange different kinds of data files on the Internet: audio, video, images, application programs, and other kinds, as well as the ASCII text handled in the original protocol, the Simple Mail Transport Protocol (SMTP).

# Hypertext Transport Protocol and HTTPS:

- HTTP is the underlying protocol used by the World Wide Web and this protocol defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands.
- **HTTPS:** A similar abbreviation, HTTPS means Hyper Text Transfer Protocol Secure. Basically, it is the secure version of HTTP.
- Communications between the browser and website are encrypted by Transport Layer Security (TLS), or its predecessor, Secure Sockets Layer (SSL).

# HTML

- HTML - a markup language To describe the general form and layout of documents.
- Markup languages use sets of markup tags to characterize text elements within a document, which gives instructions to the web browsers on how the document should appear.
- HTML is not a programming language - it cannot be used to describe computations.
- An HTML document is a mix of **content** and **controls**
- **Controls** are tags and their attributes.
- **Tags** often delimit content and specify something about how the content should be arranged in the document.  
For example, `<p>Write a paragraph here </p>` is an element.
- **Attributes** provide additional information about the content of a tag
- For example, `<img src = "redhead.jpg"/><font color = "Red"/>`

# W3C

- In 1996, the World Wide Web Consortium (W3C) became the authority to maintain the HTML specifications.
- HTML also became an international standard (ISO) in 2000. HTML5 is the latest version of HTML.
- HTML5 provides a faster and more robust approach to web development.
- Apart from text, hypertext may contain tables, lists, forms, images, and other presentational elements.
- It is an easy-to-use and flexible format to share information over the Internet.

# What Is Web 1.0?

- Basically, this first version of the Web consisted of a few people creating web pages and content and web pages for a large group of readers, allowing them to access facts, information, and content from the sources.
- Here are a few characteristics found in Web 1.0:
- It's made up of static pages connected to a system via hyperlinks
- It has HTML 3.2 elements like frames and tables
- HTML forms get sent through e-mail
- The content comes from the server's filesystem, not a relational database management system
- It features GIF buttons and graphics

# What Is Web 2.0?

- If Web 1.0 was made up of a small number of people generating content for a larger audience, then Web 2.0 is many people creating even more content for a growing audience. Web 1.0 focused on reading; Web 2.0 focused on participating and contributing
- Here's Web 2.0 characteristics:
- It offers free information sorting, allowing users to retrieve and classify data collectively
- It contains dynamic content that responds to the user's input
- It employs Developed Application Programming Interfaces (API)
- It encourages self-usage and allows forms of interaction like:
  - Podcasting
  - Social media
  - Tagging
  - Blogging
  - Commenting
  - Curating with RSS
  - Social networking
  - Web content voting
- It's used by society at large and not limited to specific communities.

# What Is Web 3.0?

- Web 3.0 currently available today, which is also referred to as Web3
- Web 1.0 is the "read-only Web," Web 2.0 is the "participative social Web," and Web 3.0 is the "read, write, execute Web."
- Web 3.0 characteristics:
- where the web technology evolves into a tool that lets users create, share, and connect content via search and analysis.
- It incorporates Artificial Intelligence and Machine Learning. If these concepts are combined with [Natural Language Processing \(NLP\)](#), the result is a computer that uses Web 3.0 to become smarter and more responsive to user needs.
- It presents the connectivity of multiple devices and applications through the [Internet of Things \(IoT\)](#). Semantic metadata makes this process possible.
- It uses 3-D graphics. In fact, we already see this in computer games, virtual tours, and e-commerce.
- Decentralized finance, [Blockchain](#) games, Decentralized autonomous organizations.
- Web 3.0 ultimately lets users interact, exchange information, and securely conduct financial transactions without a centralized authority or coordinator.

# HTML

# HTML Element Syntax

- An HTML element is an individual component of an HTML document.
- It represents semantics, or meaning.
- For example, the title element represents the title of the document.



- All elements don't require the end tag or closing tag to be present. These are referred as *empty elements*, *self-closing elements* or *void elements*.

# Case Insensitivity in HTML Tags and Attributes

- In HTML, tag and attribute names are not case-sensitive (but most attribute values are case-sensitive).
- It means the tag `<P>`, and the tag `<p>` defines the same thing in HTML which is a paragraph.
- `<p>This is a paragraph.</p> <P>This is also a valid paragraph.</P>`

# Empty HTML Elements

- Empty elements (also called self-closing or void elements) are not container tags — that means, you can not write `<hr>some content</hr>` or `<br>some content</br>`.
- A typical example of an empty element, is the `<br>` element, which represents a line break. Some other common empty elements are `<img>`, `<input>`, `<link>`, `<meta>`, `<hr>`, etc.

- **Example**
- <p>This paragraph contains <br> a line break.</p>
-  <input type="text" name="username">
- Placing one element inside another is called nesting. A nested element, also called a child element, can be a parent element too if other elements are nested within it.
- HTML tags should be nested in correct order. They must be closed in the inverse order of how they are defined, that means the last tag opened must be closed first.

# Writing Comments in HTML

- Comments are usually added with the purpose of making the source code easier to understand.
- It may help other developer (or you in the future when you edit the source code) to understand what you were trying to do with the HTML.
- Comments are not displayed in the browser.  
An HTML comment begins with
- <!--, and ends with -->, as shown in the example below:
- <!-- This is an HTML comment -->
- <!-- This is a multi-line HTML comment that spans across more than one line -->
- <p>This is a normal piece of text.</p>

# HTML Elements Types

- Elements can be placed in two distinct groups: block level and inline level elements.
- The most commonly used block-level elements are `<div>`, `<p>`, `<h1>` through `<h6>`, `<form>`, `<ol>`, `<ul>`, `<li>`, and so on.
- Whereas, the commonly used inline-level elements are `<img>`, `<a>`, `<span>`, `<strong>`, `<b>`, `<em>`, `<i>`, `<code>`, `<input>`, `<button>`, etc.
- **Note:** The block-level elements should not be placed within inline-level elements. For example, the `<p>` element should not be placed inside the `<b>` element.

# What are Attributes

- Attributes define additional characteristics or properties of the element such as width and height of an image.
- Attributes are always specified in the start tag (or opening tag) and usually consists of name/value pairs like name="value".
- Attribute values should always be enclosed in quotation marks.
- Also, some attributes are required for certain elements. For instance, an **<img>** tag must contain a **src** and **alt** attributes. Let's take a look at some examples of the attributes usages:
- **Example**
- ``
- `<a href="https://www.google.com/" title="Search Engine">Google</a>`
- `<abbr title="Hyper Text Markup Language">HTML</abbr> <input type="text" value="John Doe">`
- **Note:** Both single and double quotes can be used to quote attribute values. However, double quotes are most common. In situations where the attribute value itself contains double quotes it is necessary to wrap the value in single quotes, e.g., `value='John "Williams" Jr.'`

- There are several attributes in HTML5 that do not consist of name/value pairs but consists of just name. Such attributes are called Boolean attributes.
- **Examples:** some commonly used Boolean attributes are checked, disabled, readonly, required, etc.
- <input type="email" required>
- <input type="submit" value="Submit" disabled>
- <input type="checkbox" checked>
- <input type="text" value="Read only text" readonly>

# HTML5 document.

- You can use the following markup as a template to create a new HTML5 document.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title><!-- Insert your title here --></title>
</head>
<body>
    <!-- Insert your content here -->
</body>
</html>
```

- DOCTYPE for short, is an instruction to the web browser about the version of markup language in which a web page is written.

# The HTML head Element

- The `<head>` elements, which provide extra information about the document (metadata), or reference to other resources that are required for the document to display or behave correctly in a web browser.
- The head elements collectively describes the properties of the document such as title, provide meta information like character set, instruct the browser where to find the style sheets or scripts that allows you to extend the HTML document in a highly active and interactive ways.
- The HTML elements that can be used inside the `<head>` element are: `<title>`, `<base>`, `<link>`, `<style>`, `<meta>`, `<script>` and the `<noscript>` element.

- The <title> element defines the title of the document.
- The HTML <base> element is used to define a base URL for all relative links contained in the document. A common use of link element is to link to external style sheets.
- The <style> element is used to define embedded style information for an HTML document.

# HTML Meta

- The <meta> tags are typically used to provide structured metadata such as a document's keywords, description, author name, character encoding, and other metadata.
- Any number of meta tags can be placed inside the head section of an HTML.
- Metadata will not be displayed on the web page, but will be machine parsable, and can be used by the browsers, search engines like Google or other web services.
- You can also use the meta tag to clearly define who is the author or creator of the web page.
- **Keywords and Description for Search Engines:** Some search engines use metadata especially keywords and descriptions to index web pages;

```
<head>
  <title>Declaring Character Encoding</title>
  <meta charset="utf-8">
  <meta name="author" content="Alexander Howick">
  <meta name="keywords" content="HTML, CSS, javaScript">
  <meta name="description" content="Easy to understand tutorials and references on HTML, CSS, javaScript and more...">
</head>
```

# Configuring the Viewport for Mobile Devices

- You can use the viewport meta tag to display the web pages correctly on mobile devices.
- The following demonstration shows two web pages — one *with viewport meta tag* and other *without viewport meta tag* set.

```
<head>
  <title>Configuring the Viewport</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
```



# HTML5 Tags

Tag	Description
<a href="#"><u>&lt;a&gt;</u></a>	Defines a hyperlink.
<a href="#"><u>&lt;abbr&gt;</u></a>	Defines an abbreviated form of a longer word or phrase.
<a href="#"><u>&lt;address&gt;</u></a>	Specifies the author's contact information.
<a href="#"><u>&lt;area&gt;</u></a>	Defines a specific area within an image map.
<a href="#"><u>&lt;article&gt;</u></a>	Defines an article.
<a href="#"><u>&lt;aside&gt;</u></a>	Defines some content loosely related to the page content.
<a href="#"><u>&lt;audio&gt;</u></a>	Embeds a sound, or an audio stream in an HTML document.
<a href="#"><u>&lt;button&gt;</u></a>	Creates a clickable button.
<a href="#"><u>&lt;canvas&gt;</u></a>	Defines a region in the document, which can be used to draw graphics on the fly via scripting (usually JavaScript).
<a href="#"><u>&lt;caption&gt;</u></a>	Defines the caption or title of the table.
<a href="#"><u>&lt;cite&gt;</u></a>	Indicates a citation or reference to another source.
<a href="#"><u>&lt;code&gt;</u></a>	Specifies text as computer code.
<a href="#"><u>&lt;col&gt;</u></a>	Defines attribute values for one or more columns in a table.

# 3.HTML –links,images

# HTML Links

- A link or hyperlink is a connection from one web resource to another. Links allow users to move seamlessly from one page to another, on any server anywhere in the world.
- A link has two ends, called anchors. The link starts at the source anchor and points to the destination anchor, which may be any web resource, for example, an image, an audio or video clip, a PDF file, an HTML document or an element within the document itself, and so on.
- By default, links will appear as follow in most of the browsers:
  - An unvisited link is underlined and blue.
  - A visited link is underlined and purple.
  - An active link is underlined and red.

- **HTML Link Syntax**
- `<a href="url">Link text</a>`
- **Example:** `<a href="https://www.google.com/">Google Search</a>`
- `<a href="images/kites.jpg">`
- ` </a>`
- The **href** attribute specifies the target of the link. Its value can be an absolute or relative URL.
- An absolute URL is the URL that includes every part of the URL format, such as protocol, host name, and path of the document, e.g., `https://www.google.com/`, `https://www.example.com/form.php`, etc.
- While, relative URLs are page-relative paths, e.g., `contact.html`, `images/smiley.png`, and so on. A relative URL never includes the `http://` or `https://` prefix.

# Setting the Targets for Links

- The target attribute tells the browser where to open the linked document. There are four defined targets, and each target name starts with an underscore(\_) character:
  - `_blank` — Opens the linked document in a new window or tab.
  - `_parent` — Opens the linked document in the parent window.
  - `_self` — Opens the linked document in the same window or tab as the source document. This is the default, hence it is not necessary to explicitly specify this value.
  - `_top` — Opens the linked document in the full browser window.

- **Example**

- `<a href="/about-us.php" target="_top">About Us</a>`
- `<a href="https://www.google.com/" target="_blank">Google</a>`
- `<a href="images/sky.jpg" target="_parent">`
- ``
- `</a>`

- You can even jump to a section of another web page by specifying the URL of that page along with the anchor (i.e. #elementId) in the href attribute, for example,
- [Go to TopicA](mypage.html#topicA).
- **Creating Download Links**
  - [Download Zip file](downloads/test.zip)
  - [Download PDF file](downloads/masters.pdf)
  - [Download Image file](downloads/sample.jpg)

# HTML Images

- Images enhance visual appearance of the web pages by making them more interesting and colorful.
- The `<img>` tag is used to insert images in the HTML documents. It is an empty element and contains attributes only. The syntax of the `<img>` tag can be given with
  - ``
- **Example**
  - ``
  - ``
  - ``
- The required `alt` attribute provides alternative text description for an image if a user for some reason cannot able to view it because of slow connection, image is not available at the specified URL, or if the user uses a screen reader or non-graphical browser.

# Setting the Width and Height of an Image

- 
- 
- It's a good practice to specify both the width and height attributes for an image, so that browser can allocate that much of space for the image before it is downloaded.
- Otherwise, image loading may cause distortion or flicker in your website layout.

# Using the HTML5 Picture Element

- Sometimes, scaling an image up or down to fit different devices (or screen sizes) doesn't work as expected.
- Also, reducing the image dimension using the width and height attribute or property doesn't reduce the original file size.
- To address these problems HTML5 has introduced the `<picture>` tag that allows you to define multiple versions of an image to target different types of devices.
- The `<picture>` element contains zero or more `<source>` elements, each referring to different image source, and one `<img>` element at the end.
- Also each `<source>` element has the media attribute which specifies a media condition (similar to the media query) that is used by the browser to determine when a particular source should be used.

# Example

- <picture>
- <source media="(min-width: 1000px)" srcset="logo-large.png">
- <source media="(max-width: 500px)" srcset="logo-small.png">
- 
- </picture>
- The browser will evaluate each child <source> element and choose the best match among them; if no matches are found, the URL of the <img> element's src attribute is used.
- Also, the <img> tag must be specified as the last child of the <picture> element.

# Working with Image Maps

- An image map allows you to define hotspots on an image that acts just like a [hyperlink](#).
- The basic idea behind creating image map is to provide an easy way of linking various parts of an image without dividing it into separate image files. For example, a map of the world may have each country hyperlinked to further information about that country.
- **Example**
  - 
  - <map name="objects">
  - <area shape="circle" coords="137,231,71" href="clock.html" alt="Clock">
  - <area shape="poly" coords="363,146,273,302,452,300" href="sign.html" alt="Sign">
  - <area shape="rect" coords="520,160,641,302" href="book.html" alt="Book">
  - </map>
- The image map should not be used for website navigation. They are not search engine friendly. A valid use of an image map is with a geographical map.





# Html tables and lists

WEB TECHNOLOGIES

K. KRISHNA PRIYA

# HTML Tables

- ▶ HTML table allows you to arrange data into rows and columns. They are commonly used to display tabular data like product listings, customer's details, financial reports, and so on.
- ▶ You can create a table using the `<table>` element.
- ▶ you can use the `<tr>` elements to create rows,
- ▶ create columns inside a row you can use the `<td>` elements.
- ▶ You can also define a cell as a header for a group of table cells using the `<th>` element.

> <th>Age</th>

## Example

```
<tr>
    <td>1</td>
    <td>Peter Parker</td>
    <td>16</td>
</tr>
<tr>
    <td>2</td>
    <td>Clark Kent</td>
    <td>34</td>
</tr>
</table>
```

# Borders to table

- Tables do not have any borders by default. You can use the CSS border property to add borders to the tables.
- Also, table cells are sized just large enough to fit the contents by default.
- To add more space around the content in the table cells you can use the CSS padding property.

## ► Example

```
► table, th, td {  
►     border: 1px solid black;  
► }  
► th, td {  
►     padding: 10px;  
► }
```



The image shows two side-by-side tables illustrating border styles. Both tables have three columns: ID, Name, and Age. The first table, labeled "Separate Border (Default)", shows a standard grid where each cell has a thin black border. The second table, labeled "Collapse Border", shows a similar grid but with no visible borders between the individual cells; instead, the entire row and column structures are defined by thicker borders at the intersections.

ID	Name	Age
1	John Carter	30
2	Harry Potter	11
3	Peter Parker	21

**Separate Border (Default)**

ID	Name	Age
1	John Carter	30
2	Harry Potter	11
3	Peter Parker	21

**Collapse Border**

# Adjusting Space inside Tables: Padding

- To add more space between the table cell contents and the cell borders, you can simply use the CSS padding property.

```
> table {  
>   border-collapse: collapse;  
> }  
> table, th, td {  
>   border: 1px solid black;  
> }  
> th, td {  
>   padding: 15px;  
> }
```

ID	First Name	Last Name	Email
1	John	Carter	johncarter@mail.com
2	Peter	Parker	peterparker@mail.com
3	John	Rambo	johnrambo@mail.com
4	Harry	Potter	harrypotter@mail.com

# border-spacing and text-align

- You can also adjust the spacing between the borders of the cells using the CSS border-spacing property, if the borders of your table are separated (which is default).
- Also, text inside the <th> elements are displayed in bold font, aligned horizontally center in the cell by default. To change the default alignment you can use the CSS text-align property.

```
► table {  
►     border-spacing: 10px;  
► }  
► table, th, td {  
►     border: 1px solid black;  
► }  
► th, td {  
►     padding: 10px;  
►     text-align: left;  
► }
```

ID	First Name	Last Name	Email
1	John	Carter	johncarter@mail.com
2	Peter	Parker	peterparker@mail.com
3	John	Rambo	johnrambo@mail.com
4	Harry	Potter	harrypotter@mail.com

text-align: [left](#) | [right](#) | [center](#) | [justify](#) | [initial](#) | [inherit](#)

# Spanning Multiple Rows and Columns

- ▶ Spanning allow you to extend table rows and columns across multiple other rows and columns.

```
<table>
  <tr>
    <th>Name</th>
    <th colspan="2">Phone</th>
  </tr>
  <tr>
    <td>John Carter</td>
    <td>5550192</td>
    <td>5550152</td>
  </tr>
</table>
```

```
<table>
  <tr>
    <th>Name:</th>
    <td>John Carter</td>
  </tr>
  <tr>
    <th rowspan="2">Phone:</th>
    <td>55577854</td>
  </tr>
  <tr>
    <td>55577855</td>
  </tr>
</table>
```

# Adding Captions to Tables

- ▶ You can specify a caption (or title) for your tables using the `<caption>` element.
- ▶ The `<caption>` element must be placed directly after the opening `<table>` tag. By default, caption appears at the top of the table
- ▶ `<caption>Users Info</caption>`

# Defining a Table Header, Body, and Footer

- ▶ HTML provides a series of tags `<thead>`, `<tbody>`, and `<tfoot>` that helps you to create more structured table, by defining header, body and footer regions, respectively.
- ▶ `<table>`
- ▶   `<thead>`
- ▶    `<tr>`
- ▶      `<th>Items</th>`
- ▶      `<th>Expenditure</th>`
- ▶    `</tr>`
- ▶   `</thead>`
- ▶   `<tbody>`
- ▶    `<tr>`
- ▶     `<td>Stationary</td>`
- ▶     `<td>2,000</td>`
- ▶    `</tr>`
- ▶   `</tbody>`

# Working with HTML Lists

- HTML lists are used to present list of information in well formed and semantic way. There are three different types of list in HTML and each one has a specific purpose and meaning.
- **Unordered list** — Used to create a list of related items, in no particular order.
- **Ordered list** — Used to create a list of related items, in a specific order.
- **Description list** — Used to create a list of terms and their descriptions.
- Inside a list item you can put text, images, links, line breaks, etc. You can also place an entire list inside a list item to create the nested list.

# HTML Unordered Lists

- An unordered list created using the `<ul>` element, and each list item starts with the `<li>` element.
- You can also change the bullet type in your unordered list using the CSS **list-style-type** property.
- **list-style-type:** disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-latin | upper-latin | armenian | georgian | lower-alpha | upper-alpha | none | initial | inherit
- **list-style-image:** url | none | initial | inherit
- `ul {`
- `list-style-image: url("images/arrow.png");`
- `}`

## Example

```
<ul>
  <li>Chocolate Cake</li>
  <li>Black Forest
  Cake</li>
  <li>Pineapple Cake</li>
```

# HTML Ordered Lists

- ▶ An ordered list created using the `<ol>` element, and each list item starts with the `<li>` element. Ordered lists are used when the order of the list's items is important.
- ▶ The numbering of items in an ordered list typically starts with 1. However, if you want to change that you can use the `start` attribute
- ▶ 

```
ol {  
    list-style-type: upper-roman;  
}  
▶ Note: You can also use the type attribute to change  
the numbering type e.g. type="I". However, you should  
avoid this attribute, use the CSS list-style-type property  
instead.
```

## Example

```
<ol start="10">  
    <li>Mix ingredients</li>  
    <li>Bake in oven for an  
hour</li>  
    <li>Allow to stand for ten  
minutes</li>  
</ol>
```

# HTML Description Lists

- The description list is created using `<dl>` element.
- The `<dl>` element is used in conjunction with the `<dt>` element which specify a term, and the `<dd>` element which specify the term's definition.
- Browsers usually render the definition lists by placing the terms and definitions in separate lines,

## Example

```
<dl>
  <dt>Bread</dt>
  <dd>A baked food made of flour.</dd>
  <dt>Coffee</dt>
  <dd>A drink made from roasted coffee beans.</dd>
</dl>
```

# HTML Forms

---

WEB TECHNOLOGIES

# What is HTML Form

---

HTML Forms are required to collect different kinds of user inputs, such as contact details like name, email address, phone numbers, or details like credit card information, etc.

```
<form>  
  <label>Username: <input type="text"></label>  
  <label>Password: <input type="password"></label>  
  <input type="submit" value="Submit">  
</form>
```

# Input Element

This is the most commonly used element within HTML forms.

---

It allows you to specify various types of user input fields, depending on the type attribute.

An input element can be of type text field, password field, checkbox, radio button, submit button, reset button, file select box, as well as several new input types introduced in HTML5.

# Text Fields

---

Text fields are one line areas that allow the user to input text.

```
<form>  
  <label for="username">Username:</label>  
  <input type="text" name="username" id="username">  
</form>
```

The `<label>` tag is used to define the labels for `<input>` elements. If you want your user to enter several lines you should use a `<textarea>` instead.

# Password Field

---

Password fields are similar to text fields. The only difference is; characters in a password field are masked, i.e. they are shown as asterisks or dots.

```
<form>  
  <label for="user-pwd">Password:</label>  
  <input type="password" name="user-password" id="user-pwd">  
</form>
```

# Radio Buttons

---

Radio buttons are used to let the user select exactly one option from a pre-defined set of options. It is created using an `<input>` element whose type attribute has a value of radio.

```
<form>

  <input type="radio" name="gender" id="male">
  <label for="male">Male</label>

  <input type="radio" name="gender" id="female">
  <label for="female">Female</label>

</form>
```

# Checkboxes

---

Checkboxes allows the user to select one or more option from a pre-defined set of options. It is created using an `<input>` element whose type attribute has a value of checkbox.

```
<form>

  <input type="checkbox" name="sports" id="Football">
  <label for="Football">Football</label>

  <input type="checkbox" name="sports" id="cricket">
  <label for="cricket">Cricket</label>

  <input type="checkbox" name="sports" id="baseball">
  <label for="baseball">Baseball</label>

</form>
```

If you want to make a radio button or checkbox selected by default, you can add the attribute `checked` to the `input` element, like `<input type="checkbox" checked>`.

# File Select box

---

The file fields allow a user to browse for a local file and send it as an attachment with the form data.

Web browsers such as Google Chrome and Firefox render a file select input field with a Browse button that enables the user to navigate the local hard drive and select a file.

```
<form>  
  <label for="file-select">Upload:</label>  
  <input type="file" name="upload" id="file-select">  
</form>
```

# Textarea

---

Textarea is a multiple-line text input control that allows a user to enter more than one line of text.

Multi-line text input controls are created using an <textarea> element.

```
<form>  
  <label for="address">Address:</label>  
  <textarea rows="3" cols="30" name="address" id="address"></textarea>  
</form>
```

# Select Boxes

---

A select box is a dropdown list of options that allows user to select one or more option from a pull-down list of options.

Select box is created using the `<select>` element and `<option>` element.

```
<form>
  <label for="city">City:</label>
  <select name="city" id="city">
    <option value="sydney">Sydney</option>
    <option value="melbourne">Melbourne</option>
    <option value="cromwell">Cromwell</option>
  </select>
</form>
```

# Submit and Reset Buttons

---

A submit button is used to send the form data to a web server. When submit button is clicked the form data is sent to the file specified in the form's action attribute to process the submitted data.

A reset button resets all the forms control to default values.

```
<form action="action.php" method="post">  
    <label for="first-name">First Name:</label>  
    <input type="text" name="first-name" id="first-name">  
    <input type="submit" value="Submit">  
    <input type="reset" value="Reset">  
</form>
```

# <button> element

---

You can also create buttons using the <button> element. Buttons created with the <button> element function just like buttons created with the input element, but they offer richer rendering possibilities by allowing the embedding of other HTML elements.

# Grouping Form Controls

---

You also group logically related controls and labels within a web form using the `<legend>` element.

The `<fieldset>` tag is used to group related elements in a form. The `<fieldset>` tag draws a box around the related elements.

Grouping form controls into categories makes it easier for users to locate a control which makes the form more user-friendly.

```
<form>
  <fieldset>
    <legend>Contact Details</legend>
    <label>Email Address: <input type="email" name="email"></label>
    <label>Phone Number: <input type="text" name="phone"></label>
  </fieldset>
</form>
```

---

# HTML iframes

Web Technologies

# What is iframe

- An iframe or inline frame is used to display external objects including other web pages within a web page.
- An iframe pretty much acts like a mini web browser within a web browser.
- Also, the content inside an iframe exists entirely independent from the surrounding elements.
- The basic syntax for adding an iframe to a web page can be given with:
- <iframe src="*URL*"></iframe>
- The URL specified in the src attribute points to the location of an external object or a web page.
- <iframe src="hello.html"></iframe>

- The “`iframe`” tag defines a rectangular region within the document in which the browser can display a separate document, including scrollbars and borders.

# Setting Width and Height of an iFrame

- The height and width attributes are used to specify the height and width of the iframe.
- `<iframe src="hello.html" width="400" height="200"></iframe>`
- You can also use CSS to set the width and height of an iframe, as shown here:
- `<iframe src="hello.html" style="width: 400px; height: 200px;"></iframe>`
- The width and height attribute values are specified in pixels by default, but you can also set these values in percentage, such as 50%, 100% and so on. The default width of an iframe is 300 pixels, whereas the default height is 150 pixels.

# Removing Default Frameborder

- The iframe has a border around it by default. However, if you want to modify or remove the iframe borders, the best way is to use the CSS border property.
- `<iframe src="hello.html" style="border: none;"></iframe>`
- Similarly, you can use the border property to add the borders of your choice to an iframe. The following example will render the iframe with 2 pixels blue border.
- `<iframe src="hello.html" style="border: 2px solid blue;"></iframe>`

# Using an iFrame as Link Target

- An iframe can also be used as a target for the hyperlinks.
- An iframe can be named using the name attribute. This implies that when a link with a target attribute with that name as value is clicked, the linked resource will open in that iframe.
- `<iframe src="demo-page.html" name="myFrame"></iframe>`
- `<p><a href="https://www.google.com" target="myFrame">Open mywebpage.com</a></p>`

# Sandbox attribute

- The sandbox attribute enables an extra set of restrictions for the content in the iframe. When the sandbox attribute is present, and it will:
  - treat the content as being from a unique origin
  - block form submission
  - block script execution
  - disable APIs
  - prevent links from targeting other browsing contexts
  - prevent content from using plugins (through <embed>, <object>, <applet>, or other)
  - prevent the content to navigate its top-level browsing context
  - block automatically triggered features (such as automatically playing a video or automatically focusing a form control)

# **Client Side Scripting**

**Web Technologies**

# What is Client Side Scripting

- **Client side scripting** is a process in which the code along with HTML web page is sent to the client by the server. Here, the code refers to the script.
- In other simple words, client side scripting is a process in which scripts are executed by browsers without connecting the server.
- The code executes on the browser of client's computer either during the loading of web page or after the web page has been loaded.
- Client side scripting is mainly used for dynamic user interface elements, such as pull-down menus, navigation tools, animation buttons, data validation purpose, etc.
- JavaScript and jQuery are by far the most important client-side scripting languages or **web scripting languages** and widely used to create a dynamic and responsive webpage and websites.
- The browser (temporarily) downloads the code in the local computer and starts processing it without the server. Therefore, the client side scripting is browser dependent.

# What is Client side Script?

- A **client-side script** is a small program (or set of instructions) that is embedded (or inserted) into a web page. It is processed within the client browser instead of the web server.
- The script that executes on the user's computer system is called **client**. It is embedded (or inserted) within the HTML document or can be stored in an external separate file (known as external script).
- The script files are sent to the client machine from the web server (or servers) when they are requested. The client's web browser executes the script, then displays the web page, including any visible output from the script.

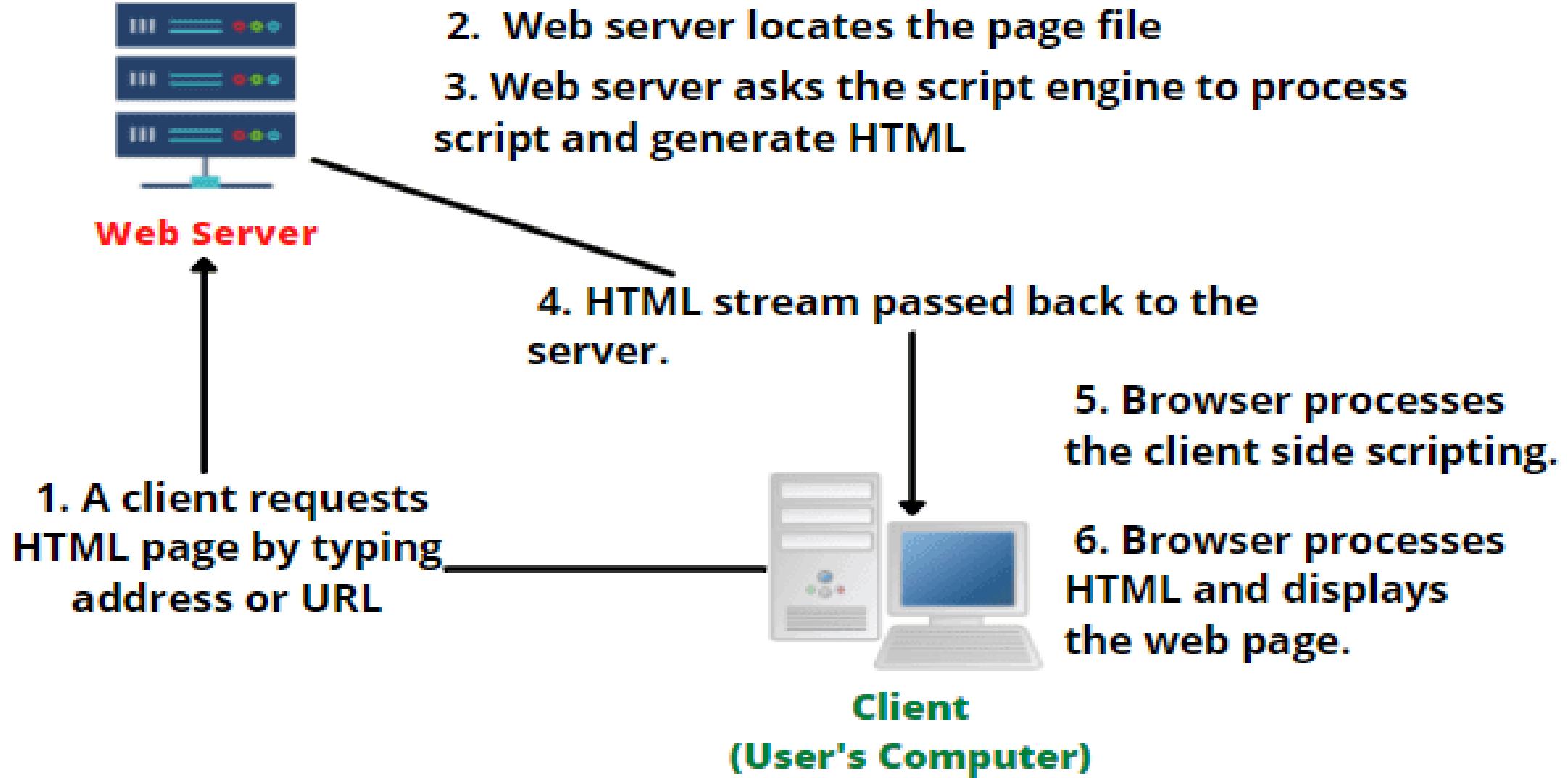


Figure: Server-side and Client-side scripting

# Popular Client side Scripting Language

- A language in which a client side script or program is written using syntax is called **client side scripting language** or client side programming.
- The most popular client side scripting languages is as follows:
  - **1. JavaScript**
  - **2. VBScript**
  - **3. jQuery**

# JavaScript

- **JavaScript:** It is the most widely client side scripting or programming language. It is based on ECMAScript standard language.
- JavaScript is an object based oriented, dynamically typed (or also called weakly typed) scripting language. It runs directly on the browser with the help of an inbuilt interpreter.
- Here, weakly typed means the variables are easily converted implicitly from one data type to another.

# VBScript and JQuery

- **VBScript:** This scripting language is developed by Microsoft, based on the Visual Basic. It is basically used to enhance the features of web pages in Internet Explorer. VBScript is interpreted by Internet Explorer web browser.
- **jQuery:** jQuery is a fast, small, lightweight JavaScript library. It is used to facilitate a lot of JavaScript code into simple-to-use-functionality.
- Most of the biggest companies such as Google, Microsoft, IBM, Netflix, etc. on the Web are using jQuery language

# Introduction to JavaScript

- **JavaScript** is a lightweight, cross-platform, and interpreted compiled programming language which is also known as the scripting language for webpages.
- It is well-known for the development of web pages, many non-browser environments also use it.
- JavaScript can be used for **Client-side** developments as well as **Server-side** developments.
- JavaScript contains a standard library of objects, like **Array**, **Date**, and **Math**, and a core set of language elements like **operators**, **control structures**, and **statements**.

Year	ECMA	Browser
1995		JavaScript was invented by Brendan Eich
1997		JavaScript became an ECMA standard (ECMA-262)
1997	ES1	ECMAScript 1 was released
1998	ES2	ECMAScript 2 was released
1999	ES3	ECMAScript 3 was released
2008	ES4	ECMAScript 4 was abandoned
2009	ES5	ECMAScript 5 was released
2015	ES6	ECMAScript 6 was released
2017	ES6	Full support for ES6 in Edge 15
2018	ES6	Full support for ES6 in all browsers **

# Features of JavaScript

- With advances in browser technology and JavaScript having moved into the server with Node.js and other frameworks, JavaScript is capable of so much more.
- few things that we can do with JavaScript:
- JavaScript was created in the first place for DOM manipulation.
- Functions in JS are objects. They may have properties and methods just like another object.
- Can handle date and time.
- Performs Form Validation although the forms are created using HTML.
- No compiler is needed.

# **Applications of JavaScript**

- **Web Development:** Adding interactivity and behavior to static sites JavaScript was invented to do this in 1995.
- **Web Applications:** With technology, browsers have improved to the extent that a language was required to create robust web applications.
- **Server Applications**
- **Games**
- **Smartwatches**
- **Art**
- **Machine Learning**
- **Mobile Applications**

# Where to put JavaScript in an HTML Document ?

- Scripts can be placed inside the **body** or the **head** section of an HTML page or inside both the head and body.
- We can also place javascript outside the HTML file which can be linked by specifying its source in the script tag.

# JavaScript in head

```
html>
<head>
    <script>
        function gfg() {
            document.getElementById("demo").innerHTML = "Welcome to
javascript";
        }
    </script>
</head>
<body>
    <h2>
        JavaScript in Head
    </h2>
    <p id="demo" style="color:green;">
        Example program
    </p>
    <button type="button" onclick="gfg()">
        click it
    </button>
</body>
</html>
```

# JavaScript in <body>

- Placing scripts at the bottom of the <body> element improves the display speed, because script interpretation slows down the display.

- <!DOCTYPE html>

- <html>
  - <body>

- <h2>Demo JavaScript in Body</h2>

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

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

- <script>

- function myFunction() {

- document.getElementById("demo").innerHTML = "Paragraph changed.;"

- }

- </script>

- </body>

- </html>

# External JavaScript

- **External file: myScript.js**
- ```
function myFunction() {  
    document.getElementById("demo").innerHTML = "Paragraph changed.";  
}
```
- External scripts are practical when the same code is used in many different web pages.
- JavaScript files have the file extension .js.
- To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:
- ```
<script src="myScript.js"></script>
```
- You can place an external script reference in <head> or <body> as you like

# External JavaScript Advantages

- Placing scripts in external files has some advantages:
- It separates HTML and code
- It makes HTML and JavaScript easier to read and maintain
- Cached JavaScript files can speed up page loads
- To add several script files to one page - use several script tags:
- **Example**
- `<script src="myScript1.js"></script>`  
`<script src="myScript2.js"></script>`

# Java Script Variables and DataTypes

Web Technologies

# **JavaScript Output : JavaScript Display Possibilities**

- JavaScript can "display" data in different ways:
  - Writing into an HTML element, using innerHTML.
  - Writing into the HTML output using document.write().
  - Writing into an alert box, using window.alert().
  - Writing into the browser console, using console.log().

# JavaScript Comments

- **Single Line Comments**

- Single line comments start with //.
- Any text between // and the end of the line will be ignored by JavaScript (will not be executed).
- let x = 5; // Declare x, give it the value of 5

- **Multi-line Comments:-**

- Multi-line comments start with /\* and end with \*/.
- Any text between /\* and \*/ will be ignored by JavaScript.

- /\*  
The code below will change  
the heading with id = "myH"  
and the paragraph with id = "myP"  
in my web page:  
\*/

# JavaScript Variables

- Variables are containers for storing data (storing data values).
- 4 Ways to Declare a JavaScript Variable:
  - Using var                    `var x = 5;`
  - Using let                    `let x = 5;`
  - Using const
  - Using nothing                `x = 5;`

# When to Use JavaScript var?

- Always declare JavaScript variables with var, let, or const.
- The **var** keyword is used in all JavaScript code from 1995 to 2015.
- The **let** and **const** keywords were added to JavaScript in 2015.
- If you want your code to run in older browsers, you must use var

# JavaScript Let

- The let keyword was introduced in ES6 (2015).
- Variables defined with let can not be redeclared.
- Variables defined with let must be declared before use.
- Variables defined with let have block scope.
- Redeclaring a variable inside a block will not redeclare the variable outside the block:

Example

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

Example

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

# Block Scope

- Before ES6 (2015), JavaScript had Global Scope and Function Scope.
- ES6 introduced two important new JavaScript keywords: let and const.
- These two keywords provide Block Scope in JavaScript.
- Variables declared inside a { } block cannot be accessed from outside the block:
- Example
- {
- let x = 2;
- }
- // x can NOT be used here

# JavaScript Const

- The const keyword was introduced in ES6 (2015).
- Variables defined with const cannot be Redeclared.
- Variables defined with const cannot be Reassigned.
- Variables defined with const have Block Scope.
- A const variable cannot be reassigned, JavaScript const variables must be assigned a value when they are declared:

- **Example**

- `const PI = 3.141592653589793;`
- `PI = 3.14; // This will give an error`
- `PI = PI + 10; // This will also give an error`

# When to use JavaScript const?

- Always declare a variable with const when you know that the value should not be changed. It does not define a constant value. It defines a constant reference to a value.
- Use const when you declare:
  - A new Array
  - A new Object
  - A new Function
  - A new RegExp
- Example
- `const cars = ["Saab", "Volvo", "BMW"];`
- `cars = ["Toyota", "Volvo", "Audi"];` // ERR

Because of this you can NOT:

Reassign a constant value  
Reassign a constant array  
Reassign a constant object  
But you CAN:  
Change the elements of constant array  
Change the properties of constant object

Example

```
// You can create a constant array:  
const cars = ["Saab", "Volvo",  
"BMW"];
```

```
// You can change an element:  
cars[0] = "Toyota";
```

```
// You can add an element:  
cars.push("Audi");
```

# JavaScript Data Types

- There are different types of data that we can use in a JavaScript program.
- Here, all data types except Object are primitive data types,

Data Types	Description	Example
String	represents textual data	'hello', "hello world!"
Number	an integer or a floating-point number	3, 3.234, 3e-2
BigInt	an integer with arbitrary precision	900719925124740999n , 1n
Boolean	Any of two values: true or false	true and false
undefined	a data type whose variable is not initialized	let a;
null	denotes a null value	let a = null;
Symbol	data type whose instances are unique and immutable	let value = Symbol('hello');
Object	key-value pairs of collection of data	let student = {};

# JavaScript BigInt

- In JavaScript, Number type can only represent numbers less than  $(2^{53} - 1)$  and more than  $-(2^{53} - 1)$ . However, if you need to use a larger number than that, you can use the BigInt data type.
- A BigInt number is created by appending n to the end of an integer.
- **// BigInt value**
  - const value1 = 900719925124740998n;
- **// Adding two big integers**
  - const result1 = value1 + 1n;
  - console.log(result1); // "900719925124740999n"
- const value2 = 900719925124740998n;
- **// Error! BigInt and number cannot be added**
  - const result2 = value2 + 1;
  - console.log(result2);

# JavaScript undefined

- The undefined data type represents value that is not assigned.
- If a variable is declared but the value is not assigned, then the value of that variable will be undefined.
- **let name;**
- **console.log(name);**
  
- **let name = undefined;**
- **console.log(name); // undefined**

# JavaScript Symbol

- This data type was introduced in a newer version of JavaScript (from ES2015).
- A value having the data type **Symbol** can be referred to as a symbol value. Symbol is an immutable primitive value that is unique. For example.
- // two symbols with the same description
- const value1 = Symbol('hello');
- const value2 = Symbol('hello');
- Though value1 and value2 both contain 'hello', they are different as they are of the Symbol type.

# JavaScript Object

- An object is a complex data type that allows us to store collections of data. For example,
- ```
const student = {  
    firstName: 'ram',  
    lastName: null,  
    class: 10  
};
```

# Types of JavaScript Operators

- There are different types of JavaScript operators:
  - Arithmetic Operators
  - Assignment Operators
  - Comparison Operators
  - String Operators
  - Logical Operators
  - Bitwise Operators
  - Ternary Operators
  - Type Operators

# JavaScript Arithmetic Operators

| Operator | Description                               |
|----------|-------------------------------------------|
| +        | Addition                                  |
| -        | Subtraction                               |
| *        | Multiplication                            |
| **       | Exponentiation ( <a href="#">ES2016</a> ) |
| /        | Division                                  |
| %        | Modulus (Division Remainder)              |
| ++       | Increment                                 |
| --       | Decrement                                 |

# JavaScript Comparison Operators

| Operator           | Description                       |
|--------------------|-----------------------------------|
| <code>==</code>    | equal to                          |
| <code>===</code>   | equal value and equal type        |
| <code>!=</code>    | not equal                         |
| <code>!==</code>   | not equal value or not equal type |
| <code>&gt;</code>  | greater than                      |
| <code>&lt;</code>  | less than                         |
| <code>&gt;=</code> | greater than or equal to          |

# JavaScript Type Operators

| Operator   | Description                                                |
|------------|------------------------------------------------------------|
| typeof     | Returns the type of a variable                             |
| instanceof | Returns true if an object is an instance of an object type |

# JavaScript Bitwise Operators

- Bit operators work on 32 bits numbers

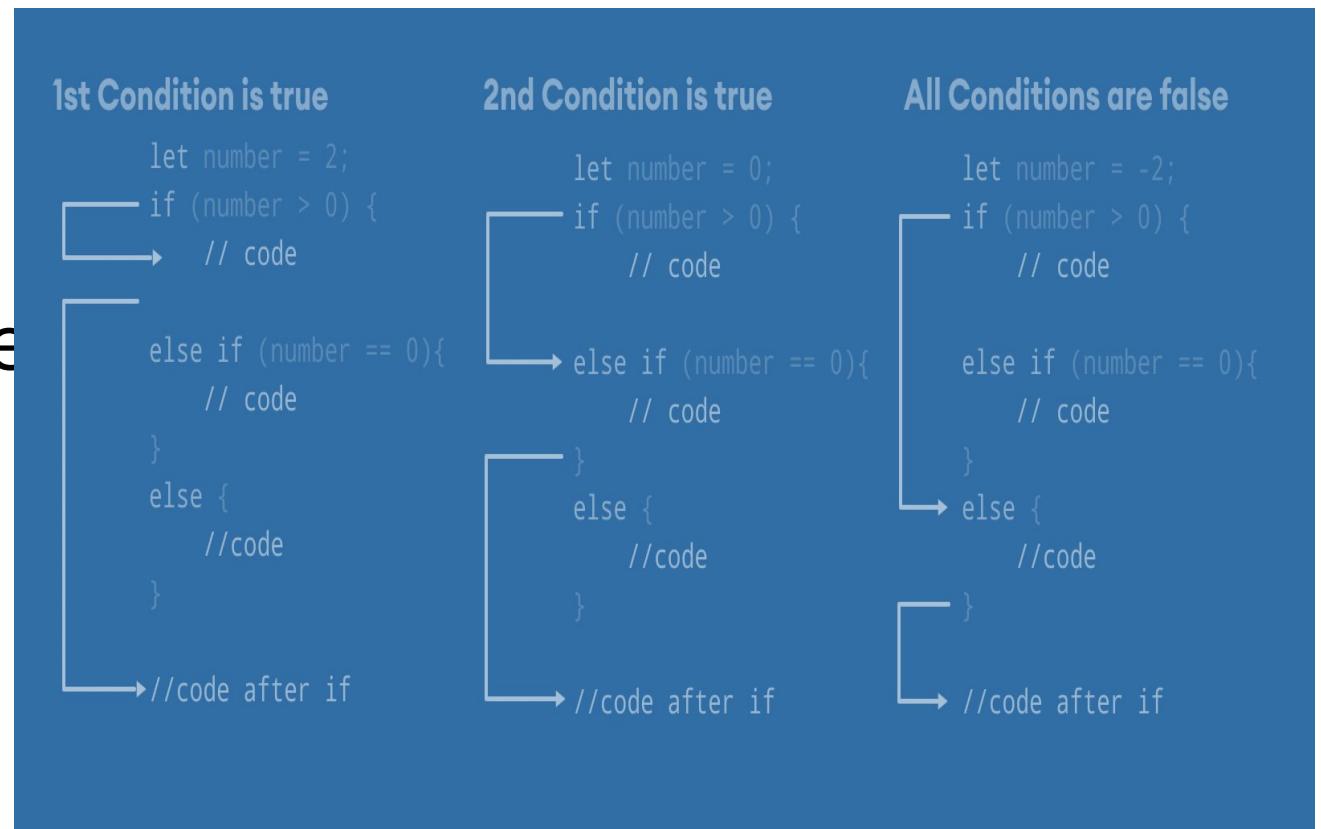
| Operator | Description | Example | Same as        | Result | Decimal |
|----------|-------------|---------|----------------|--------|---------|
| &        | AND         | 5 & 1   | 0101 &<br>0001 | 0001   | 1       |
|          | OR          | 5   1   | 0101  <br>0001 | 0101   | 5       |
| ~        | NOT         | ~ 5     | ~0101          | 1010   | 10      |
| ^        | XOR         | 5 ^ 1   | 0101 ^<br>0001 | 0100   | 4       |
| <<       | left shift  | 5 << 1  | 0101 <<<br>1   | 1010   | 10      |
| >>       | right shift | 5 >> 1  | 0101 >><br>1   | 0010   | 2       |

# JavaScript if...else Statement

- In computer programming, there may arise situations where you have to run a block of code among more than one alternatives. For example, assigning grades A, B or C based on marks obtained by a student.
- In such situations, you can use the JavaScript if...else statement to create a program that can make decisions.

- In JavaScript, there are three forms of the if...else statement.

- if statement
- if...else statement
- if...else if...else statement



# Example

```
let msg2=prompt("enter the designation ","manager");
if(msg2=='director')
    alert("welcome sir");
else if(msg2=='dean')
    alert("good morning sir");
else if(msg2=='hod')
    alert(" good day sir");
else
    alert("thank you");
```



# JavaScript for loop

- In programming, loops are used to repeat a block of code.
- Syntax:
- ```
for (initialExpression; condition; updateExpression) {  
    // for loop body  
}  
// program to display text 5 times
```
- ```
const n = 5;  
// looping from i = 1 to 5
```
- ```
for (let i = 1; i <= n; i++) {  
    console.log(`I love JavaScript. `);  
}
```

# JavaScript while Loop

- While(condition)
- {
- Statements;
- }
- While(i<10)
- {
- Text+="the number"+l;
- i++;
- }

```
Var i=1;  
do{  
Text+=‘the  
number’+l;  
i++;  
}  
while(i<10)  
;
```

# Type conversion

- 1. Implicit Type conversion
- 2. Explicit Type Conversion
- Implicit:-automatically done by interpreter
- Explicit : coversion done by the programmer.

# JAVA SCRIPT

Arrays ,Functions &Objects

# JAVASCRIPT NON-PRIMITIVE DATA TYPES

<b>Data Type</b>	<b>Description</b>
Object	represents instance through which we can access members
Array	represents group of similar values
RegExp	represents regular expression

## JAVASCRIPT FUNCTIONS ARE OBJECTS!

- In JS world a function is considered an object, and Types in JavaScript are categorized by:
  - **Primitives (string, number, null, boolean, undefined, symbol)**: these are immutable data types. They are not objects, don't have methods and they are stored in memory by value.
  - **Non-Primitives (functions, arrays and objects)**: these are mutable data types. They are objects and they are stored in memory by reference.

# JAVASCRIPT FUNCTIONS

- A JavaScript function is a block of code designed to perform a particular task.
- A JavaScript function is executed when "something" invokes it (calls it).
- A JavaScript function is defined with the `function` keyword, followed by a **name**, followed by parentheses `()`.
- Example:-`function myFunction(p1, p2)`

```
{  
  return p1 * p2; // The function returns the product of p1  
  and p2  
}
```

# SYNTAX

- function *name*(*parameter1*, *parameter2*, *parameter3*)//function definition
- {  
    *// code to be executed*  
}
- Function **parameters** are listed inside the parentheses () in the function definition.
- Function **arguments** are the **values** received by the function when it is invoked.

- **Function Expression:** When a function is assigned to a variable. The function can be named, or anonymous. Use the variable name to call a function defined in a function expression.
  - // *anonymous function expression*
  - **var doSomething = function(y) { return y + 1; };**
- // *named function expression*
- **var doSomething = function addOne(y) { return y + 1; };** // *for either of the definitions above, call the function like this:*  
doSomething(5);

# FUNCTION INVOCATION

- The code inside the function will execute when "something" **invokes** (calls) the function:
- When an event occurs (when a user clicks a button)
- When it is invoked (called) from JavaScript code
- Automatically (self invoked)

# THE () OPERATOR INVOKES THE FUNCTION

- Using the example above, toCelsius refers to the function object, and toCelsius() refers to the function result.
- Example
- ```
function toCelsius(fahrenheit)
{
    return (5/9) * (fahrenheit-32);
}
document.getElementById("demo").innerHTML = toCelsius;
```

# JAVASCRIPT ARRAYS

- JavaScript arrays are written with square brackets.
- An array is an object that can store a **collection of items**. Arrays become really useful when you need to store large amounts of data of the same type.
- Array items are separated by commas.
- The following code declares (creates) an array called cars, containing three items (car names):
  - Var cars = ["Saab", "Volvo", "BMW"];
  - const cars=["Saab", "Volvo", "BMW"];

# ARRAYS

- You can access the items in an array by referring to its index number and the index of the first element of an array is zero.
- You can also create an array using Array constructor like this:
- `var students = new Array("John", "Ann", "Kevin");`  
OR
- `var students = new Array();`
- `students[0] = "John";`
- `students[1] = "Ann"; students[2] = "Kevin";`

# JAVASCRIPT ARRAY METHODS

- The Array object has many properties and methods which help developers to handle arrays easily and efficiently.
- **length property** --> If you want to know the number of elements in an array, you can use the length property.
- **prototype property** --> If you want to add new properties and methods, you can use the prototype property.
- **reverse() method** --> You can reverse the order of items in an array using a reverse method.
- **sort() method** --> You can sort the items in an array using sort method.
- **Pop() method** --> You can remove the last item of an array using a pop method.
- **Shift() method** --> You can remove the first item of an array using shift method.
- **Push() method** --> You can add a value as the last item of the array.

# CHANGING ELEMENTS

- Array elements are accessed using their **index number**:
- Array **indexes** start with 0:
- [0] is the first array element  
[1] is the second  
[2] is the third ...
- ```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits[0] = "Kiwi";
```
- ```
fruits=["graps", "gova"];//not possible
```

# DELETING ELEMENTS IN ARRAY

- Since JavaScript arrays are objects, elements can be deleted by using the JavaScript operator delete:
- `const fruits = ["Banana", "Orange", "Apple", "Mango"];`  
`delete fruits[0];`

# MERGING (CONCATENATING) ARRAYS

- The concat() method creates a new array by merging (concatenating) existing arrays:
- Example (Merging Two Arrays)
- ```
const myGirls = ["Cecilie", "Lone"];
const myBoys = ["Emil", "Tobias", "Linus"];
```

// Concatenate (join) myGirls and myBoys

```
const myChildren = myGirls.concat(myBoys);
```

# AUTOMATIC TO STRING()

- JavaScript automatically converts an array to a comma separated string when a primitive value is expected.
- This is always the case when you try to output an array.
- These two examples will produce the same result:
- Example
- ```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML =
fruits.toString();
```

# EXAMPLE

```
<script>  
document.getElementById("demo").innerHTML =  
"The temperature is " + toCelsius(77) + " Celsius";  
  
function toCelsius(fahrenheit) {  
    return (5/9) * (fahrenheit-32);  
}  
</script>
```

# NESTED ARRAYS

- An **array** is useful because it stores multiple values into a single, organized data structure.
- *// creates a `mixedData` array with mixed data types* **var**  
mixedData = ["abcd", 1, true, undefined, null, "all the things"];
- You can even store an array in an array to create a **nested array**!  
*// creates a `arraysInArrays` array with three arrays* **var**  
arraysInArrays = [[1, 2, 3], ["Julia", "James"], [true, false, true, false]];

# NESTED ARRAYS CONT...

- Nested arrays can be particularly hard to read, so it's common to write them on one line, using a newline after each comma:
- **var arraysInArrays = [ [1, 2, 3], ["Julia", "James"], [true, false, true, false] ];**
- Example:-

```
[ [33, 91, 13, 9, 23],  
[null, "", undefined, []],  
[3.14, "pi", 3, 1, 4, "Yum, I like pie!"] ]
```

are all valid arrays. However, mixed data arrays are typically not very useful. In most cases, you'll want to use elements of the same type in your arrays.

# SPLICE

- `splice()` is another handy method that allows you to add and remove elements from anywhere within an array.
- While `push()` and `pop()` limit you to adding and removing elements from *the end of an array*, `splice()` lets you specify the index location to add new elements, as well as the number of elements you'd like to delete (if any).
- The **`splice()`** method changes the contents of an array by removing or replacing existing elements and/or adding new elements.

- `var donuts = ["glazed", "chocolate frosted", "Boston creme", "glazed cruller"];`
- `donuts.splice(1, 1, "chocolate cruller", "creme de leche");`
- *// removes "chocolate frosted" at index 1 and adds "chocolate cruller" and "creme de leche" starting at index 1*
- **Following is the syntax of splice() method:** `arrayName.splice(arg1, arg2, item1, ..... , itemX);` where,
- `arg1` = Mandatory argument. Specifies the starting index position to add/remove items.
- You can use a negative value to specify the position from the end of the array e.g., `-1` specifies the last element.
- `arg2` = Optional argument. Specifies the count of elements to be removed. If set to 0, no items will be removed.
- `item1, ..... , itemX` are the items to be added at index position `arg1`
- `splice()` method returns the item(s) that were removed.

# REMOVE ELEMENTS USING SPLICE METHOD

- Remove 1 element at index 3

Ex:-

```
let myFish = ['angel', 'clown', 'drum', 'mandarin', 'sturgeon'];
```

```
let removed = myFish.splice(3, 1);
```

- Remove 2 elements from index 0, and insert "parrot", "anemone" and "blue"

Ex:-

```
splice(0, 2, 'parrot', 'anemone', 'blue')
```

- Remove 1 element from index -2

Ex:- let myFish = ['angel', 'clown', 'mandarin', 'sturgeon'] ;

```
let removed = myFish.splice(-2, 1);
```

- // myFish is ["angel", "clown", "sturgeon"] // removed is ["mandarin"]

- Remove all elements, starting from index 2

```
let myFish = ['angel', 'clown', 'mandarin', 'sturgeon']
```

```
let removed = myFish.splice(2)
```

# FOR EACH LOOP

- Arrays have a set of special methods to help you iterate over and perform operations on collections of data.
- `var donuts = ["jelly donut", "chocolate donut", "glazed donut"];`
- `donuts.forEach(function(donut) {`
- `donut += " hole";`
- `donut = donut.toUpperCase();`
- `console.log(donut); });`
- ***Prints:***  
*JELLY DONUT HOLE*  
*CHOCOLATE DONUT HOLE*  
*GLAZED DONUT HOLE*

# METHOD

- map() method, you can take an array, perform some operation on each element of the array, and return a new array.
- The map() method accepts one argument, a function that will be used to manipulate each element in the array.

- **var** donuts = ["jelly donut", "chocolate donut", "glazed donut"];  
**var** improvedDonuts = donuts.map(**function**(donut) { donut += " hole"; donut = donut.toUpperCase(); **return** donut; }); **donu**

# JAVASCRIPT MULTIDIMENSIONAL ARRAY

- However, you can create a multidimensional array by defining an array of elements, where each element is also another array.
- To declare an empty multidimensional array, you use the same syntax as declaring one-dimensional array:
- Ex:-

```
let activities = [];
```
- ```
console.table(activities);
```
- ```
console.log(activities[0][1]); // 9
```

# ADDING ELEMENTS TO THE JAVASCRIPT MULTIDIMENSIONAL ARRAY

- You can use the Array methods such as push() and splice() to manipulate elements of a multidimensional array.
- `activities.push(['Study',2]);`
- `activities.splice(1, 0, ['Programming'], 2);`
- `console.table(activities);`

# ADDING ELEMENT TO THE OUTER ARRAY AND INNER ARRAY

- Ex:-
- let studentsData = [['Jack', 24], ['Sara', 23],];  
studentsData.push(['Peter', 24]);
- Output:-
- //[[{"Jack": 24}, {"Sara": 23}, {"Peter": 24}]
  
- let studentsData = [['Jack', 24], ['Sara', 23],]; studentsData[1][2] = 'hello'; console.log(studentsData); // [[{"Jack": 24}, {"Sara": 23, "hello"}]]

# JAVASCRIPT OBJECT

- A JavaScript object is a collection of **named values**
- It is a common practice to declare objects with the `const` keyword.
- Example
- `const person = {  
 firstName:"John",  
 lastName:"Doe",  
 age:50,  
 eyeColor:"blue"};`

# JAVASCRIPT OBJECTS

- In JavaScript, almost "everything" is an object.
- Booleans can be objects (if defined with the new keyword)
- Numbers can be objects (if defined with the new keyword)
- Strings can be objects (if defined with the new keyword)
- Dates are always objects
- Maths are always objects
- Regular expressions are always objects
- Arrays are always objects
- Functions are always objects
- Objects are always objects

# BUILT-IN JAVASCRIPT CONSTRUCTORS

- JavaScript has built-in constructors for native objects:
- `new String()` // A new String object  
`new Number()` // A new Number object  
`new Boolean()` // A new Boolean object  
`new Object()` // A new Object object  
`new Array()` // A new Array object  
`new RegExp()` // A new RegExp object  
`new Function()` // A new Function object  
`new Date()` // A new Date object

# OBJECT PROPERTIES

- The named values, in JavaScript objects, are called **properties**.
- Properties can usually be changed, added, and deleted
- Objects are sometimes called *associative arrays*, since each property is associated with a string value that can be used to access it.
- `var myCar = { "make 1": 'Ford', model: 'Mustang', year: 1969 };`
- The syntax for accessing the property of an object is:
- *objectName.property*
- `// person.age`
- or
- *objectName["property"]*
- `myCar["make 1"] = "maruthi";`
- `myCar['model'] = 'Mustang';`
- `myCar['year'] = 1969;`

# OBJECT METHODS

- Methods are **actions** that can be performed on objects.
- Object properties can be both primitive values, other objects, and functions.
- An **object method** is an object property containing a **function definition**.

| Property  | Value                                                                  |
|-----------|------------------------------------------------------------------------|
| firstName | John                                                                   |
| lastName  | Doe                                                                    |
| age       | 50                                                                     |
| eyeColor  | blue                                                                   |
| fullName  | <code>function() {return this.firstName + " " + this.lastName;}</code> |

# CREATING A JAVASCRIPT OBJECT

- With JavaScript, you can define and create your own objects.
- There are different ways to create new objects:
- Create a single object, using an object literal.
- Create a single object, with the keyword new.
- Define an object constructor, and then create objects of the constructed type.
- Create an object using Object.create().

# USING AN OBJECT LITERAL

- This is the easiest way to create a JavaScript Object.
- Using an object literal, you both define and create an object in one statement.
- An object literal is a list of name:value pairs (like age:50) inside curly braces {}.
- This example creates an empty JavaScript object, and then adds 4 properties:
- Example
- ```
const person = { };
person.firstName = "John";
person.lastName = "Doe";
person.age = 50;
person.eyeColor = "blue";
```

# USING THE JAVASCRIPT KEYWORD NEW

- The following example create a new JavaScript object using new Object(), and then adds 4 properties:
- Example
- ```
const person = new Object();
person.firstName = "John";
person.lastName = "Doe";
person.age = 50;
person.eyeColor = "blue";
```

# USING BUILT-IN METHODS

- This example uses the `toUpperCase()` method of the `String` object, to convert a text to uppercase:
- `let message = "Hello world!";  
let x = message.toUpperCase();`
- `Math.round()`
- `Math.round(x)` returns the nearest integer:
- `Math.random()`
- `Math.random()` returns a random number between 0 (inclusive), and 1 (exclusive):

# MATH OBJECT....

- The Math object is static.
- methods and properties can be used without creating a Math object first.

| Method               | Description                               |
|----------------------|-------------------------------------------|
| abs(x)               | Returns the absolute value of x           |
| acos(x)              | Returns the arccosine of x, in radians    |
| acosh(x)             | Returns the hyperbolic arccosine of x     |
| sin(x)               | Returns the arcsine of x, in radians      |
| asinh(x)             | Returns the hyperbolic arcsine of x       |
| max(x, y, z, ..., n) | Returns the number with the highest value |
| random()             | Returns a random number between 0 and 1   |

# DISPLAY JAVASCRIPT OBJECTS

- Displaying a JavaScript object will output **[object Object]**.

- ```
const person = {  
    name: "John",  
    age: 30,  
    city: "New York"  
};
```

```
document.getElementById("demo").innerHTML = person;
```

# DISPLAYING THE OBJECT IN A LOOP

- The properties of an object can be collected in a loop:
- Example
- ```
const person = {  
    name: "John",  
    age: 30,  
    city: "New York"  
};
```

```
let txt = "";  
for (let x in person) {  
    txt += person[x] + " ";  
};
```

```
document.getElementById("demo").innerHTML = txt;
```

# Pop up Boxes & HTML DOM in JavaScript

---

WEB TECHNOLOGIES

# JavaScript Popup Boxes

---

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

**Alert Box**

**Confirm Box**

**Prompt Box**

# Alert Box

---

An alert box is often used if you want to make sure information comes through to the user.

When an alert box pops up, the user will have to click "OK" to proceed.

## Syntax

```
window.alert("sometext");
```

The window.alert() method can be written without the window prefix.

```
alert("I am an alert box!");
```

# Confirm Box

---

A confirm box is often used if you want the user to verify or accept something.

When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

If the user clicks "OK", the box returns **true**. If the user clicks "Cancel", the box returns **false**.

**Syntax :** window.confirm("sometext");

The window.confirm() method can be written without the window prefix.

```
if (confirm("Press a button!")) {  
    txt = "You pressed OK!";  
} else {  
    txt = "You pressed Cancel!";  
}
```

# Prompt Box

A prompt box is often used if you want the user to input a value before entering a page.

---

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

**Syntax :** window.prompt("sometext","defaultText");

The window.prompt() method can be written without the window prefix.

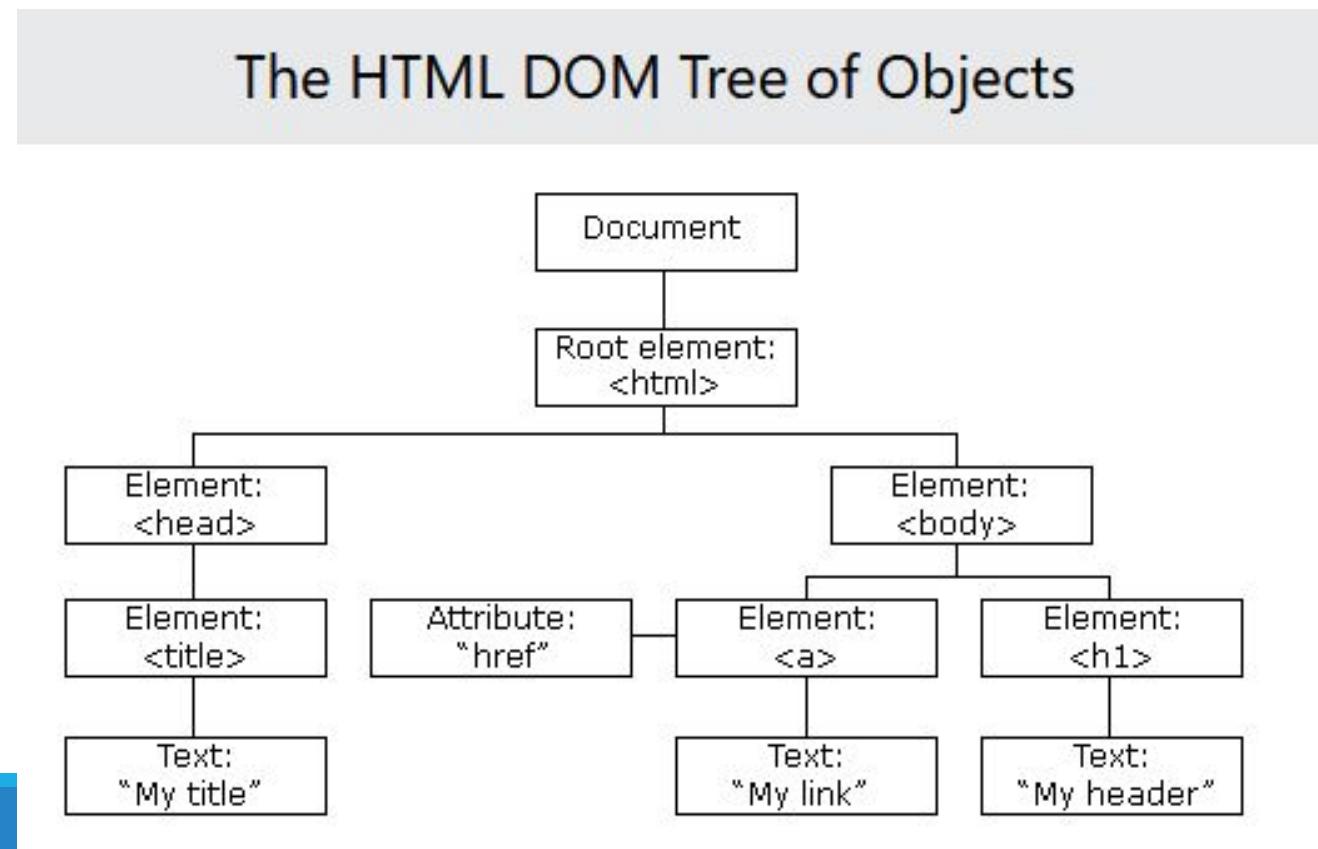
## Example

```
let person = prompt("Please enter your name", "Harry Potter");
let text;
if (person == null || person == "") {
  text = "User cancelled the prompt.";
} else {
  text = "Hello " + person + "! How are you today?";
}
```

# The HTML DOM (Document Object Model)

When a web page is loaded, the browser creates a **Document Object Model** of the page.

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



# DOM

---

With the object model, JavaScript gets all the power it needs to create dynamic HTML:

JavaScript can change all the HTML elements in the page

JavaScript can change all the HTML attributes in the page

JavaScript can change all the CSS styles in the page

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

# What is the DOM?

---

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

The DOM defines a standard for accessing documents:

*"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*

# What is the HTML DOM?

---

The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

The HTML elements as **objects**

The **properties** of all HTML elements

The **methods** to access all HTML elements

The **events** for all HTML elements

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

# HTML DOM Methods

HTML DOM methods are **actions** you can perform (on HTML Elements).

HTML DOM properties are **values** (of HTML Elements) that you can set or change.

---

## The DOM Programming Interface

The HTML DOM can be accessed with JavaScript (and with other programming languages).

In the DOM, all HTML elements are defined as **objects**.

The programming interface is the properties and methods of each object.

A **property** is a value that you can get or set (like changing the content of an HTML element).

A **method** is an action you can do (like add or deleting an HTML element).

**Example** The following example changes the content (the innerHTML) of the <p> element with id="demo":

```
document.getElementById("demo").innerHTML = "Hello World!";
```

getElementById is a method, while innerHTML is a property.

# Finding HTML Elements

---

| Method                                                    | Description                   |
|-----------------------------------------------------------|-------------------------------|
| <code>document.getElementById(<i>id</i>)</code>           | Find an element by element id |
| <code>document.getElementsByTagName(<i>name</i>)</code>   | Find elements by tag name     |
| <code>document.getElementsByClassName(<i>name</i>)</code> | Find elements by class name   |

```
const x = document.getElementById("main");
const y = x.getElementsByTagName("p");
const x = document.getElementsByClassName("intro");
const x = document.querySelectorAll("p.intro");
```

# Changing HTML Elements

| Property                                            | Description                                   |
|-----------------------------------------------------|-----------------------------------------------|
| <code>element.innerHTML = new html content</code>   | Change the inner HTML of an element           |
| <code>element.attribute = new value</code>          | Change the attribute value of an HTML element |
| <code>element.style.property = new style</code>     | Change the style of an HTML element           |
| Method                                              | Description                                   |
| <code>element.setAttribute(attribute, value)</code> | Change the attribute value of an HTML element |

# Finding HTML Elements by CSS Selectors

---

If you want to find all HTML elements that match a specified CSS selector (id, class names, types, attributes, values of attributes, etc), use the `querySelectorAll()` method.

This example returns a list of all `<p>` elements with `class="intro"`

## Example

```
const x = document.querySelectorAll("p.intro");
```

# Adding and Deleting Elements

| Method                                       | Description                                      |
|----------------------------------------------|--------------------------------------------------|
| <code>document.createElement(element)</code> | Create an HTML element                           |
| <code>document.removeChild(element)</code>   | Remove an HTML element                           |
| <code>document.appendChild(element)</code>   | Add an HTML element                              |
| <code>document.replaceChild(new, old)</code> | Replace an HTML element                          |
| <code>document.write(text)</code>            | Write into the HTML output stream                |
| <code>document.forms</code>                  | Returns all <code>&lt;form&gt;</code> elements   |
| <code>document.head</code>                   | Returns the <code>&lt;head&gt;</code> element    |
| <code>document.images</code>                 | Returns all <code>&lt;img&gt;</code> elements    |
| <code>document.scripts</code>                | Returns all <code>&lt;script&gt;</code> elements |
| <code>document.title</code>                  | Returns the <code>&lt;title&gt;</code> element   |

```
<h2>The createElement() Method</h2>  
  
<p>Create a p element and append it to "myDiv":</p>  
  
<div id="myDIV" style="padding:16px;background-color:lightgray">  
  <h3>A DIV element</h3>  
  </div>  
  
<script>  
  // Create element:  
  
  const para = document.createElement("p");  
  para.innerHTML = "This is a paragraph."  
  
  // Append to another element:  
  
  document.getElementById("myDIV").appendChild(para);  
  
</script>
```

# BROWSER ENVIRONMENT

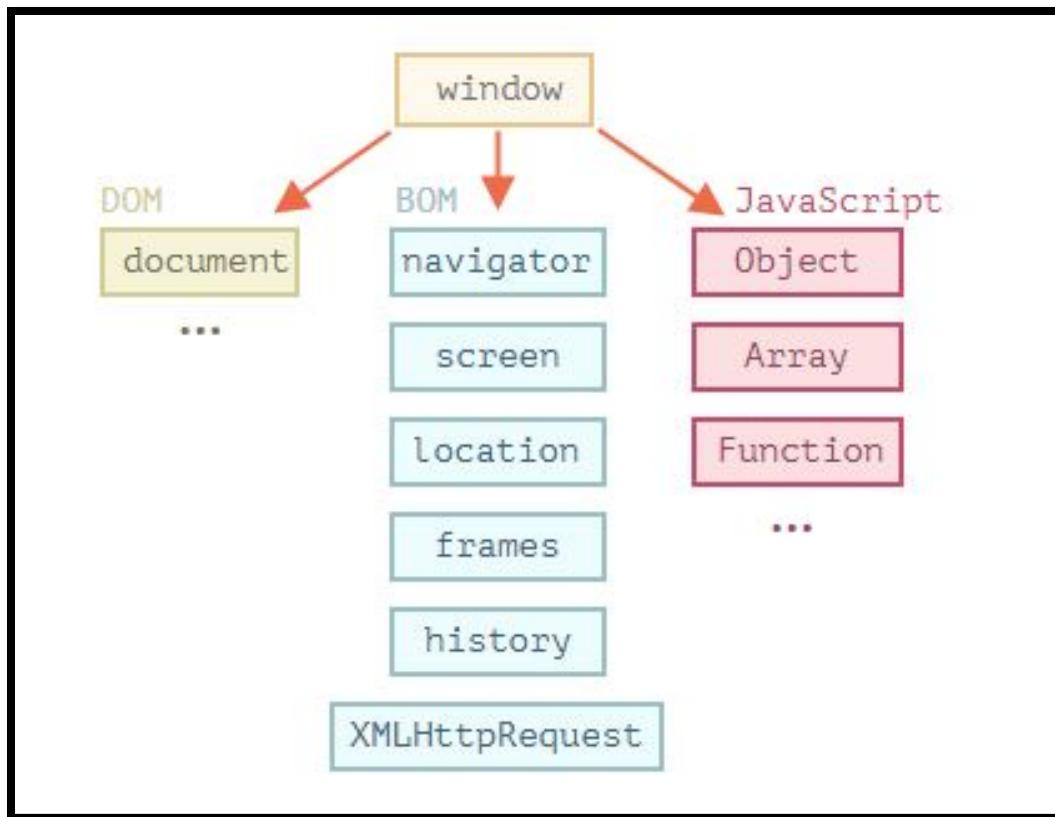
Browser Object Model (BOM)

# Introduction

- The JavaScript language was initially created for web browsers. Since then it has evolved and become a language with many uses and platforms.
- A platform may be a browser, or a web-server or another *host*.
- A host environment provides own objects and functions additional to the language core. Web browsers give a means to control web pages

# Browser Object Model

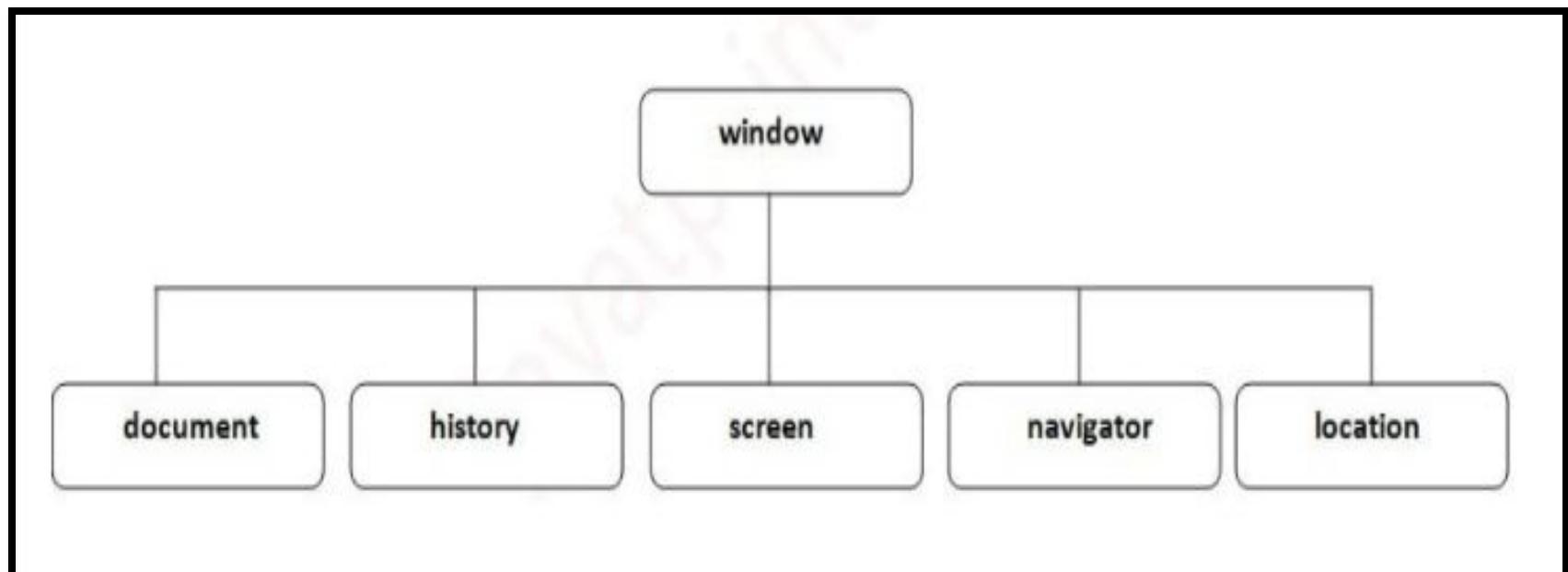
- The **Browser Object Model (BOM)** is used to interact with the browser.



# Browser window

The default object of browser is window means you can call all the functions of window by specifying window or directly.

- The BOM provides you with objects that expose the web browser's functionality.



# Section 1. Window

- It is used to interact with the browser window which displays the web page. It generally **represents a tab in browser**, but some actions like window width and height will affect the complete browser window.
- Window – understand the window object.
- Alert – display an alert dialog.
- Confirm – display a modal dialog with a question.
- Prompt – prompt the user to input some text.
- setTimeout – set a timer and execute a callback function once the timer expires.
- setInterval – execute a callback function repeatedly with a fixed delay between each call.

# window object

- The window object is supported by all browsers. It represents the browser's window.
- All global JavaScript objects, functions, and variables automatically become members of the window object.
- Global variables are properties of the window object.
- Global functions are methods of the window object.
- Even the document object (of the HTML DOM) is a property of the window object:
- `window.document.getElementById("header");`

# Window Methods

- Two properties can be used to determine the size of the browser window.
- `window.innerHeight` - the inner height of the browser window (in pixels)
- `window.innerWidth` - the inner width of the browser window (in pixels)
- Other Window Methods
- Some other methods:
- `window.open()` - open a new window
- `window.close()` - close the current window
- `window.moveTo()` - move the current window
- `window.resizeTo()` - resize the current window

# Section 2. Location

- The `window.location` object can be used to get the current page address (URL) and to redirect the browser to a new page.
- The `window.location` object can be written without the `window` prefix.
  
- [Location](#) – manipulate the location of a document via the [location](#) object.
- [Get query string parameters](#) – learn how to retrieve query string parameters.
- [Redirect to a new URL](#) – show you how to redirect to a new URL using JavaScript.

# window.location

## Some examples:

- window.location.href returns the href (URL) of the current page
- window.location.hostname returns the domain name of the web host
- window.location.pathname returns the path and filename of the current page
- window.location.protocol returns the web protocol used (http: or https:)
- window.location.assign() loads a new document

## Example

- Display the href (URL) of the current page:
- `document.getElementById("demo").innerHTML = "Page location is " + window.location.href;`

# Section 3. Navigator

- It acts as a storehouse of all the data and **information about the Browser software used to access the webpage** and this object is used to fetch information related to the browser for example, whether the user is using Chrome browser or Safari browser, which version of browser is being used etc.
- **Navigator** – query the information and capabilities of the browser using the **navigator** object.

# window.navigator

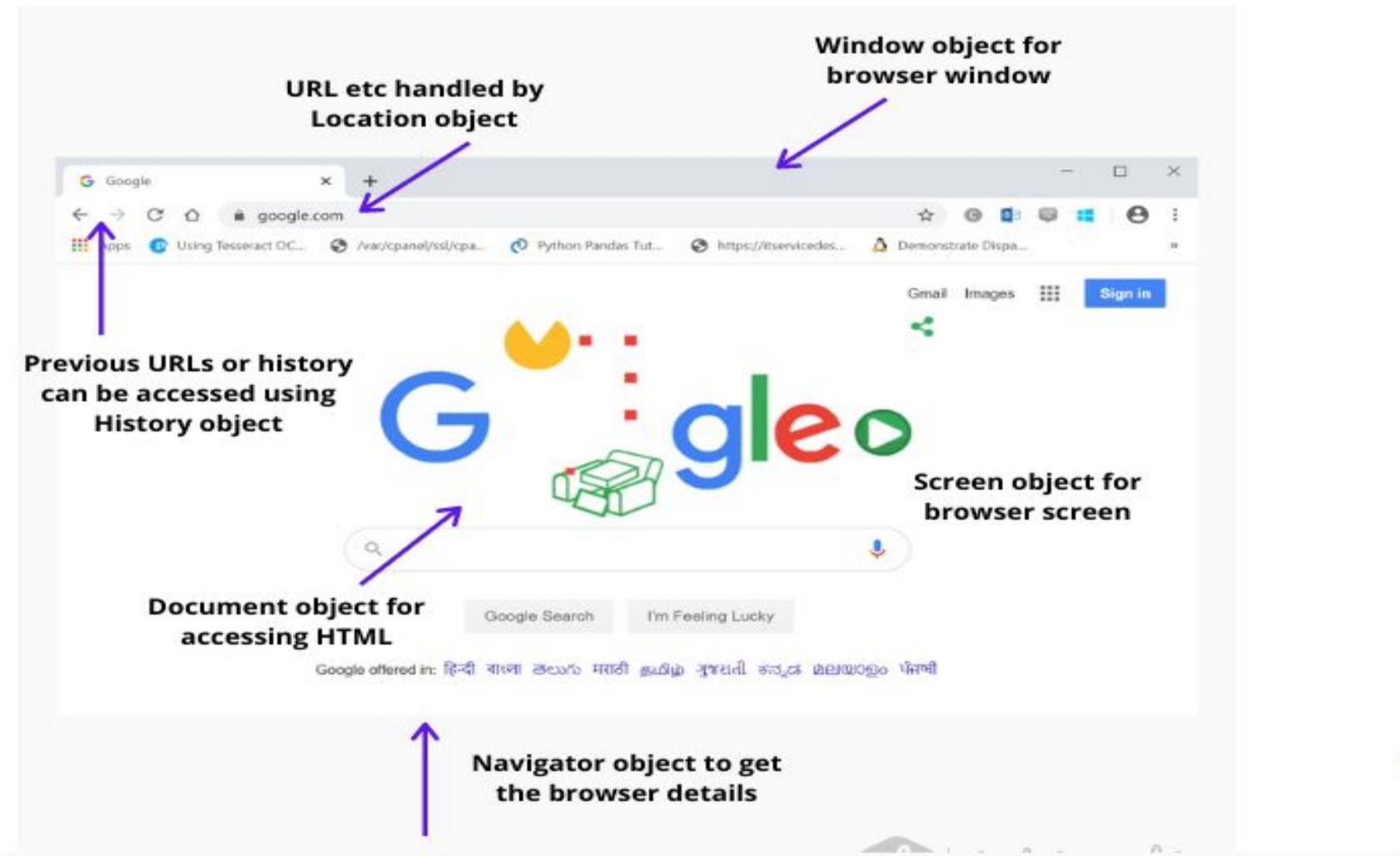
- The window.navigator object contains information about the visitor's browser.
- The window.navigator object can be written without the window prefix.
- Some examples:
- navigator.appName
- navigator.appCodeName
- navigator.platform
- **Example**
- ```
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML =
"navigator.appName is " + navigator.appName;
</script>
```

# Section 4. Screen

- The window.screen object contains information about the user's screen.
- **Screen** – get the information about the screen on which the browser is running by using the **screen** object.
- The window.screen object can be written without the window prefix.
- Properties:
  - screen.width
  - screen.height
  - screen.availWidth
  - screen.availHeight
  - screen.colorDepth
  - screen.pixelDepth
- **Example**
  - Display the width of the screen in pixels:
  - `document.getElementById("demo").innerHTML = "Screen Width: " + screen.width;`

# Section 5. History

- The window.history object contains the browsers history.
- To protect the privacy of the users, there are limitations to how JavaScript can access this object.
- **Some methods:**
  - history.back() - same as clicking back in the browser
  - history.forward() - same as clicking forward in the browser
  - History – manage the web browser's history stack with the history object.



The various objects that can be used to access various components of the browser window in the picture

# JavaScript Popup Boxes

- JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.
- **Alert Box:-**An alert box is often used if you want to make sure information comes through to the user.
- When an alert box pops up, the user will have to click "OK" to proceed.
- **Confirm Box:-**A confirm box is often used if you want the user to verify or accept something.
- When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
- If the user clicks "OK", the box returns **true**. If the user clicks "Cancel", the box returns **false**.
- **Prompt Box:-**A prompt box is often used if you want the user to input a value before entering a page.
- When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
- If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null

# Example

- `alert("I am an alert box!");`
- `if (confirm("Press a button!")) {  
 txt = "You pressed OK!";  
} else {  
 txt = "You pressed Cancel!";  
}`
- `let person = prompt("Please enter your name", "Harry Potter");  
let text;  
if (person == null || person == "") {  
 text = "User cancelled the prompt.";  
} else {  
 text = "Hello " + person + "! How are you today?";  
}`

# Form Validation

Web Technologies

# Data Validation

- Data validation is the process of ensuring that user input is clean, correct, and useful.
- Typical validation tasks are:
  - has the user filled in all required fields?
  - has the user entered a valid date?
  - has the user entered text in a numeric field?
- Most often, the purpose of data validation is to ensure correct user input.
- Validation can be defined by many different methods, and deployed in many different ways.
- **Server side validation** is performed by a web server, after input has been sent to the server.
- **Client side validation** is performed by a web browser, before input is sent to a web server.

# HTML Constraint Validation

- HTML5 introduced a new HTML validation concept called **constraint validation**.
- HTML constraint validation is based on:
  - Constraint validation **HTML Input Attributes**
  - Constraint validation **CSS Pseudo Selectors**
  - Constraint validation **DOM Properties and Methods**

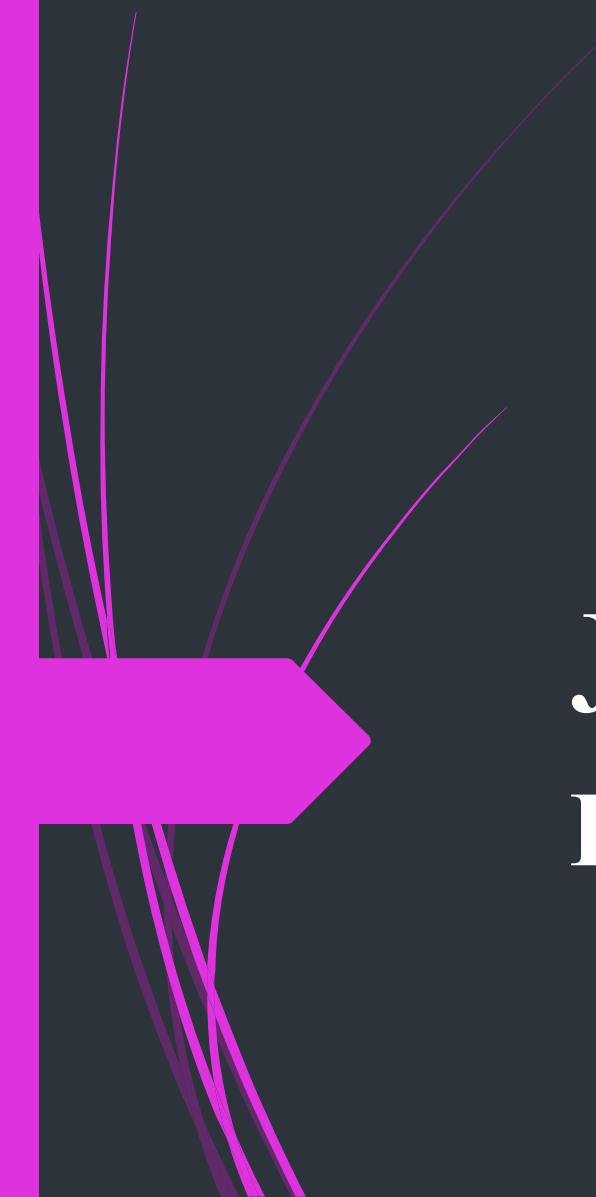
# Constraint Validation HTML Input Attributes

Attribute	Description
disabled	Specifies that the input element should be disabled
max	Specifies the maximum value of an input element
min	Specifies the minimum value of an input element
pattern	Specifies the value pattern of an input element
required	Specifies that the input field requires an element
type	Specifies the type of an input element

- The input read only attribute specifies that an input field is read-only.
- The input disabled attribute specifies that an input field should be disabled. this field is unusable and un-clickable.
- The input size attribute specifies the visible width, in characters, of an input field.
- The input max length attribute specifies the maximum number of characters allowed in an input field.
- The input placeholder attribute specifies a short hint that describes the expected value of an input field (a sample value or a short description of the expected format).
- The input required attribute specifies that an input field must be filled out before submitting the form.
- The input autofocus attribute specifies that an input field should automatically get focus when the page loads.
- The input autocomplete attribute specifies whether a form or an input field should have autocomplete on or off.







# Java Script Events

## Event Listeners

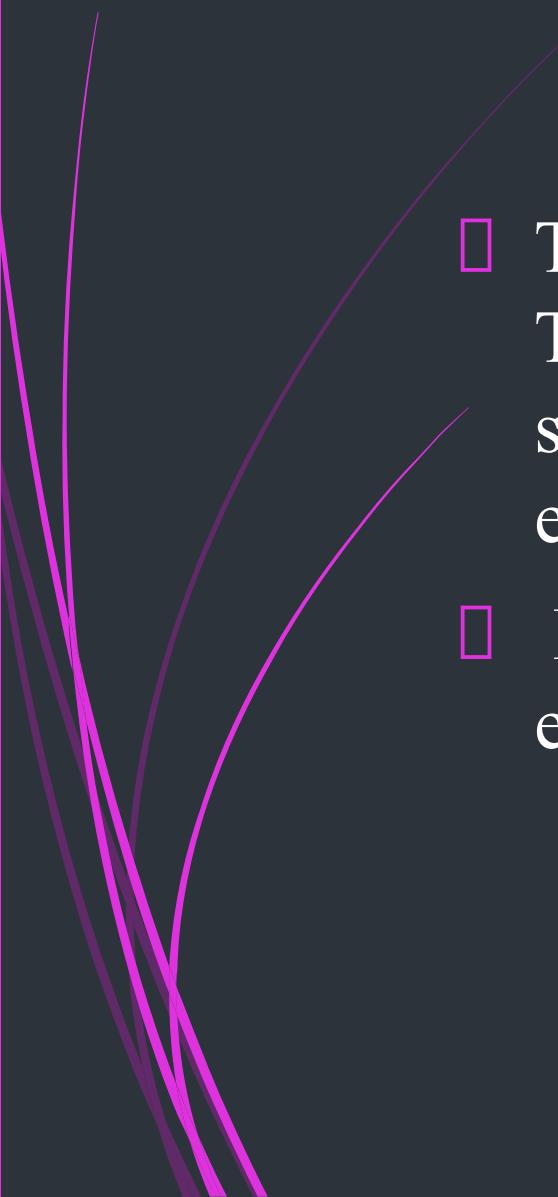
# Understanding Events

- An event is something that happens when user interact with the web page, such as
  - when he clicked a link or button,
  - entered text into an input box or textarea,
  - made selection in a select box,
  - pressed key on the keyboard,
  - moved the mouse pointer,
  - submits a form
- When an event occur, you can use a JavaScript event handler (or an event listener) to detect them and perform specific task or set of tasks.



# Reacting to Events

- A JavaScript can be executed when an event occurs, like when a user clicks on an HTML element.
- **Examples of HTML events:**
- When a user clicks the mouse
- When a web page has loaded
- When an image has been loaded
- When the mouse moves over an element
- When an input field is changed
- When an HTML form is submitted
- When a user strokes a key

- 
- There are several ways to assign an event handler. The simplest way is to add them directly to the start tag of the HTML elements using the special event-handler attributes.
  - For example, to assign a click, handler for a button element, we can use onclick attribute.

# Example

- <button type="button" onclick="alert('Hello World!')>Click Me</button>
- You can set up the event handler in an external Javascript file or with in the script.
  - Example:-
  - <button type="button" id="myBtn">Click Me</button>
  - <script>
  - function sayHello() {
  - alert('Hello World!');
  - }
  - document.getElementById("myBtn").onclick = sayHello;
  - </script>



## events groups

- In general, the events can be categorized into four main groups
  - mouse events,
  - keyboard events,
  - form events
  - document/window events.

# Mouse Events

- A mouse event is triggered when the user click some element, move the mouse pointer over an element, etc. Here're some most important mouse events and their event handler.
- **The Click Event (onclick):-**The click event occurs when a user clicks on an element on a web page. Often, these are form elements and links.
- **The Contextmenu Event (oncontextmenu):-**This event occurs when a user clicks the right mouse button on an element to open a context menu.
- **The Mouseover Event (onmouseover):-**The mouseover event occurs when a user moves the mouse pointer over an element.
- **The Mouseout Event (onmouseout):-** The mouseout event occurs when a user moves the mouse pointer outside of an element.

# Keyboard Events

- A keyboard event is fired when the user press or release a key on the keyboard. Here're some most important keyboard events and their event handler.
- **The Keydown Event (onkeydown)**:-The keydown event occurs when the user presses down a key on the keyboard.
- **The Keyup Event (onkeyup)** The keyup event occurs when the user releases a key on the keyboard.
- **The Keypress Event (onkeypress)**:-The keypress event occurs when a user presses down a key on the keyboard that has a character value associated with it. For example, keys like Ctrl, Shift, Alt, Esc, Arrow keys, etc. will not generate a keypress event, but will generate a keydown and keyup event.
-

# Form Events

- A form event is fired when a form control receive or loses focus or when the user modify a form control value such as by typing text in a text input, select any option in a select box etc. Here're some most important form events and their event handler.
- **The Focus Event (onfocus):-**  
The focus event occurs when the user gives focus to an element on a web page. You can handle the focus event with the onfocus event handler.
- **The Blur Event (onblur):-** The blur event occurs when the user takes the focus away from a form element or a window.
- **The Change Event (onchange)** The change event occurs when a user changes the value of a form element.
- **The Submit Event (onsubmit):-** The submit event only occurs when the user submits a form on a web page. You can handle it by onsubmit event handler.
-

# Document/Window Events

- Events are also triggered in situations when the page has loaded or when user resize the browser window, etc. Here're some most important document/window events and their event handler.
- **The Load Event (onload)** The load event occurs when a web page has finished loading in the web browser.
- **The Unload Event (onunload)** The unload event occurs when a user leaves the current web page.
- **The Resize Event (onresize)** The resize event occurs when a user resizes the browser window. The resize event also occurs in situations when the browser window is minimized or maximized.
-

# JavaScript Event Listeners

- The event listeners are just like event handlers, except that you can assign as many event listeners as you like to a particular event on particular element.
- For example. Suppose that you've created two functions and you try to execute both of them on click of the button using the `onclick` event handler
- If you run the above example and click the button element, only `secondFunction()` will be executed, because assigning the second event handler overwrites the first.
- In fact, you can only assign one event handler to a particular event on a particular element i.e. a single function per event per element. To deal with this problem W3C introduced more flexible event-model called *event listeners*.
- In addition to the event type and listener function parameter the `addEventListener()` accepts one more Boolean parameter `useCapture`. This is an optional parameter which specifies whether to use event bubbling or event capturing. Its basic syntax is:
- `target.addEventListener(event, function, useCapture);`



# Adding Event Listeners for Different Event Types

- Like event handler, you can assign different event listeners to different event types on the same element.
- The addEventListener() method attaches an event handler to an element without overwriting existing event handlers.
- You can add many event handlers to one element.
- You can add many event handlers of the same type to one element, i.e two "click" events.
- The following example will assign different event-listener functions to the "click", "mouseover", and "mouseout" events of a button element.
- Example: [different-Listeners.html](#)

# Adding Event Listeners to Window Object

- The addEventListener() method allows you to add event listeners to any HTML DOM elements, the document object, the window object, or any other object that support events,
- e.g, Here's an example that attaches an event listener to the window
- "resize" event:
- Example:- event-to-window

# Event Bubbling or Event Capturing?

- There are two ways of event propagation in the HTML DOM, bubbling and capturing.
- Event propagation is a way of defining the element order when an event occurs. If you have a `<p>` element inside a `<div>` element, and the user clicks on the `<p>` element, which element's "click" event should be handled first?
- In *bubbling* the inner most element's event is handled first and then the outer: the `<p>` element's click event is handled first, then the `<div>` element's click event. ---false
- In *capturing* the outer most element's event is handled first and then the inner: the `<div>` element's click event will be handled first, then the `<p>` element's click event. ----true
- `addEventListener(event, function, useCapture);`
- **Example**
- `document.getElementById("myP").addEventListener("click", myFunction, true);`
- `document.getElementById("myDiv").addEventListener("click", myFunction, true);`

# Removing Event Listeners

- You can use the `removeEventListener()` method to remove an event listener that have been previously attached with the `addEventListener()`.
  
- `element.removeEventListener("mousemove", myFunction);`
- Here's an example: Remove-event

# HTML DOM Events

- HTML DOM events allow JavaScript to register different event handlers on elements in an HTML document.
- Events are normally used in combination with functions, and the function will not be executed before the event occurs (such as when a user clicks a button).

Event	Description
abort	The event occurs when the loading of a media is aborted
afterprint	The event occurs when a page has started printing, or if the print dialogue box has been closed
blur	The event occurs when an element loses focus
canplay	The event occurs when the browser can start playing the media (when it has buffered enough to begin)
change	The event occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>)
click	The event occurs when the user clicks on an element
contextmenu	The event occurs when the user right-clicks on an element to open a context menu
copy	The event occurs when the user copies the content of an element

# jQuery

---

# jQuery Introduction

- The purpose of jQuery is to make it much easier to use JavaScript on your website.
- jQuery is a lightweight, "write less, do more", JavaScript library.
- The purpose of jQuery is to make it much easier to use JavaScript on your website.
- jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.
- It is most widely used JS Library
- Cross browser compatibility
- Open source and free to use

# What is jQuery?

**jQuery is a fast, small, and feature-rich JavaScript Library.**

---

- ◆ **jQuery is a lightweight, "write less, do more", JavaScript library.**
- ◆ **The purpose of jQuery is to make it much easier to use JavaScript on your website.**
- ◆ **jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.**
- ◆ **jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.**
- ◆ **The jQuery library contains the following features:**
  - 1. HTML/DOM manipulation
  - 2. CSS manipulation
  - 3. Utilities
  - 4. HTML event methods
  - 5. Effects and animations
  - 6. AJAX

# Why jQuery?

---

There are lots of other JavaScript libraries out there, but jQuery is probably the most popular, and also the most extendable.

Many of the biggest companies on the Web use jQuery, such as:

Google

Microsoft

IBM

Netflix

# jQuery Syntax

- ❖ With jQuery you select (query) HTML elements and perform "actions" on them. The jQuery syntax is tailor-made for **selecting** HTML elements and performing some **action** on the element(s).
- 

- ❖ Basic syntax is: **`$(selector).action()`**

- ❖ A \$ sign to define/access jQuery

- ❖ A *(selector)* to "query (or find)" HTML elements

- ❖ A jQuery *action()* to be performed on the element(s)

- ❖ **Examples:**

- ❖ `$(this).hide()` - hides the current element.

- ❖ `$("p").hide()` - hides all `<p>` elements.

- ❖ `$(".test").hide()` - hides all elements with `class="test"`.

- ❖ `$("#test").hide()` - hides the element with `id="test"`.

# The Document Ready Event

---

You might have noticed that all jQuery methods in our examples, are inside a document ready event:

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

or

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

This is to prevent any jQuery code from running before the document is finished loading (is ready).

# jQuery Selectors

---

- ✓ jQuery selectors are one of the most important parts of the jQuery library.
- ✓ jQuery selectors allow you to select and manipulate HTML element(s).
- ✓ jQuery selectors are used to "find" (or select) HTML elements based on their name, id, classes, types, attributes, values of attributes and much more.
- ✓ All selectors in jQuery start with the dollar sign and parentheses: `\$()`.

# More Examples of jQuery Selectors

Syntax	Description
<code>\$("*")</code>	Selects all elements
<code>\$(this)</code>	Selects the current HTML element
<code>\$(".p.intro")</code>	Selects all <code>&lt;p&gt;</code> elements with class="intro"
<code>\$(".p:first")</code>	Selects the first <code>&lt;p&gt;</code> element
<code>\$(".ul li:first")</code>	Selects the first <code>&lt;li&gt;</code> element of the first <code>&lt;ul&gt;</code>
<code>\$("[href]")</code>	Selects all elements with an href attribute
<code>\$(".a[target='_blank']")</code>	Selects all <code>&lt;a&gt;</code> elements with a target attribute value equal to "_blank"
<code>\$(".a[target!='_blank']")</code>	Selects all <code>&lt;a&gt;</code> elements with a target attribute value NOT equal to "_blank"
<code>\$(":button")</code>	Selects all <code>&lt;button&gt;</code> elements and <code>&lt;input&gt;</code> elements of type="button"
<code>\$(".tr:even")</code>	Selects all even <code>&lt;tr&gt;</code> elements
<code>\$(".tr:odd")</code>	Selects all odd <code>&lt;tr&gt;</code> elements

# Functions In a Separate File

---

If your website contains a lot of pages, and you want your jQuery functions to be easy to maintain, you can put your jQuery functions in a separate .js file.

## Example

```
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script src="my_jquery_functions.js"></script>
</head>
```

# jQuery Event Methods

---

- jQuery is tailor-made to respond to events in an HTML page.
- All the different visitors' actions that a web page can respond to are called events.
- An event represents the precise moment when something happens.
- Examples:
  - moving a mouse over an element
  - selecting a radio button
  - clicking on an element
- The term "**fires/fired**" is often used with events. Example: "The keypress event is fired, the moment you press a key".

# Here are some common DOM events:

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	Unload

# Commonly Used jQuery Event Methods

---

`$(document).ready():-`

The `$(document).ready()` method allows us to execute a function when the document is fully loaded.

`click():-`The `click()` method attaches an event handler function to an HTML element.

`dblclick():-`The `dblclick()` method attaches an event handler function to an HTML element.

`mouseenter():-` method attaches an event handler function to an HTML element.

# The on() Method

The on() method attaches one or more event handlers for the selected elements.

Example

---

Example

```
$("p").on({  
    mouseenter: function(){  
        $(this).css("background-color", "lightgray");  
    },  
    mouseleave: function(){  
        $(this).css("background-color", "lightblue");  
    },  
    click: function(){  
        $(this).css("background-color", "yellow");    }    });
```

# jQuery Effects - Hide and Show

---

Hide, Show, Toggle, Slide, Fade, and Animate.

jQuery hide() and show()

With jQuery, you can hide and show HTML elements with the hide() and show() methods.

Example

```
$("#hide").click(function(){
  $("p").hide();
});
```

```
$("#show").click(function(){
  $("p").show();
});
```

# Syntax:

---

`$(selector).hide(speed,callback);`

`$(selector).show(speed,callback);`

The optional speed parameter specifies the speed of the hiding/showing, and can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after the hide() or show() method completes (you will learn more about callback functions in a later chapter).

# jQuery toggle()

---

You can also toggle between hiding and showing an element with the toggle() method.

Syntax:

```
$(selector).toggle(speed,callback);
```

The optional speed parameter can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after toggle() completes.

Example

```
$("button").click(function(){
  $("p").toggle();
});
```

# jQuery Effects - Fading

---

With jQuery you can fade elements in and out of visibility.

## jQuery Fading Methods

With jQuery you can fade an element in and out of visibility.

jQuery has the following fade methods:

`fadeIn()`

`fadeOut()`

`fadeToggle()`

`fadeTo()`

# Example

---

```
$("button").click(function(){
  $("#div1").fadeIn();
  $("#div2").fadeIn("slow");
  $("#div3").fadeIn(3000);
});
```

# jQuery

- Style Properties
  - Effects

# Style Properties

- The jQuery library supports nearly all of the selectors included in Cascading Style Sheet (CSS) specifications 1 through 3, as outlined on the World Wide Web Consortium's site.
- Using JQuery library developers can enhance their websites without worrying about browsers and their versions as long as the browsers have JavaScript enabled.
- Most of the JQuery CSS Methods do not modify the content of the jQuery object and they are used to apply CSS properties on DOM elements.
- Apply CSS Properties
- This is very simple to apply any CSS property using JQuery method
- **css( PropertyName, PropertyValue ).**

# CSS-related properties

- These methods get and set CSS-related properties of elements.
- **.css():-**Get the value of a computed style property for the first element in the set of matched elements or set one or more CSS properties for every matched element.
- **.height():-**Get the current computed height for the first element in the set of matched elements or set the height of every matched element.
- **width():-**Get the current computed width for the first element in the set of matched elements or set the width of every matched element.
- **.innerHeight():-**Get the current computed inner (including padding but not border) for the first element in the set of matched elements or set the inner width of every matched element.
- **.innerWidth():-**Get the current computed inner width (including padding but not border) for the first element in the set of matched elements or set the inner width of every matched element.

# Cont...

- **jQuery.cssNumber**:-An object containing all CSS properties that may be used without a unit. The .css() method uses this object to see if it may append px to unitless values.
- **.offset()**:-Get the current coordinates of the first element, or set the coordinates of every element, in the set of matched elements, relative to the document.
- **.outerHeight()**:-Get the current computed outer height (including padding, border, and optionally margin) for the first element in the set of matched elements or set the outer height of every matched element.
- **.outerWidth()**
- **.position()**:- *Get the current coordinates of the first element in the set of matched elements, relative to the offset parent.*
- **.scrollLeft()**:-Get the current horizontal position of the scroll bar for the first element in the set of matched elements or set the horizontal position of the scroll bar for every matched element.
- **.scrollTop()**:-Get the current vertical position of the scroll bar for the first element in the set of matched elements or set the vertical position of the scroll bar for every matched element.

# jQuery css() Method

- The css() method sets or returns one or more style properties for the selected elements.
- Return a CSS Property
- To return the value of a specified CSS property, use the following syntax:
- `css("propertyname");`
- The following example will return the background-color value of the FIRST matched element:
- Example
- `$("p").css("background-color");`

# Set a CSS Property

- To set a specified CSS property, use the following syntax:
- `css("propertyname","value");`
- The following example will set the background-color value for ALL matched elements:
- Example
- `$(".p").css("background-color", "yellow");`
- [.css\( propertyName, value \)](#)
- [.css\( propertyName, function \)](#)
- [.css\( properties \)](#)

# Set Multiple CSS Properties

- You can apply multiple CSS properties using a single JQuery method **CSS( {key1:val1, key2:val2....})**. You can apply as many properties as you like in a single call.
- To set multiple CSS properties, use the following syntax:
- Selector.css({"*propertynname*": "*value*", "*propertynname*": "*value*", ...});
- Example
- `$( "p" ).css({ "background-color": "yellow", "font-size": "200%" })`

## . Setting Element Width & Height

- The **width( val )** and **height( val )** method can be used to set the width and height respectively of any element.
- Get the current computed height for the first element in the set of matched elements or set the height of every matched element.
- **height():-** This method does not accept any argument.
- **This method is also able to find the height of the window and document.**
  
- **// Returns height of browser viewport**
- **\$( window ).height();**
- 
- **// Returns height of HTML document**
- **\$( document ).height();**

# Cont...

- **.height( value )**
- Here:- value
- Type: String or Number
- An integer representing the number of pixels, or an integer with an optional unit of measure appended (as a string).
- **.height( function ):-**
- function
- Type: Function( Integer index, Integer height ) => String or Number
- A function returning the height to set. Receives the index position of the element in the set and the old height as arguments. Within the function, this refers to the current element in the set.

# width()

- .width() not accept any arguments
- .width( value ) set the width
- ..width( function ):-Type: Function( Integer index, Integer value ) => String or Number
  - A function returning the width to set. Receives the index position of the element in the set and the old width as arguments. Within the function, this refers to the current element in the set.
- // Returns width of browser viewport
- \$( window ).width();
- // Returns width of HTML document
- \$( document ).width();

# .innerHeight()

- **.innerHeight()**
- *current computed height for the first element in the set of matched elements, including padding but not border.*
- **.innerHeight( value ), .innerHeight( function )**
- *Set the CSS inner height of each element in the set of matched elements.*

# .position()

- *Get the current coordinates of the first element in the set of matched elements, relative to the offset parent.*
- The .position() method allows us to retrieve the current position of an element (specifically its margin box) relative to the offset parent (specifically its padding box, which excludes margins and borders).

# jQuery Effects

- jQuery provides a trivially simple interface for doing various kind of amazing effects.
- jQuery is a collection of several useful methods, which can be used to carry out many common tasks in JavaScript.
- Developers can use jQuery to do cool animations like fade in or fade out. They can also change the CSS class of a component dynamically e.g. making a component active or inactive.
- The jQuery library provides several methods for adding the animation or effects to the selected HTML element on the web page to the selected HTML elements.

# The following table lists the common jQuery effects.

jQuery Method	Description
hide()	Hide the selected HTML element
show()	Display the selected HTML elements
toggle()	Display or hide the selected HTML elements
fadeIn()	Perform the fade-in effect on the selected HTML elements
fadeOut()	Perform the fade-out effect on the selected HTML elements
fadeTo()	Perform the fade-out effect on the selected HTML elements to a given opacity
fadeToggle()	Perform the fade-in or fade-out effect on the selected HTML elements.
slideDown()	Perform the sliding-down motion effect to display the selected HTML elements
slideUp()	Perform the sliding-up motion effect to hide the selected HTML elements
slideToggle()	Perform the sliding-up or sliding-down motion effect to hide or show the selected HTML elements
animate()	Perform the custom animation using the element's CSS properties
delay()	Using the timer set, it delays the execution of the items in the queue
stop()	Stop the current animation which is running on the selected HTML elements

# Showing and Hiding Elements

- The commands for showing and hiding elements are pretty much what we would expect – **show()** to show the elements in a wrapped set and **hide()** to hide them.
- **Syntax**
- Here is the simple syntax for show() method –
- **[selector].show( speed, [callback] );**
- Here is the description of all the parameters –
- **speed** – A string representing one of the three predefined speeds ("slow", "normal", or "fast") or the number of milliseconds to run the animation (e.g. 1000).
- **callback** – This optional parameter represents a function to be executed whenever the animation completes; executes once for each element animated against.

# hide() method –

- [selector].hide( speed, [callback] );
- Here is the description of all the parameters –
  - speed – A string representing one of the three predefined speeds ("slow", "normal", or "fast") or the number of milliseconds to run the animation (e.g. 1000).
  - callback – This optional parameter represents a function to be executed whenever the animation completes; executes once for each element animated against.

# Toggling the Elements

- jQuery provides methods to toggle the display state of elements between revealed or hidden. If the element is initially displayed, it will be hidden; if hidden, it will be shown.
- **Syntax**
- Here is the simple syntax for one of the toggle() methods –
- **[selector].toggle([speed][, callback]);**
- Here is the description of all the parameters –
- speed – A string representing one of the three predefined speeds ("slow", "normal", or "fast") or the number of milliseconds to run the animation (e.g. 1000).
- callback – This optional parameter represents a function to be executed whenever the animation completes; executes once for each element animated against.

# slideDown()

- The slideDown() method reveals all matched elements by adjusting their height and firing an optional callback after completion.
- **Syntax**
- Here is the simple syntax to use this method –
- **selector.slideDown( speed, [callback] );**
- Parameters
- speed – A string representing one of the three predefined speeds ("slow", "normal", or "fast") or the number of milliseconds to run the animation (e.g. 1000).
- callback – This is optional parameter representing a function to call once the animation is complete.

# slideToggle()

- The slideToggle() method toggles the visibility of all matched elements by adjusting their height and firing an optional callback after completion.
- **Syntax**
- Here is the simple syntax to use this method –
- selector.slideToggle( speed, [callback] );
- Parameters
- speed – A string representing one of the three predefined speeds ("slow", "normal", or "fast") or the number of milliseconds to run the animation (e.g. 1000).
- callback – This is optional parameter representing a function to call once the animation is complete.

# stop()

- The stop( [clearQueue, gotoEnd] ) method stops all the currently running animations on all the specified elements.
- **Syntax**
- Here is the simple syntax to use this method –
- **selector.stop( [clearQueue], [gotoEnd] ) ;**
- Parameters
- clearQueue – This is optional boolean parameter. When set to true clears the animation queue, effectively stopping all queued animations.
- gotoEnd – This is optional boolean parameter. A Boolean (true/false) that when set to true causes the currently playing animation to immediately complete, including resetting original styles on show and hide and calling the callback function.

# jQuery

# Traversing the DOM

# What is Traversing?

- ▶ jQuery traversing, which means "move through", are used to "find" (or select) HTML elements based on their relation to other elements.
- ▶ Start with one selection and move through that selection until you reach the elements you desire.

# Traversing the DOM

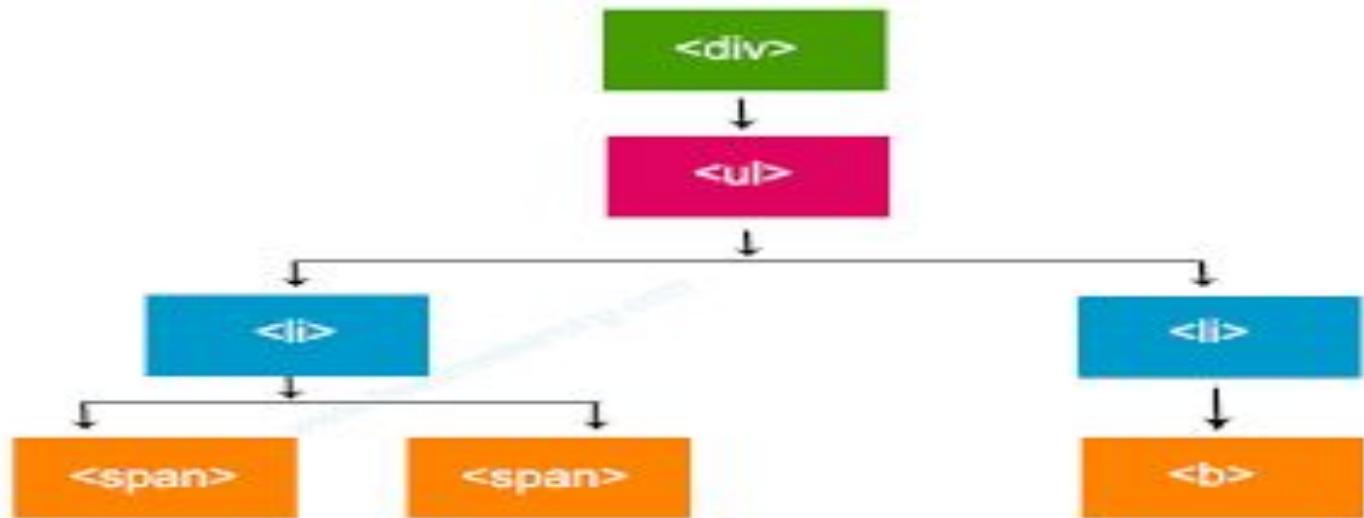
- ▶ jQuery provides a variety of methods that allow us to traverse the DOM.
- ▶ The largest category of traversal methods are tree-traversal.

HTML page as a tree (DOM tree).

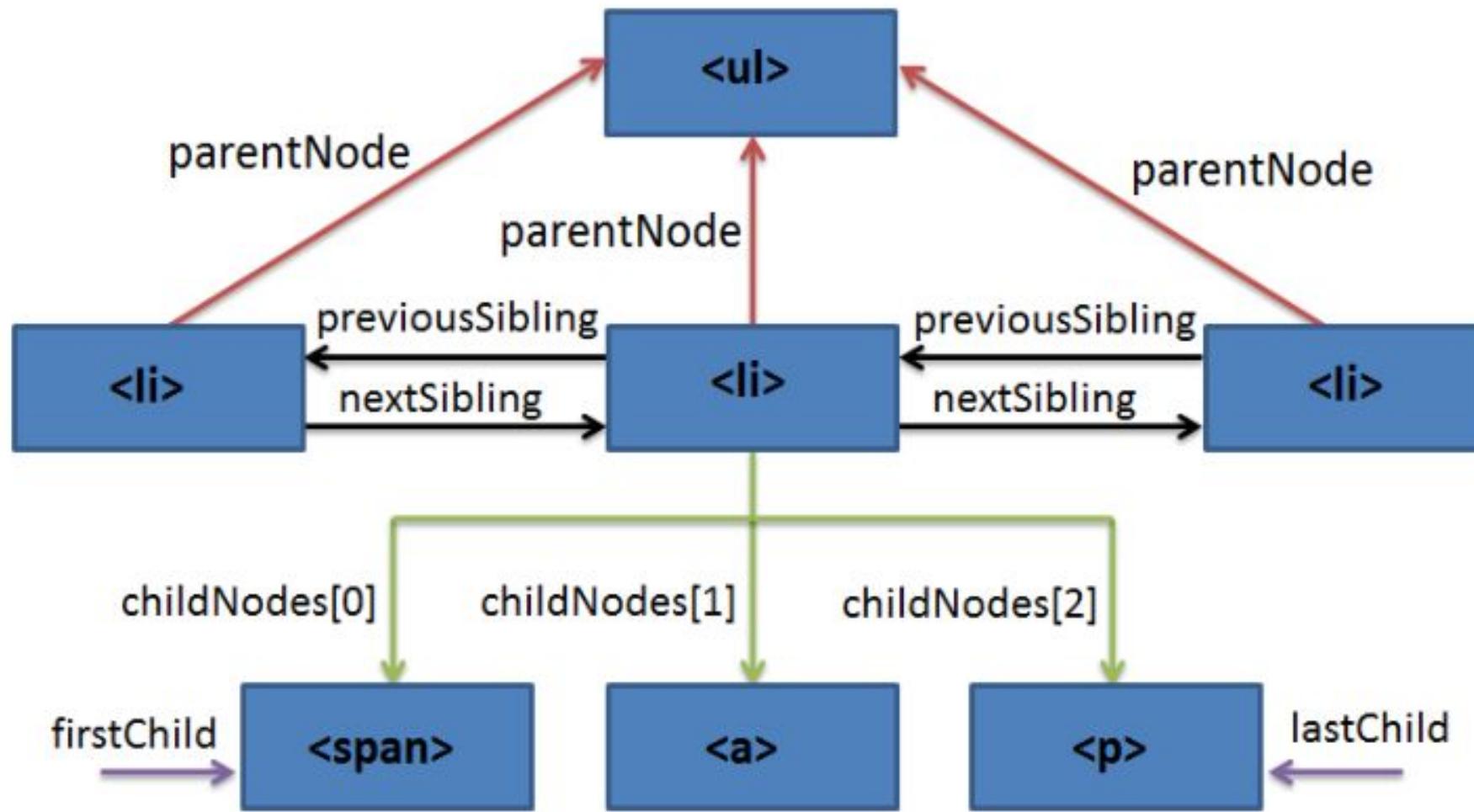
With jQuery traversing, you can easily move up (ancestors), down (descendants) and sideways (siblings) in the tree, starting from the selected (current) element. This movement is called traversing - or moving through - the DOM tree.

## Cont...

- ▶ The `<div>` element is the **parent** of `<ul>`, and an **ancestor** of everything inside of it
- ▶ The `<ul>` element is the **parent** of both `<li>` elements, and a **child** of `<div>`
- ▶ The left `<li>` element is the **parent** of `<span>`, **child** of `<ul>` and a **descendant** of `<div>`
- ▶ The `<span>` element is a **child** of the left `<li>` and a **descendant** of `<ul>` and `<div>`
- ▶ The two `<li>` elements are **siblings** (they share the same parent)
- ▶ The right `<li>` element is the **parent** of `<b>`, **child** of `<ul>` and a **descendant** of `<div>`
- ▶ The `<b>` element is a **child** of the right `<li>` and a **descendant** of `<ul>` and `<div>`



# DOM traversing Ex



# jQuery Traversing - Ancestors

- ▶ With jQuery you can traverse up the DOM tree to find ancestors of an element.
- ▶ An ancestor is a parent, grandparent, great-grandparent, and so on.
- ▶ Traversing Up the DOM Tree
- ▶ Three useful jQuery methods for traversing up the DOM tree are:
- ▶ `parent()`
- ▶ `parents()`
- ▶ `parentsUntil()`

# jQuery Traversing - Descendants

- ▶ With jQuery you can traverse down the DOM tree to find descendants of an element.
- ▶ A descendant is a child, grandchild, great-grandchild, and so on.
- ▶ Traversing Down the DOM Tree
- ▶ Two useful jQuery methods for traversing down the DOM tree are:
- ▶ `children()`
- ▶ `find()`
- ▶ `jQuery children()` Method
- ▶ The `children()` method returns all direct children of the selected element.
- ▶ This method only traverses a single level down the DOM tree.

# jQuery Traversing Methods

Method	Description
add()	Adds elements to the set of matched elements
addBack()	Adds the previous set of elements to the current set
children()	Returns all direct children of the selected element
closest()	Returns the first ancestor of the selected element
contents()	Returns all direct children of the selected element (including text and comment nodes)
eq()	Returns an element with a specific index number of the selected elements
filter()	Reduce the set of matched elements to those that match the selector or pass the function's test
find()	Returns descendant elements of the selected element
first()	Returns the first element of the selected elements
parent()	Returns the direct parent element of the selected element
parents()	Returns all ancestor elements of the selected element
siblings()	Returns all sibling elements of the selected element
prev()	Returns the previous sibling element of the selected element

# add()

- ▶ The add() method adds elements to an existing group of elements.
- ▶ This method adds elements on the whole document, or just inside context elements if the *context* parameter is specified.
- ▶ Syntax
- ▶ `$selector.add(element,context)`
  
- ▶ Example
- ▶ Add `<p>` and `<span>` elements to an existing group of elements (`<h1>`):
- ▶ `$("h1").add("p").add("span")`

# jQuery children() Method

- ▶ The `children()` method returns all direct children of the selected element.
- ▶ **The DOM tree:** This method only traverse a single level down the DOM tree. To traverse down multiple levels (to return grandchildren or other descendants), use the [find\(\)](#) method.
- ▶ **Tip:** To traverse a single level up the DOM tree, or all the way up to the document's root element (to return parents or other ancestors), use the [parent\(\)](#) or [parents\(\)](#) method.
- ▶ Syntax
- ▶ `$(selector).children(filter)`

# jQuery each() Method

- ▶ The each() method specifies a function to run for each matched element.
- ▶ Syntax
- ▶ `$(selector).each(function(index,element))`
- ▶ Example
- ▶ Alert the text of each `<li>` element:
- ▶ 

```
$("button").click(function(){
    $("li").each(function(){
        alert($(this).text())
    });
});
```

# jQuery find() Method

- ▶ The `find()` method returns descendant elements of the selected element.
- ▶ A descendant is a child, grandchild, great-grandchild, and so on.
- ▶ Syntax
- ▶ `$(selector).find(filter)`
- ▶ Example
- ▶ Return all `<span>` elements that are descendants of `<ul>`:
- ▶ 

```
$(document).ready(function(){  
    $("ul").find("span").css({"color": "red", "border": "2px solid red"});  
});
```

# jQuery prev() Method

- ▶ The prev() method returns the previous sibling element of the selected element.
- ▶ Sibling elements are elements that share the same parent.
- ▶ The DOM tree: This method traverse backwards along the previous sibling of DOM elements.
- ▶ Related methods:
  - ▶ prevAll() - returns all previous sibling elements of the selected element
  - ▶ prevUntil() - returns all previous sibling elements between two given arguments
- ▶ Syntax
- ▶ `$(selector).prev(filter)`

# Introduction to Jquery Plugin



# Plugins

- ▶ A plug-in is piece of code written in a standard JavaScript file. These files provide useful jQuery methods which can be used along with jQuery library methods.
- ▶ A jQuery plugin is simply a new method that we use to extend jQuery's prototype object.
- ▶ By extending the prototype object you enable all jQuery objects to inherit any methods that you add.
- ▶ As established, whenever you call `jQuery()` you're creating a new jQuery object, with all of jQuery's methods inherited.
- ▶ You can make your own plugins and use them privately in your code or you can release them into the wild.
- ▶ There are thousands of jQuery plugins available online. The barrier to creating a plugin of your own is so low that you'll want to do it straight away!

# Finding & Evaluating Plugins

- ▶ One of the most celebrated aspects of jQuery is its extensive plugin ecosystem. From table sorting to form validation to auto completion – if there's a need for it, chances are good that someone has written a plugin for it.
- ▶ The easiest way to find plugins is to search Google or the [jQuery Plugins Registry](#).
- ▶ Once you choose a plugin, you'll need to add it to your page.
- ▶ Download the plugin, unzip it if necessary, place it within your application's directory structure, then include the plugin in your page using a script tag (after you include jQuery lib file).

# How to use Plugins

- To make a plug-in's methods available to us, we include plug-in file very similar to jQuery library file in the <head> of the document.
- We must ensure that it appears after the main jQuery source file, and before our custom JavaScript code.
- Following example shows how to include **jquery.plug-in.js** plugin –
- <head>
- <title>The jQuery Example</title>
- 
- <script type = "text/javascript"
- src = "https://www.tutorialspoint.com/jquery/jquery-3.6.0.js">
- </script>
- 
- <script src = "jquery.plug-in.js" type = "text/javascript"></script>
- <script src = "custom.js" type = "text/javascript"></script>
- </head>

# 3 ways to use plugin

- ▶ Using third party plugin
- ▶ Implementing plugin
- ▶ Internal method creating

# How to Create a Basic Plugin

- ▶ Before we write our own plugins, we must first understand a little about how jQuery works.
- ▶ Whenever you use the `$` function to select elements, it returns a jQuery object. This object contains all of the methods you've been using (`.css()`, `.click()`, etc.).
- ▶ The jQuery object gets these methods from the **`$.fn. object`**.
- ▶ This object contains all of the jQuery object methods, and if we want to write our own methods, it will need to contain those as well.

# Basic Plugin Authoring

- Let's start with simple Plugin that adds some basic styles to retrieved elements.
- All we need to do is to add a function called myStyle to \$.fn and it will be available just like any other jQuery Object method.

Ex:-

```
<div>
<a href="#">MY Style</a>
</div>
<script>
$.fn.myStyle=function() {
this.css("color","White");
this.css("background-color","#ff9900");
return this;
};
$("a").myStyle();
```

# Chaining

- ▶ One of jQuery's features is chaining, when you link five or six actions onto one selector. This is accomplished by having all jQuery object methods return the original jQuery object again.
- ▶ Making our plugin method chainable takes one line of code:
- ▶ Add final line “return this”.

# \$ Alias and Adding Scope

- \$ variable is very popular among JavaScript libraries, and if you're using another library with jQuery, you will have to make jQuery not use the \$ with `jQuery.noConflict()`.
- To over come such problem and work well with other plugins, and still use the jQuery \$ alias, we need to put all of our code inside of **an Immediately Invoked Function Expression[IIFE]**, and then pass the function `jQuery`, and name the parameter \$.
- And another use of such Immediately Invoked Function is to allow us to have our own private variables.
- For example: In our plugin we want a different color green, and we want to store it in a variable.

```
(function ( $ ) {  
    $.fn.greenify = function() {  
        this.css( "color", "green" );  
        return this;  
    };  
  
}( jQuery ));
```

```
(function ( $ ) {  
    var shade = "#556b2f";  
    $.fn.greenify = function() {  
        this.css( "color", shade );  
        return this;  
    };}( jQuery ));
```

# Minimizing Plugin Footprint

- It's good practice when writing plugins to only take up one slot within `$.fn`. This reduces both the chance that your plugin will be overridden, and the chance that your plugin will override other plugins.

```
(function( $ ) {  
    $.fn.openPopup = function() {  
        // Open popup code.  
    };  
  
    $.fn.closePopup = function() {  
        // Close popup code.  
    };  
}( jQuery ));
```

```
(function( $ ) {  
    $.fn.popup = function( action ) {  
        if ( action === "open" ) {  
            // Open popup code.  
        }  
        if ( action === "close" ) {  
            // Close popup code.  
        }  
    };  
}( jQuery ));
```

# Using the each() Method

- ▶ Your typical jQuery object will contain references to any number of DOM elements, and that's why jQuery objects are often referred to as collections.
- ▶ If you want to do any manipulating with specific elements (e.g. getting a data attribute, calculating specific positions) then you need to use `.each()` to loop through the elements.

```
(function ($) {
  $.fn.myNewPlugin = function() {

    return this.each(function() {
      // Do something to each element here.
    });
  };
})(jQuery);
```

# Accepting Options

- As your plugins get more and more complex, it's a good idea to make your plugin customizable by accepting options. The easiest way to do this, especially if there are lots of options, is with an object literal. Let's change our greenify plugin to accept some options.

```
(function ( $ ) {  
    $.fn.greenify = function( options ) {  
        // This is the easiest way to have default options.  
        var settings = $.extend({  
            // These are the defaults.  
            color: "#556b2f",  
            backgroundColor: "white"  
        }, options );  
        // Greenify the collection based on the settings variable.  
        return this.css({  
            color: settings.color,  
            backgroundColor: settings.backgroundColor  
        });  
    };  
});
```

# Jquery Events

Web Technologies

K.Krishna Priya

# What are Events?

- All the different visitors' actions that a web page can respond to are called events.
- An event represents the precise moment when something happens.
- jQuery is tailor-made to respond to events in an HTML page.
- Examples:
- moving a mouse over an element
- selecting a radio button
- clicking on an element
- The term "**fires/fired**" is often used with events. Example: "The keypress event is fired, the moment you press a key".

# Here are some common DOM events:

A jQuery Event is the result of an action that can be detected by jQuery (JavaScript). When these events are triggered, you can then use a custom function to do pretty much whatever you want with the event. These custom functions are called **Event Handlers**.

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload
Mouse Events	Keyboard Events	Form Events	Document/Window Events

# jQuery Syntax For Event Methods

- Following are the examples of some common events –
  - A mouse click
  - A web page loading
  - Taking mouse over an element
  - Submitting an HTML form
  - A keystroke on your keyboard, etc
- In jQuery, most DOM events have an equivalent jQuery method.
- ```
$("p").click(function(){  
    // action goes here!!  
});
```

# Commonly Used jQuery Event Methods

- **The Document Ready Event**
- This is to prevent any jQuery code from running before the document is finished loading (is ready).
- It is good practice to wait for the document to be fully loaded and ready before working with it. This also allows you to have your JavaScript code before the body of your document, in the head section.
- `$(document).ready(function(){`

*// jQuery methods go here...*

`});`

# Example of focus() and blur()

- \$(document).ready(function(){
  - \$("input").focus(function(){
    - \$(this).css("background-color", "yellow");
    - });
    - \$("input").blur(function(){
      - \$(this).css("background-color", "green");
      - });
      - });

# Methods.....

- **click():-** method attaches an event handler function to an HTML element.
- **dblclick(), mouseenter(), mouseleave(), mousedown(), mouseup(), blur(), on()**
- **hover():-** The hover() method takes two functions and is a combination of the mouseenter() and mouseleave() methods.

# Example of on()

- The **on()** method attaches one or more event handlers for the selected elements.
- Example
- ```
$("p").on({
  mouseenter: function(){
    $(this).css("background-color", "lightgray");
  },
  mouseleave: function(){
    $(this).css("background-color", "lightblue");
  },
  click: function(){
    $(this).css("background-color", "yellow");
  }
});
```

# Hover()

- \$(document).ready(function(){
  - \$("#p1").hover(function(){
    - alert("You entered p1!");
    - },
    - function(){
      - alert("Bye! You now leave p1!");
      - });
      - });



# Bootstrap

Introduction

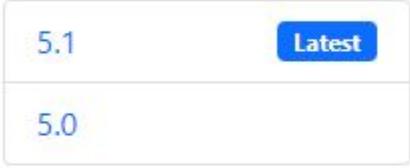
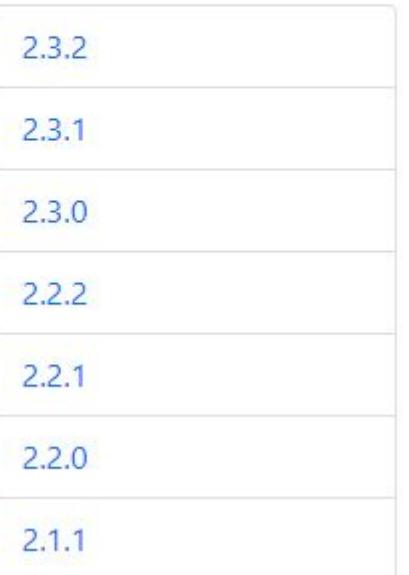
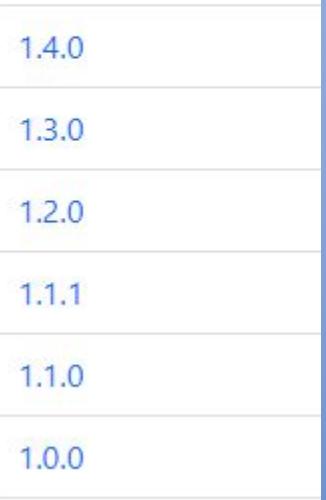


# What is Twitter Bootstrap?

- Bootstrap is a sleek, intuitive, and powerful, mobile first front-end framework for faster and easier web development. It uses HTML, CSS and Javascript.
- Bootstrap was developed by *Mark Otto* and *Jacob Thornton* at *Twitter*. It was released as an open source product in August 2011 on GitHub.

# Bootstrap

- Bootstrap is an Open-Source front-end Framework.
- Bootstrap, from v1 through v5.
- **Bootstrap 5.0.0 is the current version of bootstrap**

v5.x	v4.x	v3.x	v2.x	v1.x
Current major release. Last update was v5.1.0. 	Our previous major release with its minor releases. Last update was v4.6.0. 	Every minor and patch release from v3 is listed below. Last update was v3.4.1. 	Every minor and patch release from v2 is listed below. 	Every minor and patch release from v1 is listed below. 

## What's new in Bootstrap 5 (in compare to Bootstrap 4)

- **jQuery isn't required anymore** (however you can still use jQuery methods if you wish)
- Dropped support for **Internet Explorer**
- The use of JavaScript is **minimized in favor of CSS (CSS custom properties)**
- Almost each option is **available as data-attribute** (it can be set manually without JS)
- Enhanced **Grid system**
- Enhanced **modularity**
- Enhanced **customization**
- New components/helpers/utilities/variations
- Some components were removed (i.e. Jumbotron)
- Enhanced **Icons** (SVG Icon Library)

# What is Twitter Bootstrap?

- ✓ Twitter Bootstrap is Front End Frame work for Uis and themes
  - Bootstrap is the most popular HTML, CSS and JavaScript framework for developing a responsive and mobile friendly website.
  - It is absolutely free to download and use.
  - It includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many others.
  - It can also use JavaScript plug-ins.
  - It facilitates you to create responsive designs.

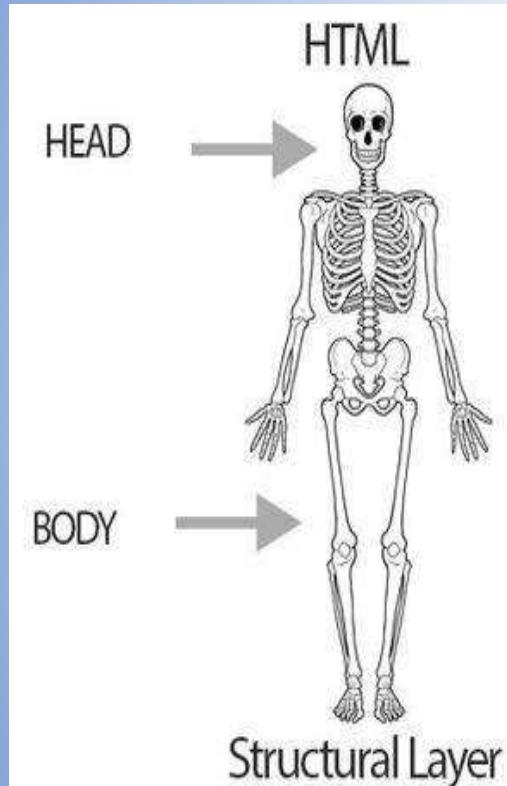
# Applications of Bootstrap

- **Scaffolding** – Bootstrap provides a basic structure with Grid System, link styles, and background. This is covered in detail in the section Bootstrap Basic Structure
- **CSS** – Bootstrap comes with the feature of global CSS settings, fundamental HTML elements styled and enhanced with extensible classes, and an advanced grid system. This is covered in detail in the section Bootstrap with CSS.
- **Components** – Bootstrap contains over a dozen reusable components built to provide iconography, dropdowns, navigation, alerts, pop-overs, and much more. This is covered in detail in the section Layout Components.
- **JavaScript Plugins** – Bootstrap contains over a dozen custom jQuery plugins. You can easily include them all, or one by one. This is covered in details in the section Bootstrap Plugins.
- **Customize** – You can customize Bootstrap's components, LESS variables, and jQuery plugins to get your very own version.

# Why Use Bootstrap?

- Advantages of Bootstrap:
- **Easy to use:** Anybody with just basic knowledge of HTML and CSS can start using Bootstrap
- **Responsive features:** Bootstrap's responsive CSS adjusts to phones, tablets, and desktops
- **Mobile-first approach:** In Bootstrap, mobile-first styles are part of the core framework
- **Browser compatibility:** Bootstrap 5 is compatible with all modern browsers (Chrome, Firefox, Edge, Safari, and Opera). **Note** that if you need support for IE11 and down, you must use either BS4 or BS3.

# Basic Difference between HTML,CSS,BOOTSTRAP



I AM HTML



I AM HTML  
AND CSS



I AM HTML,CSS,JAVASCRIPT  
(BOOTSTRAP)

# EXAMPLE OF HTML,CSS,BOOTSTRAP

Login With HTML Only

Email:

Password:

Login With HTML & CSS

Email:

Password:

I AM BOOTSTRAP

Create an account:

Email address  

Password  

Name your first app  

Company name (optional)  

If you continue, we will use your information to provide you with personalized services and offers.

## Bootstrap components & helpers

- Alerts
- Badges
- Breadcrumbs
- Buttons
- Cards
- Carousel
- Collapse
- Dropdowns
- List group
- Modal
- Navbars
- Panels
- Paginations
- Popovers
- Progress bars
- Scrollspy
- Spinners
- Toasts
- Tooltips
- Clearfix
- Icons
- Tables
- Responsive Ut
- Gutters

# Purpose of Bootstrap

- Decrease cost of website
- Save time
- Excellent look and feel
- Decrease code of line
- Maximum use of code
- It is easy andnot complex code (it is understood non programmer)
- It consume low space which help to execute fast in server

# Install Methods

- There are two ways...
  - Manually Download Source
  - CDN
- 
- To download bootstrap5 use this link:
  - <https://mdbootstrap.com/docs/standard/bootstrap-5/>

# Installation

- There are two main options to add Bootstrap to your web project. You can link to publicly available sources, or download the framework directly.
- **Download/Install**
  - You can download and install the Bootstrap source files with Bower, Composer, Meteor, or npm.
  - This allows greater control and the option to include or exclude modules as needed.

# Where to Get Bootstrap

- There are two ways to start using Bootstrap on your own web site.
  - Download Bootstrap from [getbootstrap.com](http://getbootstrap.com)
    - If you want to download and host Bootstrap yourself, go to [getbootstrap.com](http://getbootstrap.com), and follow the instructions there.
  - Include Bootstrap from a CDN
    - If you don't want to download and host Bootstrap yourself, you can include it from a CDN (Content Delivery Network).
    - MaxCDN provides CDN support for Bootstrap's CSS and JavaScript. You must also include jQuery.

# Bootstrap CDN

- You must include the following Bootstrap's CSS, JavaScript, and jQuery from MaxCDN into your web page.

```
<!-- Latest compiled and minified Bootstrap CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
```

```
<!-- Latest compiled Bootstrap JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
```

```
<!-- latest jQuery library -->
<script src="https://code.jquery.com/jquery-latest.js"></script>
```

- Advantage of using the Bootstrap CDN:

- Many users already have downloaded Bootstrap from MaxCDN 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 CDN

- If you don't want to download and host Bootstrap 5 yourself, you can include it from a CDN (Content Delivery Network).
- jsDelivr provides CDN support for Bootstrap's CSS and JavaScript:
  - **MaxCDN:**
  - <!-- Latest compiled and minified CSS -->  
`<link  
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css"  
rel="stylesheet">`
  - <!-- Latest compiled JavaScript -->  
`<script  
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"  
></script>`

# Create Web Page with Bootstrap

- Add the HTML5 doctype
  - Bootstrap uses HTML elements and CSS properties that require the HTML5 doctype.
  - Always include the HTML5 doctype at the beginning of the page, along with the lang attribute and the correct character set:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
  </head>
</html>
```

# Create Web Page with Bootstrap (2)

- Bootstrap is mobile-first
  - Bootstrap 3 is designed to be responsive to mobile devices. Mobile-first styles are part of the core framework.
  - To ensure proper rendering and touch zooming, add the following <meta> tag inside the <head> element:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

- The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).
- The initial-scale=1 part sets the initial zoom level when the page is first loaded by the browser.

The screenshot shows a web browser displaying the Bootstrap documentation at <https://getbootstrap.com/docs/5.3/getting-started/download/>. The page has a purple header with navigation links like 'Docs', 'Examples', 'Icons', 'Themes', 'Blog', a search bar, and social sharing icons. A sidebar on the left contains sections for 'Getting started' (with 'Download' highlighted) and 'Customize'. The main content area features a large heading 'Compiled CSS and JS' and a paragraph about downloading pre-compiled code. It includes a 'Download' button and a note about what's included. Below this is a section for 'Source files' with instructions for compiling Bootstrap using Sass and Autoprefixer.

Docs Examples Icons Themes Blog

Search CTRL K

v5.3

On this page

Compiled CSS and JS

Source files

Examples

CDN via jsDelivr

Package managers

npm

yarn

RubyGems

Composer

NuGet

## Compiled CSS and JS

Download ready-to-use compiled code for **Bootstrap v5.3.0-alpha3** to easily drop into your project, which includes:

- Compiled and minified CSS bundles (see [CSS files comparison](#))
- Compiled and minified JavaScript plugins (see [JS files comparison](#))

This doesn't include documentation, source files, or any optional JavaScript dependencies like Popper.

[Download](#)

## Source files

Compile Bootstrap with your own asset pipeline by downloading our source Sass, JavaScript, and documentation files. This option requires some additional tooling:

- [Sass compiler](#) for compiling Sass source files into CSS files
- [Autoprefixer](#) for CSS vendor prefixing

# What Is Responsive Web Design?

- As the name suggests, Responsive Web Design is the technique for developing websites and web portals in a more interactive way with the robust CX/UX (customer/user experience) optimal view solutions on a web page with the best browser compatibility that can run and operate in various types of devices.
- Responsive layouts automatically accommodate and adjust to any device's screen size, using a desktop, a laptop, a tablet, or a mobile phone.

# Creating Responsive Layout With Bootstrap

- Bootstrap is responsive and mobile-friendly from the beginning. Its six-tier grid classes deliver more reasonable control over the layout as well as decide how it will be rendered on various types of devices such as desktops, laptops, tablets, mobile phones, etc.
- The below example will build a responsive UI design layout that is furnished with:
  - 4 column layout in extra-large devices ( $\text{viewport} \geq 1200\text{px}$ ), and
  - 3 column layout in large devices ( $992\text{px} \leq \text{viewport} < 1200\text{px}$ ), whereas
  - 2 column layout in medium devices ( $768\text{px} \leq \text{viewport} < 992\text{px}$ ), and
  - 1 column layout in small and extra-small devices ( $\text{viewport} < 768\text{px}$ ).

# Bootstrap Responsive Design

GRID SYSTEM

# What is Responsive Web Design?

- ▶ Responsive web design makes your web page look good on all devices.
- ▶ Responsive web design uses only HTML and CSS.
- ▶ Responsive web design is not a program or a JavaScript.
- ▶ A responsive website is a website that resizes and rearranges that items on the page depending on the size of your browser. With a responsive website, if you resize your browser, you can see the changes occur in real time. Bootstrap makes your website responsive for you.

# Responsive Design



# Bootstrap standard devices resolution

- ▶ Extra small devices i.e *Phones (<768px)*, Symbol (-xs-), column width(Auto).
- ▶ Small devices i.e *Tablets ( $\geq 768px$ )*, Symbol (-sm-), column width(60px).
- ▶ Medium devices i.e *Desktop ( $\geq 992px$ )*, Symbol (-md-), column width(78px).
- ▶ Large devices i.e *Desktop ( $\geq 1200px$ )*, Symbol (-lg-), column width(95px)

# RESPONSIVE DESIGN ALL DEVICES

**col-xs-\***

- Extra small devices
- Mobile devices

**col-sm-\***

- Small devices
- tablets

**col-md-\***

- medium devices
- laptops , desktops

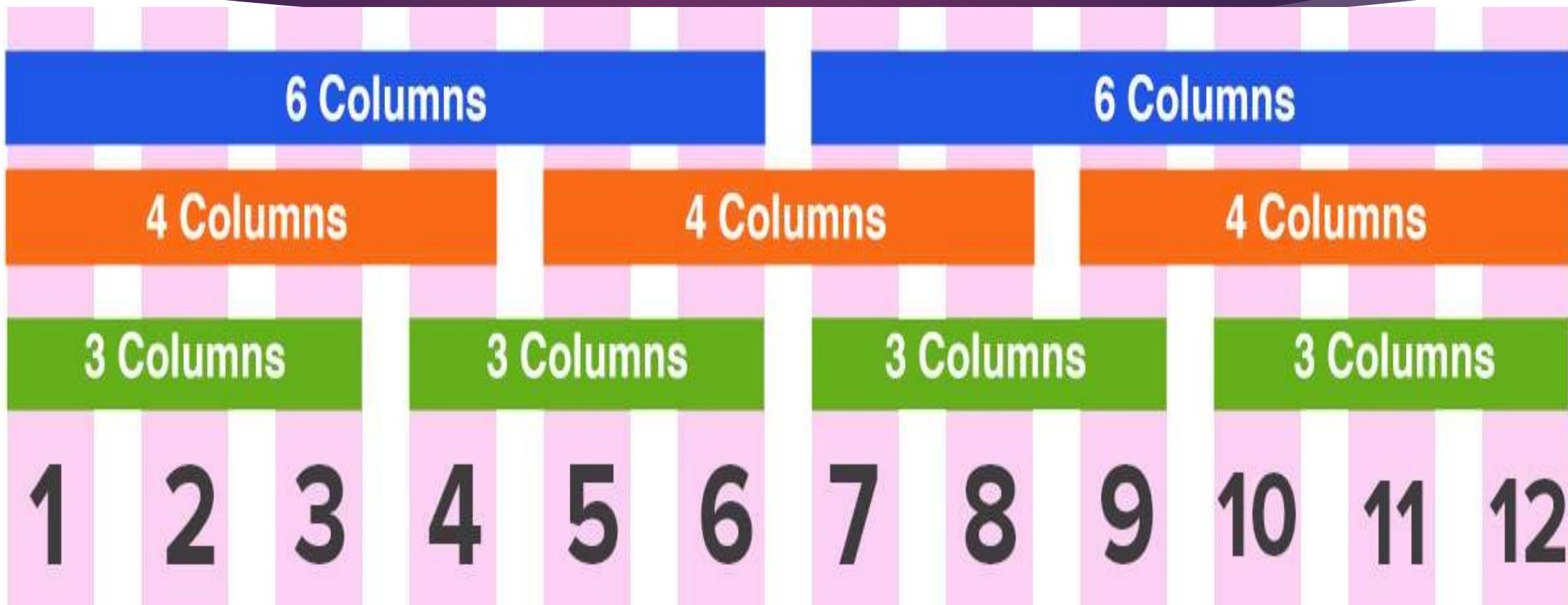
**col-lg-\***

- extra large desktops devices

# Grid System

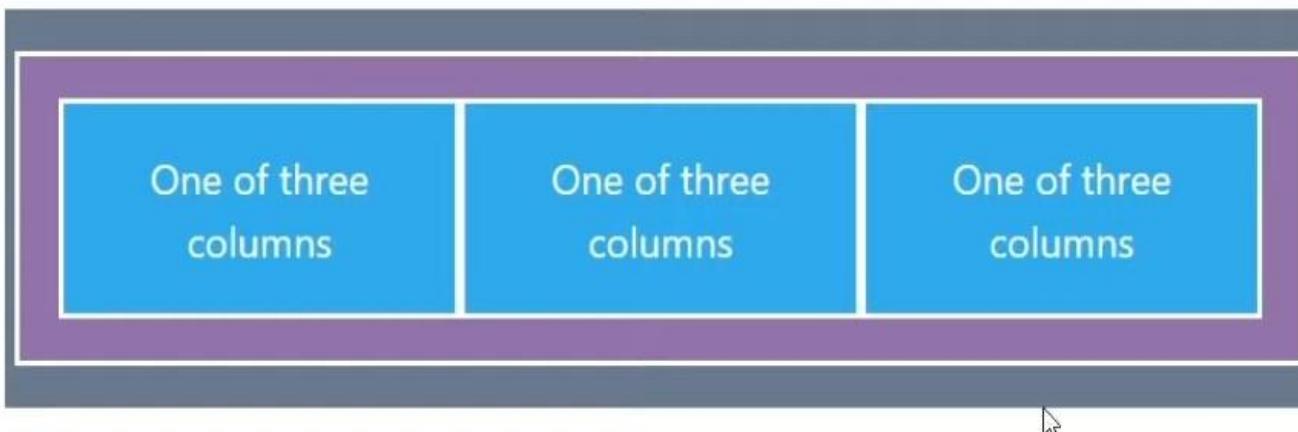
- ▶ Bootstrap includes a responsive, mobile first fluid grid system that appropriately scales up to 12 columns as the device or viewport size increases. It includes predefined classes for easy layout options.
- ▶ Grid systems are used for creating page layouts through a series of rows and columns that house your content.
- ▶ Grid classes apply to devices with screen widths greater than or equal to the breakpoint sizes, and override grid classes targeted at smaller devices. Therefore, applying any - md- class to an element will not only affect its styling on medium devices but also on large devices if a -lg- class is not present.

# Bootstrap Grid System Example



In order to create layout we use:

- Container (grey)
- Row (violet)
- Columns (blue)



# Containers, rows & columns

# Grid system

The Bootstrap Grid System allows users to create responsive layout.

We can use it to rearrange layout of the website depending on it's screen size.

Example:

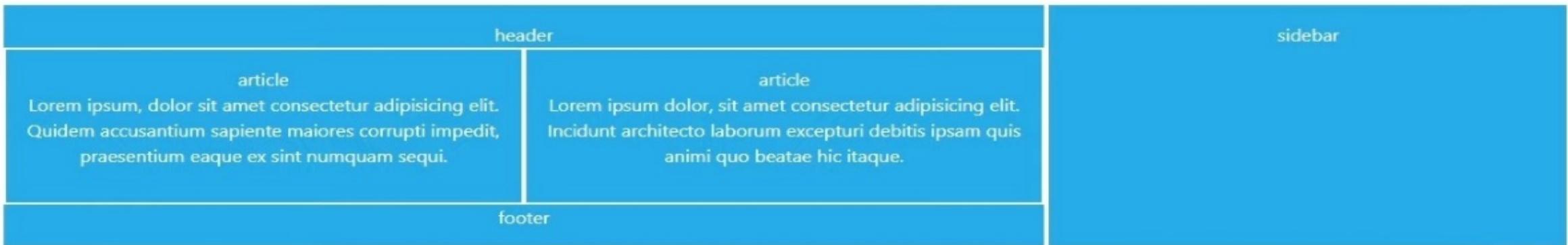
Using Bootstrap Grid System we can create page which will have different layout on desktop (top right corner) and tablet/mobile device (bottom right corner).



## Example 1



## Example 2



### Example 3



# BOOTSTRAP NAVIGATION BAR

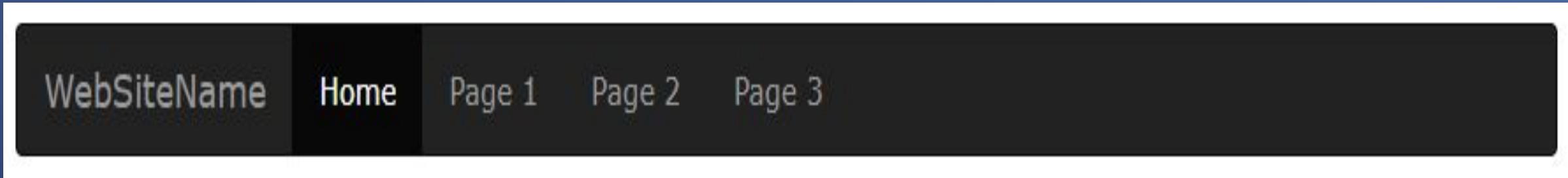


# NAVIGATION BARS

- A navigation bar is a navigation header that is placed at the top of the page:
- With Bootstrap, a navigation bar can extend or collapse, depending on the screen size.
- The navigation bar will be on one single line on large screens - because Bootstrap is responsive
- A standard navigation bar is created with

```
<nav class="navbar navbar-default">
```

# ALL OF THE EXAMPLES



- ▶ Inverted Navigation Bar Just change the
- ▶ .navbar-default class into **.navbar-inverse:**

# Navigation Bar With Dropdown

WebSiteName

Home

Page 1 ▾

Page 2

Page 3

## NAVIGATION BAR WITH DROPDOWN

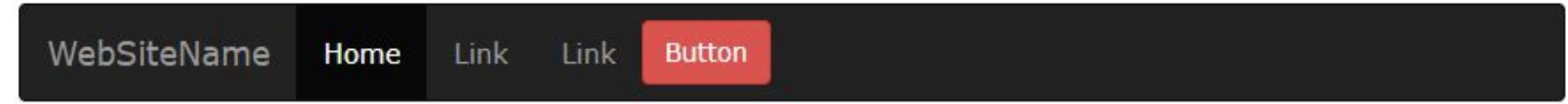
navigation bars can also hold dropdown menus.

# Right-Aligned Navigation Bar



The `.navbar-right` class is used to right-align navigation bar buttons.

## Navbar Buttons



To add buttons inside the navbar, add the `.navbar-btn` class on a Bootstrap button:

# Navbar Forms



To add form elements inside the navbar, add the **.navbar-form** class to a form element and add an input(s). note that we have added a .form-group class to the div container holding the input.

Navbar



## Collapsible Navbar

COLLAPSING THE NAVIGATION BAR

Navbar Home Features Pricing About

Search

Search

Navbar Home Features Pricing About

Search

Search

Navbar Home Features Pricing About

Search

Search

choose from **.navbar-light** for use with light background colors,  
or **.navbar-dark** for dark background colors. then, customize with **.bg-\***  
utilities. Like **bg-dark**, **bg-primary**

**Basic Navbar**

<nav>	navbar
<nav>	navbar-expand-xl lg md sm
<ul>	navbar-nav
<li>	nav-item
<a>	nav-link

**Collapse**

<button>	navbar-toggler
	data-toggle
	data-target
<span>	navbar-toggler-icon
<div>	collapse
<div>	navbar-collapse
	id
	target_name

**Appearance**

<nav>	navbar-dark
<nav>	navbar-light
<nav>	bg-dark
<nav>	bg-light
<span>	navbar-text
<a>	navbar-brand

**Dropdown**

<li>	dropdown
<a>	dropdown-toggle
	data-toggle
	id
	target_name
<div>	dropdown-menu
<div>	dropdown-divider
	aria-labelledby
<a>	dropdown-item

**Position**

<nav>	fixed-top
<nav>	fixed-bottom
<nav>	sticky-top



# WHAT ARE ICON FONTS

- ▶ Icon fonts are fonts that contain symbols and glyphs instead of letters or numbers.
- ▶ They're popular for web designers since you can style them with **CSS** the same way as regular text.
- ▶ Also, since they're vector's they're easily scale-able. They're small, so they load quickly and (bonus!) they're supported in all browsers.
- ▶ You can get icon fonts from a bunch of different places. Three of the most popular are [Entypo](#), [Font Awesome](#), and [IcoMoon](#). You're to use these fonts on your site through the CSS declaration `@font-face`.

# FONT AWESOME

- ▶ Font Awesome is free and open source. It was originally designed to work with Bootstrap, but it works well with all frameworks. To use Font Awesome icons, you'll have to use @fontface and put them inside a <span> or <i> element and then assign them classes.

# Bootstrap CSS components

# All are the Bootstrap css components

CSS Typography

CSS Buttons

CSS Forms

CSS Helpers

CSS Images

CSS Tables

CSS Dropdowns

CSS Navs

Glyphicons

# Typography

- The elements below are HTML elements that will be styled a little bit differently by Bootstrap than browser defaults. Look at the examples to see the result/differences.

Element/Class	Description
<h1> - <h6> or .h1 - .h6	h1 - h6 headings
<small>	Creates a lighter, secondary text in any heading
.small	Indicates smaller text (set to 85% of the size of the parent): Smaller text
.lead	Makes a text stand out: Stand out text
<mark> or .mark	Highlights text: Highlighted text
<del>	Indicates deleted text: Deleted text
<s>	Indicates no longer relevant text: <del>No longer relevant text</del>

# Bootstrap CSS Buttons

## Button Colors

[Basic](#)[Default](#)[Primary](#)[Success](#)[Info](#)[Warning](#)[Danger](#)[Link](#)

---

## Button Sizes

[Large](#)[Small](#)[XSmall](#)

# Cont..

## Active/Disabled Buttons

Info Button

Active Info Button

Disabled Info Button

## Block-level Button

Block-level Button

## Button Groups

Apple

Samsung

Sony

# Button Classes

Class	Description
.btn	Adds basic styling to any button
.btn-default	Indicates a default/standard button
.btn-primary	Provides extra visual weight and identifies the primary action in a set of buttons
.btn-success	Indicates a successful or positive action
.btn-info	Contextual button for informational alert messages
.btn-warning	Indicates caution should be taken with this action
.btn-danger	Indicates a dangerous or potentially negative action
btn-link	Makes a button look like a link (will still have button behavior)
.btn-lg	Makes a large button
.btn-sm	Makes a small button
.active	Makes the button appear pressed
.disabled	Makes the button disabled
navbar-btn	Vertically aligns a button inside a navbar

# Button Group Classes

The classes below can be used to style any <a>, <button>, or <input> element:

Class	Description
.btn-group	Groups buttons together on a single line
.btn-group-justified	Makes a group of buttons span the entire width of the screen
.btn-group-lg	Large button group (makes all buttons in a button group larger - increased font-size and padding)
.btn-group-sm	Small button group (makes all buttons in a button group smaller)
.btn-group-xs	Extra small button group (makes all buttons in a button group extra small)
.btn-group-vertical	Makes a button group appear vertically stacked

# Bootstrap CSS Helper Classes

- Text
- Add meaning through text-colors with the classes below. Links will darken on hover:

Class	Description
.text-muted	Text styled with class "text-muted"
.text-primary	Text styled with class "text-primary"
.text-success	Text styled with class "text-success"
.text-info	Text styled with class "text-info"
.text-warning	Text styled with class "text-warning"

# Background

- Add meaning through background-colors with the classes below.  
Links will darken on hover just like text classes.

Class	Description
.bg-primary	Table cell is styled with class "bg-primary"
.bg-success	Table cell is styled with class "bg-success"
.bg-info	Table cell is styled with class "bg-info"
.bg-warning	Table cell is styled with class "bg-warning"
.bg-danger	Table cell is styled with class "bg-danger"

# Bootstrap CSS Images

## Bootstrap Images

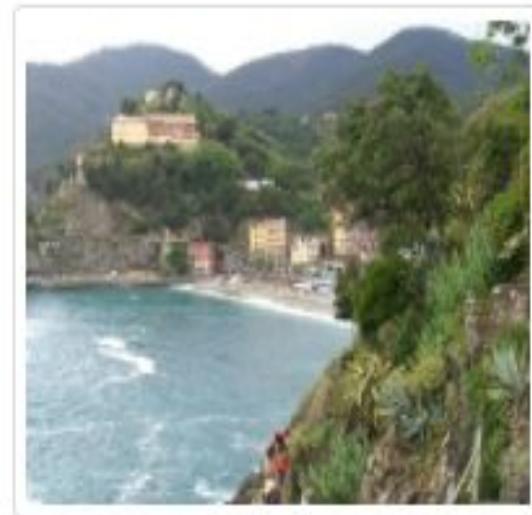
Rounded Corners:



Circle:



Thumbnail:



# <img> Classes

Class	Description
.img-rounded	Adds rounded corners to an image (not available in IE8)
.img-circle	Shapes the image to a circle (not available in IE8)
.img-thumbnail	Shapes the image to a thumbnail
.img-responsive	Makes an image responsive (will scale nicely to the parent element)

## Responsive Images

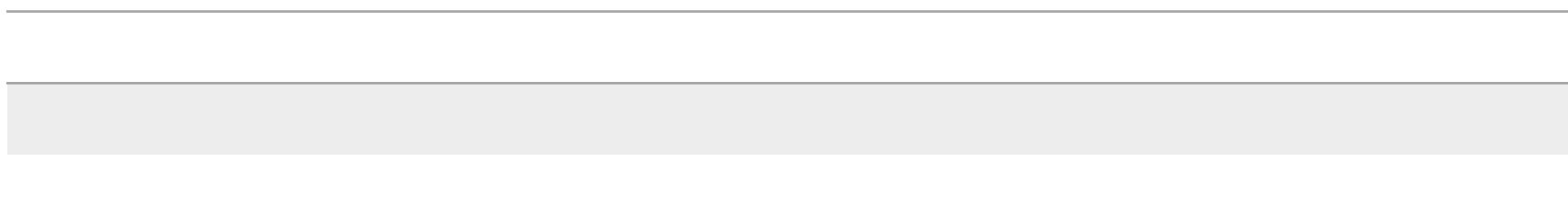
Create responsive images by adding an **.img-responsive** class to the <img> tag. The image will then scale nicely to the parent element.

The **.img-responsive** class applies **max-width: 100%, height: auto**, and **display:block** to the image

# Bootstrap CSS Tables

- <table> Classes

Class	Description
.table	Adds basic styling (light padding and only horizontal dividers) to any <table>
.table-striped	Adds zebra-striping to any table row within <tbody> (not available in IE8)
.table-bordered	Adds border on all sides of the table and cells
.table-hover	Enables a hover state on table rows within a <tbody>
.table-condensed	Makes table more compact by cutting cell padding in half



# <tr>, <th> and <td> Classes

Class	Description
.active	Applies the hover color (light-grey) to a particular row or cell
.success	Indicates a successful or positive action
.info	Indicates a neutral informative change or action
.warning	Indicates a warning that might need attention
.danger	Indicates a dangerous or potentially negative action

# Bootstrap Dropdown

Class	Description
.dropdown	Indicates a dropdown menu
.dropdown-menu	Builds the dropdown menu
.dropdown-menu-right	Right-aligns a dropdown menu
.dropdown-header	Adds a header inside the dropdown menu
.dropup	Indicates a dropup menu
.disabled	Disables an item in the dropdown menu
.divider	Separates items inside the dropdown menu with a horizontal line

# Bootstrap Glyphicon Components

- Bootstrap includes 260 glyphs from the Glyphicon Halflings set. Glyphicons Halflings are normally not available for free, but their creator has made them available for Bootstrap free of cost. As a thank you, you should include a link back to [Glyphicons](#) whenever possible.
- Use glyphicons in text, buttons, toolbars, navigation, or forms

# Exmp...

Glyph	Description
*	glyphicon glyphicon-asterisk
+	glyphicon glyphicon-plus
-	glyphicon glyphicon-minus
€	glyphicon glyphicon-euro
☁	glyphicon glyphicon-cloud
✉	glyphicon glyphicon-envelope
✎	glyphicon glyphicon-pencil
酾	glyphicon glyphicon-glass
♫	glyphicon glyphicon-music

🔍	glyphicon glyphicon-zoom-out
ⓧ	glyphicon glyphicon-off
📶	glyphicon glyphicon-signal
⚙	glyphicon glyphicon-cog
🗑	glyphicon glyphicon-trash
🏠	glyphicon glyphicon-home
📄	glyphicon glyphicon-file
🕒	glyphicon glyphicon-time
🛣	glyphicon glyphicon-road
⬇	glyphicon glyphicon-download-alt
⬇	glyphicon glyphicon-download

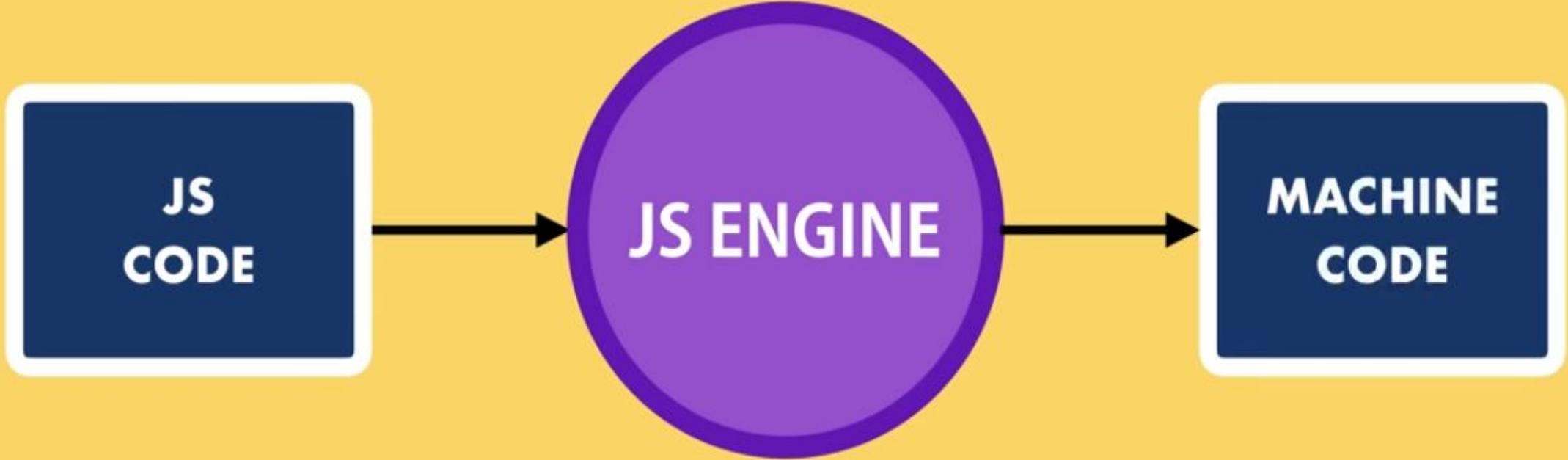
🔍	glyphicon glyphicon-search
❤	glyphicon glyphicon-heart
★	glyphicon glyphicon-star
☆	glyphicon glyphicon-star-empty
👤	glyphicon glyphicon-user
🎞	glyphicon glyphicon-film
☒	glyphicon glyphicon-th-large
☒	glyphicon glyphicon-th
☒	glyphicon glyphicon-th-list
✓	glyphicon glyphicon-ok
✗	glyphicon glyphicon-remove





# What is Node.js?

- Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine).
- Node.js was developed by Ryan Dahl in 2009
- Node.js is an open source server environment.
- Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.
- Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- Node.js uses JavaScript on the server



# What Can Node.js Do?

- Node.js can generate dynamic page content
- Node.js can create, open, read, write, delete, and close files on the server
- Node.js can collect form data
- Node.js can add, delete, modify data in your database
- A runtime Environment for executing JavaScript code.
- We Often use Node to build back-end services.
- Also called Application Programming Interface.
- Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications.

# What is a Node.js File?

- Node.js files contain tasks that will be executed on certain events
- A typical event is someone trying to access a port on the server
- Node.js files must be initiated on the server before having any effect
- Node.js files have extension ".js"

*Node.js* is an open-source, cross-platform runtime environment that allows developers to create all kinds of server-side tools and applications in [JavaScript](#).

# javaScript Engine for respective web browsers



Chakra



SpiderMonkey



v8

# Features of Node.js

- **Asynchronous and Event Driven** – All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data.
- **Very Fast** – Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
- **Single Threaded but Highly Scalable** – Node.js uses a single threaded model with event looping.
- Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
- **No Buffering** – Node.js applications never buffer any data. These applications simply output the data in chunks.



**REQUEST**



**HTTP://URL**



**RESPONSE**

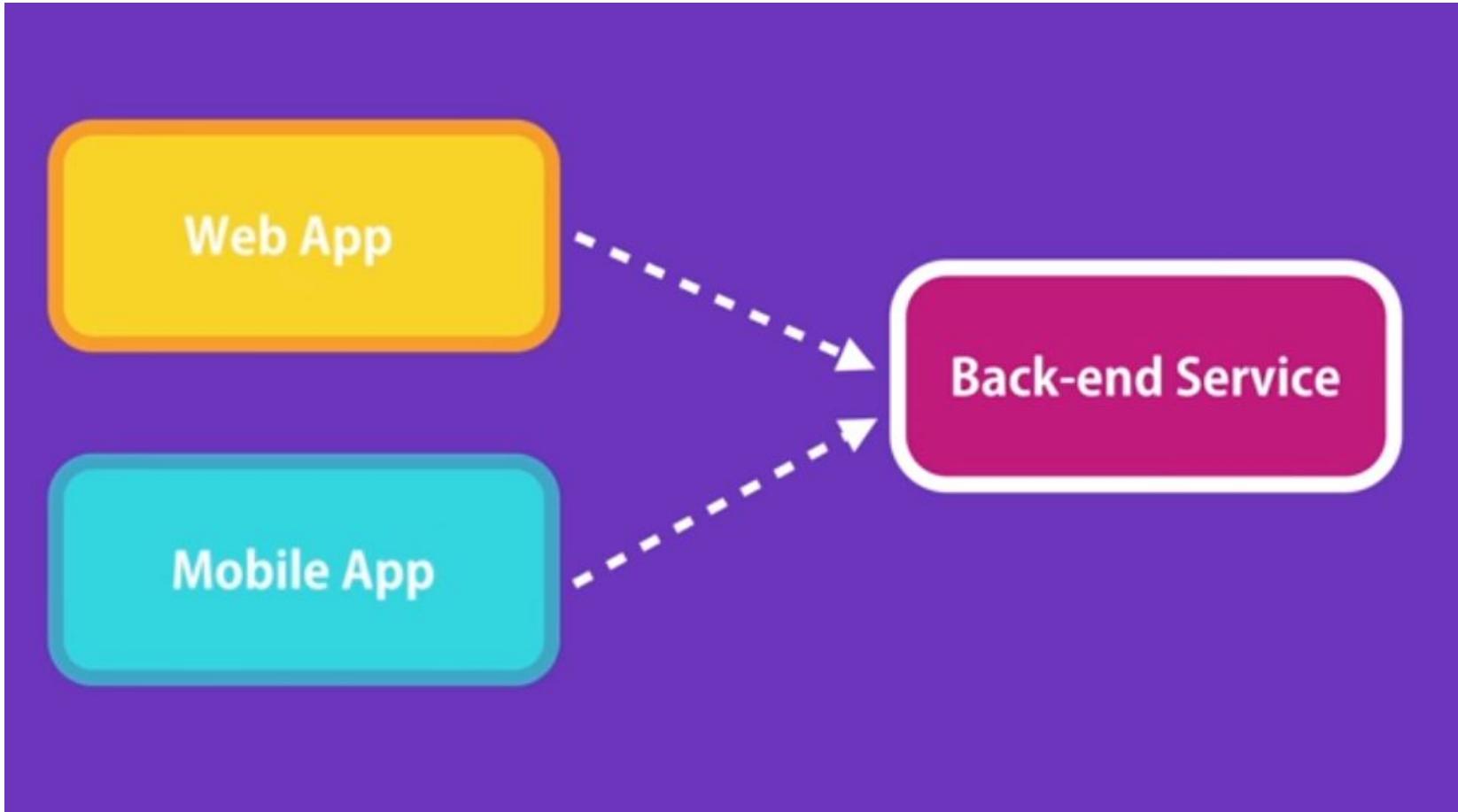


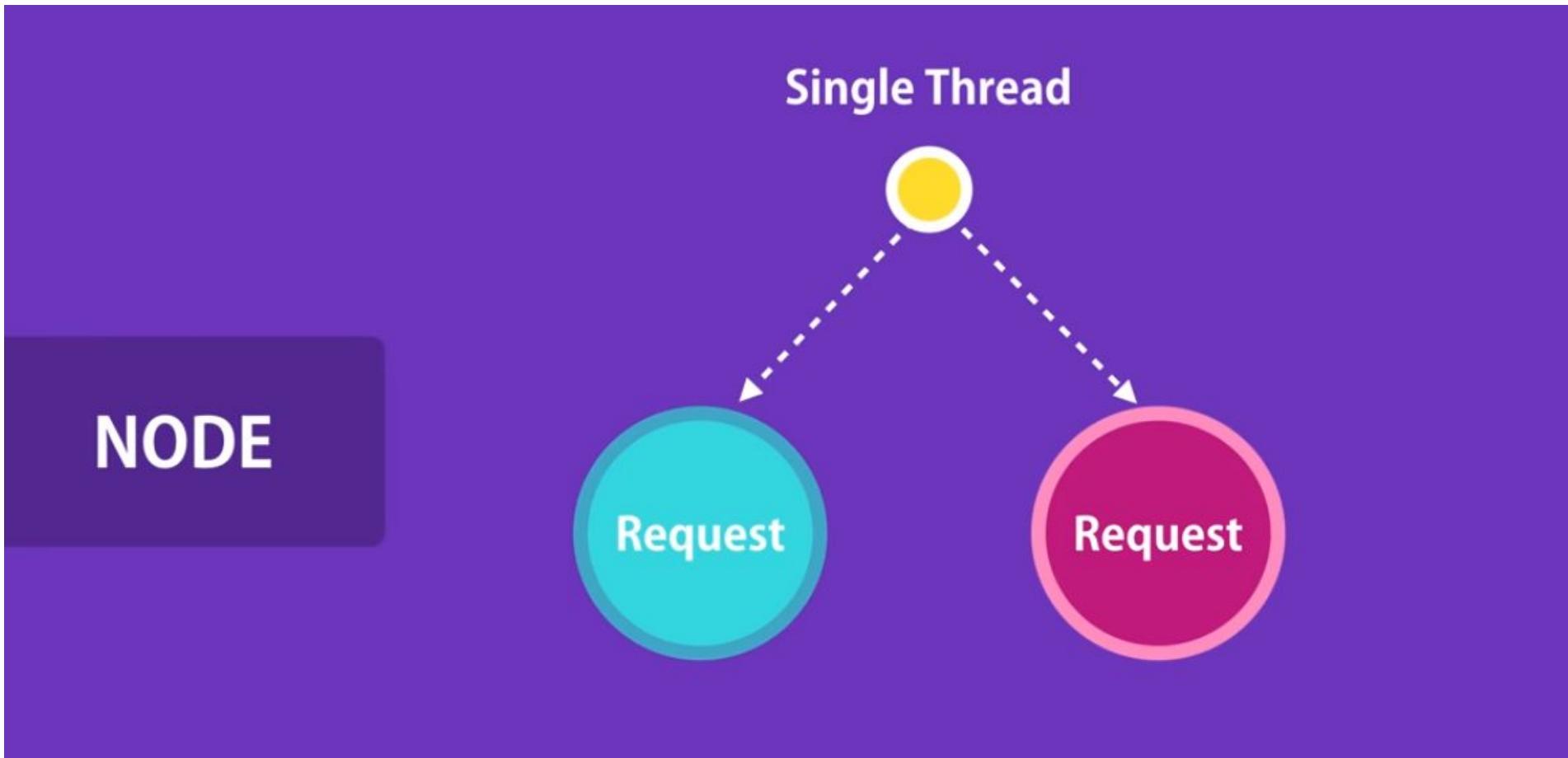
**HTTP://URL**



# Non-blocking I/O

- ✓ Works on a single thread using non-blocking I/O calls
- ✓ Supports tens of thousands concurrent connections
- ✓ Optimizes throughput and scalability in web applications with many I/O operations
- ✓ This makes Node.js apps extremely **fast** and **efficient**





# Where to Use Node.js?

- Following are the areas where Node.js is proving itself as a perfect technology partner.
- I/O bound Applications
- Data Streaming Applications
- Data Intensive Real-time Applications (DIRT)
- JSON APIs based Applications
- Single Page Applications

# Cont..

- Here is how Node.js handles a file request:
- Sends the task to the computer's file system.
- Ready to handle the next request.
- When the file system has opened and read the file, the server returns the content to the client.
- Node.js eliminates the waiting, and simply continues with the next request.
- Node.js runs single-threaded, non-blocking, asynchronously programming, which is very memory efficient.

- A Node.js application consists of the following three important components –
- **Import required modules** – We use the **require** directive to load Node.js modules.
- **Create server** – A server which will listen to client's requests similar to Apache HTTP Server.
- **Read request and return response** – The server created in an earlier step will read the HTTP request made by the client which can be a browser or a console and return the response.

# Creating Node.js Application

- Step 1 - Import Required Module
- We use the require directive to load the http module and store the returned HTTP instance into an http variable as follows –
- `var http = require("http");`

## Step 2 - Create Server

- We use the created http instance and call **http.createServer()** method to create a server instance and then we bind it at port 8081 using the **listen** method associated with the server instance. Pass it a function with parameters request and response.

- `http.createServer(function (request, response) {`
- `// Send the HTTP header`
- `// HTTP Status: 200 : OK`
- `// Content Type: text/plain`
- `response.writeHead(200, {'Content-Type': 'text/plain'});`
- `// Send the response body as "Hello World"`
- `response.end('Hello World\n');`
- `}).listen(8081);`
- `// Console will print the message`
- `console.log('Server running at http://127.0.0.1:8081/');`

# Step 3 - Testing Request & Response combining step1 and step2

- var http = require("http");
- http.createServer(function (request, response) {
- // Send the HTTP header
- // HTTP Status: 200 : OK
- // Content Type: text/plain
- response.writeHead(200, {'Content-Type': 'text/plain'});
- 
- // Send the response body as "Hello World"
- response.end('Hello World\n');
- }).listen(8081);
- // Console will print the message
- console.log('Server running at <http://127.0.0.1:8081/>');

## Step:4

- Now execute the main.js to start the server as follows –
- \$ node main.js

# Node.js Module

- Module in Node.js is a simple or complex functionality organized in single or multiple JavaScript files which can be reused throughout the Node.js application.
- Each module in Node.js has its own context, so it cannot interfere with other modules or pollute global scope. Also, each module can be placed in a separate .js file under a separate folder.

# Node.js Core Modules

- Node.js is a light weight framework. The core modules include bare minimum functionalities of Node.js. These core modules are compiled into its binary distribution and load automatically when Node.js process starts. However, you need to import the core module first in order to use it in your application.

Core Module	Description
<a href="#"><u>http</u></a>	http module includes classes, methods and events to create Node.js http server.
<a href="#"><u>url</u></a>	url module includes methods for URL resolution and parsing.
<a href="#"><u>querystring</u></a>	querystring module includes methods to deal with query string.
<a href="#"><u>path</u></a>	path module includes methods to deal with file paths.
<a href="#"><u>fs</u></a>	fs module includes classes, methods, and events to work with file I/O.
<a href="#"><u>util</u></a>	util module includes utility functions useful for programmers.

# Example: Load and Use Core http Module

```
var http = require('http');

var server = http.createServer(function(req, res){

    //write code here

});

server.listen(5000);
```

- In the above example, `require()` function returns an object because `http` module returns its functionality as an object, you can then use its properties and methods using dot notation e.g. `http.createServer()`.

# Node.js Local Module

- Local modules are modules created locally in your Node.js application. These modules include different functionalities of your application in separate files and folders.
- write simple logging module which logs the information, warning or error to the console.

```
var log = {
    info: function (info) {
        console.log('Info: ' + info);
    },
    warning: function (warning) {
        console.log('Warning: ' + warning);
    },
    error: function (error) {
        console.log('Error: ' + error);
    }
};

module.exports = log
```

# Export Module in Node.js

- The module.exports is a special object which is included in every JavaScript file in the Node.js application by default.
- The module is a variable that represents the current module, and exports is an object that will be exposed as a module.
- So, whatever you assign to module.exports will be exposed as a module.
- Note: You must specify ./ as a path of root folder to import a local module. However, you do not need to specify the path to import Node.js core modules or NPM modules in the require() function.

```
Message.js  
module.exports = 'Hello world';
```

```
app.js  
var msg = require('./Messages.js');  
  
console.log(msg);
```

```
C:\> node app.js  
Hello World
```

# Node.js - REPL Terminal

# REPL stands

- REPL stands for **Read Evaluate Print Loop** and it represents a computer environment like a Windows console or Unix/Linux shell where a command is entered and the system responds with an output in an interactive mode. Node.js or Node comes bundled with a REPL environment. It performs the following tasks –
- REPL for, and it is a programming language environment (basically a console window) that takes single expression as user input and returns the result back to the console after execution. The REPL session provides a convenient way to quickly test simple JavaScript code.
- Read – Reads user's input, parses the input into JavaScript data-structure, and stores in memory.
- Eval – Takes and evaluates the data structure.
- Print – Prints the result.
- Loop – Loops the above command until the user presses ctrl-c twice.
- The REPL feature of Node is very useful in experimenting with Node.js codes and to debug JavaScript codes.

# Starting REPL

- REPL can be started by simply running node on shell/console without any arguments as follows.
- \$ node
- You will see the REPL Command prompt > where you can type any Node.js command –
  - \$ node
  - >

# Simple Expression

- Let's try a simple mathematics at the Node.js REPL command prompt
  -
- \$ node
- > 1 + 3
- 4
- > 1 + ( 2 \* 3 ) - 4
- 3
- >

# Use Variables

- You can make use variables to store values and print later like any conventional script. If var keyword is not used, then the value is stored in the variable and printed. Whereas if var keyword is used, then the value is stored but not printed. You can print variables using console.log().
- \$ node
- > x = 10
- 10
- > var y = 10
- undefined
- > x + y
- 20
- > console.log("Hello World")
- Hello World
- undefined

# Multiline Expression

- Node REPL supports multiline expression similar to JavaScript
- \$ node
- > var x = 0
- undefined
- > do {
- ... x++;
- ... console.log("x: " + x);   ... }
- while ( x < 5 );
- x: 1
- x: 2
- x: 3
- x: 4
- x: 5
- undefined... comes automatically when you press Enter after the opening bracket. Node automatically checks the continuity of expressions.

# REPL Commands

- `ctrl + c` – terminate the current command.
- `ctrl + c` twice – terminate the Node REPL.
- `ctrl + d` – terminate the Node REPL.
- Up/Down Keys – see command history and modify previous commands.
- tab Keys – list of current commands.
- `.help` – list of all commands.
- `.break` – exit from multiline expression.
- `.clear` – exit from multiline expression.
- `.save filename` – save the current Node REPL session to a file.
- `.load filename` – load file content in current Node REPL session.

# Stopping REPL

- As mentioned above, you will need to use ctrl-c twice to come out of Node.js REPL.
- \$ node
- >
- (^C again to quit)
- >

# NPM - Node Package Manager

- Node Package Manager (NPM) is a command line tool that installs, updates or uninstalls Node.js packages in your application.
- It is also an online repository for open-source Node.js packages.
- It has now become a popular package manager for other open-source JavaScript frameworks like AngularJS, jQuery, Gulp, Bower etc.
- NPM is included with Node.js installation.
- After you install Node.js, verify NPM installation by writing the following command in terminal or command prompt.
- C:\> npm -v  
2.11.3

# Updating the NPM

- If you have an older version of NPM then you can update it to the latest version using the following command.
- C:\> npm install npm -g

# What is NPM?

- NPM is a package manager for Node.js packages, or modules if you like.
- It hosts thousands of free packages to download and use.
- The NPM program is installed on your computer when you install Node.js

# Install Package Locally

- Use the following command to install any third party module in your local Node.js project folder.
- C:\>npm install <package name>
- For example, the following command will install ExpressJS into MyNodeProj folder.

```
C:\MyNodeProj> npm install express
```

- Now let's try installing the express module using **global installation**.

# Install Package Globally

- NPM can also install packages globally so that all the node.js application on that computer can import and use the installed packages. NPM installs global packages into
  - /<User>/local/lib/node\_modules folder.
- Apply -g in the install command to install package globally. For example, the following command will install ExpressJS globally.
- C:\MyNodeProj> npm install -g express

```
$ npm install express -g
```

# Uninstall Packages

- Use the following command to remove a local package from your project.
- C:\>npm uninstall <package name>

The following command will uninstall ExpressJS from the application.

```
C:\MyNodeProj> npm uninstall express
```

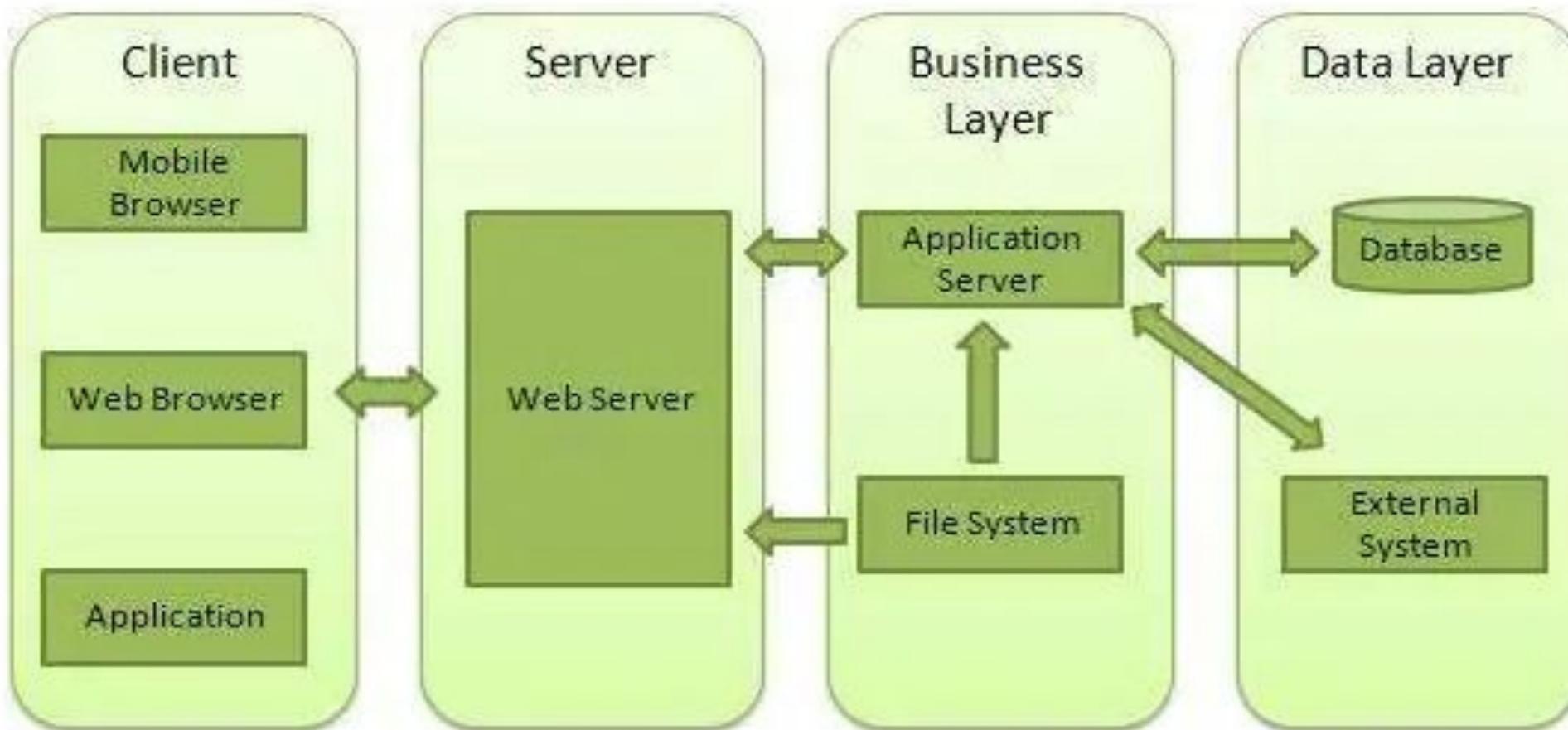
# Node.js Web Module

- The web module is the smallest deployable and usable unit of web resources. A web module contains web components and static web content files, such as images, which are called web resources.

# What is a web server and How it works?

- Web server is a software program that deals with HTTP (hypertext transfer protocol) requests sent by HTTP clients (web browser like Google, Internet Explorer, Mozilla Firefox) and render web pages in response to the clients. Web server usually returns HTML documents along with images, style sheets, and scripts.
- Most of the web server support server-side scripts and get data from the database and performs complex logic and then send the corresponding result to the HTTP client browser.

# Web Application Architecture Model



# **Web application architecture is divided into 4 layers:-**

- **Client** – The client layer sends HTTP requests to the web server via mobile Browser or application, web Browser.
- **Server** – The server layer intercepts the request made by clients and passes them the response.
- **Business Layer** – This layer interacts with the data layer through the database. The business layer contains the application server which is utilised by the web server to do the required processing.
- **Data** – data layer contains the database and send corresponding data to the server.

# How to create web server using Node.js

- With the help of Node.js, we can create HTTP client of the server.
- Node.js provide us HTTP module, HTTP module is mainly used to build HTTP server.
- We are going to write a code for creating HTTP server which listens at 3030 port.

```
var http = require('http');
var url = require('url');
var fs = require('fs');
http.createServer(function (req, res) {
  var q = url.parse(req.url, true);
  var filename = "index.html" + q.pathname;
  fs.readFile(filename, function (err, data) {
    if (err) {
      res.writeHead(404, { 'Content-Type': 'text/html' });
      return res.end("404 Not Found");
    }
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.write(data);
    return res.end();
  });
}).listen(3030);
console.log('server running at localhost:3030');
```

# Create a index.html

- Next, create an HTML file named index.html having the following code in the same directory where you have created server.js
- <!DOCTYPE html>
- <html>
- <body>
  - <h1>Hello World!</h1>
  - </body>
- </html>

# Node.js - Express Framework

# Express

- Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications.
- It facilitates the rapid development of Node based Web applications.
- **Following are some of the core features of Express framework –**
- Allows to set up middlewares to respond to HTTP Requests.
- Defines a routing table which is used to perform different actions based on HTTP Method and URL.
- Allows to dynamically render HTML Pages based on passing arguments to templates.

# WHAT IS EXPRESS?

Express is a fast, unopinionated and minimalist web framework for Node.js

Express is a “**server-side**” or “**back-end**” framework. It is not comparable to client-side frameworks like React, Angular & Vue. It can be used in combination with those frameworks to build full stack applications

# WHY USE EXPRESS?

- Makes building web applications with Node.js MUCH easier
- Used for both server rendered apps as well as API/Microservices
- Extremley light, fast and free
- Full control of request and response
- By far the most popular Node framework
- Great to use with client side frameworks as it's all JavaScript

# BASIC SERVER SYNTAX

```
const express = require('express');

// Init express
const app = express();

// Create your endpoints/route handlers
app.get('/', function(req, res) {
  res.send('Hello World!');
});

// Listen on a port
app.listen(5000);
```

# BASIC ROUTE HANDLING

- Handling requests/routes is simple
- app.get(), app.post(), app.put(), app.delete(), etc
- Access to params, query strings, url parts, etc
- Express has a router so we can store routes in separate files and export
- We can parse incoming data with the Body Parser

```
app.get('/', function(req, res) {  
  // Fetch from database  
  // Load pages  
  // Return JSON  
  // Full access to request & response  
});
```

# EXPRESS MIDDLEWARE

**Middleware functions** are functions that have access to the **request** and **response** object. Express has built in middleware but middleware also comes from 3rd party packages as well as custom middleware

- Execute any code
- Make changes to the request/response objects
- End response cycle
- Call next middleware in the stack

# Installing Express

- Firstly, install the Express framework globally using NPM so that it can be used to create a web application using node terminal.
- \$ npm install express –save
- The above command saves the installation locally in the **node\_modules** directory and creates a directory express inside node\_modules.

- You should install the following important modules along with express
- **body-parser** – This is a node.js middleware for handling JSON, Raw, Text and URL encoded form data.
- **cookie-parser** – Parse Cookie header and populate req.cookies with an object keyed by the cookie names.
- **multer** – This is a node.js middleware for handling multipart/form-data.
- \$ npm install body-parser --save
- \$ npm install cookie-parser --save
- \$ npm install multer --save

# Hello world Example

- Following is a very basic Express app which starts a server and listens on port 8081 for connection. This app responds with **Hello World!** for requests to the homepage. For every other path, it will respond with a **404 Not Found**.

```
var express = require('express');
var app = express();

app.get('/', function (req, res) {
  res.send('Hello World');
})

var server = app.listen(8081, function () {
  var host = server.address().address
  var port = server.address().port

  console.log("Example app listening at http://%s:%s", host,
  port)
})
```

# Request & Response

- Express application uses a callback function whose parameters are request and response objects.
- app.get('/', function (req, res) {
  - // --
  - })
- Request Object – The request object represents the HTTP request and has properties for the request query string, parameters, body, HTTP headers, and so on.
- Response Object – The response object represents the HTTP response that an Express app sends when it gets an HTTP request.
- You can print req and res objects which provide a lot of information related to HTTP request and response including cookies, sessions, URL, etc.

# BasicRouting

- We have seen a basic application which serves HTTP request for the homepage.
- Routing refers to determining how an application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method (GET, POST, and so on).
- ```
app.get('/', (req, res) => {  
  res.send('Hello World!')  
});
```

# Serving Static Files

- Express provides a built-in middleware **express.static** to serve static files, such as images, CSS, JavaScript, etc.
- You simply need to pass the name of the directory where you keep your static assets, to the **express.static** middleware to start serving the files directly. For example, if you keep your images, CSS, and JavaScript files in a directory named public, you can do this –
- `app.use(express.static('public'));`

# "Hello Word" app to add the functionality to handle static files.

- var express = require('express');
- var app = express();
- app.use(express.static('public'));
- app.get('/', function (req, res) {
  - res.send('Hello World');
  - })
- var server = app.listen(8081, function () {
  - var host = server.address().address
  - var port = server.address().port
  - console.log("Example app listening at http://%s:%s", host, port)
  - })

- Any files in the public directory are served by adding their filename (*relative* to the base "public" directory) to the base URL. So for example:
  - `http://localhost:3000/images/dog.jpg`
  - `http://localhost:3000/css/style.css`
  - `http://localhost:3000/js/app.js`
  - `http://localhost:3000/about.html`

# Using middleware

- Express middleware are **functions that execute during the lifecycle of a request to the Express server**. Each middleware has access to the HTTP request and response for each route (or path) it's attached to.
- Middleware is used extensively in Express apps, for tasks from serving static files to error handling, to compressing HTTP responses.
- Whereas route functions end the HTTP request-response cycle by returning some response to the HTTP client.
- middleware functions *typically* perform some operation on the request or response and then call the next function in the "stack"
- which might be more middleware or a route handler. The order in which middleware is called is up to the app developer

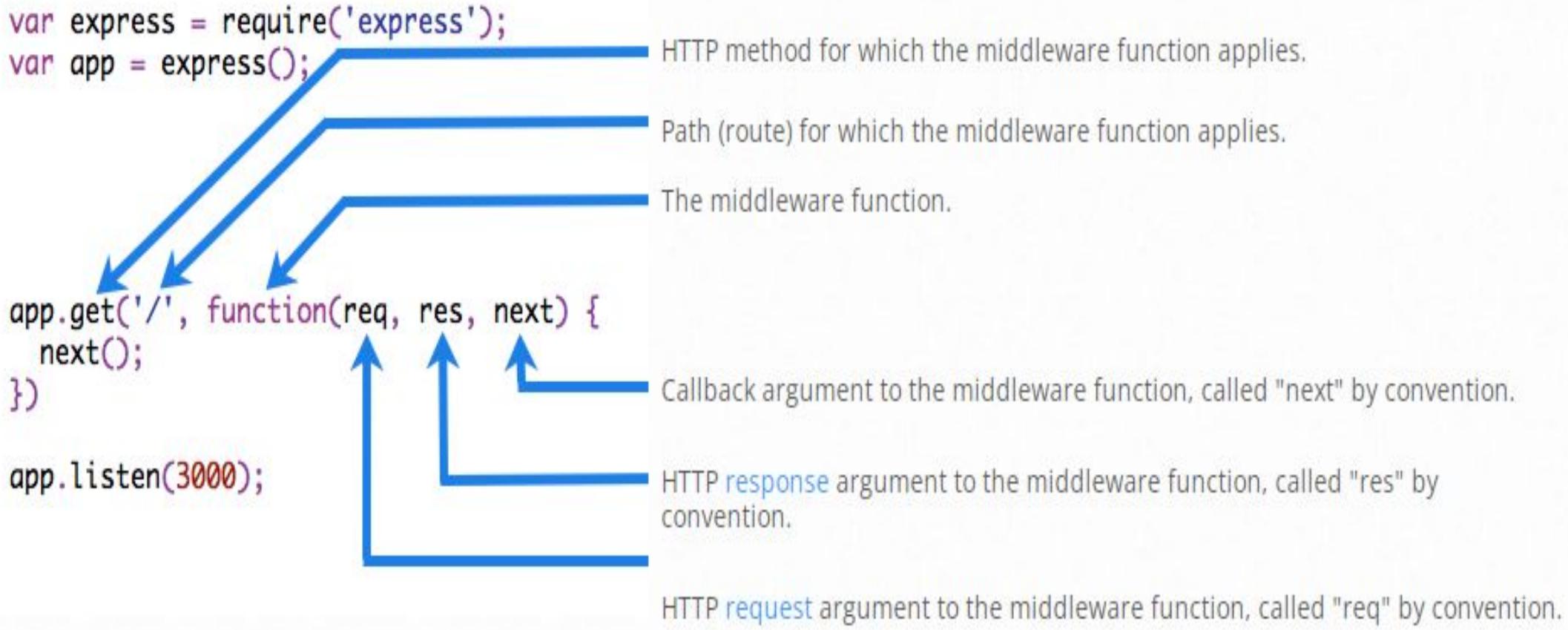
# Difference b/w middleware and route handler

- The only difference between a middleware function and a route handler callback is that middleware functions have a third argument next.
- which middleware functions are expected to call if they are not that which completes the request cycle (when the middleware function is called, this contains the next function that must be called).

# Middleware functions can perform the following tasks:

- Execute any code.
  - Make changes to the request and the response objects.
  - End the request-response cycle.
  - Call the next middleware in the stack.
- 
- If the current middleware function does not end the request-response cycle, it must call `next()` to pass control to the next middleware function.

# The following figure shows the elements of a middleware function call:



# An example middleware function

- const express = require('express');
- const app = express();
- // An example middleware function
- let a\_middleware\_function = function(req, res, next) {
  - // ... perform some operations
  - next(); // Call next() so Express will call the next middleware function in the chain.
- }
- // Function added with use() for all routes and verbs
- app.use(a\_middleware\_function);
- // Function added with use() for a specific route
- app.use('/somerule', a\_middleware\_function);
- // A middleware function added for a specific HTTP verb and route
- app.get('/', a\_middleware\_function);
- app.listen(3000);

# Nodejs-RESTFull-API

Representational State Transfer



# What is REST

- REST stands for REpresentational State Transfer.(REST).
- RESTful API is an interface that two computer systems use to exchange information securely over the internet.
- Most business applications have to communicate with other internal and third-party applications to perform various tasks.
- For example, to generate monthly payslips, your internal accounts system has to share data with your customer's banking system to automate invoicing and communicate with an internal timesheet application.
- RESTful APIs support this information exchange because they follow secure, reliable, and efficient software communication standards.

# REST....

- REST was first introduced by Roy Fielding in 2000
- REST is web standards based architecture and uses HTTP Protocol.
- It revolves around resource where every component is a resource.
- A resource is accessed by a common interface using HTTP standard methods..
- A REST Server simply provides access to resources.
- REST client accesses and modifies the resources using HTTP protocol.
- Here each resource is identified by URIs/ global IDs.
- REST uses various representation to represent a resource like text, JSON, XML but JSON is the most popular one.

# What is an API?

- An application programming interface (API) defines the rules that you must follow to communicate with other software systems. Developers expose or create APIs so that other applications can communicate with their applications programmatically.
- Application programming interfaces (APIs) are everywhere. They enable software to communicate with other pieces of software—internal or external—consistently, which is a key ingredient in scalability, not to mention reusability.
- It's quite common nowadays for online services to have public-facing APIs. These enable other developers to easily integrate features like social media logins, credit card payments, and behavior tracking.

# RESTful API

- APIs that follow the REST architectural style are called REST APIs.
- Web services that implement REST architecture are called RESTful web services.
- The term RESTful API generally refers to RESTful web APIs.
- However, you can use the terms REST API and RESTful API interchangeably.

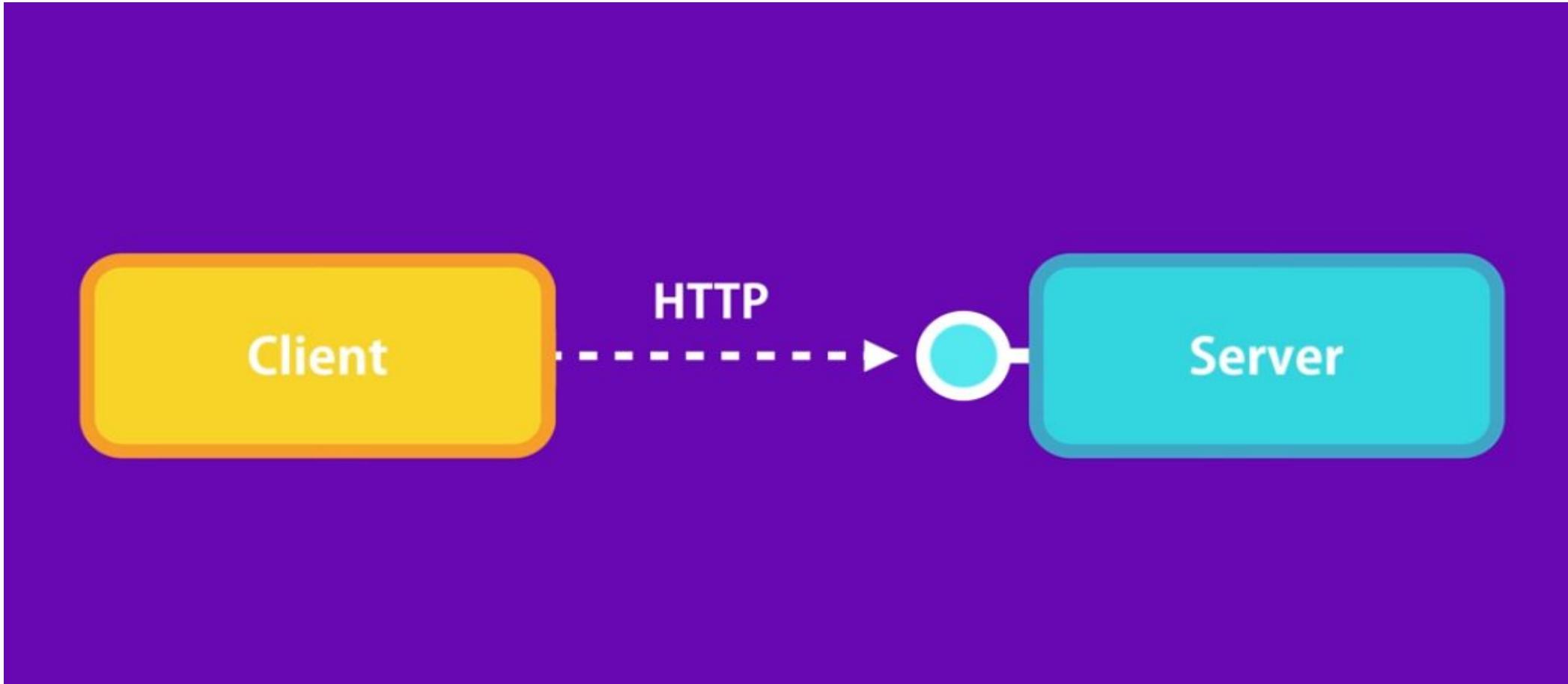
# What are the benefits of RESTful APIs?

- **Scalability** :- REST APIs can scale efficiently because REST optimizes client-server interactions. Statelessness removes server load because the server does not have to retain past client request information.
- **Flexibility**:- RESTful web services support total client-server separation. They simplify and decouple various server components so that each part can evolve independently.
- Platform or technology changes at the server application do not affect the client application. For example, developers can make changes to the database layer without rewriting the application logic.
- **Independence**
- REST APIs are independent of the technology used. You can write both client and server applications in various programming languages without affecting the API design

# How do RESTful APIs work?

- The basic function of a RESTful API is the same as browsing the internet. The client contacts the server by using the API when it requires a resource.
- **These are the general steps for any REST API call:**
  - The client sends a request to the server. The client follows the API documentation to format the request in a way that the server understands.
  - The server authenticates the client and confirms that the client has the right to make that request.
  - The server receives the request and processes it internally.
  - The server returns a response to the client. The response contains information that tells the client whether the request was successful. The response also includes any information that the client requested.
  - The REST API request and response details vary slightly depending on how the API developers design the API.

# Client and server communication



The main functions used in any REST-based architecture are: RESTful API

Create

Read

Update

Delete



In **HTTP** there are five methods that are commonly used in a REST-based Architecture i.e., POST, GET, PUT, PATCH, and DELETE

- **GET:** The HTTP GET method is used to **read** (or retrieve) a representation of a resource. In the safe path, GET returns a representation in XML or JSON and an HTTP response code of 200 (OK). In an error case, it most often returns a 404 (NOT FOUND) or 400 (BAD REQUEST).
- **POST:** The POST verb is most often utilized to **create** new resources. On successful creation, return HTTP status 201, returning a Location header with a link to the newly-created resource with the 201 HTTP status.
- **PUT/PATCH:** It is used for **updating** the capabilities. However, PUT can also be used to **create** a resource in the case where the resource ID is chosen by the client instead of by the server. On successful update, return 200 (or 204 if not returning any content in the body) from a PUT. If using PUT for create, return HTTP status 201 on successful creation.
- **DELETE:** It is used to **delete** a resource identified by a URI. On successful deletion, return HTTP status 200 (OK) along with a response body.

# Example

|                                                                                                   |                                      |                                                                                       |
|---------------------------------------------------------------------------------------------------|--------------------------------------|---------------------------------------------------------------------------------------|
|  <b>GET</b>      | <b>/users</b> finds all users        |    |
|  <b>POST</b>     | <b>/users</b> creates a user         |    |
|  <b>GET</b>      | <b>/users/:id</b> finds user details |    |
|  <b>DELETE</b> | <b>/users/:id</b> deletes a user     |  |
|  <b>PATCH</b>  | <b>/users/:id</b> updates a user     |  |

# What is Postman?

- Postman is an [API platform](#) for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster.
- Postman is one of the most popular software testing tools which is used for API testing. With the help of this tool, developers can easily create, test, share, and document APIs.





**Git**



**GitHub**

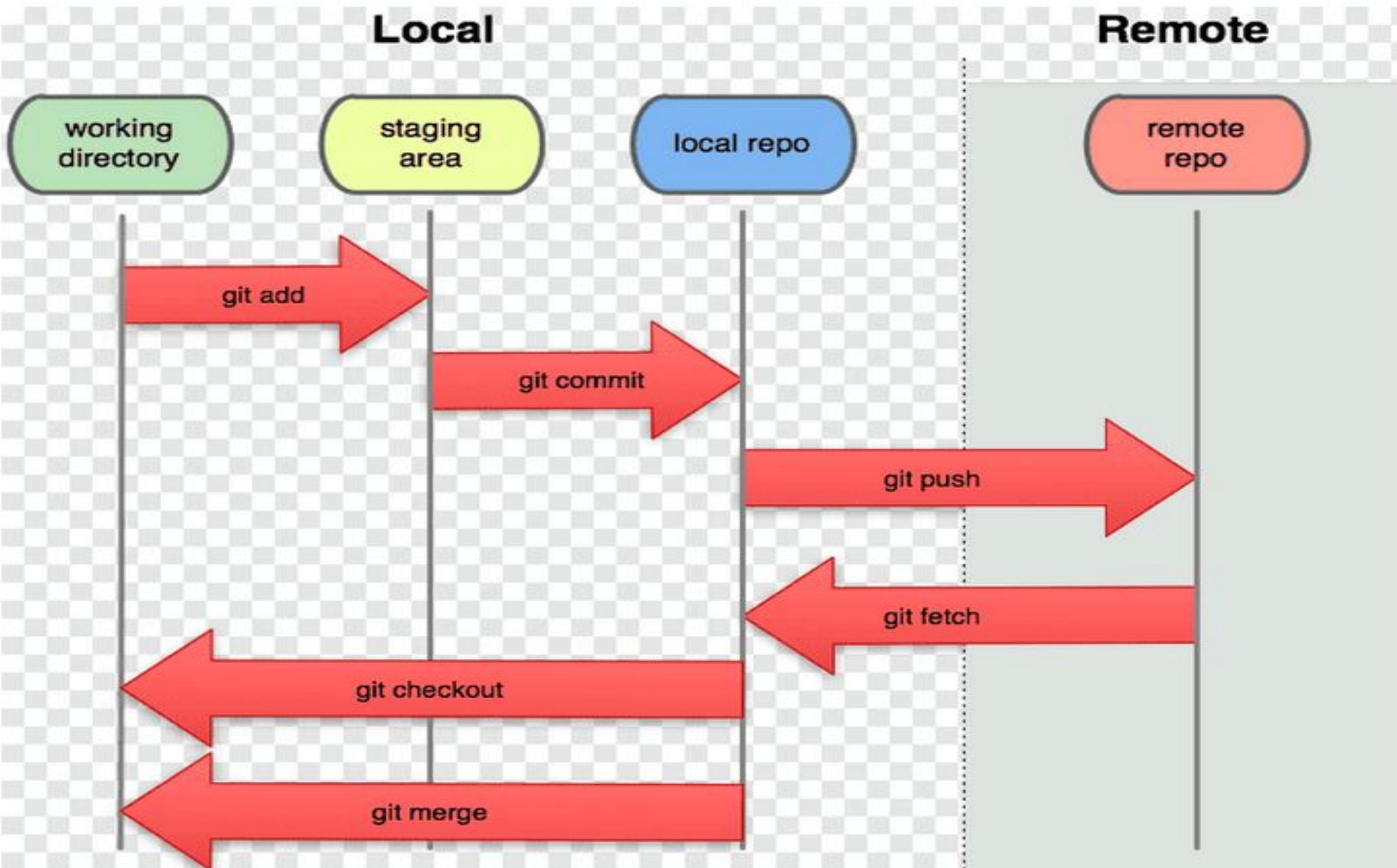
# What is Git?

- Git is a popular version control system. It was created by Linus Torvalds in 2005, and has been maintained by Junio Hamano since then.
- It is used for:
  - Tracking code changes
  - Tracking who made changes
  - Coding collaboration



# What does Git do?

- Manage projects with **Repositories**
- **Clone** a project to work on a local copy
- Control and track changes with **Staging** and **Committing**
- **Branch** and **Merge** to allow for work on different parts and versions of a project
- **Pull** the latest version of the project to a local copy
- **Push** local updates to the main project



# Working with Git

- Initialize Git on a folder, making it a **Repository**
- Git now creates a hidden folder to keep track of changes in that folder
- When a file is changed, added or deleted, it is considered **modified**
- You select the modified files you want to **Stage**
- The **Staged** files are **Committed**, which prompts Git to store a **permanent** snapshot of the files
- Git allows you to see the full history of every commit.
- You can revert back to any previous commit.
- Git does not store a separate copy of every file in every commit, but keeps track of changes made in each commit!

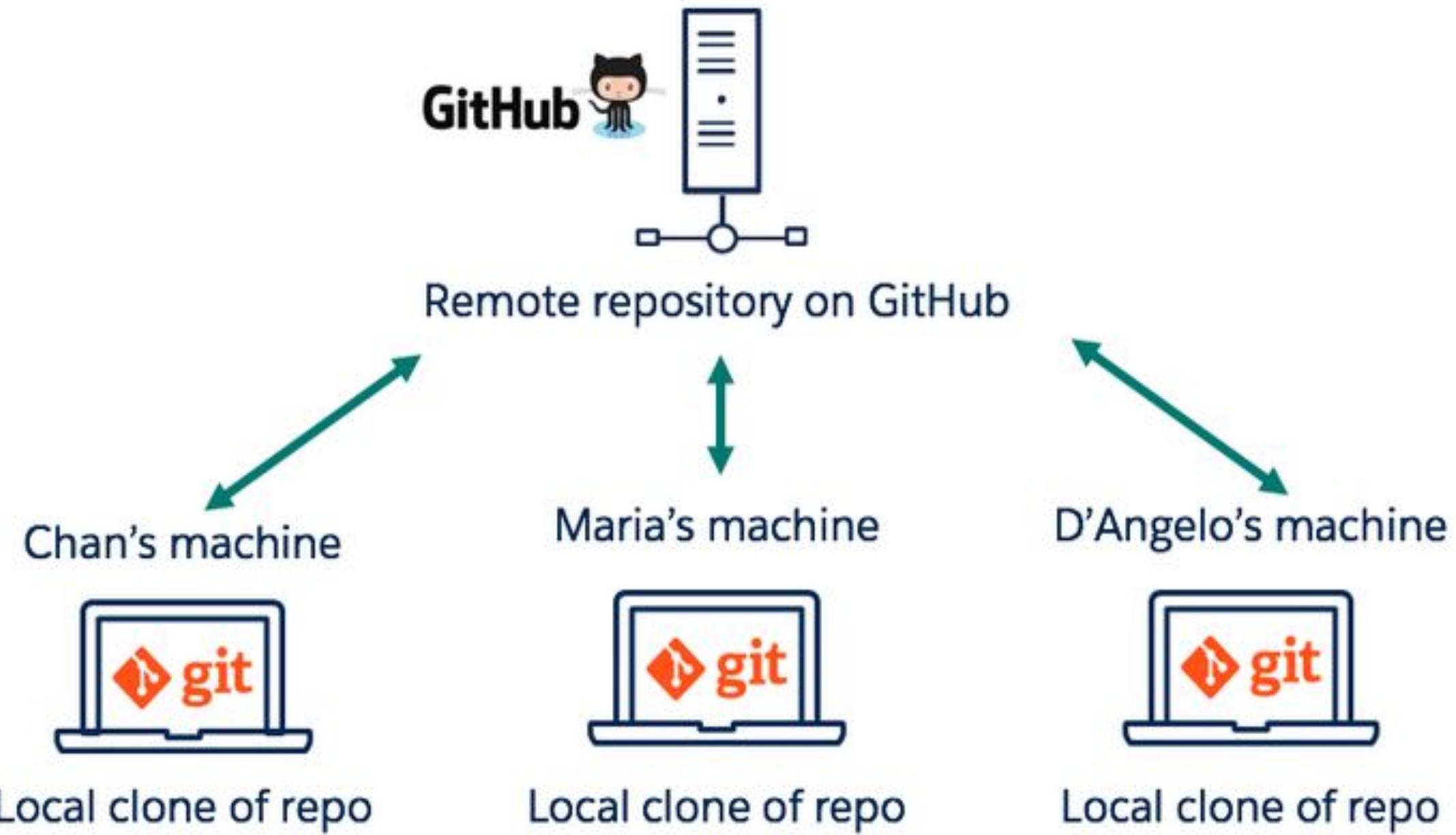
# Why Git?

- Over 70% of developers use Git!
- Developers can work together from anywhere in the world.
- Developers can see the full history of the project.
- Developers can revert to earlier versions of a project.

# What is GitHub?

- GitHub is an online hosting service for Git repositories.
- GitHub lets you store your repo on their platform.
- Another awesome feature that comes with GitHub is the ability to collaborate with other developers from any location.





# How to configure Git

- To verify this, you can run this command on the command line:  
**> git --version.**  
This shows you the current version installed on your PC.
- The next thing you'll need to do is to set your username and email address. Git will use this information to identify who made specific changes to files.
- To set your username, type and execute these commands:  
**> git config --global user.name "YOUR\_USERNAME"**  
**>git config --global user.email "YOUR\_EMAIL"**

**Note:** Use global to set the username and e-mail for every repository on your computer.  
If you want to set the username/e-mail for just the current repo, you can remove global

# How to Create and Initialize a Project in Git

- Now to initialize your project, simply run  
`> git init`
- This will tell Git to get ready to start watching your files for every change that occurs.

# How to add files in Git

- When we first initialized our project, the file was not being tracked by Git. To do that, we use this command `git add .`. The period or dot that comes after add means all the files that exist in the repository. If you want to add a specific file, maybe one named `about.txt`, you use

**`>git add about.txt`**

**`$ git add .`**

Now our file is in the staged state. You will not get a response after this command, but to know what state your file is in, you can run the **`git status`** command.

# Committed state and Modified state

- A file is in the **committed** state when all the changes made to the file have been saved in the local repo. Files in the committed stage are files ready to be pushed to the remote repo (on GitHub).
- A file in the **modified** state has some changes made to it but it's not yet saved. This means that the state of the file has been altered from its previous state in the committed state.

# Staged state

- A file in the **staged** state means it is ready to be committed. In this state, all necessary changes have been made so the next step is to move the file to the commit state.

# How to commit files in Git

- The next state for a file after the staged state is the committed state.  
To commit our file, we use the

**>git commit -m "first commit"**

- The first part of the command git commit tells Git that all the files staged are ready to be committed so it is time to take a snapshot.
- The second part -m "first commit" is the commit message. -m is shorthand for message while the text inside the parenthesis is the commit message.

# Push the repository to GitHub

- The first command
- `>git remote add origin  
https://github.com/ihechikara/git-and-github-tutorial.git`
- creates a connection between your local repo and the remote repo on Github.
- the project already exists locally so we will use commands in the "...or push an existing repository from the command line" section. These are the commands:
- `git remote add origin  
https://github.com/ihechikara/git-and-github-tutorial.git`
- `git branch -M main`
- `git push -u origin main`

# Cont....

- The first command `git remote add origin https://github.com/ihechikara/git-and-github-tutorial.git` creates a connection between your local repo and the remote repo on Github.
- The second command `git branch -M main` changes your main branch's name to "main". The default branch might be created as "master", but "main" is the standard name for this repo now.
- The last command > **`git push -u origin main`** pushes your repo from your local device to GitHub. You should get a response similar to this:

```
IHECHIKARA@DESKTOP-KGB10SU MINGW64 ~/Desktop/Git and GitHub tutorial (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 325 bytes | 325.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ihechikara/git-and-github-tutorial.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

# Cont...

- We first add the file by using
- **git add .** which adds all the files in the folder (one file in our case).
- Then we commit the file by running
- **git commit -m "added new task"** followed by
- **git push -u origin main.**
- Those are the three steps to pushing your modified files to GitHub.

# How to Use Branches in Git

- With branches, you can create a copy of a file you would like to work on without messing up the original copy. You can either merge these changes to the original copy or just let the branch remain independent.
- At this point, I want to add more tasks to the list but I am not yet sure whether I want them on my main list. So I will create a new branch called test to see what my list would look like with more tasks included.

# create a new branch

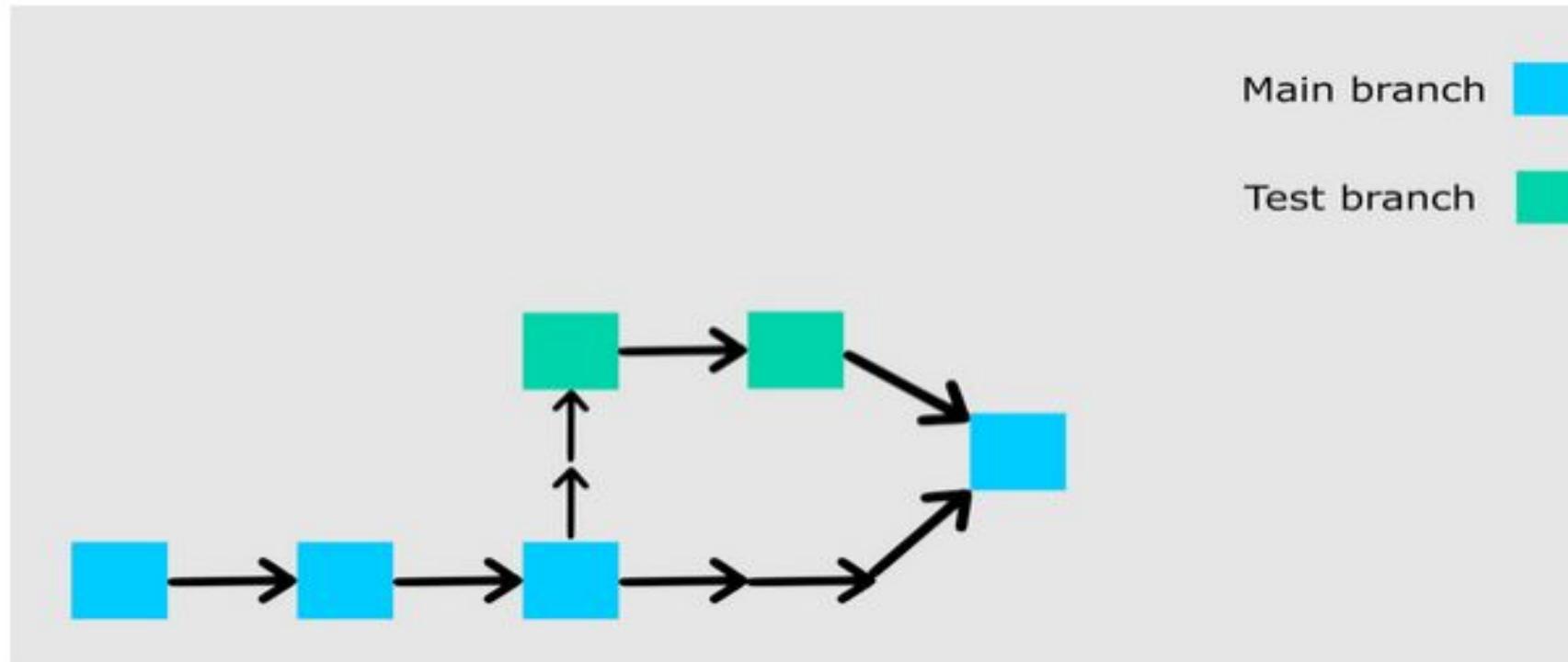
- To create a new branch, run this command:
- **\$ git checkout -b test .**
- I will break it down. checkout tells Git it is supposed to switch to a new branch. **-b** tells Git to create a new branch. **test** is the name of the branch to be created and switched to.
- After committing your test branch, switch back to the main branch by running this command:
- **\$git checkout main.**
- You can check all the branches that exist in your repo by running the **\$git branch** command.
- Now we can merge the changes we made in the test branch into the main branch by running.
- **\$git merge test.**

# Push new branch to git

- If you would like to push your test branch, switch to the branch using
- **\$git checkout test** and then run
- **\$ git push -u origin test**

# To view all branches in repository

- \$git branch -a /-r
- -a represents all the braches from local and remote
- It display all the branches in remote repository.



git merge

# How to Pull a Repository in Git

- To pull in Git means to clone a remote repository's current state into your computer/repository.
- This comes in handy when you want to work on your repo from a different computer or when you are contributing to an open source project online.
- Go to GitHub, and on your repository's main page you should see a green button that says "Code".
- When you click on the button, you should see some options in a dropdown menu. Go on and copy the HTTPS URL. After that, run
- **\$git clone YOUR\_HTTPS\_URL.**
- This command pulls the remote repository into your local computer in a folder

# NPM Scripts

# Content

- Learn what an npm script is in Node
- Learn how to run scripts with npm
- Explain the difference between custom and default scripts

# What Are NPM Scripts?

- An npm script is a convenient way to bundle common shell commands for your project.
- They are typically commands, or a string of commands, which would normally be entered at the command line in order to do something with your application.
- Scripts are stored in a project's *package.json* file, which means they're shared amongst everyone using the codebase.
- They help automate repetitive tasks

# Why Use npm Scripts?

- You can add scripts to package.json, run them on the command line, use pre and post hooks, and pass line arguments.
- npm scripts are a powerful way of automating tasks in JavaScript projects. They can improve your workflow and save you time by providing you with consistent commands to run multiple tasks.
- Common use cases for npm scripts include building your Node project, starting a development server, compiling CSS, linting, minifying, or anything else you find yourself typing into your terminal frequently that's related to your project.

# scripts

- The "scripts" property of your package.json file supports a number of built-in scripts and their preset life cycle events as well as arbitrary scripts.
- These all can be executed by running
- **\$npm run-script <stage>** or **npm run <stage>** for short.
- Pre and post commands with matching names will be run for those as well (e.g. premyscript, myscript, postmyscript).
- Scripts from dependencies can be run with
- **\$ npm explore <pkg> -- npm run <stage>**.

- 1. Run npm init on the terminal. The command will ask you some questions about your project. Answer them to create an appropriate package.json file.
- npm init
- In the package.json file, locate the scripts field. Here, you can add the name of a script and the command it should run as key/value pairs. For example, the script below, named hello-world, prints “Hello world” in the terminal when run.
- {
- "scripts": {
- "hello-world": "echo \\\"Hello world\\\""
- }
- }

# common scripts for JavaScript projects:

- Here are some common scripts for JavaScript projects:
- **start**: This script starts the development server. For example, in a Node project, it can [run the server using nodemon](#).
- **build**: Generates the production code for your application and may use a tool like webpack to minify and bundle the code.
- **test**: This script runs the tests defined in your project. It may run a testing framework like Jest.
- **lint**: A lint script runs a linting tool such as ESLint to check the code for potential errors.
- **watch**: This script watches the source code for changes and then runs a command. It's useful for re-running tests or rebuilding the application on code change.
- **deploy**: Runs a command that deploys the application to the specified environment like production or staging.

# Pre and Post Scripts

- npm supports pre and post scripts. Pre scripts run before a specific script while post scripts run afterward. You can create pre and post scripts for any script, just prefix "pre" or "post" to your script name.
- For example, below are pretest and posttest scripts that will run before and after the test script respectively.
- {
- "scripts": {
- "pretest": "npm run lint",
- "test": "jest",
- "posttest": "npm run build"
- }
- }

# Running npm Scripts From package.json

- Once you've added an npm script to package.json, you can run it using the npm run command.
- **\$npm run <script-name>**
- For example, to run the start script defined earlier, use:  
**\$npm run start**

# Running Multiple npm Scripts

- You can run multiple npm scripts in two ways:
  - Sequentially
  - In Parallel
- To run multiple npm scripts sequentially use `&&`, for example:  
**`$npm run start && npm test`**
- To run multiple npm scripts in parallel use `&`, for example:  
**`$npm run server & npm run client`**

# Passing Command Line Arguments to Scripts

- You can pass command line arguments to an npm script using two dashes "--" after the script name. For example, the following command runs the test script with the watch argument:
- **\$npm run test -- --watch**
- You can also send a command line argument to an npm script like this:
- **\$npm run my-port --PORT=3000**
- Then access it in the script as follows.
- "scripts": {
- "my-port": "echo \\\"Port: %npm\_config\_PORT%\\\""
- }
- The script should print "Port: 3000" when you run it.

# PHP Introduction

Web Technologies

IIIT -Nuzvid



# What is PHP

- PHP is an open-source, interpreted, and object-oriented scripting language that can be executed at the server-side.
- PHP is well suited for web development. Therefore, it is used to develop web applications (an application that executes on the server and generates the dynamic page.).
- PHP was created by **Rasmus Lerdorf** in **1994** but appeared in the market in 1995.



## Cont....

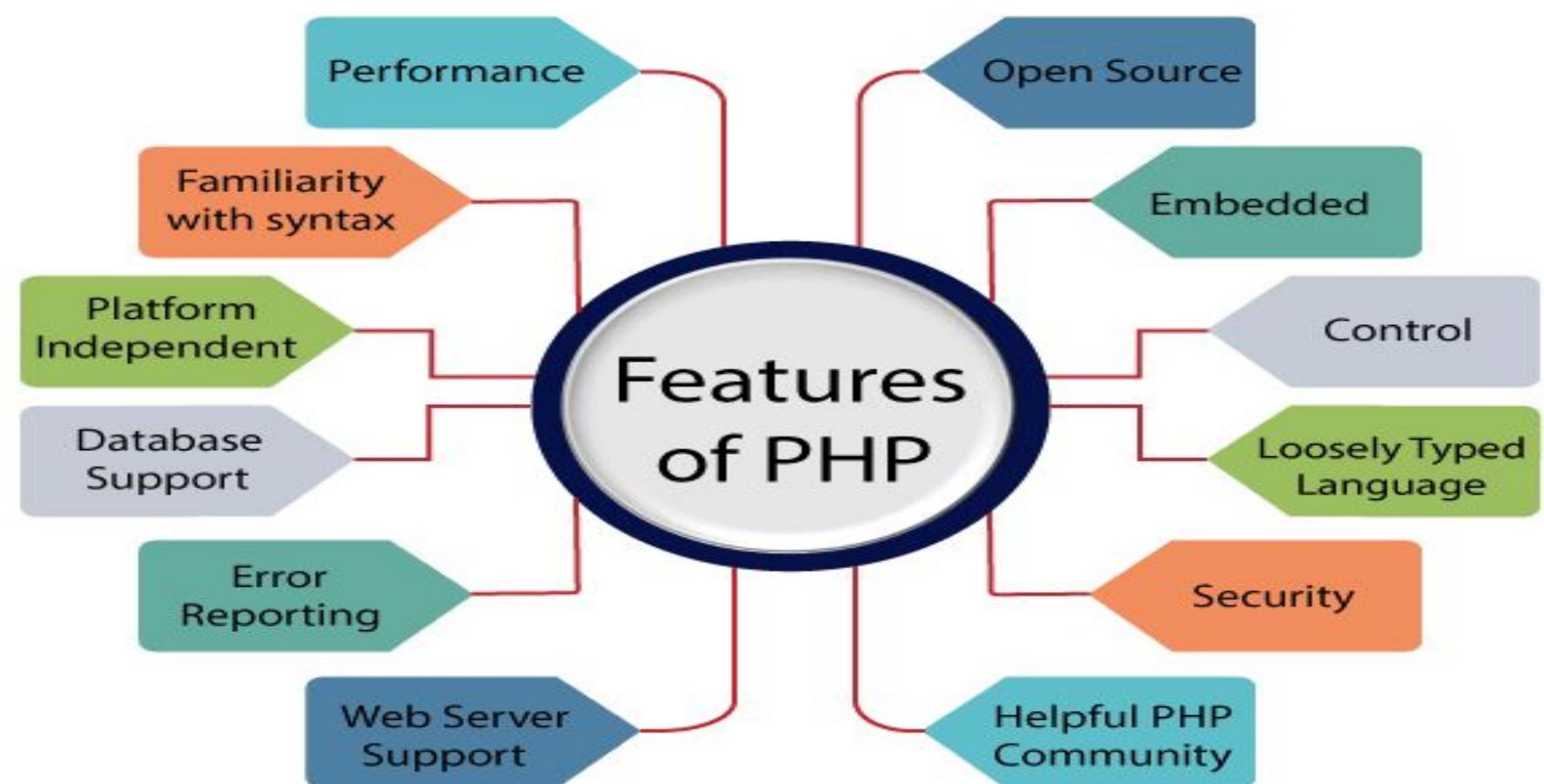
- PHP stands for Hypertext Preprocessor.
- PHP is an interpreted language, i.e., there is no need for compilation.
- PHP is faster than other scripting languages, for example, ASP and JSP.
- PHP is a server-side scripting language, which is used to manage the dynamic content of the website.
- PHP can be embedded into HTML.
- PHP is an object-oriented language.
- PHP is an open-source scripting language.
- PHP is simple and easy to learn language.



# Why use PHP

- PHP is a server-side scripting language, which is used to design the dynamic web applications with MySQL database.
- It handles dynamic content, database as well as session tracking for the website.
- You can create sessions in PHP.
- It can access cookies variable and also set cookies.
- It helps to encrypt the data and apply validation.
- PHP supports several protocols such as HTTP, POP3, SNMP, LDAP, IMAP, and many more.
- As PHP is easy to install and set up, this is the main reason why PHP is the best language to learn.
-

# Features of PHP





# Web Development

- PHP is widely used in web development nowadays. PHP can develop dynamic websites easily. But you must have the basic knowledge of following technologies for web development as well.
- HTML
- CSS
- JavaScript
- Ajax
- XML and JSON
- jQuery



# What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data

# Install PHP

- To install PHP, we will suggest you to install AMP (Apache, MySQL, PHP) software stack. It is available for all operating systems. There are many AMP options available in the market that are given below:
  - WAMP for Windows
  - LAMP for Linux
  - MAMP for Mac
  - XAMPP (Cross, Apache, MySQL, PHP, Perl) for Cross Platform: It includes some other components too such as FileZilla, OpenSSL, Webalizer, Mercury Mail, etc.
  - If you are on Windows and don't want Perl and other features of XAMPP, you should go for WAMP



# How to run PHP code in XAMPP

- Generally, a PHP file contains HTML tags and some PHP scripting code. It is very easy to create a simple PHP example. To do so, create a file and write HTML tags + PHP code and save this file with .php extension.
- All PHP code goes between the php tag. It starts with <?php and ends with ?>. The syntax of PHP tag is given below:
  - <?php
  - //your code here
  - ?>



# First program

```
□ <!DOCTYPE>
□ <html>
□ <body>
□ <?php
□ echo "<h2>Hello First PHP</h2>";
□ ?>
□ </body>
□ </html>
```

## Variables

By now we should know how to use variables and know what variables are.

To quickly sum up what a variable is, it is a name that holds information - functions, strings, integers, etc.

With Javascript we wrote variables by writing `var myName;` but in PHP, we do things a little differently. We use a dollar sign.

`$myName` is a variable.



## Rules

All variables must start with a \$

Variables cannot start with a number.

All variables must start with a letter or an underscore \_

No spaces are allowed. Use underscores instead.

Numbers need no quotes (Just like JS), Quote everything else.

PHP variables are CaSe-SeNsItIvE!!!

Remember, long variable names can be camel cased. So instead of..

`$asuperlongvariablename`

you can write

`$aSuperLongVariableName` or `$A_Super_Long_Variable_Name`



# PHP Data Types

- PHP supports the following data types:
- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL

# PHP String

- A string is a sequence of characters, like "Hello world!".
- A string can be any text inside quotes. You can use single or double quotes:
- ```
<?php
$x = "Hello world!";
$y = 'Hello world!';

echo $x;
echo "<br>";
echo $y;
?>
```

# PHP Integer and other types

- An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.
- The PHP var\_dump() function returns the data type and value:
  - <?php
  - \$x = 5985;
  - \$y = 10.365;
  - \$x1 = true;
  - \$y1 = false;
  - var\_dump(\$x);
  - Var\_dump(\$y);
  - var\_dump(\$x1);
  - Var\_dump(\$y1);
  - ?>



# PHP Array

- An array stores multiple values in one single variable.
- <?php  
\$cars = array("Volvo","BMW","Toyota");  
var\_dump(\$cars);  
?>

# PHP String Functions

- **strlen()** - Return the Length of a String
- **str\_word\_count()** - Count Words in a String
- **strrev()** - Reverse a String
- **strpos()** - Search For a Text Within a String
- **str\_replace()** - Replace Text Within a String
- **ucwords()** – the first letter of each word turned to uppercase.
- **strtoupper()**- change the entire string into uppercase
- **strtolower()**– changed the string into lowe case letters.

## Example to string functions

- <?php  
echo strlen("Hello world!"); // outputs 12
- echo str\_word\_count("Hello world!");
- echo strrev("Hello world!");
- echo strpos("Hello world!", "world"); // outputs 6
- echo str\_replace("world", "Dolly", "Hello world!"); //  
outputs Hello Dolly!  
?>

# PHP Constants

- Constants are like variables except that once they are defined they cannot be changed or undefined.
- A constant is an identifier (name) for a simple value. The value cannot be changed during the script.
- A valid constant name starts with a letter or underscore (no \$ sign before the constant name).
- **Note:** Unlike variables, constants are automatically global across the entire script.
- **Syntax**
- `define(name, value, case-insensitive)`
- Create a constant with a **case-sensitive** name:  
`<?php  
define("GREETING", "Welcome to W3Schools.com!");  
echo GREETING;  
?>`
- `<?php  
define("GREETING", "Welcome to W3Schools.com!", true); // case-insensitive  
echo greeting;  
?>`

# PHP Operators

- Operators are used to perform operations on variables and values.
- PHP divides the operators in the following groups:
- Arithmetic operators □ +,-,\* ,/,%(modulus)
- Assignment operators □ "=""    x += y, x -= y, x \*= y, x % = y
- Comparison operators □ >,<,>=,<=,==,!=,<>,!== (not identical)
- Increment/Decrement operators □ ++,--(pre and post)
- Logical operators
- String operators- □ .(concatenate), .= Concatenation assignment
- \$txt1 .= \$txt2      Appends \$txt2 to \$txt1
- Array operators   =>
- Conditional assignment operators

Operator	Name	Example	Result
+	Union	$\$x + \$y$	Union of $\$x$ and $\$y$
$==$	Equality	$\$x == \$y$	Returns true if $\$x$ and $\$y$ have the same key/value pairs
$==$	Identity	$\$x === \$y$	Returns true if $\$x$ and $\$y$ have the same key/value pairs in the same order and of the same types
$!=$	Inequality	$\$x != \$y$	Returns true if $\$x$ is not equal to $\$y$
$!=$	Inequality	$\$x != \$y$	Returns true if $\$x$ is not equal to $\$y$
$!=$	Inequality	$\$x != \$y$	Returns true if $\$x$ is not equal to $\$y$

# PHP Conditional Assignment Operators

Operator	Name	Example	Result
:?	Ternary	$\$x = expr1 ? expr2 : expr3$	Returns the value of \$x. The value of \$x is <i>expr2</i> if <i>expr1</i> = TRUE. The value of \$x is <i>expr3</i> if <i>expr1</i> = FALSE
??	Null coalescing	$\$x = expr1 ?? expr2$	Returns the value of \$x. The value of \$x is <i>expr1</i> if <i>expr1</i> exists, and is not NULL. If <i>expr1</i> does not exist, or is NULL, the value of \$x is <i>expr2</i> . Introduced in PHP 7

Php

- **Conditional Statements**
- **PHP Loops**

# Conditional Statements

- Very often when you write code, you want to perform different actions for different conditions
- In PHP we have the following **conditional statements**:
  1. **if** statement - executes some code if one condition is true
  2. **if...else** statement - executes some code if a condition is true and another code if that condition is false
  3. **if...elseif...else** statement - executes different codes for more than two conditions
  4. **switch statement** - selects one of many blocks of code to be executed

# Simple if

- `if (condition) {  
 code to be executed if condition is true;  
}`
- `<?php  
$t = date("H");  
  
if ($t < "20") {  
 echo "Have a good day!";  
}  
?>`

# The if...else Statement

- `if (condition) {  
 code to be executed if  
 condition is true;  
} else {  
 code to be executed if  
 condition is false;  
}`

```
<?php  
$t = date("H");  
  
if ($t < "20") {  
    echo "Have a good day!";  
} else {  
    echo "Have a good night!";  
}  
?>
```

# if...elseif...else Statement

- `if (condition) {  
 code to be executed if  
 this condition is true;  
} elseif (condition) {  
 code to be executed if  
 first condition is false and  
 this condition is true;  
} else {  
 code to be executed if all  
 conditions are false;  
}`

```
<?php  
$t = date("H");  
  
if ($t < "10") {  
    echo "Have a good morning!";  
} elseif ($t < "20") {  
    echo "Have a good day!";  
} else {  
    echo "Have a good night!";  
}  
?>
```

# switchS

- Use the switch statement to select one of many blocks of code to be executed.
- ```
switch (n) {  
    case label1:  
        code to be executed if n=label1;  
        break;  
    case label2:  
        code to be executed if n=label2;  
        break;  
    case label3:  
        code to be executed if n=label3;  
        break;  
    ...  
    default:  
        code to be executed if n is  
        different from all labels;  
}
```

```
<?php  
$favcolor = "red";  
  
switch ($favcolor) {  
    case "red":  
        echo "Your favorite color is red!";  
        break;  
    case "blue":  
        echo "Your favorite color is blue!";  
        break;  
    case "green":  
        echo "Your favorite color is green!";  
        break;  
    default:  
        echo "Your favorite color is neither red,  
        blue, nor green!";  
}  
?>
```

# PHP Loops

- In PHP, we have the following loop types:
- **while** - loops through a block of code as long as the specified condition is true
- **do...while** - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- **for** - loops through a block of code a specified number of times
- **foreach** - loops through a block of code for each element in an array

# loops

- The while loop executes a block of code as long as the specified condition is true.
- `while (condition is true) {  
 code to be executed;  
}`
- `do {  
 code to be executed;  
} while (condition is true);`

```
<?php  
$x = 1;  
while($x <= 5) {  
    echo "The number is: $x  
<br>";  
    $x++;  
}  
?>
```

# for Loop

- *for (init counter; test counter; increment counter) {  
  code to be executed for each iteration;  
}*
- <?php  
  for (\$x = 0; \$x <= 10; \$x++) {  
    echo "The number is: \$x <br>";  
  }  
?>

# foreach Loop

- The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.
- <?php
- \$colors = array("red", "green", "blue", "yellow");
- foreach (\$colors as \$value) {
- echo "\$value <br>";
- }
- ?>

- <?php
- \$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
- **foreach**(\$age as \$x => \$val) {
  - echo "\$x = \$val<br>";
  - }
  - ?>

# Break and Continue

- The break statement can also be used to jump out of a loop.

```
<?php  
for ($x = 0; $x < 10; $x++) {  
    if ($x == 4) {  
        break;  
    }  
    echo "The number is: $x <br>";  
}  
?>
```

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

```
<?php  
for ($x = 0; $x < 10; $x++) {  
    if ($x == 4) {  
        continue;  
    }  
    echo "The number is: $x <br>";  
}  
?>
```

# PHP Functions

- function *functionName()* {  
  *code to be executed*;  
}
- <?php  
  function writeMsg() {  
    echo "Hello world!";  
  }  
  
  writeMsg(); // call the function  
?>

## Function Arguments

```
<?php  
function familyName($fname, $year) {  
  echo "$fname Refsnes. Born in $year  
<br>";  
}  
  
familyName("Hege", "1975");  
familyName("Stale", "1978");  
familyName("Kai Jim", "1983");  
?>
```

# PHP GET and POST

Web Technologies

IIIT Nuzvid

# What is HTTP?

- The Hypertext Transfer Protocol (HTTP) is designed to enable communications between clients and servers.
- HTTP works as a request-response protocol between a client and server.
- Example: A client (browser) sends an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.
- **GET, POST ,PUT ,HEAD ,DELETE ,PATCH ,OPTIONS ,CONNECT,TRACE**

# superglobals

- Some predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.
- The PHP superglobal variables are:
- `$GLOBALS`
- `$_SERVER`
- `$_REQUEST`
- `$_POST`
- `$_GET`
- `$_COOKIE`
- `$_SESSION`

# PHP \$GLOBALS

- `$GLOBALS` is a PHP super global variable which is used to access global variables from anywhere in the PHP script (also from within functions or methods).
- PHP stores all global variables in an array called `$GLOBALS[index]`. The *index* holds the name of the variable.

```
• php
$x = 75;
$y = 25;

function addition() {
    $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}

addition();
echo $z;
?>
```

# PHP \$\_SERVER

- `$_SERVER` is a PHP super global variable which holds information about headers, paths, and script locations.
- ```
<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
?>
```

# The GET Method

- GET is used to request data from a specified resource.
- Note that the query string (name/value pairs) is sent in the URL of a GET request:
  - /test/demo\_form.php?name1=value1&name2=value2
- **Some notes on GET requests:**
  - GET requests can be cached
  - GET requests remain in the browser history
  - GET requests can be bookmarked
  - GET requests should never be used when dealing with sensitive data
  - GET requests have length restrictions
  - GET requests are only used to request data

# `$_REQUEST`

- PHP `$_REQUEST` is a PHP super global variable which is used to collect data after submitting an HTML form.

```
<html>
<body>

<form method="post"
action="<?php echo
$_SERVER['PHP_SELF'];?>">
    Name: <input type="text"
name="fname">
    <input type="submit">
</form>
```

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST")
{
    // collect value of input field
    $name = $_REQUEST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>
</body>
</html>
```

# POST Method

- PHP `$_POST` is a PHP super global variable which is used to collect form data after submitting an HTML form with `method="post"`. `$_POST` is also widely used to pass variables.
- POST is used to send data to a server to create/update a resource.
- The data sent to the server with POST is stored in the request body of the HTTP request:
- The PHP provides `$_POST` associative array to access all the sent information using POST method.
- The POST method can be used to send ASCII as well as binary data.
- **Some notes on POST requests:**
- POST requests are never cached
- POST requests do not remain in the browser history
- POST requests cannot be bookmarked
- POST requests have no restrictions on data length

# Compare GET vs. POST

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL  Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs

```
<html>
  <body>
    <form action = "gettest.php" method = "GET">
      Username: <input type = "text" name = "username" /> <br>
      Blood Group: <input type = "text" name = "bloodgroup" /> <br>
      <input type = "submit" />
    </form>
  </body>
</html>
```

# Save this file with gettest.php

```
<html>
  <body>
```

Welcome <?php echo \$\_GET["username"]; ?> </br>

Your blood group is: <?php echo \$\_GET["bloodgroup"]; ?>

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

When the user will click on Submit button after filling the form, the URL sent to the server could look something like this:

# `$_cookies`

- A cookie is a small file of 4KB that a server embeds in the user's computer.
- Every time a user requests the browser for a webpage, it also sends the cookies.
- These cookies are taken into use to identify the user.
- Cookies allow the creation of a personalized surfing environment for use.
- It helps in tracking the sites visited by the user. PHP allows creating and also retrieving a cookie value.
- All of the data in the cookie is automatically sent to the server each time the browser requests a page from the server.

# Syntax for creating Cookie

- `setcookie("first_cookie", "username_of_the account is anon", time() + 3600, "d:/testcoookie/", "", 0);`
- First cookie located here, is the name of the cookie that stores the meaning as the username of the account is anon and the time between logged in and 3600 milliseconds. Following that, it will be deleted from the screen.

# `$_session`

- a session is stored information in variables on the server. It is also considered a global variable stored on the server. When a session is created, a cookie is generated with a unique ID of the session. The cookie is later stored in the user's computer. Sessions are generally created in an application where temporary information larger than 4KB has to be stored.

# PHP File Inclusion & File system

---

WEB TECHNOLOGIES

# File Inclusion

---

You can include the content of a PHP file into another PHP file before the server executes it. There are two PHP functions which can be used to include one PHP file into another PHP file.

- The **include()** Function
- The **require()** Function

This is a strong point of PHP which helps in creating functions, headers, footers, or elements that can be reused on multiple pages.

This will help developers to make it easy to change the layout of complete website with minimal effort. If there is any change required then instead of changing thousand of files just change included file.

# The include() Function

---

The `include()` function takes all the text in a specified file and copies it into the file that uses the `include` function.

**Both `include` and `require` are identical to each other, except failure.**

`include` only generates a warning, i.e., `E_WARNING`, and continue the execution of the script.

`require` generates a fatal error, i.e., `E_COMPILE_ERROR`, and stop the execution of the script.

# PHP Filesystem

---

- ❖ The filesystem functions allow you to access and manipulate the filesystem.
- ❖ The filesystem functions are part of the PHP core. There is no installation needed to use these functions.
- ❖ For accessing the files on the system, the file path will be used.

# File System functions

---

- PHP File Formats Support
- PHP file() Function
- PHP file\_exists() Function
- PHP fopen() Function
- PHP fwrite() Function
- PHP fclose() Function
- PHP fgets() Function
- PHP copy() Function
- Deleting a file
- PHP file\_get\_contents() Function

# PHP File Formats Support

---

PHP file functions support a wide range of file formats that include:

File.txt

File.log

File.custom\_extension i.e. file.xyz

File.csv

File.gif, file.jpg etc

Files provide a permanent cost effective data storage solution for simple data compared to databases that require other software and skills to manage DBMS systems.

You want to store simple data such as server logs for later retrieval and analysis

You want to store program settings i.e. program.ini

# PHP file() Function

---

PHP provides a convenient way of working with files via its rich collection of built in functions.

**PHP file\_exists() Function:-**his function is used to determine whether a file exists or not.

It comes in handy when we want to know if a file exists or not before processing it.

You can also use this function when creating a new file and you want to ensure that the file does not already exist on the server.

```
<?php  
file_exists($filename);  
?>
```

# PHP fopen() Function

---

The fopen function is used to open files. It has the following syntax

```
<?php  
fopen($file_name, $mode, $use_include_path, $context);  
?>
```

HERE,

“fopen” is the PHP open file function

“\$file\_name” is the name of the file to be opened

“\$mode” is the mode in which the file should be opened, the table below shows the modes

Mode	description
r	<ul style="list-style-type: none"><li>•Read file from beginning. Returns false if the file doesn't exist. Read only</li></ul>
r+	<ul style="list-style-type: none"><li>•Read file from beginning Returns false if the file doesn't exist. Read and write</li></ul>
w	<ul style="list-style-type: none"><li>•Write to file at beginning truncate file to zero length, If the file doesn't exist attempt to create it.</li><li>•Write only</li></ul>
w+	<ul style="list-style-type: none"><li>•Write to file at beginning, truncate file to zero length, If the file doesn't exist attempt to create it. Read and Write</li></ul>
a	<ul style="list-style-type: none"><li>•Append to file at end ,If the file doesn't exist attempt to create it. Write only</li></ul>
a+	<ul style="list-style-type: none"><li>•Php append to file at end .If the file doesn't exist attempt to create it,Read and write</li></ul>

# PHP fwrite() Function

---

The fwrite function is used to write files.

It has the following syntax

```
<?php  
fwrite($handle, $string, $length);  
?>
```

HERE,

“fwrite” is the PHP function for writing to files

“\$handle” is the file pointer resource

“\$string” is the data to be written in the file.

“\$length” is optional, can be used to specify the maximum file length.

# Create File

---

```
<?php  
$fh = fopen("my_settings.txt", 'w') or die("Failed to create file");  
$text = <<<_END  
localhost;root;pwd1234;my_database _END;  
fwrite($fh, $text) or die("Could not write to file");  
fclose($fh);  
echo "File 'my_settings.txt' written successfully"; ?>
```

# PHP fclose() Function

---

The fclose() function is used to close a file in php which is already open

It has the following syntax.

```
<?php  
fclose($handle);  
?>
```

HERE,

“fclose” is the PHP function for closing an open file

“\$handle” is the file pointer resource.

# PHP copy() Function

---

The PHP copy function is used to copy files. It has the following basic syntax. `copy($file,$copied_file);`

HERE,

```
<?php  
copy('my_settings.txt', 'my_settings_backup.txt') or die("Could not  
copy file");  
echo "File successfully copied to 'my_settings_backup.txt"';  
?>
```

# Deleting a file

---

```
<?php  
if (!unlink('my_settings_backup.txt'))  
{   echo "Could not delete file";  
}  
else  
{   echo "File 'my_settings_backup.txt' successfully deleted";  
}  
?>
```

# PHP file\_get\_contents() Function

---

The file\_get\_contents function is used to read the entire file contents.

```
<?php  
echo "<pre>"; // Enables display of line feeds  
echo file_get_contents("my_settings.txt");  
echo "</pre>"; // Terminates pre tag  
?>
```

# PHP file size and type

---

The filesize function returns the size of the given file. The size is specified in bytes.

```
<?php  
$filename = "fruits.txt";  
$filesize = filesize($filename);  
echo "The file size is: $filesize bytes\n";  
?>
```

---

Function	Description
File_exists	Used to determine if a file exists or not
fopen	Used to open a file. Returns a pointer to the opened file
fwrite	Used to write to files
fclose	Used to close files
fgets	Used to read a file line by line
copy	Used to copy an existing file
unlink	Used to delete an existing file
file_get_contents	Used to return the contents of a file as a string

# PHP-DIRECTORIES PHP-FILE UPLOAD

Web Technologies  
RGUKT, IIIT Nuzvid.

# PHP PARSING DIRECTORIES

PHP also allows you to work with directories on the file system, for example, you can open a directory and read its contents, create or delete a directory, list all files in the directory, and so on.

- **Creating a New Directory**
- **Copying Files from One Location to Another**
- **Listing All Files in a Directory**
- **Listing All Files of a Certain Type**

# CREATING A NEW DIRECTORY

you can create a new and empty directory by calling the PHP `mkdir()` function with the path and name of the directory to be created, as shown in the example below:

To make the `mkdir()` function work, the parent directories in the directory path parameter has to exist already, for example, if you specify the directory path as `testdir/subdir` than the `testdir` has to exist otherwise PHP will generate an error.

```
<?php
// The directory path
$dir = "testdir";
// Check the existence of directory
if(!file_exists($dir)){
    // Attempt to create directory
    if(mkdir($dir)){
        echo "Directory created
successfully.";
    } else{
        echo "ERROR: Directory could not
be created.";
    }
} else{
    echo "ERROR: Directory already
exists.";
}
```

# COPYING FILES FROM ONE LOCATION TO ANOTHER

You can copy a file from one location to another by calling PHP `copy()` function with the file's source and destination paths as arguments. If the destination file already exists it'll be overwritten.

the target directory which is *backup* and the source file i.e. "example.txt" has to exist already; otherwise PHP will generate an error.

```
<?php  
// Source file path  
$file = "example.txt";  
// Destination file path  
$newfile = "backup/example.txt";  
  
// Check the existence of file  
if(file_exists($file)){  
    // Attempt to copy file  
    if(copy($file, $newfile)){  
        echo "File copied successfully.";  
    } else{  
        echo "ERROR: File could not be copied.";  
    }  
} else{  
    echo "ERROR: File does not exist."  
}  
?>
```

# LISTING ALL FILES IN A DIRECTORY

You can use the PHP `scandir()` function to list files and directories inside the specified path. Now we're going to create a custom function that will recursively list all files in a directory using PHP. This script will be helpful if you're working with deeply nested directory structure.

```
echo "ERROR: No files found in the directory.";  
    }  
} else {  
    echo "ERROR: The directory does not exist."  
}  
  
// Call the function  
outputFiles("mydir");  
?>
```

```
<?php  
// Define a function to output files in a directory  
function outputFiles($path){  
    // Check directory exists or not  
    if(file_exists($path) && is_dir($path)){  
        // Scan the files in this directory  
        $result = scandir($path);  
        // Filter out the current (.) and parent (..) directories  
        $files = array_diff($result, array('.', '..'));  
        if(count($files) > 0){  
            // Loop through retuned array  
            foreach($files as $file){  
                if(is_file("$path/$file")){  
                    // Display filename  
                    echo $file . "<br>";  
                } else if(is_dir("$path/$file")){  
                    // Recursively call the function if directories found  
                    outputFiles("$path/$file");  
                }  
            }  
        } else{
```

# LISTING ALL FILES OF A CERTAIN TYPE

While working on directory and file structure, sometimes you might need to find out certain types of files within the directory, for example, listing only .text or .png files, etc. You can do this easily with the PHP `glob()` function, which matches files based on the pattern.

```
<?php
/* Search the directory and loop through
returned array containing the matched files */
foreach(glob("documents/*.txt") as $file){
    echo basename($file) . " (size: " . filesize($file) . " bytes)" . "<br>";
}
?>
```

The `glob()` function can also be used to find all the files within a directory or its subdirectories. The function defined in the following example will recursively list all files

<?php within a directory.

```
// Define a function to output files in a directory
function outputFiles($path){
    // Check directory exists or not
    if(file_exists($path) && is_dir($path)){
        // Search the files in this directory
        $files = glob($path . "*");
        if(count($files) > 0){
            // Loop through retuned array
            foreach($files as $file){
                if(is_file("$file")){
                    // Display only filename
                    echo basename($file) . "<br>";
                } else if(is_dir("$file")){
                    // Recursively call the function if directories found
                    outputFiles("$file");
                }
            }
        }
    }
}
```

```
else{
    echo "ERROR: No such file found in the
directory.";
}
} else {
    echo "ERROR: The directory does not exist.";
}
}

// Call the function
outputFiles("mydir");
?>
```

# **PHP FILE UPLOAD**

You can upload files on remote server using a Simple HTML form and PHP. You can upload any kind of file like images, videos, ZIP files, Microsoft Office documents, PDFs, as well as executables files and a wide range of other file types.

**Step 1: Creating an HTML form to upload the file**

**Step 2: Processing the uploaded file**

**Step3: simply display the details of the uploaded file**

# CREATING AN HTML FORM TO UPLOAD THE FILE

```
<body>
<form action="upload-manager.php" method="post" enctype="multipart/form-data">
    <h2>Upload File</h2>
    <label for="fileSelect">Filename:</label>
    <input type="file" name="photo" id="fileSelect">
    <input type="submit" name="submit" value="Upload">
    <p><strong>Note:</strong> Only .jpg, .jpeg, .gif, .png formats allowed to a max size
of 5 MB.</p>
</form>
</body>
```

Note: In addition to a file-select field the upload form must use the HTTP post method and must contain an `enctype="multipart/form-data"` attribute. This attribute ensures that the form data is encoded as multipart MIME data — which is required for uploading the large quantities of binary data such as image, audio, video, etc.

# PROCESSING THE UPLOADED FILE

- It will store the uploaded file in a "upload" folder on permanent basis as well as implement some basic security check like file type and file size to ensure that users upload the correct file type and within the allowed limit.
- Here's the complete code of our "upload-manager.php" file.(given in google classroom of your respective class)
- Note: The above script prevents uploading a file with the same name as an existing file in the same folder. However, if you want to allow this just prepend the file name with a random string or timestamp, like \$filename = time() . '\_' .  
\$\_FILES["photo"]["name"];

# DISPLAY THE DETAILS OF THE UPLOADED FILE

The PHP code in the following example will simply display the details of the uploaded file and stores it in a temporary directory on the web server.

```
<?php
if($_FILES["photo"]["error"] > 0){
    echo "Error: " . $_FILES["photo"]["error"] . "<br>";
} else{
    echo "File Name: " . $_FILES["photo"]["name"] . "<br>";
    echo "File Type: " . $_FILES["photo"]["type"] . "<br>";
    echo "File Size: " . ($_FILES["photo"]["size"] / 1024) . " KB<br>";
    echo "Stored in: " . $_FILES["photo"]["tmp_name"];
}
?>
```

Once a file has been successfully uploaded, it is automatically stored in a temporary directory on the server. To store this file on a permanent basis, you need to move it from the temporary directory to a permanent location using the PHP's `move_uploaded_file()` function.

# PHP –MYSQL-CURD operations

WEB TECHNOLOGIES

IIIT NUZVID

- ▶ We mainly use HTML forms when collecting user input in web-based applications.
- ▶ They range from contact forms, login forms, and also registration forms.
- ▶ Forms are the fundamental interface between the user and the server.
- ▶ Creating a form on a web application is achieved using HTML.
- ▶ PHP connects the web application with the database server.

# create HTML forms

- ▶ Here is an example of a form that submits data to a file named index.php. To have a complete source code, use this HTML code and change the method to either POST or GET where necessary.
- ▶ 

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>HTML Form</title>
</head>
<body>
    <h1>HTML Form</h1>
    <form method="" action="index.php">
        Name: <input type="text" name="name"><br><br/>
        Email: <input type="text" name="email"><br/>
        <br/>
        <input type="submit" value="submit" >
    </form>
</body>
</html>
```

# HTML forms and MySQL database CRUD operations

- ▶ First, create a MySQL database and name it crud. Create a table with three columns, name it user. The columns are: id ,name , email.
- ▶ **Create:** To create a record in the database, Once the form is submitted via the POST method, it is processed, and a record is created in the table user.
- ▶ **Read :** retrieves the inserted data and displays it in an HTML table.
- ▶ **Update :** The update form is displayed when we click the edit button in the table cell. the item to be edited is sent to the script update.php. Either the GET or POST method can be used.
- ▶ **Delete :** To delete a record in the table, the user clicks the delete button in the HTML table. If the id is not empty, then the record with the submitted id is deleted.

# Php and Mysql connection statements



## PHP & MySQL Programming Steps :

### 1) Connection

```
mysqli_connect(Server Name, User Name, Password, Database Name)
```

### 2) Run SQL Query

```
mysqli_query(Connection Name, SQL Query)
```



### 3) Close Connection

```
mysqli_close(Connection Name)
```

# Processing the form data

```
▶  <?php  
▶  # Check if name and email fields are empty  
▶  if(empty($_POST['name']) && empty($_POST['email'])){  
▶      # If the fields are empty, display a message to the user  
▶      echo "<br/> Please fill in the fields";  
▶  # Process the form data if the input fields are not empty  
▶ }else{  
▶     $name= $_POST['name'];  
▶     $email= $_POST['email'];  
▶     echo ('Your Name is:  '. $name. '<br/>');  
▶     echo ('Your Email is:' . $email. '<br/>');  
▶ }  
▶ ?>
```

# Create a database server connection

- ▶ Create a file named connect.php and place in it the following code. The scripts make a connection to the MySQL database server.
- ▶ <?php
- ▶     \$servername = "localhost";
- ▶     \$username = "root"; # MySQL user
- ▶     \$password = ""; # MySQL Server root password
- ▶     \$dbname='crud'; # Database name
- ▶     \$conn = new mysqli(\$servername, \$username, \$password, \$dbname);
- ▶     if (\$conn->connect\_error) {
- ▶         # Display an error message if the connection fails
- ▶         die("Connection failed: " . \$conn->connect\_error);
- ▶     }
- ▶ ?>

# To create a record in the database

- Once the form is submitted via the POST method, it is processed, and a record is created in the table user.

```
<?php  
// Include script to make a database connection  
include("connect.php");  
// Check if input fields are empty  
if(empty($_POST['name']) &&  
empty($_POST['email'])){  
    // If the fields are empty, display a message to the user  
    echo "Please fill in the fields";  
// Process the form data if the input fields are not empty  
}
```

```
else{  
    $name= $_POST['name'];  
    $email= $_POST['email'];  
    echo ('Your Name is: ' . $name. '<br/>');  
    echo ('Your Email is:' . $email. '<br/>');  
    # Insert into the database  
    $query = "INSERT INTO user(name,email) VALUES  
    ('$name','$email')";  
    if (mysqli_query($conn, $query)) {  
        echo "New record created successfully !";  
    } else {  
        echo "Error inserting record: " . $conn->error;  
    }  
}  
?>
```

# Update

- ▶ The update form is displayed when we click the edit button in the table cell.
- ▶ The update button is a submit button for a form with hidden input fields.
- ▶ Once the edit button is clicked, the id of the item to be edited is sent to the script update.php. Either the GET or POST method can be used.
- ▶ 

```
<td>
    <form action='update.php' method='post'>
        <input name='id' value='".$row["id"]."' type='hidden'>
        <button type='submit' name='update' value='update'>Edit</button>
    </form>
</td>
```
- ▶ In the update.php script, a form with data matching the submitted information is displayed for editing. Once the edit is complete, the updated data is resubmitted to the script for processing.

# Delete

- ▶ To delete a record in the table, the user clicks the delete button in the HTML table.
- ▶ **<td>**
- ▶   **<form action='form-post.php' method='post'>**
- ▶   **<input name='id' value='".\$row["id"]."' type='hidden'>**
- ▶   **<button type='submit' name='delete' value='delete'>Delete</button>**
- ▶   **</form>**
- ▶ **</td>**
- ▶ The submit button has a hidden id field. The data is sent to a form and processed by the PHP script below. If the id is not empty, then the record with the submitted id is deleted.

# Record with the submitted id is deleted.

```
> if(!empty($_POST['delete']) && !empty($_POST['id'])){  
>     $query3 = "DELETE FROM user WHERE id='".$_.POST['id']."' ";  
>     if (mysqli_query($conn, $query3)) {  
>         echo "Record deleted successfully !";  
>     } else {  
>         // Display an error message if unable to delete the record  
>         echo "Error deleting record: " . $conn->error;  
>     }  
> }
```

# What is JSON?

- ▶ JavaScript Object Notation (JSON) is a standard text format for storing and transmitting data over a network.
- ▶ JSON comes from JavaScript and has a syntax similar to JavaScript but can be used separately from it.
- ▶ JSON is used for client/server communication in mobile and web applications written in many programming languages, including JavaScript, Python, Java, C++, C#, Go, **PHP**, and many others.

# JSON Functions

► PHP has some built-in functions to handle JSON.  
First, we will look at the following two functions:  
**json\_encode()**  
**json\_decode()**

Function	Libraries
json_encode	Returns the JSON representation of a value.
json_decode	Decodes a JSON string.

# Encoding JSON in PHP (`json_encode`)

- ▶ The `json_encode()` function is used to encode a value to JSON format.
- ▶ **Example**
- ▶ The following example shows how to convert an array into JSON with PHP –
  - ▶ <?php
  - ▶     \$arr = array('a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e' => 5);
  - ▶     echo json\_encode(\$arr);
  - ▶ ?>
- ▶ While executing, this will produce the following result –
  - ▶ {"a":1,"b":2,"c":3,"d":4,"e":5}

- ▶ <!DOCTYPE html>
  - ▶ <html>
  - ▶ <body>
  - ▶ <?php
  - ▶ \$age = array("Peter"=>35, "Ben"=>37, "Joe"=>43);
  - ▶ echo json\_encode(\$age);
  - ▶ ?>
  - ▶ </body>
  - ▶ </html>
- 
- ▶ {"Peter":35,"Ben":37,"Joe":43}

# PHP - json\_decode()

- ▶ The json\_decode() function is used to decode a JSON object into a PHP object or an associative array.
- ▶ The json\_decode() function returns an object by default.
- ▶ The json\_decode() function has a second parameter, and when set to true, JSON objects are decoded into associative arrays.
- ▶ **Example**
- ▶ This example decodes JSON data into a PHP object:
- ▶ 

```
<?php
$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';

var_dump(json_decode($jsonobj));
?>
```
- ▶ 

```
object(stdClass)#1 (3) { ["Peter"]=> int(35) ["Ben"]=> int(37) ["Joe"]=> int(43) }
```

# PHP - MySql

Web Technologies

# PHP Connect to MySQL

- PHP 5 and later can work with a MySQL database using:
- **MySQLi extension** (the "i" stands for improved)
- **PDO (PHP Data Objects)**
- PDO will work on 12 different database systems, whereas MySQLi will only work with MySQL databases.
- Both are object-oriented, but MySQLi also offers a procedural API.

# Example (MySQLi Object-Oriented)

- <?php  
\$servername = "localhost";  
\$username = "username";  
\$password = "password";
- // Create connection  
\$conn = new mysqli(\$servername, \$username, \$password);  
  
// Check connection  
if (\$conn->connect\_error) {  
 die("Connection failed: " . \$conn->connect\_error);  
}  
echo "Connected successfully";  
?>

# Example (MySQLi Procedural)

- <?php  
\$servername = "localhost";  
\$username = "username";  
\$password = "password";
- // Create connection  
\$conn = mysqli\_connect(\$servername, \$username, \$password);  
  
// Check connection  
if (!\$conn) {  
 die("Connection failed: " . mysqli\_connect\_error());  
}  
echo "Connected successfully";  
?>

# Close the Connection

- **MySQLi Object-Oriented:**

- \$conn->close();

- **MySQLi Procedural:**

- mysqli\_close(\$conn);

# PHP Create a MySQL Database

- <?php  
  \$servername = "localhost";  
  \$username = "username";  
  \$password = "password";  
  
  // Create connection  
  \$conn = mysqli\_connect(\$servername, \$username, \$password);  
  // Check connection
- if (!\$conn) {  
  die("Connection failed: " . mysqli\_connect\_error());  
}  
  // Create database  
  \$sql = "**CREATE DATABASE myDB**";
- if (mysqli\_query(\$conn, \$sql)) {
- echo "Database created successfully";  
} else {  
  echo "Error creating database: " . mysqli\_error(\$conn);  
}
- mysqli\_close(\$conn);  
?>

# Create a table in MySql and inserting

- ```
CREATE TABLE MyGuests (
    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    firstname VARCHAR(30) NOT NULL,
    lastname VARCHAR(30) NOT NULL,
    email VARCHAR(50),
    reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
    CURRENT_TIMESTAMP
)
```
- ```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}
$conn->close();
?>
```

# PHP MySQL Update Data

- UPDATE table\_name  
SET column1=value, column2=value2,...  
WHERE some\_column=some\_value

# Example (MySQLi Procedural)

- <?php  
    \$servername = "localhost";  
    \$username = "username";  
    \$password = "password";  
    \$dbname = "myDB";  
  
    // Create connection  
    \$conn = mysqli\_connect(\$servername, \$username, \$password, \$dbname);  
    // Check connection  
    if (!\$conn) {  
        die("Connection failed: " . mysqli\_connect\_error());  
    }  
  
    \$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";  
  
    if (mysqli\_query(\$conn, \$sql)) {  
        echo "Record updated successfully";  
    } else {  
        echo "Error updating record: " . mysqli\_error(\$conn);  
    }  
  
    mysqli\_close(\$conn);  
?>

# PHP MySQL Delete Data

- **Delete Data From a MySQL Table Using MySQLi**
- `DELETE FROM table_name  
WHERE some_column = some_value`

- <?php  
    \$servername = "localhost";  
    \$username = "username";  
    \$password = "password";  
    \$dbname = "myDB";  
  
    // Create connection  
    \$conn = mysqli\_connect(\$servername, \$username, \$password, \$dbname);  
    // Check connection  
    if (!\$conn) {  
        die("Connection failed: " . mysqli\_connect\_error());  
    }  
  
    // sql to delete a record  
    \$sql = "DELETE FROM MyGuests WHERE id=3";  
  
    if (mysqli\_query(\$conn, \$sql)) {  
        echo "Record deleted successfully";  
    } else {  
        echo "Error deleting record: " . mysqli\_error(\$conn);  
    }  
  
    mysqli\_close(\$conn);  
?>