**Lecture notes**

**on**

# Introduction to Web Designing and GIS

**India Meteorological Department**

# 1. Introduction to Web Designing and GIS

## What is Web Designing?

- Web designing is the process of creating and organizing the layout, structure, and content of a website.
- It involves both the aesthetic and functional aspects of websites to provide an engaging user experience.
- Tools used in web designing include HTML, CSS, JavaScript, and various web design software (e.g., Adobe Dreamweaver, Figma).

2. Importance of Web Designing

A well-designed website is crucial for businesses, organizations, and individuals who want to establish an online presence. Here are some key reasons why web designing is important:

- **First Impressions Matter**: A website is often the first interaction a user has with a brand. A well-designed site creates a positive impression and builds credibility.
- **Enhanced User Experience (UX)**: Good web design ensures intuitive navigation, clear content layout, and accessibility, leading to a seamless user experience.
- **Search Engine Optimization (SEO)**: Websites with proper design elements, optimized images, and structured content rank better on search engines, attracting more visitors.
- **Mobile Responsiveness**: With the increasing use of smartphones, websites must be mobile-friendly to provide a consistent experience across all devices.

## How the Web Works?

The web operates as a vast network of computers and servers communicating with each other through the internet. When a user enters a URL in their web browser, several processes occur to retrieve and display the requested webpage.

### a) Components of the Web

1. **Web Browsers**: Software applications like Google Chrome, Mozilla Firefox, and Safari that retrieve and display web pages.

2. **Web Servers**: Computers that store and manage website files and respond to user requests.
3. **Internet Protocol (IP) & Domain Name System (DNS)**: IP addresses identify servers, while DNS translates human-readable domain names (e.g., www.example.com) into IP addresses.
4. **HTTP/HTTPS**: HyperText Transfer Protocol (Secure) is the protocol used for communication between web browsers and servers.
5. **Client-Side & Server-Side**:
    - **Client-Side (Frontend)**: Code that runs in the browser, including HTML, CSS, and JavaScript.
    - **Server-Side (Backend)**: Code that runs on the server, including PHP, Python, Java, or Node.js.

**b) How a Web Page Loads**

1. **User Request**: A user enters a website URL in their browser.
2. **DNS Resolution**: The domain name is translated into an IP address.
3. **Server Communication**: The browser sends an HTTP request to the web server.
4. **Processing Request**: The server retrieves the requested HTML, CSS, and JavaScript files.
5. **Rendering the Page**: The browser processes and displays the webpage to the user.

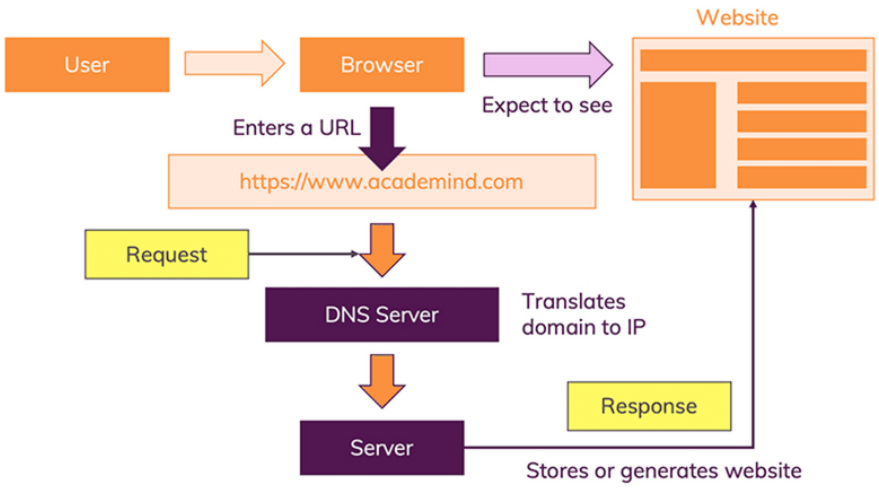



*Figure 1 Source academia.com*

## Static website vs Dynamic Website

**Websites can be classified into two main types: Static and Dynamic.**

**a) Static Websites**

Static websites consist of fixed content that does not change unless the source code is modified manually.

Characteristics:

- Built using HTML, CSS, and minimal JavaScript.
- Content remains the same for every visitor unless updated manually.
- No interaction with databases.
- Loads quickly as no server-side processing is required.

Advantages:

- Fast loading speed.
- Easy to develop and host.
- More secure as there is no database or backend processing.

Disadvantages:

- Not suitable for websites requiring frequent content updates.
- Limited interactivity and personalization.

Examples:

- Personal blogs (without comment sections).
- Small business portfolio websites.
- Landing pages.

**b) Dynamic Websites**

Dynamic websites generate content dynamically based on user interactions and database queries.

Characteristics:

- Uses server-side languages like PHP, Python, Node.js, or ASP.NET.
- Content is retrieved from databases in real time.
- Supports user interaction, login systems, and content management.
- Requires a web server for backend processing.

Advantages:

- Content can be updated easily via a CMS.
- More interactive and personalized for users.
- Suitable for e-commerce, social media, and large websites.

Disadvantages:

- Slower load times due to server-side processing.
- More complex to develop and maintain.
- Requires database security measures.

Examples:

- Social media platforms (Facebook, Twitter).
- E-commerce websites (Amazon, eBay).
- News websites with frequently updated content.

**Web development:**

- Frontend (what users see)
- Backend (server, database, logic)
- Full-stack(both)

**Frontend Technologies:**

Responsible for the user interface (UI)

1. HTML – Structure of the webpage
2. CSS – Styling and layout
3. JavaScript – Interactivity and dynamic content

**Frontend Frameworks & Libraries**

- React.js – By Facebook, for dynamic Uis
- Vue.js – Lightweight and beginner-friendly
- Angular – Robust framework by Google

- Bootstrap – Pre-designed UI components

**Backend Technologies**

Responsible for the server-side logic and database interaction

Languages:

- Node.js (JavaScript)
- Python (Django, Flask)
- PHP (Laravel, WordPress)
- Ruby (Rails)
- Java (Spring Boot)
- .NET (C#)

**Databases:**

Used to store and retrieve data

- Relational (SQL):
    - MySQL, PostgreSQL, SQL Server
- NoSQL:
    - MongoDB, Firebase, CouchDB

**Web Servers & Hosting:**

- **Web Servers:**
    - **Apache**
    - **NGINX**
- **Cloud Hosting:**
    - **AWS**
    - **Google Cloud**
    - **Microsoft Azure**
    - **Vercel, Netlify (for frontend)**

**Development Tools:**

- VS Code, Sublime Text, notepad++ – Code editors

- Postman – API testing
- Chrome DevTools – Debugging
- Figma – UI design and prototyping

**Web Development Stacks**

- LAMP – Linux, Apache, MySQL, PHP
- MEAN – MongoDB, Express, Angular, Node.js
- MERN – MongoDB, Express, React, Node.js
- JAMstack – JavaScript, APIs, Markup (static-first)

**Modern Trends in Web Development**

- Progressive Web Apps (PWA)
- Single Page Applications (SPA)
- Serverless architecture
- AI integration

**Progressive Web Apps (PWA)**

A Progressive Web App is a web application that behaves like a native mobile app.

Key Features:

- Works offline
- Can be installed on the user's home screen
- Push notifications
- Fast and responsive
- Built with HTML, CSS, JavaScript

Benefits:

- No need for app store approval
- Improved performance and user engagement
- One codebase for web and mobile-like experience

Example: Twitter Lite, Starbucks PWA

**Single Page Applications (SPA)**

A Single Page Application is a web app that loads a single HTML page and dynamically updates content without refreshing the entire page.

Key Technologies:

- JavaScript frameworks: React, Angular, Vue.js
- API communication (usually REST or GraphQL)

Benefits:

- Faster navigation
- Smooth user experience
- No full page reloads

Example: Gmail, Facebook

**Serverless Architecture:**

Serverless doesn’t mean "no servers" — it means developers don’t manage the server infrastructure. The cloud provider handles it.

How it works:

- You write functions (code) that run on-demand.
- Hosted by providers like AWS Lambda, Azure Functions, Google Cloud Functions.

Benefits:

- Scalable automatically
- Cost-effective (pay only for what you use)
- Simplifies backend development

Use Cases: APIs, form handling, real-time processing

**AI Integration in Web Development**

Using Artificial Intelligence to enhance web functionality and user experience.

Common Uses:

- Chatbots (e.g., support bots)
- Personalized content and recommendations
- Image recognition
- Natural language processing

Benefits:

- Improves user interaction
- Automates repetitive tasks
- Offers data-driven personalization

Example: Netflix suggestions, AI-powered search, Shopify chatbots

## Content Management System (CMS):

- A Content Management System (CMS) is software that allows users to create, manage, and publish digital content—usually for websites—without needing to write code.
- Provides a user-friendly interface (like a dashboard)Separates content creation from the website's code
- Often includes features like
    - Text editing (WYSIWYG editor)
    - Media uploads
    - Page templates
    - Plugins/extensions

**Types of CMS**

- Traditional CMS:
    - Frontend and backend are connected
    - Example: WordPress, Joomla, Drupal
- Headless CMS:
    - Only manages content (no built-in frontend)

- Sends content via API to any platform
- Example: Strapi, Contentful, Sanity

**Advantages of Using a CMS**

- No coding needed for content updates
- Speeds up website development
- Enables team collaboration
- Supports SEO, media, and user roles

**Popular CMS Platforms**

- WordPress (most widely used)
- Shopify (for e-commerce)
- Drupal
- Wix / Squarespace (easy website builders)

## Introduction to XML

- **XML (eXtensible Markup Language) is a markup language designed to store and transport data in a structured, readable format.**
- **Unlike HTML, XML focuses on data representation, not how it looks.**

**Key Features**

- **Uses custom tags defined by the user**
- **Designed to be self-descriptive**
- **Both human-readable and machine-readable**
- **Used for data exchange between systems (e.g., web services, APIs)**

**Main Uses of XML**

- **Web services (e.g., SOAP APIs)**
- **Data configuration and storage**
- **Office documents (e.g., DOCX, XLSX are XML-based)**
- **Communication between different platforms**

**Important Notes:**

- **XML does not display data – it structures it**
- **Tags must be properly nested and closed**
- **It's case-sensitive**

**XML vs HTML**

| Feature | XML | HTML |
|---|---|---|
| Purpose | Store & transport data | Display content |
| Tags | Custom/user-defined | Predefined (e.g., <p>) |
| Structure | Strict & well-formed | More lenient |

## Key Elements of Web Design

1. **Layout**:
    - Refers to the structure of the website, how elements are arranged visually. Common layouts include grid-based, single-page, and multi-column layouts.
2. **Color Scheme**:
    - The choice of colors should reflect the brand identity, invoke emotions, and ensure readability.
3. **Typography**:
    - The style, size, and spacing of fonts are crucial to the design and readability of text content.
4. **Navigation**:
    - Refers to the ease with which users can move through the website, usually via menus, links, or buttons.
5. **Graphics & Imagery**:
    - Using visuals like images, icons, and videos that complement the text to enhance the appeal of the site.
6. **White Space**:
    - Also called negative space, it helps separate elements and gives breathing room, improving the site's readability and aesthetic appeal.

## Essential Tools for Web Designing

Web designers use various tools to create, prototype, and develop websites:

- **Design Software**: Adobe XD, Figma, Sketch, Canva for UI/UX design.
- **Code Editors**: Visual Studio Code, Sublime Text, Atom for writing HTML, CSS, and JavaScript.
- **Content Management Systems (CMS)**: WordPress, Joomla, Drupal for managing website content.
- **Frameworks & Libraries**: Bootstrap, Tailwind CSS, jQuery for faster and more responsive design.

## Web Design vs. Web Development

Although web designing and web development go hand in hand, they are distinct processes:

| Web Designing | Web Development |
|---|---|
| Focuses on visuals and layout | Focuses on functionality and coding |
| Uses tools like Photoshop, Figma | Uses programming languages like HTML, CSS, JavaScript |
| Ensures usability and aesthetics | Ensures backend logic and interactivity |

## Principles of Effective Web Design

- **Simplicity**: Clean layouts and minimal distractions improve user experience.
- **Consistency**: Maintaining uniformity in fonts, colors, and layout enhances usability.
- **Accessibility**: Websites should be accessible to all, including individuals with disabilities.
- **Mobile-Friendliness**: Responsive design ensures seamless display across devices.
- **Fast Load Time**: Optimized images, caching, and efficient coding improve website speed.

## Basic Technologies Used in Web Designing

1. **HTML (HyperText Markup Language)**:
    - The foundation of web pages, HTML defines the structure of the content using elements like headings, paragraphs, links, images, etc.
2. **CSS (Cascading Style Sheets)**:
    - A style sheet language used to control the visual appearance of web pages (colors, fonts, layout, etc.).
3. **JavaScript**:
    - A scripting language that adds interactivity to web pages, such as sliders, form validations, or dynamic updates without refreshing the page.
4. **Responsive Web Design**:
    - An approach to web design that makes web pages render well on a variety of devices and window/screen sizes. This is achieved through media queries in CSS.

## Web Design Process

1. **Planning**:
    - Understand the purpose, target audience, and content of the website.
2. **Wireframing**:
    - Create a basic layout or skeleton of the site to outline the placement of elements without the final design details.
3. **Design**:
    - Apply branding, choose color schemes, typography, and design elements to create mockups of the site.
4. **Development**:
    - Write the HTML, CSS, and JavaScript code based on the design.
5. **Testing**:
    - Test the website across different browsers, devices, and screen sizes to ensure compatibility and functionality.
6. **Launch**:

    - Once tested, the site is deployed and made live on the internet.
7. **Maintenance**:
    - Regular updates and improvements are necessary to keep the site relevant and functional.

## Web Design Trends

- **Dark Mode**: More sites are offering a dark mode to reduce eye strain and save battery life.
- **Micro-Animations**: Small, subtle animations that improve user interaction without being distracting.
- **Minimalism**: Flat design and minimalist layouts that focus on content with fewer distractions.
- **3D Elements**: The use of 3D graphics to add depth and interaction to web design.
- **Voice User Interface (VUI)**: Designing sites optimized for voice search and interaction with voice assistants.

## Tools for Web Designing

- **Adobe XD / Figma**: For prototyping and creating design mockups.
- **WordPress**: A popular content management system (CMS) that allows easy creation of websites without much coding.
- **Bootstrap**: A front-end framework for building responsive websites quickly.
- **Sublime Text / Visual Studio Code**: Code editors for writing HTML, CSS, and JavaScript.

## Introduction to HTML

HTML stands for Hypertext Markup Language, and it is the most widely used language to write Web Pages.

- Hypertext refers to the way in which Web pages (HTML documents) are linked together. Thus, the link available on a webpage is called Hypertext.
- As its name suggests, HTML is a Markup Language which means you use HTML to simply "mark-up" a text document with tags that tell a Web browser how to structure it to display.

HTML was developed with the intent of defining the structure of documents like headings, paragraphs, lists, and so forth to facilitate the sharing of scientific information between researchers. Now, HTML is being widely used to format web pages with the help of different tags available in HTML language.

Basic HTML Document

Following is an example of an HTML document:

***<!DOCTYPE html>***

***<html>***

***<head>***

***<title>This is document title</title>***

***</head>***

***<body>***

***<h1>This is a heading</h1>***

*** Document content goes here..... ***

***</body>***

***</html>***

Save it in an HTML file Test.html using your any text editor and open it using a web browser like Internet Explorer or Google Chrome, or Firefox etc.

## HTML Tags

HTML is a markup language and makes use of various tags to format the content. These tags are enclosed within angle braces <Tag Name>. Except few tags, most of the tags have their corresponding closing tags. For example, <html> has its closing tag</html> and <body> tag has its closing tag </body> tag etc.

To learn HTML, you will need to study various tags and understand how they behave, while formatting a textual document. Learning HTML is simple as users have to learn the usage of different tags in order to format the text or images to make a beautiful webpage. World Wide Web Consortium (W3C) recommends to use lowercase tags starting from HTML 4.

The opening and closing tags use the same command except the closing tag contains and additional forward slash /

For example, the expression <B> Warning </B> would cause the word 'Warning' to appear in bold face on a Web page

**HTML Document structure**

***Document declaration tag***

***<html>***

***<head>***

***Document header related tags***

***</head>***

***<body>***

***Document body related tags***

***</body>***

***</html>***

**The <!DOCTYPE> Declaration**

The declaration tag is used by the web browser to understand the version of the HTML used in the document. Current version of HTML is 5 and it makes use of **<!DOCTYPE html>** declaration.

**Basic Tags**

**HTML Headings**

HTML headings are defined with the <h1> to <h6> tags.

<h1> defines the most important heading. <h6> defines the least important heading.

<h1>This is heading 1</h1>

**HTML Paragraphs**

HTML paragraphs are defined with the  tag

This is a paragraph.

**HTML Links**

HTML links are defined with the <a> tag

<a href="https://www.w3schools.com">This is a link</a>

The link's destination is specified in the href attribute.

Attributes are used to provide additional information about HTML elements.

**HTML Images**

HTML images are defined with the <img> tag.

The source file (src), alternative text (alt), width, and height are provided as attributes

**HTML Elements**

An HTML element is defined by a start tag, some content, and an end tag.

The HTML element is everything from the start tag to the end tag:

<tagname>Content goes here...</tagname>

Examples of some HTML elements:

<h1>My First Heading</h1>

My first paragraph.

**Nested HTML Elements**

HTML elements can be nested (this means that elements can contain other elements).

All HTML documents consist of nested HTML elements.

The following example contains four HTML elements (<html>, <body>, <h1> and ):

**<!DOCTYPE html>**

**<html>**

**<body>**

**<h1>My First Heading</h1>**

** My first paragraph. **

**</body>**

**</html>**

**HTML Attributes**

HTML attributes provide additional information about HTML elements.

- All HTML elements can have attributes
- Attributes provide additional information about elements
- Attributes are always specified in the start tag
- Attributes usually come in name/value pairs like: name="value"

**href Attribute**

The <a> tag defines a hyperlink. The href attribute specifies the URL of the page the link goes to

**<a href="https://www.w3schools.com">Visit W3Schools</a>**

**src Attribute**

**The <img> tag is used to embed an image in an HTML page. The src attribute specifies the path to the image to be displayed:**

**<img src="img_girl.jpg">**

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

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

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

**width and height Attributes**

The <img> tag should also contain the width and height attributes, which specify the width and height of the image (in pixels):

<img src="img_girl.jpg" width="500" height="600">

**alt Attribute**

The required alt attribute for the <img> tag specifies an alternate text for an image, if the image for some reason cannot be displayed. This can be due to a slow connection, or an error in the src attribute, or if the user uses a screen reader.

Example

<img src="img_girl.jpg" alt="Girl with a jacket">

**style Attribute**

The style attribute is used to add styles to an element, such as color, font, size, and more.

Example

This is a red paragraph.

**More can be explored from https://www.w3schools.com/html/default.asp**

**HTML Styles**

The HTML style attribute is used to add styles to an element, such as color, font, size, and more.

**The HTML Style Attribute**

Setting the style of an HTML element, can be done with the style attribute.

The HTML style attribute has the following syntax:

<*tagname* style="*property*:*value;*">

**Background Color**

The CSS background-color property defines the background color for an HTML element.

Set the background color for a page to powderblue:

<body style="background-color:powderblue;">

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

</body>

Set background color for two different elements:

```
<body>

<h1 style="background-color:powderblue;">This is a heading</h1>
<p style="background-color:tomato;">This is a paragraph.</p>

</body>
```

**Text Color**

The CSS color property defines the text color for an HTML element:

Example

```
<h1 style="color:blue;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>
```

**Fonts**

The CSS font-family property defines the font to be used for an HTML element:

Example

```
<h1 style="font-family:verdana;">This is a heading</h1>
<p style="font-family:courier;">This is a paragraph.</p>
```

**Text Size**

The CSS font-size property defines the text size for an HTML element:

Example

```
<h1 style="font-size:300%;">This is a heading</h1>
<p style="font-size:160%;">This is a paragraph.</p>
```

**Text Alignment**

The CSS text-align property defines the horizontal text alignment for an HTML element:

Example

```
<h1 style="text-align:center;">Centered Heading</h1>
<p style="text-align:center;">Centered paragraph.</p>
```

## HTML Formatting Elements

Formatting elements were designed to display special types of text:

- <b> - Bold text
-  - Important text
- <i> - Italic text
-  - Emphasized text
- <mark> - Marked text
- <small> - Smaller text
- <del> - Deleted text
- <ins> - Inserted text
- <sub> - Subscript text
- <sup> - Superscript text

**HTML comments are not displayed in the browser, but they can help document your HTML source code.**

## HTML Comment Tag

You can add comments to your HTML source by using the following syntax:

<!-- Write your comments here -->

Notice that there is an exclamation point (!) in the start tag, but not in the end tag.

Example

<!-- This is a comment -->

This is a paragraph.

## Introduction to CSS

**What is CSS?**

**Cascading Style Sheets (CSS) is used to format the layout of a webpage.**

## HTML Styles - CSS

CSS stands for Cascading Style Sheets.

CSS saves a lot of work. It can control the layout of multiple web pages all at once.

With CSS, you can control the color, font, the size of text, the spacing between elements, how elements are positioned and laid out, what background images or background colors are to be used, different displays for different devices and screen sizes, and much more!

The word cascading means that a style applied to a parent element will also apply to all children elements within the parent. So, if you set the color of the body text to "blue", all headings, paragraphs, and other text elements within the body will also get the same color (unless you specify something else)!

## Using CSS

**CSS can be added to HTML documents in 3 ways:**

**Inline** - by using the style attribute inside HTML elements

**Internal** - by using a <style> element in the <head> section

**External** - by using a <link> element to link to an external CSS file

The most common way to add CSS, is to keep the styles in external CSS files.

## Inline CSS

An inline CSS is used to apply a unique style to a single HTML element.

An inline CSS uses the style attribute of an HTML element.

The following example sets the text color of the <h1> element to blue, and the text color of the <p> element to red:

Example

```
<h1 style="color:blue;">A Blue Heading</h1>


<p style="color:red;">A red paragraph.</p>
```

**Internal CSS**

An internal CSS is used to define a style for a single HTML page.

An internal CSS is defined in the <head> section of an HTML page, within a <style> element.

The following example sets the text color of ALL the <h1> elements (on that page) to blue, and the text color of ALL the <p> elements to red. In addition, the page will be displayed with a "powderblue" background color:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color: powderblue;}
h1   {color: blue;}
p    {color: red;}
</style>
</head>
<body>
```

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

</body>
</html>
```

**External CSS**

An external style sheet is used to define the style for many HTML pages.

To use an external style sheet, add a link to it in the <head> section of each HTML page:

Example

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

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

</body>
</html>
```

The external style sheet can be written in any text editor. The file must not contain any HTML code, and must be saved with a .css extension.

Here is what the "styles.css" file looks like:

"styles.css":

```
body {
  background-color: powderblue;
}
h1 {
  color: blue;
}
p {
  color: red;
}
```

With an external style sheet, you can change the look of an entire web site, by changing one file!

**CSS Colors, Fonts and Sizes**

The CSS color property defines the text color to be used.

The CSS font-family property defines the font to be used.

The CSS font-size property defines the text size to be used.

Example

Use of CSS color, font-family and font-size properties:

```
<!DOCTYPE html>
<html>
<head>
<style>
```

```
h1 {
  color: blue;
  font-family: verdana;
  font-size: 300%;
}
p {
  color: red;
  font-family: courier;
  font-size: 160%;
}
</style>
</head>
<body>

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

</body>
</html>
```

**CSS Border**

The CSS border property defines a border around an HTML element.

Tip: You can define a border for nearly all HTML elements.

Example

Use of CSS border property:

```
p {
  border: 2px solid powderblue;
}
```

**CSS Padding**

The CSS padding property defines a padding (space) between the text and the border.

Example

Use of CSS border and padding properties:

```
p {
  border: 2px solid powderblue;
  padding: 30px;
}
```

**CSS Margin**

The CSS margin property defines a margin (space) outside the border.

Example

Use of CSS border and margin properties:

```
p {
  border: 2px solid powderblue;
  margin: 50px;
}
```

**Link to External CSS**

External style sheets can be referenced with a full URL or with a path relative to the current web page.

Example

This example uses a full URL to link to a style sheet:

```
<link rel="stylesheet" href="https://www.w3schools.com/html/styles.css">
```

Example

This example links to a style sheet located in the html folder on the current web site:

```
<link rel="stylesheet" href="/html/styles.css">
```

Example

This example links to a style sheet located in the same folder as the current page:

```
<link rel="stylesheet" href="styles.css">
```

**CSS selectors**

CSS selectors are used to “find” (or select) HTML elements based on their element name, id, class, attribute, and more.

**THE UNIVERSAL SELECTORS**: Rather than selecting elements of a specific type, the universal selector quite simply matches the name of any element type

```
* {
    color: #000000;
  }
```

This rule renders the content of every element in our document in black.

**THE ELEMENT SELECTOR:** The element selector selects elements based on the element name. You can select all p elements on a page like this (in this case, all p elements will be center-aligned, with a red text color)

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

**THE DESCENDANT SELECTOR:** Suppose you want to apply a style rule to a particular element only when it lies inside a particular element. As given in the following example, the style rule will apply to the em element only when it lies inside the ul tag.

```
ul em {
    color: #000000;
}
```

**THE ID SELECTOR**

The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element should be unique within a page, so the id selector is used to select one unique element!

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

```
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the style.</p>

</body>
</html>
```

## THE CLASS SELECTORS

The class selector selects elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the name of the class.

In the example below, all HTML elements with class=”center” will be red and center-aligned:

You can apply more than one class selector to a given element.

```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<h1 class="center">Red and center-aligned heading</h1>
<p class="center">Red and center-aligned paragraph.</p>

</body>
</html>
```

**GROUPING SELECTORS**

**It will be better to group the selectors, to minimize the code. To group selectors, separate each selector with a comma.**

```
h1, h2, p {
   text-align: center;
   color: blue;
}
```

# Introduction to Java Script

**JavaScript** is a **high-level**, **interpreted programming language** primarily used to create **interactive and dynamic behavior** on websites.

**Key Features:**

- **Client-side language**: Runs directly in the user's web browser.
- **Interpreted**: No need to compile before running.
- **Dynamically typed**: Variable types are determined at runtime.
- **Object-oriented**: Uses objects to model data and functionality.
- **Event-driven**: Reacts to user actions like clicks, hovers, and keystrokes.

**What Can JavaScript Do?**

| Task | Example |
|---|---|
| Manipulate content | Change text, images, or styles on a webpage |
| Respond to events | React to clicks, form submissions, keypresses |
| Validate forms | Check input values before submission |
| Communicate with servers | Load data without reloading (AJAX, fetch) |
| Create animations | Fade-ins, slides, interactive effects |
| Build web applications | Games, tools, dashboards, and full web apps |

**Where JavaScript Runs**

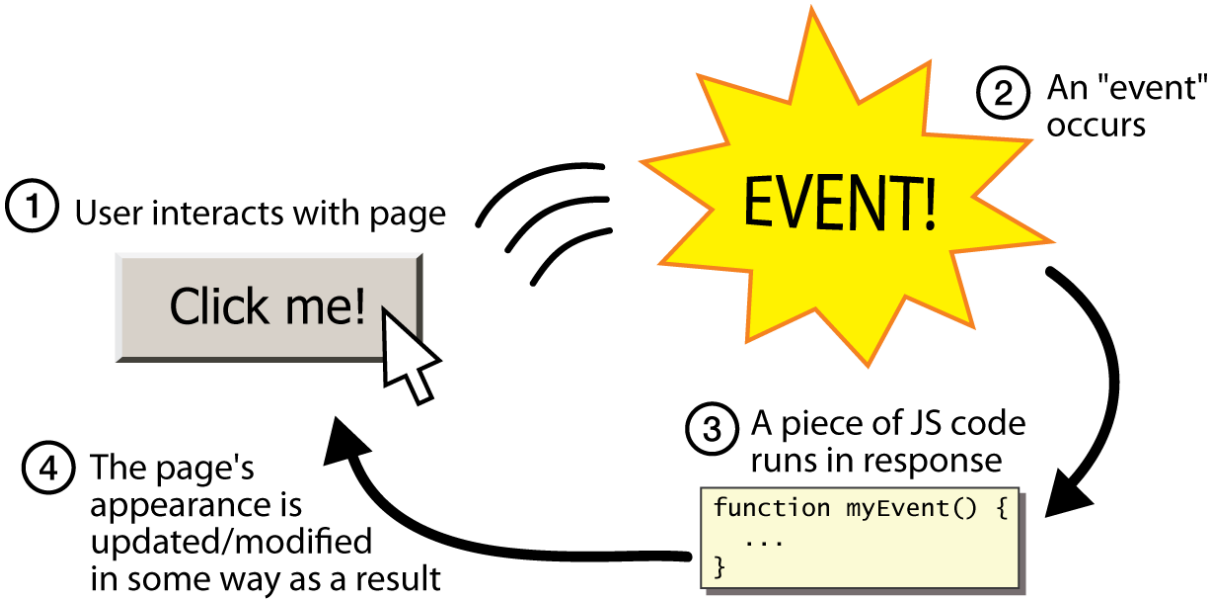
- **Browsers** (e.g., Chrome, Firefox, Safari)
- **Server-side** (via Node.js)
- **Mobile apps** (via frameworks like React Native)

Linking to a JavaScript file: script

```HTML
<script src="filename" type="text/javascript"></script>
```

- script tag should be placed in HTML page's head
- script code is stored in a separate .js file
- JS code can be placed directly in the HTML file's body or head (like CSS) but this is bad style (should separate content, presentation, and behavior)

**JavaScript is an event-driven programming language**




Event-driven programming means that the flow of the program is determined by events, such as:

- User interactions (clicks, key presses, mouse movements)
- Browser events (page load, resize)
- Network responses (data loaded from a server)
- Timers (delayed or repeated actions)

Instead of the code running from top to bottom in a straight line, it waits for events, and responds to them using event handlers (functions that are called when an event occurs).

Example:

```
<button onclick="sayHello()">Click Me</button>


<script>
  function sayHello() {
    alert("Hello, World!");
  }
</script>
```

**Adding Event Listeners with JavaScript**

```
<button id="myBtn">Click Me</button>


<script>
  document.getElementById("myBtn").addEventListener("click", function() {
    alert("Button was clicked!");
  });
</script>
```

**Common Events in JavaScript**

| **Event** | **Description** |
|---|---|
| click | When an element is clicked |
| mouseover | When mouse hovers over an element |
| keydown | When a key is pressed |
| load | When a page or resource finishes loading |
| submit | When a form is submitted |
| change | When a form element value changes |

**Document Object Model (DOM)**

The Document Object Model (DOM) is a programming interface for HTML and XML documents.
It represents the structure of a webpage as a tree of objects.

```
<html>
  <body>
    <h1>Hello</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```

The DOM turns this into a tree like:

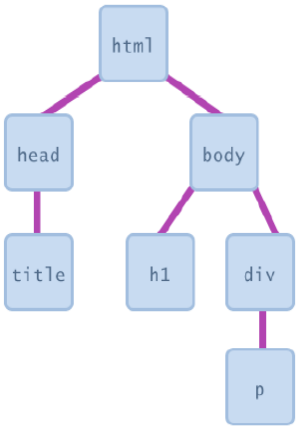
```
document
└── html
    └── body
        ├── h1
        └── p
```

- most JS code manipulates elements on an HTML page
- one can examine elements' state

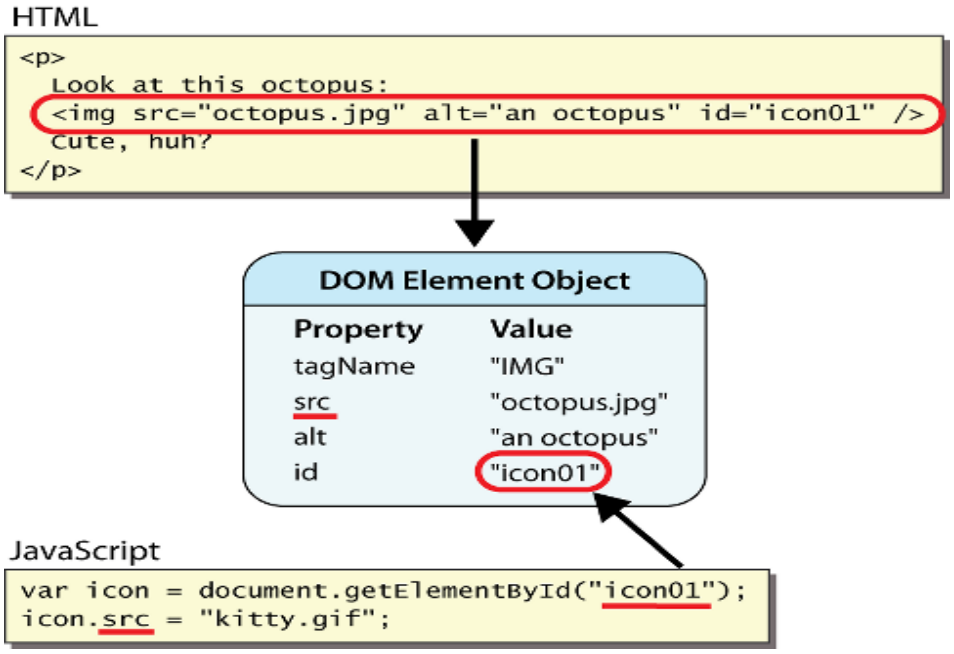
  e.g. see whether a box is checked
- one can change state

  e.g. insert some new text into a div
- one can change styles

  e.g. make a paragraph red



## DOM element objects




## Accessing elements

- document.getElementById returns the DOM object for an element with a given id
- can change the text inside most elements by setting the innerHTML property
- can change the text in form controls by setting the value property

```
var name = document.getElementById("id");
```
JS

```
<button onclick="changeText();">Click me!</button>
<span id="output">replace me</span>
<input id="textbox" type="text" />
```
HTML

```
function changeText() {
	var span = document.getElementById("output");
	var textBox = document.getElementById("textbox");

	 textbox.style.color = "red";

}
```
JS

# Introduction to PHP

**PHP** stands for **Hypertext Preprocessor** (it's a recursive acronym). It is a **server-side scripting language** designed for **web development** but also used as a general-purpose language.

PHP code is executed **on the server**, and the result is **sent to the browser** as plain HTML.

**Key Features of PHP**

- Open-source and free to use
- Server-side execution
- Works well with databases (especially MySQL)
- Easy to embed into HTML
- Platform-independent (Windows, Linux, Mac)

**Basic Syntax:**

**A PHP script starts with:**

```
<?php
  // PHP code here
?>
```

**Example:**

```
<?php
  echo "Hello, world!";
?>
```

Output in browser:

Hello, world!

**Embedding PHP in HTML**

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

```
<h1>Welcome</h1>

<p>The time is: <?php echo date("h:i A"); ?></p>

</body>
</html>
```

**PHP Variables**

- Start with $ symbol
- No need to declare type

```
<?php
  $name = "Alice";
  $age = 25;
  echo "Name: $name, Age: $age";
?>
```

**Common Data Types**

- String
- Integer
- Float
- Boolean
- Array
- Object
- NULL

**Control Structures**

**If-Else:**

```
<?php
  if ($age >= 18) {
    echo "Adult";
  } else {
    echo "Minor";
  }
?>
```

**Loops:**

```
<?php
  for ($i = 0; $i < 5; $i++) {
    echo $i . "<br>";
  }
?>
```

**Functions**

```
<?php
  function greet($name) {
    return "Hello, $name!";
  }
  echo greet("Alice");
?>
```

**Working with Forms**

**HTML:**

```
<form method="post" action="welcome.php">
  Name: <input type="text" name="username">
```

```
  <input type="submit">
</form>
```

**PHP (welcome.php):**

```
<?php
  $user = $_POST["username"];
  echo "Welcome, $user!";
?>
```

**Connecting to a Database (MySQL)**

```
<?php
$conn = mysqli_connect("localhost", "root", "", "testdb");
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully!";
?>
```

**What You Need to Run PHP**

- A **web server** (like Apache)
- **PHP installed**
- A local server stack like
    - **XAMPP** (Windows)
    - **MAMP** (Mac)
    - **WAMP** (Windows)

# Introduction to Database management System (DBMS)

## What is a Database?

A database is an organized collection of data, typically stored electronically. It allows for easy storage, retrieval, and management of data.

## What is DBMS?

A Database Management System (DBMS) is software that interacts with the database, users, and applications to manage and organize data.

It provides tools to create, read, update, and delete (CRUD) data in a structured and secure way.

## Examples of Popular DBMS Software

| DBMS | Type |
|---|---|
| MySQL | Relational |
| Oracle DB | Relational |
| Microsoft SQL Server | Relational |
| MongoDB | NoSQL (Document-based) |
| PostgreSQL | Relational |
| SQLite | Relational (lightweight) |

## Key Features of DBMS

- Data Storage & Access
- Data Security
- Backup & Recovery
- Multi-user Access

- Data Integrity
- Data Independence

**Components of DBMS**

| Component | Role |
|---|---|
| DBMS Engine | Manages data storage and access |
| Data | Actual data stored |
| Data Dictionary | Metadata (data about data) |
| Query Processor | Interprets and runs user commands |

**Types of Databases**

| **Type** | **Description** |
|---|---|
| **Relational DB** | **Uses tables with rows and columns** |
| **NoSQL DB** | **Non-tabular (key-value, document, etc.)** |
| **Distributed DB** | **Data stored on multiple servers** |
| **Cloud DB** | **Database services over the internet** |

**SQL – Structured Query Language**

SQL is the standard language for interacting with relational databases.

**Example SQL Commands:**

```
-- Create a table
CREATE TABLE users (
  id INT PRIMARY KEY,
  name VARCHAR(50),
  age INT
);
```

```
-- Insert data
INSERT INTO users (id, name, age) VALUES (1, 'Alice', 24);
-- Retrieve data
SELECT * FROM users;
-- Update data
UPDATE users SET age = 25 WHERE id = 1;
-- Delete data
DELETE FROM users WHERE id = 1;
```

**Advantages of DBMS**

- Data security
- Data consistency
- Centralized control
- Efficient data handling
- Backup and recovery

**Disadvantages:**

- Cost of setup and maintenance
- Complexity (learning curve)
- Hardware and software requirements

# Introduction to GIS

GIS is a computer-based system designed for capturing, storing, manipulating, analyzing, managing, and presenting spatial or geographic data. It combines **location data (where things are)** with **attribute data (what things are)** to help visualize and analyze real-world phenomena.

**Example:** Mapping flood-prone areas using rainfall data, topography, and land use.

### Components of GIS

A fully functioning GIS integrates five key components**:**

| Component | Description |
|---|---|
| Hardware | Computers, GPS devices, servers, scanners used to run GIS software. |
| Software | Tools like ArcGIS, QGIS, GRASS GIS that perform spatial analysis and map creation. |
| Data | Core of GIS—spatial (geographic) and attribute (descriptive) data. |
| People | Analysts, developers, planners, researchers using GIS tools and interpreting results. |
| Methods | Procedures and models for data analysis and decision-making. |

### GIS data model

A **GIS Data Model** defines how **spatial** and **attribute information** is organized, stored, and processed in a GIS system. Understanding GIS data models is fundamental to effectively handling geospatial data for mapping, analysis, and decision-making.

A **GIS data model** is a **framework** that defines the logical structure for representing geographic features and their attributes in a GIS. It determines how **real-world phenomena** (like roads, rivers, elevation) are abstracted into digital format for storage, analysis, and visualization.

Types of GIS Data Models:

GIS data is categorized into **two primary types** based on how spatial features are represented:

### 1. Vector Data Model

The **vector model** represents geographic features using **discrete geometric shapes**:

| Feature | Representation | Example |
|---|---|---|
| **Point** | Single x, y coordinate | Wells, trees, poles |
| **Line** | Series of connected points | Roads, rivers, pipelines |
| **Polygon** | Closed loops of lines | Land parcels, lakes, districts |

**Structure:**

- Each feature is **linked to attributes** stored in tables.
- Data stored in formats like: **Shapefile, GeoJSON, GPKG, KML**.

**Advantages:**

- High precision for discrete features
- Suitable for network analysis (roads, utilities)

### 2. Raster Data Model

- The **raster model** represents the world as a **grid of cells (pixels)**, each with a value representing a specific attribute (e.g., temperature, elevation).

**Structure:**

- Regular rows and columns
- Resolution defined by **cell size**
- Stored in formats like: **GeoTIFF, NetCDF, IMG, HDF**

**Advantages:**

- Ideal for continuous data (elevation, rainfall)
- Efficient for spatial modeling and overlay analysis

**Layers**: A **layer** is a **collection of geographic data representing a particular theme or feature**. Each layer contains either vector or raster data, and shows **one type of information**, such as roads, rivers, land use, elevation, or rainfall.

Layers are comprised of two data types

- *Spatial data* which describes location (where)
- *Attribute data* specifing what, how much,when

Layers may be represented in two ways:

- in *vector* format as points and lines
- in *raster(or image)* format as pixels

All geographic data has 4 properties:

projection, scale, accuracy and resolution

**Common Layer Formats:**

| **Format** | **Type** | **Description** |
|---|---|---|
| .shp | Vector | Shapefile—widely used format |
| .geojson | Vector | Web-friendly text-based format |
| .tif / .tiff | Raster | GeoTIFF for elevation or satellite data |
| .kml | Vector | Google Earth layer format |
| .nc | Raster | NetCDF for gridded scientific data |
| .gdb / .gpkg | Both | Geodatabase or GeoPackage |

**Projection**

- A map projection is a mathematical transformation used to represent the curved surface of the Earth (3D) on a flat map (2D).

- Since the Earth is a sphere (or more precisely, an oblate spheroid), projecting it onto a flat surface always introduces distortion—in shape, area, distance, or direction.

**Types of Projections**

Projections are chosen based on the purpose of the map:

| Projection Type | Preserves | Example Use |
|---|---|---|
| **Conformal** | Shape | Navigation (e.g., Mercator) |
| **Equal Area** | Area | Thematic maps (e.g., Albers Equal Area) |
| **Equidistant** | Distance | Radio range maps |
| **Azimuthal** | Direction | Air route maps |

**Common Projection Systems**

- WGS 84 – Global GPS system (Lat/Long)
- UTM (Universal Transverse Mercator) – Divides the world into 60 zones, commonly used in GIS
- State Plane – For detailed mapping in U.S. states

**Scale:**

Scale defines the relationship between distance on a map and the corresponding distance on the ground.

Example: 1:50,000 means 1 unit on the map equals 50,000 units on Earth.

**Types of Scale**

- Large Scale (e.g., 1:10,000): More detail, smaller area (e.g., city map)
- Small Scale (e.g., 1:1,000,000): Less detail, larger area (e.g., world map)

**Importance**

- Determines the level of detail.
- Affects the size of features and symbols.
- Essential in choosing the right map for the right purpose.

**Accuracy:**

Accuracy refers to how close the mapped or measured values are to the true values on Earth.

**Types of Accuracy**

**Positional Accuracy**: How close coordinates or locations are to their real-world positions.

**Attribute Accuracy**: How correct the descriptive data is (e.g., land use, population).

**Temporal Accuracy**: Whether the data reflects the current state (important for dynamic datasets).

**Logical Consistency**: Whether the data structure and relationships are valid.

**Importance**

In critical applications like flood mapping or urban planning, low accuracy can lead to poor decisions or risks.

**Resolution**

**Resolution** refers to the **smallest unit of measurement or detail** that a dataset can capture.

**Types of Resolution**

| Type | Meaning | Example |
|---|---|---|
| **Spatial Resolution** | Size of the smallest feature that can be detected | 30 m pixel in a Landsat image |
| **Temporal Resolution** | Frequency of data capture or update | Daily vs. monthly rainfall data |
| **Spectral Resolution** | Number and width of spectral bands | RGB vs. multispectral images |
| **Radiometric Resolution** | Ability to distinguish small differences in intensity | 8-bit (256 levels), 16-bit (65536 levels) |

**Importance**

- High resolution = more detail but larger data size.
- Choose based on application (e.g., high spatial resolution for urban mapping, high temporal resolution for weather monitoring).

**Functions of GIS**

GIS enables:

- **Data Visualization**: Displaying spatial information on maps.
- **Spatial Analysis**: Measuring distances, areas, patterns, overlays, etc.
- **Data Integration**: Combining data from various sources (satellite, field, sensor).
- **Modeling and Simulation**: Predicting changes, like urban growth or flood risk.
- **Decision Support**: For planning, resource allocation, and emergency response.

Applications of GIS

GIS is used in:

- **Urban planning**: Zoning, infrastructure planning.
- **Disaster management**: Risk mapping, emergency response.
- **Environmental monitoring**: Deforestation, pollution tracking.
- **Agriculture**: Crop monitoring, soil mapping.
- **Transport and logistics**: Route optimization, traffic analysis.
- **Utilities**: Managing water, electricity, telecom networks.

**WebGIS (Web-based Geographic Information System)**

WebGIS is the extension of GIS technology to the web platform, allowing users to access, interact with, and analyze spatial data via a web browser. It decentralizes GIS capabilities and makes them available to a wider audience without the need for specialized software.

Example: Google Maps, Bhuvan (ISRO), or a weather dashboard showing real-time rainfall and temperature.

**Key Features**

| Feature | Description |
|---|---|
| Accessibility | Access GIS tools/data anywhere using the internet. |
| Interactivity | Zoom, pan, query, measure directly on web maps. |
| Sharing and Collaboration | Multiple users can view, update, or share geospatial data in real-time. |
| Integration | Connects with other web services, APIs, and databases (e.g., REST APIs, WMS/WFS). |

**Components of WebGIS**

- Client: Web browser, mobile app, or desktop application.
- Web Server: Hosts the GIS web application (e.g., Apache, NGINX).
- GIS Server: Processes spatial data and handles map rendering (e.g., ArcGIS Server, GeoServer).
- Database: Stores spatial and attribute data (e.g., PostgreSQL/PostGIS, Oracle Spatial).
- Web Services: Standards like WMS (Web Map Service), WFS (Web Feature Service), REST APIs.

**Tools & Technologies**

- Frontend: JavaScript libraries like Leaflet, OpenLayers, Mapbox GL, CesiumJS.
- Backend: Python (Django), Node.js, PHP, Java with GIS libraries.
- Spatial Databases: PostGIS, MySQL Spatial, MongoDB with GeoJSON.
- Open-source Platforms: GeoServer, QGIS Server, MapServer.
- Cloud GIS: Google Earth Engine, ArcGIS Online, Amazon Location Services.

**Advantages of WebGIS**

- No installation required for end-users.
- Real-time data access and updates.
- Scalability—support for thousands of users.

- Cost-effective deployment.
- Enables citizen engagement in smart governance.

**Applications of WebGIS**

- **Smart Cities**: Real-time traffic and utility maps.
- **E-Governance**: Property tax maps, public feedback systems.
- **Disaster Management**: Live maps for flood alerts, cyclone tracking.
- **Environmental Monitoring**: Web dashboards for pollution, forest cover.
- **Public Health**: COVID-19 tracking dashboards.
- **Tourism & Culture**: Interactive heritage site maps.