Graphics And Web Designing

HTML , CSS, JAVASCRIPT AND PHOTOSHOP

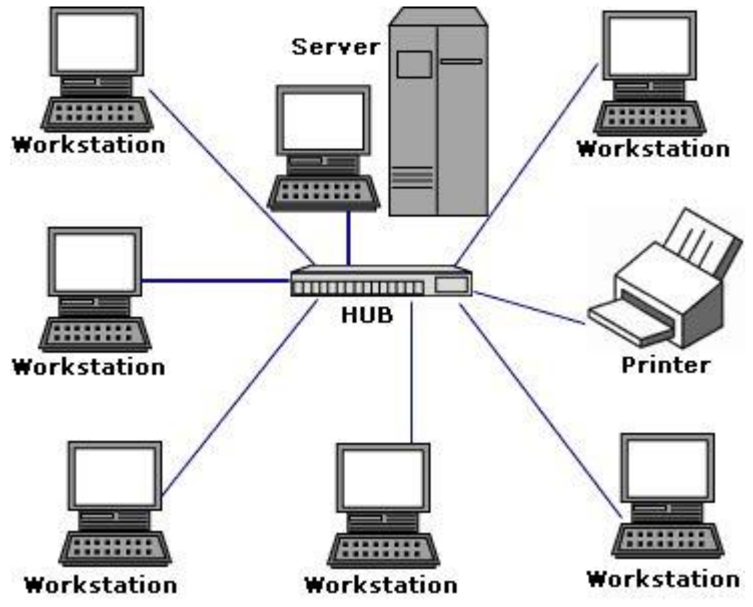
What Is Internet

- The Internet is a global network of interconnected computers and computer networks. It allows communication and the exchange of information between users worldwide. The Internet encompasses a vast array of services and resources, including websites, email, file sharing, social media, online gaming, and more.

- The Internet relies on a set of protocols, such as the Transmission Control Protocol (TCP) and the Internet Protocol (IP), to enable devices to communicate with each other.

How Internet Works

The Internet is a global network that connects billions of devices worldwide. It works by breaking data into smaller packets, routing them through networks of routers and switches, and reassembling them at their destination. This process relies on protocols like TCP/IP for data transmission and involves layers of security measures to protect privacy and ensure secure communication.



What is web

The term "web" typically refers to the World Wide Web (WWW), which is a system of interconnected hypertext documents and other resources that are accessed over the Internet. These documents are linked together through hyperlinks and can contain text, images, videos, and other multimedia elements.

The web operates on a set of protocols, including HTTP (Hypertext Transfer Protocol) and HTTPS (HTTP Secure), which allow for the transfer of data between servers and clients securely. Web pages are typically created using technologies such as HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), and JavaScript.

WWW

- The World Wide Web (WWW) is a major component of the Internet, providing a system of interlinked hypertext documents accessed via the Internet. Web browsers, such as Chrome, Firefox, Safari, and others, allow users to navigate and interact with the content on the World Wide Web.

- Internet Application that used for sharing and accessing information over the internet using Web Browser

- Component Of WWW(Web)
- HTML
- URL(Uniform Resource Locators)
- Web Browsers
- Web Servers and Web Hosting etc

How it works ?

The WWW operates on the Hypertext Transfer Protocol (HTTP), allowing users to navigate between web pages by clicking on hyperlinks.

Web Browsers

Web browsers like Chrome, Firefox & Safari enable users to access & view web pages on their devices.

Importance

The WWW transformed the way we communicate, conduct business, learn, and entertain ourselves, impacting every aspect of modern life.

What Is Graphic Designing

- Graphic design is the art and practice of planning and creating visual content to communicate ideas and messages to a viewer.
- The main tools used are image and typography.
- Graphic designers use visual hierarchy and page layout techniques to meet users' specific needs and focus on the logic of displaying elements in interactive designs to optimize the user experience.
- Effective designs communicate information that inspires and informs consumers, making it critical for any business' success.

What Is Website/Web Designing

- Website is the collection of web pages, different multimedia content such as text, images, and videos which can be accessed by the URL which you can see in the address bar of the browser. For example: JavaTPoint,GeeksForGeeks,TutorialsPoint

- Web designing is the process of planning, conceptualizing, and implementing the plan for designing a website in a way that is functional and offers a good user experience.
- It involves creating a website's visuals, including color schemes, font choices, page layouts, and more.
- Web design is concerned with what the user actually sees on their computer screen or mobile device, and less so about the mechanisms beneath the surface that make it all work. Through the use of color, images, typography, and layout, web designers bring a digital experience to life.
- As a web designer, you'll focus on planning the user experience of the website, wireframe layouts, organize content and images in a way that tells a story, and design the final UI

Types Of Web Developer

- Front-end Developer:
- Back-end Developer:
- Full-stack Developer:

Front End

- Specializes in the user interface and user experience (UI/UX) of a website.
- Works with HTML, CSS, and JavaScript to create visually appealing and interactive web pages.
- Familiar with front-end frameworks/libraries like React, Angular, or Vue.js.

Backend

- Focuses on the server-side logic and database management of a website.
- Works with server-side languages such as PHP, Python, Ruby, Java, or Node.js.
- Manages databases and ensures data is stored and retrieved efficiently.

Full Stack

- Proficient in both front-end and back-end development.
- Capable of handling the entire web development process from the user interface to server-side logic and database management.

Web Page And Website

- Webpage:
- A webpage is a single document or page of content that is part of a larger website.
- It is typically written in HTML (Hypertext Markup Language) and may include other technologies such as CSS (Cascading Style Sheets) for styling and JavaScript for interactivity.
- Webpages are what you see and interact with when you visit a website. Each webpage is a separate file containing information like text, images, multimedia, and links.
- Website:
- A website is a collection of related webpages that are typically identified by a common domain name and are accessible over the internet.
- It consists of a homepage and other interconnected pages, forming a navigable structure.
- Websites can include various types of content, such as text, images, videos, and interactive elements.
- Websites are created to serve a specific purpose, whether it's providing information, selling products, offering services, or any other function.
- Eg https://www.tutorialspoint.com/java/index.htm

Component Of Website

- **HTML** (Hypertext Markup Language): is the backbone of webpages. It defines the structure and content of a webpage through a series of elements and tags. Elements can include headings, paragraphs, images, links, forms, and more.

- **CSS** (Cascading Style Sheets):CSS is used to style and format the HTML content. It controls the layout, colors, fonts, and overall visual presentation of the webpage. CSS can be applied inline, in a separate file, or internally within the HTML document.

- **JavaScript**:(scripting language) adds interactivity and dynamic behavior to webpages. It allows for client-side processing, user input validation, and the manipulation of webpage elements in real-time. Various JavaScript libraries and frameworks, such as jQuery, React, Angular, and Vue.js, are commonly used to enhance functionality.

- **Images and Multimedia**:Images, videos, and audio files are essential components for enhancing visual and multimedia content on webpages. The <img>, <video>, and <audio> tags are commonly used to embed multimedia elements.

Component Of Website

- **Links and Navigation:**Hyperlinks (<a> tag) enable navigation between different pages or sections within a webpage. Navigation menus, buttons, and other UI elements contribute to a user-friendly browsing experience.

- **Forms** (created using the <form> tag) enable users to input data, submit information, and interact with the webpage. Form elements include text inputs, radio buttons, checkboxes, dropdowns, and buttons.

- **Server-side scripting languages**, such as PHP, Python, Ruby, or Node.js, are used to handle server-side logic. They process user requests, interact with databases, and generate dynamic content before sending it to the user's browser.

- **Databases** store and retrieve data used by web applications. Common databases include MySQL, PostgreSQL, MongoDB, and Firebase. Server-side scripts often interact with databases to dynamically generate content.

Other Component

- Responsive Design:
- Cookies and Local Storage:
- Security features such as HTTPS (SSL/TLS)

Type of Website

1.Static Website:

- **Content**: The content of a static website remains fixed and does not change unless the webmaster manually edits the HTML files.
- **Technology**: Static websites are built using HTML and CSS primarily. They may include some client-side scripting using JavaScript.
- **Server Interaction**: No server-side processing is involved. Each page is a separate HTML file stored on the server, and the server sends this file directly to the user's browser upon request.
- **Advantages**:
- Faster loading times because there is no need for server processing.
- Simplicity and ease of hosting.
- **Disadvantages:**
- Limited interactivity; interactions are predefined and cannot be customized based on user input.
- Updating content requires manual editing of HTML files.
- Examples: Brochure websites, personal blogs, portfolio websites.

Type of Website

2.Dynamic Website:

- **Conten**t: The content of a dynamic website is generated on-the-fly, often using a combination of server-side scripting languages, a database, and client-side scripts.
- **Technology**: Dynamic websites use server-side languages such as PHP, Python, Ruby, or Node.js, and they often involve databases like MySQL, PostgreSQL, or MongoDB. Front-end technologies like HTML, CSS, and JavaScript are still used.
- **Server Interaction**: When a user requests a page, the server processes the request, retrieves data from the database, generates the content, and sends it to the user's browser.
- **Advantages:**
- Content can be customized based on user interactions or preferences.
- Easier to manage and update, especially for large websites with dynamic content.
- Allows for user authentication and personalized user experiences.
- **Disadvantages**:
- Generally requires more server resources compared to static websites.
- May have slightly slower loading times due to server-side processing.
- Examples: Social media sites, content management systems (CMS), e-commerce platforms, forums.

HTML (Hypertext Markup Language)

- **HTML**

HTML, or Hypertext Markup Language, is the standard markup language used to create the structure and content of webpages. It plays a fundamental role in web development by providing a set of elements and tags that define the various components of a webpage. Here are some key aspects of

(HyperText: type of text that contains links (or hyperlinks) to other texts, allow users navigate between different pieces of information by clicking on the links)some key aspects of HTML)

- **Structure**:HTML documents are structured as a tree of elements, each represented by tags. The basic structure of an HTML document includes the <html>, <head>, and <body> elements.
- **Elements and Tags**:HTML elements are represented by tags, which come in pairs (opening and closing tags). The content of an element is placed between these tags. For example:
- <p>This is a paragraph.</p>
- **Attributes**:HTML elements can have attributes that provide additional information about the element. Attributes are added to the opening tag. For example: <a href="https://www.example.com">Visit Example</a>

HTML 5(HyperText Markup Language version 5)

- HTML5:is the latest version of HTML, introducing new elements, attributes, and APIs. It focuses on enhancing multimedia support, improving semantics, and providing better support for mobile devices.

Html Structure

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Document</title>
</head>
<body>
  <!--- Body and Some Contain Of Website--->

</body>
</html>
```

API stands for "Application Programming Interface." It is a set of rules and protocols that allows one software application to interact with anothe

Formatting

- Formatting in the context of web development refers to the visual presentation and styling of content on a webpage. It involves using HTML and CSS to structure and design elements for improved readability

- HTML is used for structuring the content, defining headings, paragraphs, lists, images, and other elements. CSS (Cascading Style Sheets) is then applied to control the appearance of these HTML elements, specifying attributes like colors, fonts, spacing, and layout.

- In simpler terms, formatting helps make your webpage visually appealing and organized, enhancing the user experience by presenting information in a clear and aesthetically pleasing manner.

Importance Of Formatting

- **Ease of Maintenance**: Well-formatted code is easier to maintain and update. Developers can quickly understand the structure of the code and make changes without introducing errors.
- **Collaboration**: If multiple developers are working on a project, consistent and readable code makes collaboration smoother. Team members can understand each other's code more easily.
- **Debugging**: When issues or bugs arise, it's easier to identify and fix problems in well-formatted code. Clear indentation and structure help in locating errors.
- **Scalability**: As a project grows, maintaining a consistent and readable coding style becomes increasingly important. It helps manage complexity and ensures that the codebase remains maintainable.

What is Code Editor (IDE)

A code editor or IDE (Integrated Development Environment) is a software application that developers use to write and edit code. It provides various features and tools to assist in the software development process. Here's a brief explanation of each:

Code Editor:

- A code editor is a lightweight software tool specifically designed for writing and editing code. It typically offers features like syntax highlighting, code folding, auto-indentation, and basic code completion. Examples of popular code editors include Visual Studio Code, Sublime Text, Atom, and Notepad++.

IDE (Integrated Development Environment):

- An IDE is a more comprehensive software package that combines a code editor with additional features and tools for software development. In addition to editing code, an IDE typically includes features such as code debugging, build automation, version control integration, project management, and more. IDEs are often tailored to specific programming languages or platforms. Examples of popular IDEs include Visual Studio (for .NET development), IntelliJ IDEA (for Java development), Eclipse, PyCharm (for Python development), and Xcode (for iOS/macOS development).

Advantages Of Code Editor

1. **Lightweight and Fast:** Code editors are generally lightweight and fast, providing a smooth editing experience without unnecessary overhead.
2. **Customizability:** Most code editors offer extensive customization options, allowing developers to tailor the environment to their preferences with themes, plugins, and extensions.
3. **Focused on Code Editing:** Code editors are designed primarily for code editing, providing features such as syntax highlighting, code folding, and basic code completion to enhance productivity.
4. **Portability:** Code editors are often platform-independent, making them suitable for use on various operating systems without significant differences in functionality.
5. **Learning Curve:** Code editors tend to have a shorter learning curve compared to full-fledged IDEs, making them more accessible to beginners and developers working on smaller projects.

Disadvantages

1. Beginners may miss use the features like auto correct and auto suggestions by remembering it instead of learning how it work
2. May requires set up for code editor

Tags /Non Empty Tag

non-empty" tags refer to tags that can contain content between their opening and closing tags. In contrast, "empty" or "void" tags do not have content between opening and closing tags and usually represent self-contained elements.

- **Anchor** : Th**e <a>** tag is used to create hyperlinks in a web page

Eg: <a href="https://www.example.com">Visit Example.com</a>

The **href** attribute (hypertext reference) specifies the URL (Uniform Resource Locator) of the page or resource to which the link points.

Image(Void Tag): The <img> tag in HTML is used to embed images in a web page

Eg: <img src="image.jpg" alt="An example image">

Table: The <table> tag in HTML is used to create tables on a web page.

```
<table border="1">
 <tr>
   <th>Header 1</th>
   <tr>
   <td>Row 1, Cell 1</td>
</tr>
 <tr>
   <td>Row 2, Cell 1</td>
 </tr>
</table>
```

- <table>: This is the container for the entire table.
- <tr>: Stands for "table row" and is used to define a row in the table.
- <th>: Stands for "table header" and is used to define a header cell in a table. Text in <th> elements is typically bold and centered.
- <td>: Stands for "table data" and is used to define a standard cell in a table. This is

List

Ordered List: An ordered list is a list where each item is numbered

```
<ol>
  <li>Item A</li>
  <li>Item B</li>
  <li>Item C</li>
</ol>
```

OutPut

1. Item A
2. Item B
3. Item C

Unordered List: An unordered list is a list where each item is marked with a bullet point

```
<ul>
  <li>Item A</li>
  <li>Item B</li>
  <li>Item C</li>
</ul>
```

OutPut

- Item A
- Item B
- Item C

Block

- Every HTML element has a default display value, depending on what type of element it is.
- The two most common display values are block and inline.

1.Block-level Elements

- A block-level element always starts on a new line, and the browsers automatically add some space (a margin) before and after the element.
- A block-level element always takes up the full width available (stretches out to the left and right as far as it can
- Two commonly used block elements are: <p> and <div>.

Div

- The <div> element is often used as a container for other HTML elements.
- The <div> element has no required attributes, but style, class and id are common.
- When used together with CSS, the <div> element can be used to style blocks of content:

1.Block-level Elements

div Tag

Example

<div> A</div>

<div> B</div>

Output

A

B

p Tag

Example

<p> a</p>

<p> b</p>

Output

a

b

Other Block Level Elements

- <address><article><aside>
- <blockquote>
- <canvas>
- <dd><div><dl><dt>
- <fieldset><figcaption><figure>
- <footer><form>
- <h1><h6><header><hr>
- <main><nav><noscript>
- <ol><ul><li><p>
- <pre>
- <section>
- <table><tfoot>
- <video>

Inline Elements

- An inline element does not start on a new line.
- An inline element only takes up as much width as necessary.
- This is a <span> element inside a paragraph.

Other TAGS

- <a><abbr><acronym><b><bdo><big><br><button><cite><code><dfn><em><i><img><input><kbd><label><map><object><output><q><samp><script><select><small><span><strong><sub><sup><textarea><time><tt><var>

Span

- The <span> element is an inline container used to mark up a part of a text, or a part of a document.
- The <span> element has no required attributes, but style, class and id are common.
- When used together with CSS, the <span> element can be used to style parts of the text:

- **Quotation :** <q> tag is used to define a short inline quotation

Eg: <p>This is a <q>short inline quotation</q> in a paragraph.</p>

- **iframe:**

The <iframe> (short for inline frame) tag in HTML is used to embed another HTML document within the current HTML document

Eg: <iframe src="https://www.example.com" width="600" height="400" title="Example Website"></iframe>

- **form**:<form> tag in HTML is used to create an HTML form that collects user input

```
<form action="/submit-form" method="post">
 <form action="/submit-form" method="post">
  <label for="email">Email:</label>
  <input type="email" id="email" name="email" required>
  <label for="message">Message:</label>
  <textarea id="message" name="message" rows="4" required></textarea>
  <button type="submit">Submit</button>
 </form>
```

<form action="/submit-form" method="post">

- **action Attribute:**
- The action attribute specifies the URL or endpoint to which the form data will be sent when the user submits the form.
- In this example, action="/submit-form" indicates that the form data will be sent to the "/submit-form" URL.
- **method Attribute:**
- The method attribute specifies the HTTP method to be used when submitting the form.
- method="post" indicates that the data will be sent using the HTTP POST method.
- The two primary methods used in forms are:
- **POST**: Sends data to the server to be processed (e.g., for form submissions that may modify server data).
- **GET**: Appends data to the URL, typically used for retrieving data (not suitable for sensitive information).

Form Elements

- **<form>**:The root element for creating a form.
- **<input>**:A versatile element with various types, such as text, password, checkbox, radio, etc.
- **<label>**:Represents a label for an <input>, <select>, <textarea>, or <button> element.
- **<select>**:Creates a dropdown list.

```
<select name="country">
  <option value="us">United States</option>
  <option value="ca">Canada</option>
</select>
```

- **<textarea>:**Represents a multiline text input control.
- **<button>**:Represents a clickable button.
- **CheckBox :**A checkbox allows the user to select one or more options from a set of choices. Each checkbox operates independently of others, meaning you can select multiple checkboxes at the same time.
- **Radio Button:**Radio buttons are used when you want the user to make a single selection from a set of options.

Input Tag Types and Attribute

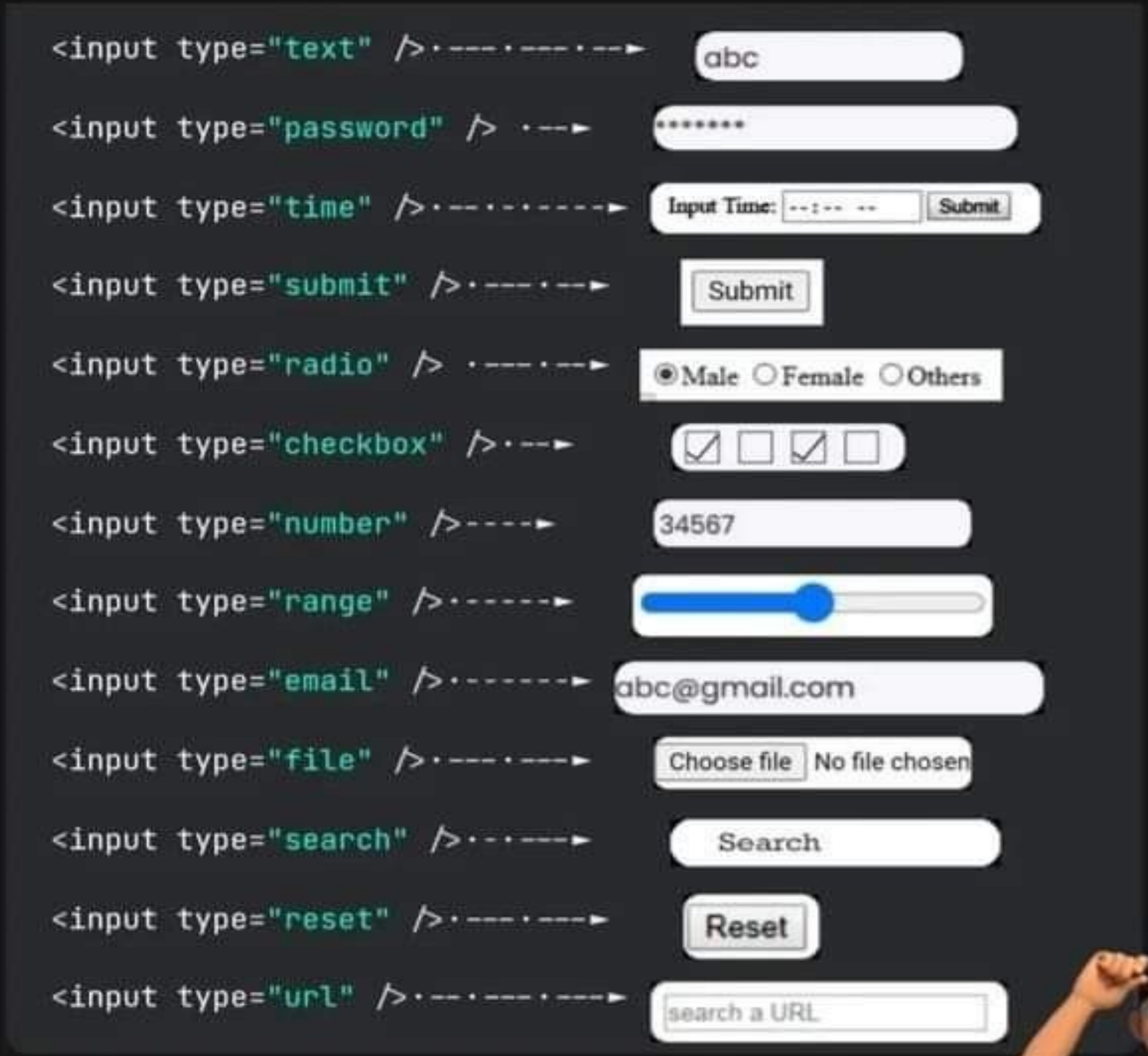
- **Text Input (type="text")**:
- <input type="text" name="username" placeholder="Enter your username">
- **Password Input (type="**password**"):**
- <input type="password" name="password" placeholder="Enter your password">
- **Checkbox (type="checkbox"):**
- **<**input type="checkbox" name="subscribe" value="yes"> Subscribe to newsletter
- **Radio Button (type="radio"):**

<input type="radio" name="gender" value="male"> Male

<input type="radio" name="gender" value="female"> Female

- **Number Input (type="number"):**
- **Email Input (type="email"):**
- **File Input (type="file"):**
- **Submit Button (type="submit"):**
- **Reset Button (type="reset"):**
- **Hidden Input (type="hidden"):**

HTML Input Types



```
<input type="text" />
<input type="password" />
<input type="time" />
<input type="submit" />
<input type="radio" />
<input type="checkbox" />
<input type="number" />
<input type="range" />
<input type="email" />
<input type="file" />
<input type="search" />
<input type="reset" />
<input type="url" />
```

KHALID – WEB DEV
@khalid.js

Why Name Attribute

- **Form Submission**:name attribute is crucial in this process as it identifies the name-value pair associated with the input field
- **JavaScript Access:**in case of js for interaction with form naem attribute used to reference and manipulate the input field var usernameValue = document.forms["myForm"]["username"].value**;**
- **Radio Buttons and Checkboxes:**name attribute is used to group related buttons or checkboxes together. Only one radio button within a group with the same name can be selected at a time.
- **To Increase Readability of Code**

Multimedia

- Multimedia refers to the integration of different types of media elements, such as text, images, audio, video, animations, and interactive content, to convey information or entertain users.
- The term "multimedia" is a combination of "multi" (many) and "media" (different forms of communication)

- Text: Written or displayed information that can include titles, captions, and descriptive content.

- Images: Still pictures or graphics that can be photographs, illustrations, diagrams, or any other visual representation.

- Audio: Sound or music that can enhance the overall experience. This includes spoken words, background music, or sound effects.

- Video: Moving images and animations that are often a sequence of frames displayed in rapid succession to create the illusion of motion.

- Animations: Dynamic visual elements that move or change over time. This can include animated graphics, GIFs, or other visual effects.

- Interactive Content: Elements that allow users to actively engage with the multimedia, such as clickable buttons, hyperlinks, forms, and interactive interfaces.

Code And Pre Tag

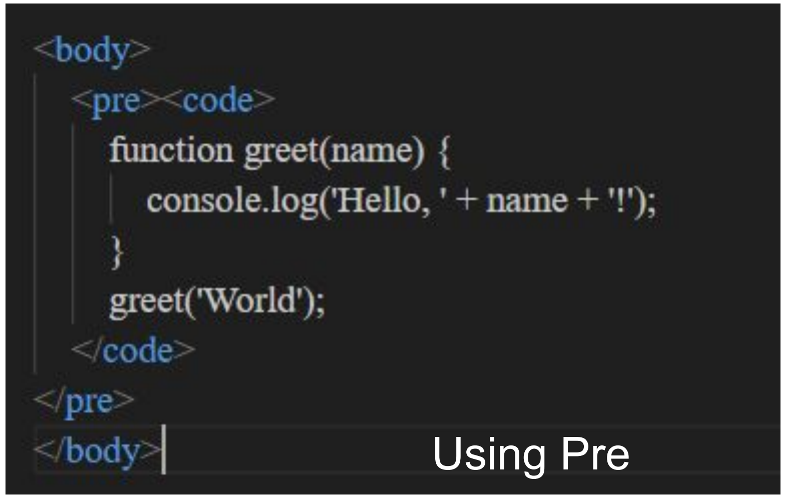
We can embed code snippets using various elements and attributes

The <code> element is typically used for inline code snippets, while the <pre> element is used for displaying blocks of code or preformatted text. These elements help maintain the formatting and structure of the code.

Why Code

1. **Documentation and Examples**: Including code snippets in HTML can be a helpful way to document code, demonstrate how to use certain functions or features, and provide examples for educational or reference purposes. Developers often include code snippets in tutorials, documentation, or blog posts to illustrate concepts or provide guidance.
2. **Presentation**: Sometimes, you might want to display code within an HTML document for presentation purposes, such as showcasing code samples or highlighting syntax. Using <pre> and <code> elements preserves the formatting and structure of the code, making it easier to read and understand.
3. **Copy and Paste**: Users can easily copy code snippets from HTML documents and paste them into their code editor or development environment for further exploration or use. By providing code samples directly in HTML, you make it convenient for users to access and work with the code.
4. **Styling and Highlighting**: HTML elements like <pre> and <code> can be styled using CSS to enhance their appearance, such as changing the font, color, or background. Additionally, you can use syntax highlighting libraries or plugins to improve the presentation of code snippets and make them more visually appealing.

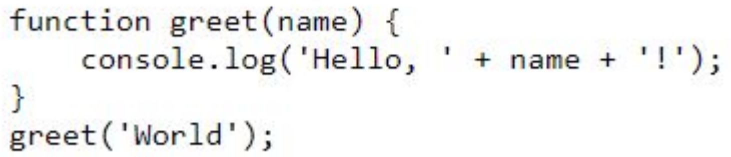
Code And Pre Tag



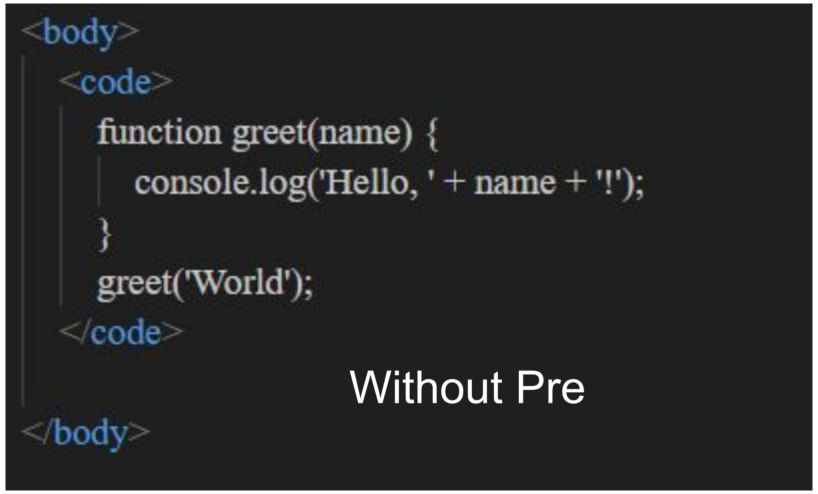
```
<body>
  <pre><code>
    function greet(name) {
      console.log('Hello, ' + name + '!');
    }
    greet('World');
  </code>
</pre>
</body>
```

Using Pre

Output

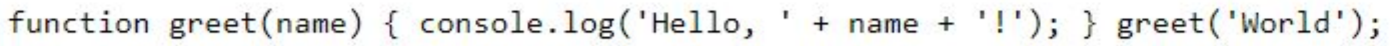


```
function greet(name) {
    console.log('Hello, ' + name + '!');
}
greet('World');
```



```
<body>
  <code>
    function greet(name) {
      console.log('Hello, ' + name + '!');
    }
    greet('World');
  </code>

</body>
```

Without Pre



```
function greet(name) { console.log('Hello, ' + name + '!'); } greet('World');
```

Sup and Sub

<sub> and <sup> elements are used to define subscript and superscript text, respectively. Subscript and superscript are commonly used in mathematical expressions, chemical formulas, footnotes, and various other contexts where text needs to be displayed below or above the baseline of surrounding text.

Example

```
<body>
   <p>This is a chemical formula: H<sub>2</sub> O</p>
   <p>This is an equation: x<sup>2</sup> + y<sup>2</sup> = r<sup>2</sup></p>
   <p>This is a footnote<sup>*</sup></p>
</body>
```

Output

This is a chemical formula: H_2O

This is an equation: $x^2 + y^2 = r^2$

This is a footnote*

Charset(set of Character) attribute

In HTML, the charset attribute is used within the <meta> element to specify the character encoding used in the document. Character encoding determines how characters are represented as binary data and decoded by web browsers when rendering HTML documents.

UTF-8 covers almost all of the characters and symbols in the world!

The charset attribute is placed within a <meta> tag within the <head> section of an HTML document.

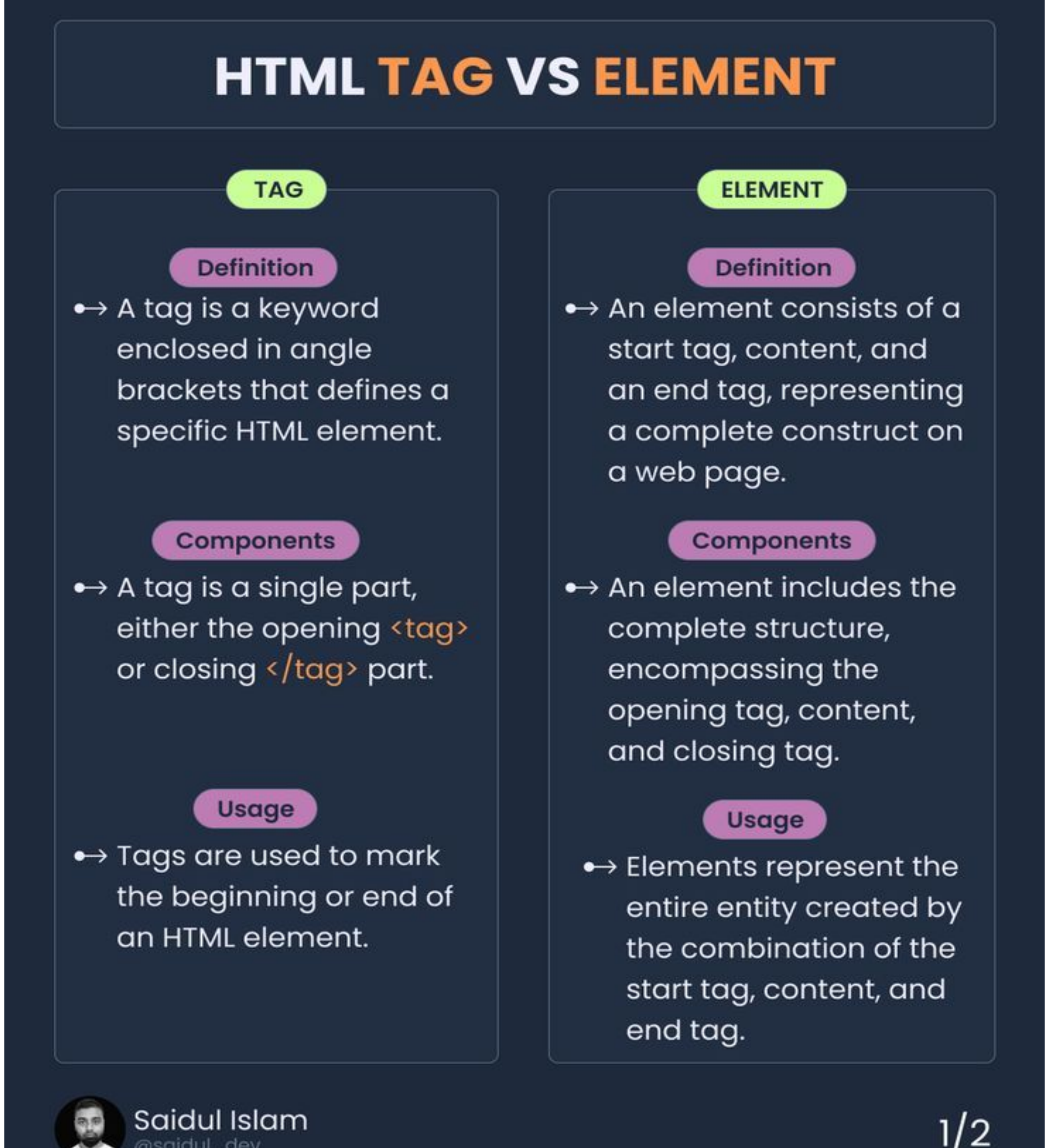
```
<head>

   <meta charset="UTF-8">

   <title>Document Title</title>
</head>
```

In this example, charset="UTF-8" indicates that the character encoding for the document is UTF-8(Unicode Transformation Format-8), which is a widely used encoding that supports a vast range of characters from various languages and scripts.

The charset attribute is essential for ensuring that characters in the HTML document are displayed correctly by the browser. It's recommended to always specify the character encoding, typically UTF-8, in your HTML documents to avoid potential rendering issues or character encoding mismatches.

Entities

- HTML entities are special codes used to represent characters that have a specific meaning in HTML, such as reserved characters or characters that might not be easily typable on a standard keyboard. HTML entities start with an ampersand (&) and end with a semicolon (;). Here are some common HTML entities:



HTML TAG VS ELEMENT

TAG

Definition

- A tag is a keyword enclosed in angle brackets that defines a specific HTML element.

Components

- A tag is a single part, either the opening `<tag>` or closing `</tag>` part.

Usage

- Tags are used to mark the beginning or end of an HTML element.

ELEMENT

Definition

- An element consists of a start tag, content, and an end tag, representing a complete construct on a web page.

Components

- An element includes the complete structure, encompassing the opening tag, content, and closing tag.

Usage

- Elements represent the entire entity created by the combination of the start tag, content, and end tag.

Saidul Islam

@saidul_dev

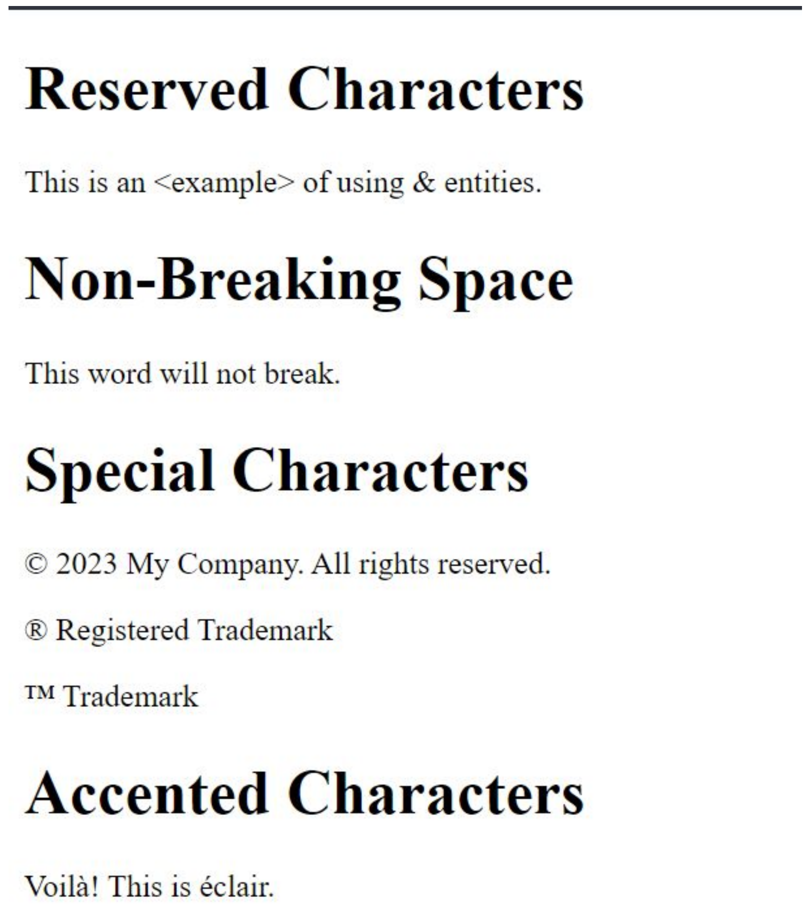
Entities

- **Reserved Characters:**
- &lt; represents < (less than)
- &gt; represents > (greater than)
- &amp; represents & (ampersand)
- &quot; represents " (double quote)
- &apos; represents ' (apostrophe/single quote)
- **Non-Breaking Space:**
- &nbsp; represents a non-breaking space.
- **Special Characters:**
- &copy; represents © (copyright)
- &reg; represents ® (registered trademark)
- &trade; represents ™ (trademark)
- **Accented Characters:**
- &eacute; represents é
- &ouml; represents ö
- &auml; represents ä

Example

```
<body>

    <h1>Reserved Characters</h1>
    <p>This is an &lt;example&gt; of using &amp; entities.</p>

    <h1>Non-Breaking Space</h1>
    <p>This&nbsp;word&nbsp;will&nbsp;not&nbsp;break.</p>

    <h1>Special Characters</h1>
    <p>&copy; 2023 My Company. All rights reserved.</p>
    <p>&reg; Registered Trademark</p>
    <p>&trade; Trademark</p>

    <h1>Accented Characters</h1>
    <p>Voil&agrave;! This is &eacute;clair.</p>

</body>
```

Output



Reserved Characters

This is an <example> of using & entities.

Non-Breaking Space

This word will not break.

Special Characters

© 2023 My Company. All rights reserved.

® Registered Trademark

™ Trademark

Accented Characters

Voilà! This is éclair.

HTML5 New Element

- **<header>:**Represents a header section that typically contains a group of introductory or navigational aids
- **<nav>:**Defines a navigation menu for a webpage.
- **<article>:**Represents a self-contained piece of content that could be distributed and reused independently, such as a news article or blog post
- **<section>:**Represents a thematic grouping of content within a document, such as a chapter, section, or topic
- **<footer>:**Represents a footer for its nearest section or article, often containing metadata, copyright information, or links to related pages
- **<figure>:**
- **<audio> And <Video>**
- **New Input Type :** date And email

Audio And Video

```
• <audio controls>
  <source src="audio.mp3" type="audio/mp3">
  Your browser does not support the audio element.
</audio>

<video width="320" height="240" controls>
  <source src="video.mp4" type="video/mp4">
  Your browser does not support the video element.
</video>
```

More tags:https://www.w3schools.com/tags/

HTML SVG

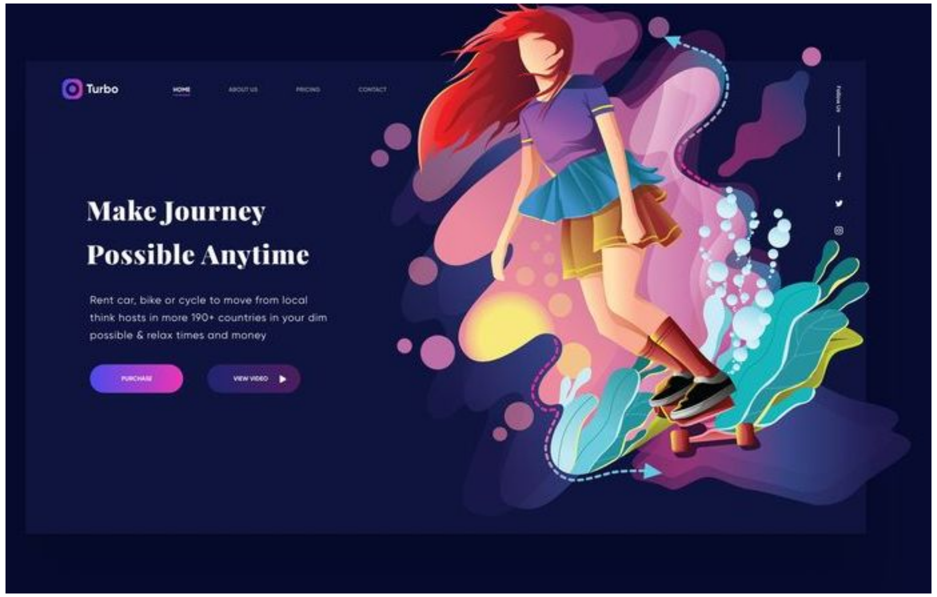
- SVG (Scalable Vector Graphics) is an XML-based markup language for describing vector graphics. It's commonly used for creating graphics on the web that can scale easily without losing quality

```
<svg width="200" height="100">
    <!-- Draw a rectangle -->
    <rect width="100" height="50" fill="#FF0000" />
          <!-- Draw a circle -->
    <circle cx="150" cy="75" r="25" fill="#00FF00" />
          <!-- Draw text -->
    <text x="10" y="20" fill="#000000">SVG Example</text>
 </svg>
```

IN Example

- The <svg> element is used to create an SVG container.
- The <rect> element draws a rectangle with a width of 100 units, a height of 50 units, and a red fill.
- The <circle> element draws a circle with a center at (150, 75), a radius of 25 units, and a green fill.
- The <text> element places text at coordinates (10, 20) with a black fill.

Exapmle



Make Journey Possible Anytime

Rent car, bike or cycle to move from local think hosts in more 190+ countries in your dim possible & relax times and money

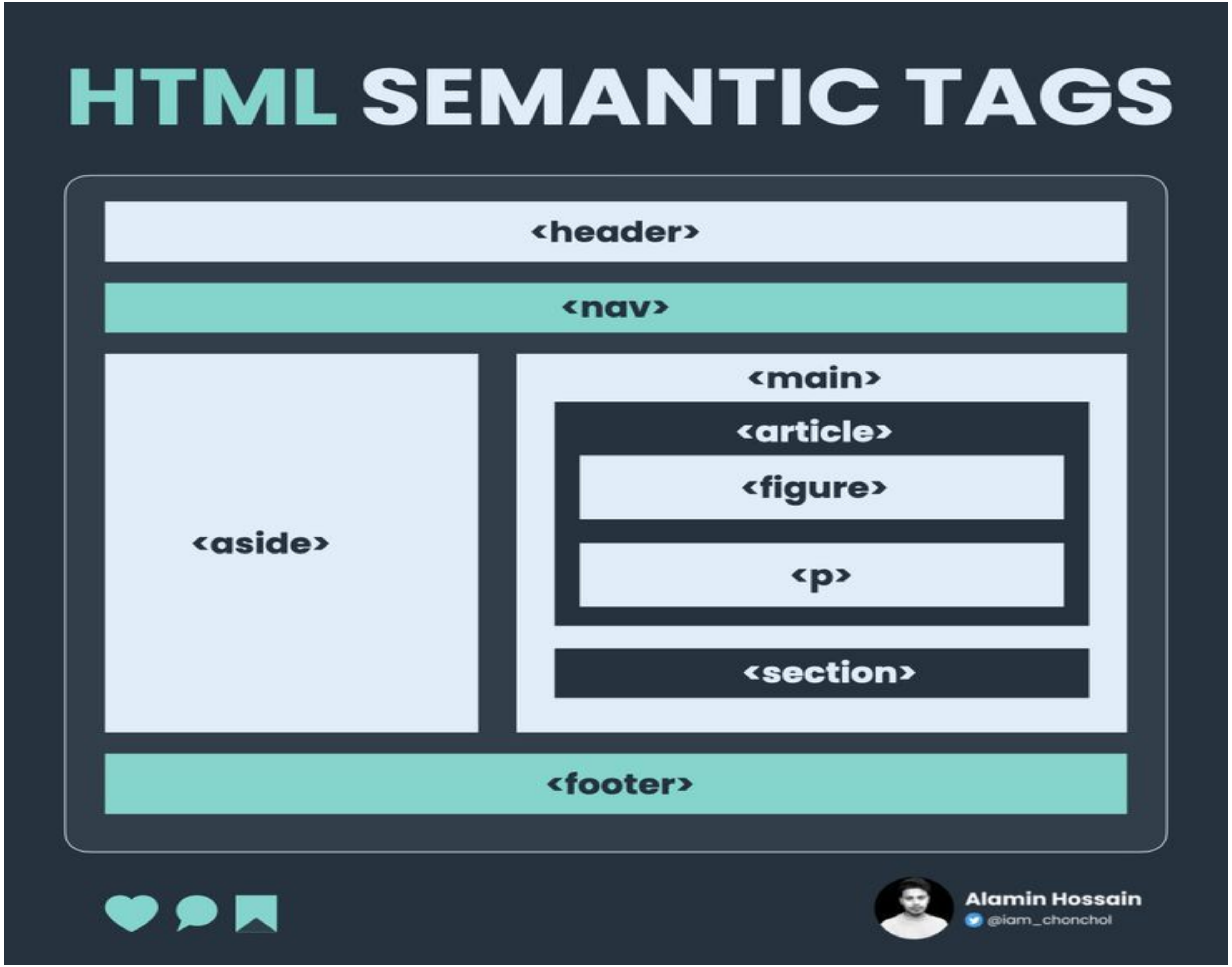
The Internet's Undersea Backbone.

Sub-Marine Cables

Ninety-nine percent of international data is transmitted by wires at the bottom of the ocean called submarine communications cables. In total, they are hundreds of

Semantic Tag

- Semantic tags in HTML are elements that carry meaning about the structure and content of a web page, making it more understandable for both browsers and developers. These tags provide context and information about the role of the enclosed content. Semantic HTML is important for accessibility, search engine optimization (SEO), and maintaining a clean and meaningful document structure.


HTML SEMANTIC TAGS
<header>
<nav>
<aside>
<main>
<article>
<figure>
<p>
<section>
<footer>
Alamin Hossain
@iam_chonchol

<header>:

Represents a container for introductory content or a group of navigational links.

Often contains headings, logos, and navigation menus.

<nav>:

Represents a section of navigation links.

<main>:

Represents the main content of the document. It excludes content that is repeated across multiple pages, such as headers, footers, and navigation menus.

<article>:

Represents a self-contained piece of content that could be distributed and reused independently, such as a news article or blog post.

<section>:

Represents a generic section of a document. It can be a thematic grouping of content, such as chapters, or a grouping of related content.

<aside>:

Represents content that is tangentially related to the content around it. It is often used for sidebars, pull quotes, or related links.

<figure> is used to encapsulate media content (like images, diagrams, or videos) along with their captions.

<figcaption> is used to provide a caption for the content inside a <figure>.

<footer>:

Represents the footer of a section or a page. Typically contains metadata, copyright information, and links to related documents.

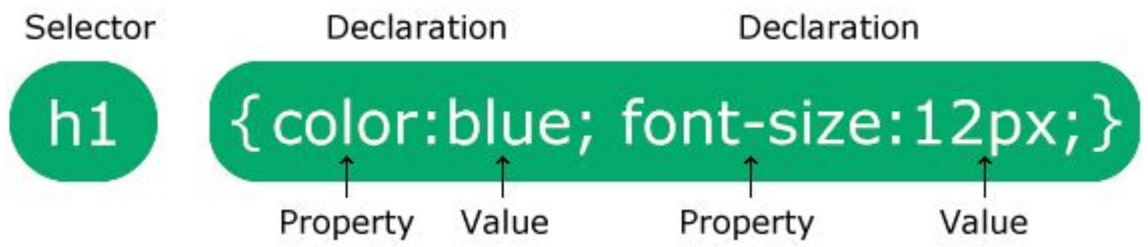
<blockquote>:

<time>:

<mark>:

CSS(Cascading Style Sheets)

- CSS, or Cascading Style Sheets, is a style sheet language used for describing the presentation of a document written in HTML or XML (including XML dialects such as SVG or XHTML). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.



- Css Syntax
- **Selectors:** CSS uses selectors to target HTML elements and apply styles to them. Selectors can target elements based on their type, class, ID, attributes, and more.
- **Properties and Values**: CSS properties define the visual styles of the selected elements, and each property is assigned a specific value.

Ways to Insert(ADD) CSS

- External CSS

External styles are defined within the <link> element, inside the <head> section of an HTML page we write separate css file for external css

<head><link rel="stylesheet" href="mystyle.css"></head>

- Internal CSS

Internal styles are defined within the <style> element, inside the <head> section of an HTML page:*<head><style>body { background-color: linen;}</style></head>*

- Inline CSS

Inline styles are defined within the "style" attribute of the relevant element:

<p style="color:red;">This is a paragraph.</p> . Write css with in html tag

Extarnal CSS

MyExample.html

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

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

•  </body>
•  </html>
```

mystyle.css

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

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

Internal CSS

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

</head><body>
<h1>This is a heading</h1>
</body></html>
```

Inline CSS

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

More:https://www.w3schools.com/css/css_howto.asp

Naming Convention

- naming conventions are guidelines for naming classes and IDs in your HTML and CSS code. Consistent and meaningful naming conventions help improve code readability, maintainability, and collaboration among developers. There are several popular naming conventions, and you can choose the one that best fits your project or team preferences.

Naming Convention

Naming rules

- block-name__elem-name_mod-name_mod-val
- Names are written in lowercase Latin letters.
- Words are separated by a hyphen (-).
- The block name defines the namespace for its elements and modifiers.
- The element name is separated from the block name by a double underscore (__).
- The modifier name is separated from the block or element name by a single underscore (_).
- The modifier value is separated from the modifier name by a single underscore (_).
- For boolean modifiers, the value is not included in the name.

Type Of Convenstion

- BEM
- SMACSS
- ITCSS
- OOCSS
- AMCSS

Selector Types

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

Selector

Selector	Example	Example description
#id	#firstname	Selects the element with id="firstname"
.class	.intro	Selects all elements with class="intro"
element.class	p.intro	Selects on ly <p> elements with class="intro"
*	*	Selects all elements
element	p	Selects all <p> elements
element,element,..	div, p	Selects all <div> elements and all <p> elements

More: https://www.w3schools.com/css/css_selectors.asp

Pseudo Selector

- Pseudo-selectors are used to select and style elements based on their state or position in the document. Some common examples include

a. **:hover** used to select and style an element when the user hovers over it with the mouse

b. **:focus** is applied when an element is currently in focus. It is often used with form elements like input fields and buttons to indicate that they are currently selected or being interacted with.

c. **:first-child** first child element of its parent. It is useful when you want to style the first child element differently from its siblings.

d. **:nth-child** allows you to select elements based on their position within a parent element

CSS Pseudo-classes

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

For example, it can be used to:

- Style an element when a user mouses over it
- Style visited and unvisited links differently
- Style an element when it gets focus

The syntax of pseudo-classes:

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

Note: a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective!

Anchor Pseudo-classes

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

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

/* mouse over link */
a:hover {
  color: #FF00FF;
}

/* selected link */
a:active {
  color: #0000FF;
}
```

@CodeWithNayeem

Example

```
a:hover {
  color: red;
}
input:focus {
  border: 2px solid blue;
}
li:first-child {
  font-weight: bold;
}
li:nth-child(2n) {
  background-color: #f0f0f0;
}
```

Pseudo Element(Selector)

- A pseudo-element is a CSS feature that allows you to style a specific part of an element. Pseudo-elements are denoted by double colons :: in CSS.
- ::before and ::after are pseudo-elements that allow you to insert content before and after an element's actual content, respectively

```
p::before {
  content: ">>";
  color: blue;
}
p::after {
  content: "<<";
  color: red;
}
```

Css Inheritance

CSS inheritance refers to the mechanism by which certain properties of CSS styles are passed down from parent elements to their child elements in the HTML document DOM (Document Object Model). This inheritance system simplifies styling by allowing child elements to inherit properties from their parent elements, reducing the need to specify styles for every single element individually.

Inherited Properties: Some CSS properties are automatically inherited by child elements from their parent elements. Common inherited properties include font properties

(e.g., font-family, font-size, font-weight), text properties (e.g., color, line-height, text-align), and list properties (e.g., list-style-type, list-style-position).

Css Inheritance

```
.parent {
        font-family: Arial, sans-serif;
        color: blue;
    }

    /* Child element style */
    .child {
        font-style: italic;
    }
```

```
<body>
    <div class="parent">
        This is the parent element.
        <div class="child">
            This is the child element.
        </div>
    </div>
</body>
```

- The `.parent` class applies a font-family of Arial or sans-serif and a color of blue to the parent `<div>` element.
- The `.child` class applies a font-style of italic to the child `<div>` element.

Since the child `<div>` is nested inside the parent `<div>`, it inherits some styles from its parent:

Class And ID and Its Selector

Class

In HTML and CSS, a class is a way to define a reusable style or a grouping mechanism for HTML elements. Multiple elements can share the same class, and styles defined for that class will be applied to all elements with that class.

Id

- An ID is a unique identifier for an HTML element. Unlike classes, IDs should be unique within a page. They are often used to select and style a specific element on a page

HTML CLASS VS ID ATTRIBUTE

Class

```html
<p
class="highlight">Highlig
hted Text-1</p>

<p
class="highlight">Highli
ghted Text-2</p>
```

```css
.highlight {
background-color: yellow;
font-weight: bold;
}
```

- The class attribute is used to assign one or more class names to an HTML element.
- Multiple elements can share the same class, allowing you to apply the same styles or behavior to multiple elements.
- Classes are useful when you want to style or target groups of elements with similar characteristics.
- CSS styles targeting a class start with a dot (.) followed by the class name (e.g., .my-class).

ID

```html
<p id="mark">Marked
Text</p>

<p id="header">This
is Header Text</p>
```

```css
#mark {
background-color: blue;
color: white;
}
```

- The id attribute is used to assign a unique identifier to a single HTML element.
- Each element on a page can have only one unique ID.
- IDs are useful when you want to style or target a specific element uniquely or when you need to reference it in JavaScript.
- CSS styles targeting an ID start with a hash (#) followed by the ID name (e.g., #my-id).

@saidul_dev

Example

Class Html Usage

<div *class="example"*>This is a div with a class.</div>

Css Usage

```
.example {
  color: blue;
  font-size: 16px;
}
```

ID HTML Usage

<p id="unique">This is a paragraph with an ID.</p>

Css Usage

```
#unique {
  font-weight: bold;
  color: green;
}
```

Css Property

- **Color :**is used to set color of text , background , border color etc

<h1 style="background-color:DodgerBlue;">Hello World</h1>

<p style="color:Tomato;">Lorem ipsum...</p>

- **Background** properties are used to add background effects for elements. such as
- background-color:*Color Value* ;
- background-image:url("Image Location/Path")
- background-repeat: repeate mode of background
- background-attachment
- background-position: x and y posiotion of background image
- background-Size: Sized of image

Font Property

- **Font:** Used To set and style of text such as Size , Type , Font_Family
- **font-family:** use the font-family property to specify the font of a text eg..

p1 { font-family: "Times New Roman", Times, serif;}

- **font-style** property is mostly used to specify italic text

This property has three values:

- normal - The text is shown normally
- italic - The text is shown in italics
- oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

Font Property

- **font-family:** Specifies the font family for text.
- **font-size:** Sets the size of the font.
- f**ont-weight**: Defines the thickness of the font (e.g., normal, bold).
- **font-style**: Specifies the font style (e.g., normal, italic).
- **line-height**: Sets the height of a line of text. /* Values: normal, number, length, percentage */

Text Property

Text Decoration: Use the text-decoration property to add or remove text decorations (underline, overline, line-through). eg

```
a {
  text-decoration: none; /* Values: none, underline, overline, line-through */
}
```

Text Align: Use the text-align property to set the horizontal alignment of text.

```
p {
  text-align: center; /* Values: left, right, center, justify */
}
```

Opacity Property

- The opacity property in CSS is used to control the transparency of an element. The value for opacity ranges from 0 (completely transparent) to 1 (completely opaque)
- Eg

```
.element {
 opacity: 0.7; /* Set the opacity to 70% */
}.
```

Comment In Css

Single Line

- /*This is Single Line Comment */

Multi Line Comment

- /*

this is multiline Comment

example

*/

Border Property

- border property is used to define the border of an element
- Eg

```
.element {
 border-width: 2px;
 border-style: solid;
/*value :dotted ,dashed .double */
border-color: #333;
}
```

- Shorthand

border: [border-width] [border-style] [border-color];

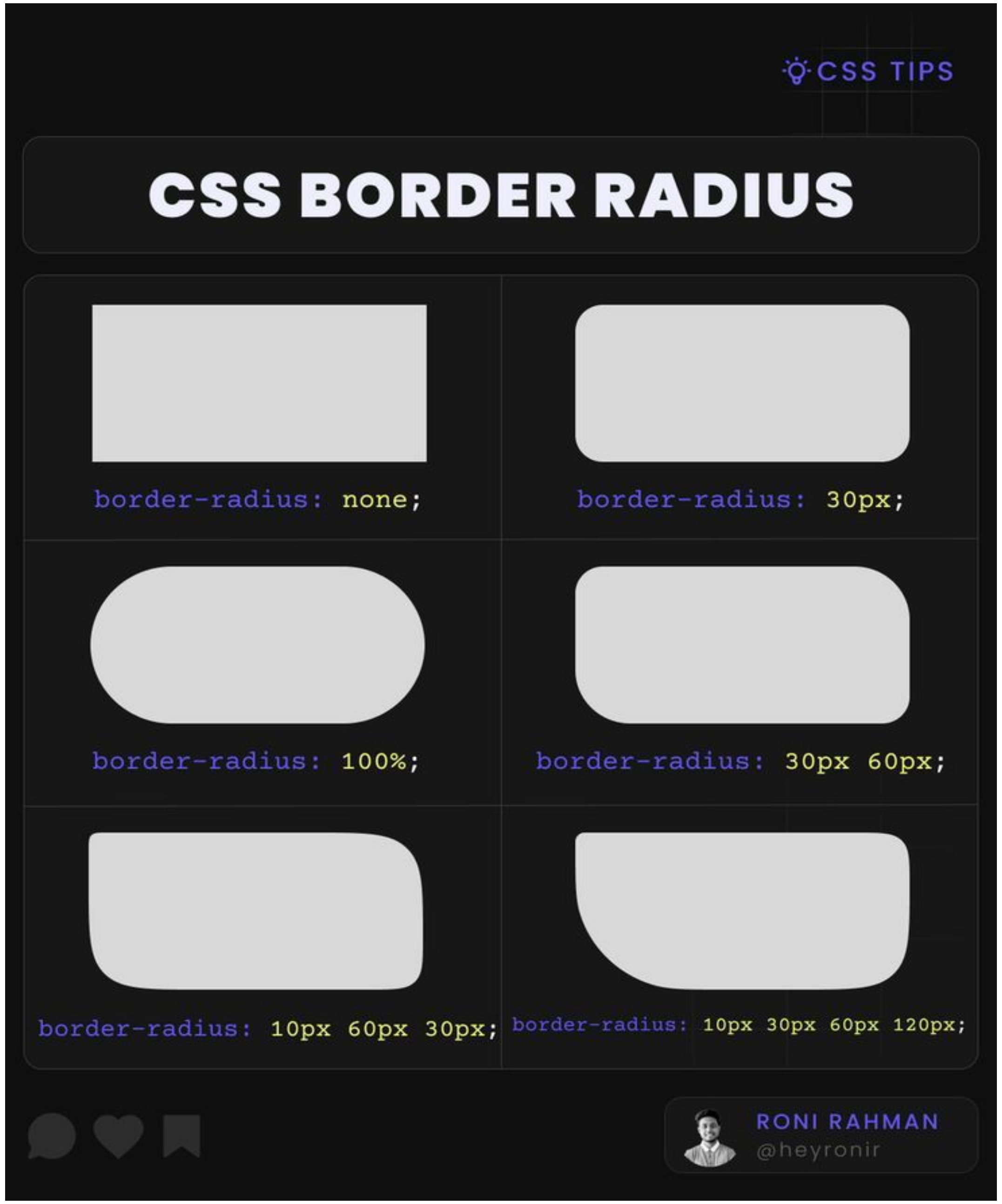
Eg:

```
.element {
 border: 2px solid #333;
}
```

- you can set the border for specific sides using properties like border-top, border-right, border-bottom, and border-left

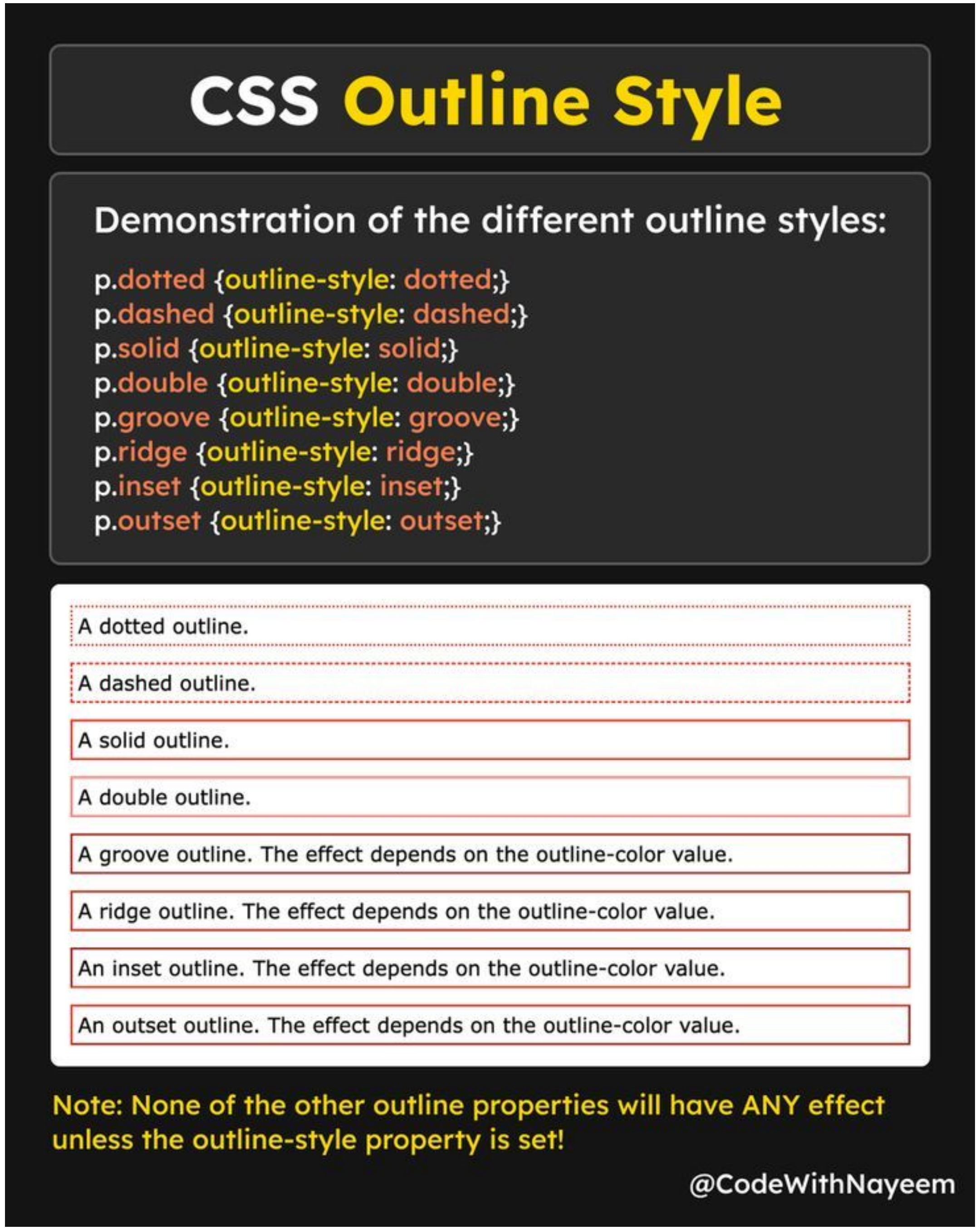
```
.element {
  border-top: 1px dotted #999;
  border-right: 2px dashed #666;
  border-bottom: 3px double #ccc;
  border-left: 4px solid #000;
}
```

Short Hande - border :



CSS BORDER RADIUS

```
border-radius: none;
border-radius: 30px;
border-radius: 100%;
border-radius: 30px 60px;
border-radius: 10px 60px 30px;
border-radius: 10px 30px 60px 120px;
```



CSS Outline Style

Demonstration of the different outline styles:

```
p.dotted {outline-style: dotted;}
p.dashed {outline-style: dashed;}
p.solid {outline-style: solid;}
p.double {outline-style: double;}
p.groove {outline-style: groove;}
p.ridge {outline-style: ridge;}
p.inset {outline-style: inset;}
p.outset {outline-style: outset;}
```

```
A dotted outline.
A dashed outline.
A solid outline.
A double outline.
A groove outline. The effect depends on the outline-color value.
A ridge outline. The effect depends on the outline-color value.
An inset outline. The effect depends on the outline-color value.
An outset outline. The effect depends on the outline-color value.
```

Note: None of the other outline properties will have ANY effect unless the outline-style property is set!

@CodeWithNayeem

Outline

- The outline CSS shorthand property allows drawing a line around the element, outside the border. It is used to set all the properties of the outline in a single declaration

```
.element {
  outline-width: 2px;
/*medium|thin|thick|length|initial|inherit*/
  outline-style: solid;
/*value :dotted ,dashed .double */
outline-color: #333;
Outline-offset:medium|thin|thick
|length|initial|inherit;
}
```

Margin Property

The margin property sets the space outside the border of an element. It defines the clearance around an element's outer edge. You can set the margin for all sides at once or individually for each side.

Short Hand

```
selector {
  margin: top right bottom left;
}
```

Eg:

```
.box {
  margin: 10px 20px 10px 20px; /* top right bottom left */
}
```

Padding

The padding property sets the space between the content of an element and its border. It defines the inner clearance within the element.

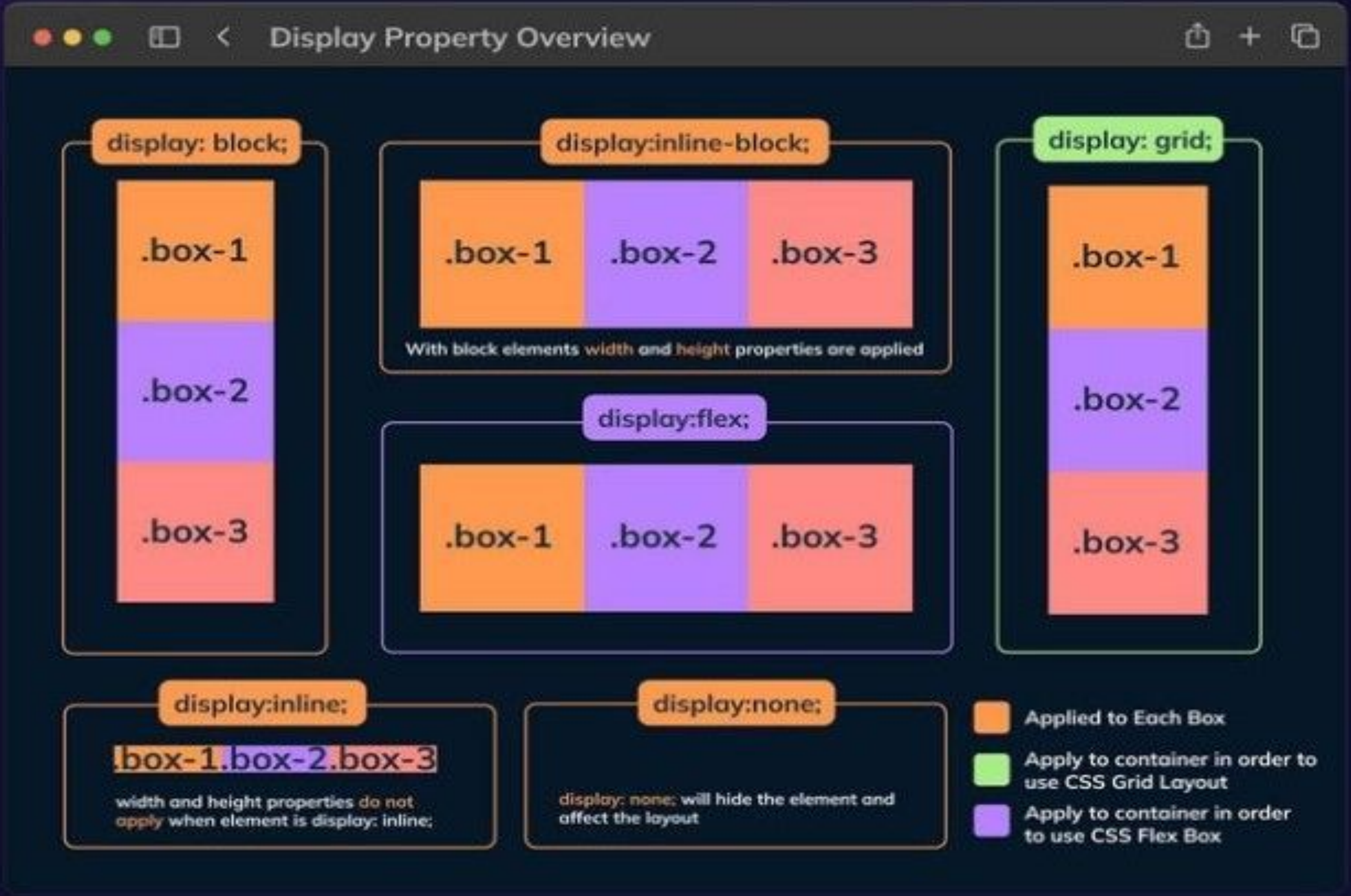
Shorthand

```
selector {
  padding: top right bottom left;
}
Eg.  .box {
  padding: 15px; /* Applies to all sides */
}
```

CSS Display

- In CSS (Cascading Style Sheets), the display property is used to control the layout behavior of an element. It determines how an element is rendered in the document flow. The display property can take various values, each influencing the element's layout differently

Display Property Cheat Sheet



With block elements width and height properties are applied

width and height properties do not apply when element is display: inline;

display: none; will hide the element and affect the layout

🟧 Applied to Each Box

🟩 Apply to container in order to use CSS Grid Layout

🟪 Apply to container in order to use CSS Flex Box

Diaplay:Block

The element will generate a block-level box. It starts on a new line and takes up the full width of its container, extending to the left and right edges.

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

Display:Inline

The element will generate an inline-level box. The element will not start on a new line and will only take up as much width as necessary.

```
span {
  display: inline;
}
```

Display:Inline-Block

The element will generate an inline-level box, but it will behave like a block-level box in terms of layout. It will flow with surrounding content and allow for setting width and height.

```
img {
  display: inline-block;
}
```

Display:None

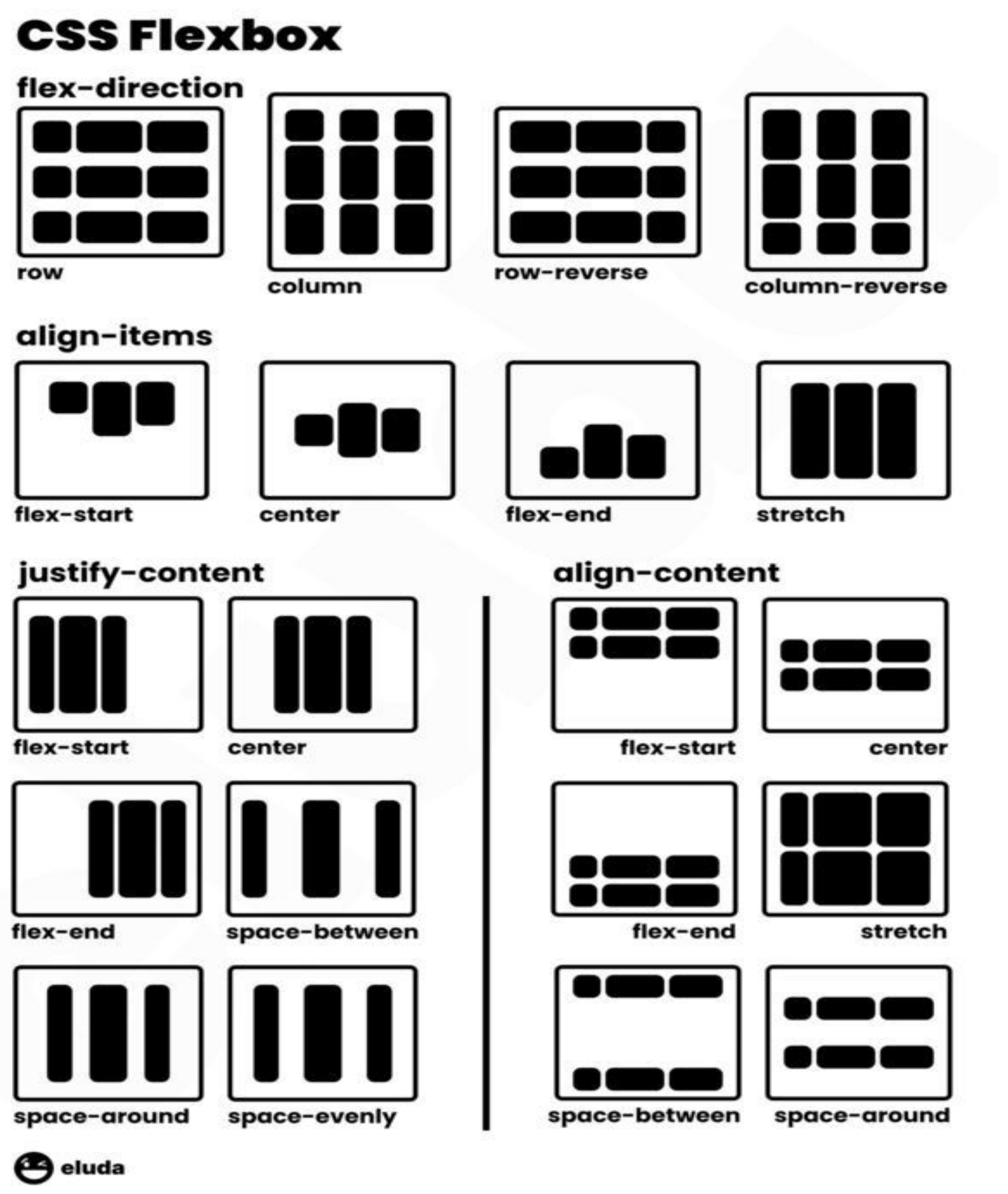
The element will be completely removed from the layout and will not take up any space. It's often used for hiding elements.

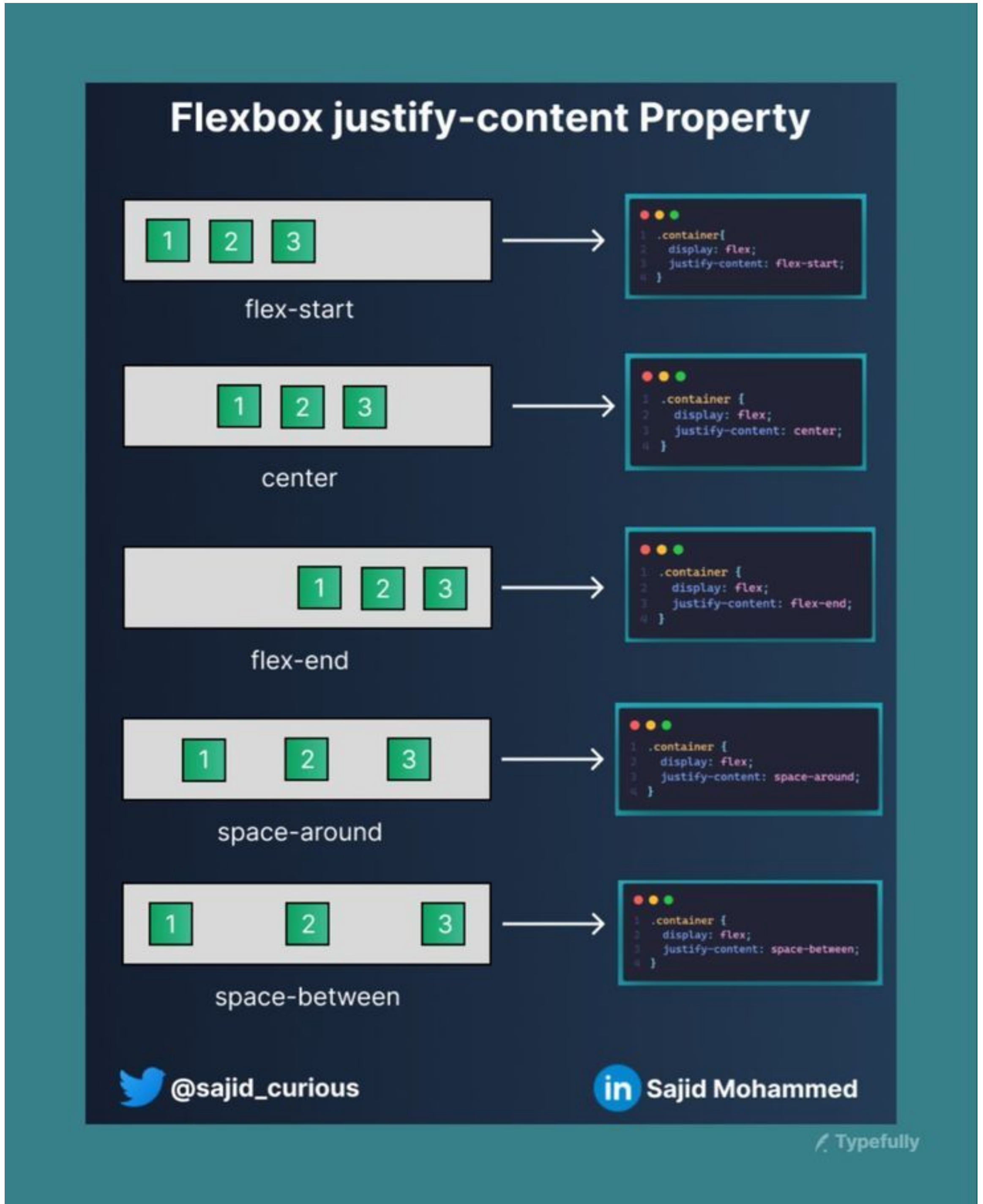
```
.hidden {
  display: none;
}
```

Display:Flex

The element becomes a flex container, and its direct children become flex items. This allows you to create flexible and responsive layouts.

```
.flex-container {
  display: flex;
}
```





```css
.container{
  display: flex;
  justify-content: flex-start;
}
```

```css
.container {
  display: flex;
  justify-content: center;
}
```

```css
.container {
  display: flex;
  justify-content: flex-end;
}
```

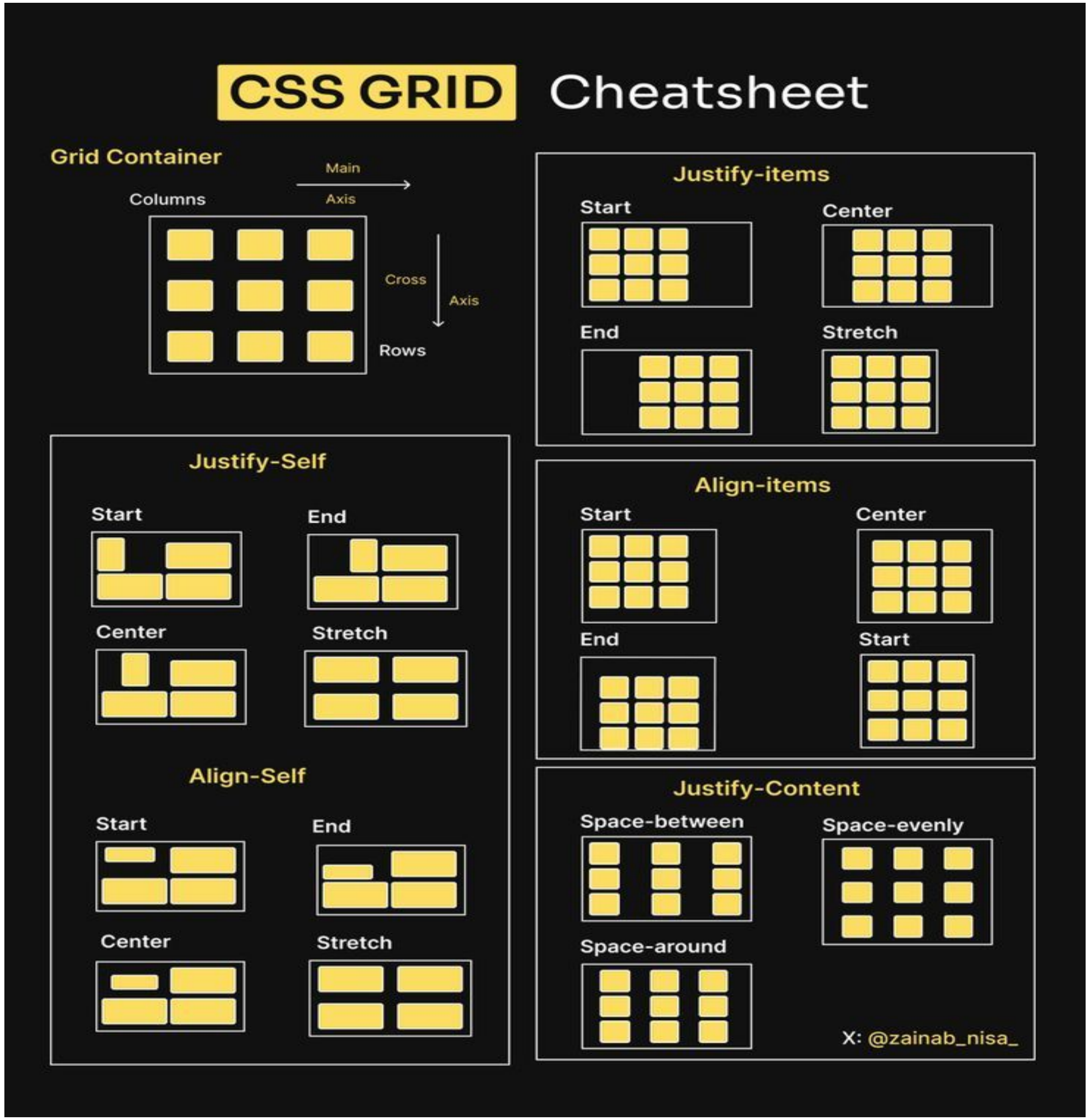
```css
.container {
  display: flex;
  justify-content: space-around;
}
```

```css
.container {
  display: flex;
  justify-content: space-between;
}
```

Display:Grid

- The element becomes a grid container, and its direct children become grid items. This is used for creating grid-based layouts.

```
.grid-container {
  display: grid;
}
```

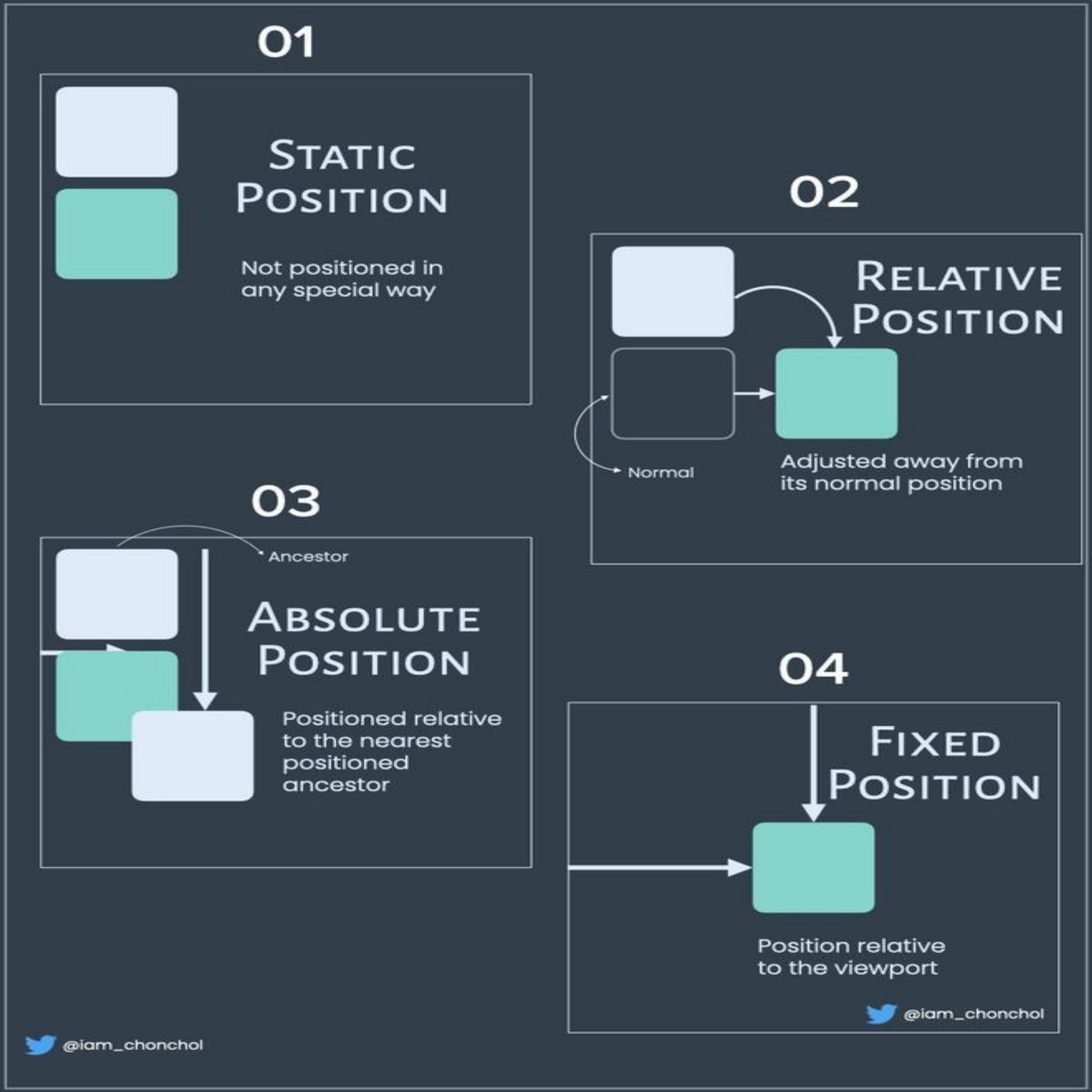

CSS GRID Cheatsheet
Grid Container
Main Axis
Columns
Cross Axis
Rows
Justify-items
Start
Center
End
Stretch
Justify-Self
Start
End
Center
Stretch
Align-items
Start
Center
End
Start
Align-Self
Start
End
Center
Stretch
Justify-Content
Space-between
Space-evenly
Space-around
X: @zainab_nisa_

Float

The float property in CSS is used to specify how an element should float. Floating an element allows it to be pushed to one side of its containing element while allowing content to flow around it. It's commonly used for creating layouts where elements are positioned side by side.

```
.float-left {
  float: left;
}
.float-right {
  float: right;
}
```

CSS POSITION


01
Static Position
Not positioned in any special way
02
Relative Position
Normal
Adjusted away from its normal position
03
Ancestor
Absolute Position
Positioned relative to the nearest positioned ancestor
04
Fixed Position
Position relative to the viewport
@iam_chonchol
@iam_chonchol

Position

- The CSS position property is used to control the positioning of an element within its containing element. Its value can be

1. ***Static (position: static;):***

This is the default value. Elements with position: static; are positioned according to the normal flow of the document.

The top, right, bottom, and left properties have no effect when an element has position: static;

```
.static-example {
   position: static;
}
```

2.Relative (position: relative;):

An element with position: relative; is positioned relative to its normal position in the document flow.

You can then use the top, right, bottom, and left properties to move the element from its normal position.

```
.relative-example {
    position: relative;
    top: 10px;
    left: 20px;
}
```

3.Absolute (position: absolute;)

An element with position: absolute; is removed from the normal flow of the document and positioned relative to its nearest positioned ancestor.

If there is no positioned ancestor, it is positioned relative to the initial containing block (usually the <html> element).

It is then offset based on the values of top, right, bottom, and left.

```
.absolute-example {
  position: absolute;
  top: 50px;
  left: 100px;
}
```

4.Fixed (position: fixed;)

An element with position: fixed; is removed from the normal flow of the document and positioned relative to the browser window.

It remains in a fixed position even when the page is scrolled.

Similar to absolute, it is offset based on the values of top, right, bottom, and left.

```
.fixed-example {
   position: fixed;
   top: 10px;
   right: 10px;
}
```

5.Sticky (position: sticky;)

An element with position: sticky; is positioned based on the user's scroll position. It behaves like relative within its container until it crosses a specified point, then it becomes fixed.

The element will stick to the specified position when the user scrolls to that point.

```
.sticky-example {
   position: sticky;
   top: 50px;
}
```

Media Query

What is programming Langauge

- A programming language is a computer language that is used by programmers (developers) to communicate with computers. It is a set of instructions written in any specific language (C, C++, JavaScript, Python) to perform a specific task.
- A programming language is mainly used to develop desktop applications, websites, and mobile applications.

Java Script

- JavaScript is an object-based scripting language which is lightweight and cross-platform.(object-based" because it uses objects to organize and structure code)

- JavaScript is not a compiled language, but it is a translated language. The JavaScript Translator (embedded in the browser) is responsible for translating the JavaScript code for the web browser.(*When you write JavaScript code, you create a text file containing human-readable code.The translation happens at runtime in the user's web browser.)*
- JavaScript is used to create client-side dynamic pages

Features

- All popular web browsers support JavaScript as they provide built-in execution environments.
- JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
- JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).
- JavaScript is an object-oriented programming language that uses prototypes rather than using classes for inheritance.
- It is a light-weighted and interpreted language.
- It is a case-sensitive language.
- JavaScript is supportable in several operating systems including, Windows, macOS, etc.
- It provides good control to the users over the web browsers.

Application of JavaScript

- Client-side validation,
- Dynamic drop-down menus,
- Displaying date and time,
- Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
- Displaying clocks etc.

JS In Html

```
<script>
document.write("Hello JavaScript by JavaScript");
</script>
```

We can use <Script> tag to execute/Implement Js in our html website

its file extension is .js

JS In Html / Place To Put Js(Linked JS)

1.Between the body tag of html

```
<body><script>
document.write("Hello JavaScript by JavaScript");
</body></script>
```

2.Between the head tag of html

```
<head><script>
  document.write("Hello JavaScript by JavaScript");
</script><head>
```

3. In .js file (external javaScript)

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

Tools for js development

Text Editors/IDEs:

- Visual Studio Code (VSCode): A popular, free, and open-source code editor with a wide range of extensions for JavaScript development.
- Sublime Text: A lightweight, fast, and customizable text editor.
- Atom: A hackable text editor developed by GitHub.

Version Control:

- Git: A distributed version control system widely used for tracking changes in source code.

Package Managers:

- npm (Node Package Manager): The default package manager for Node.js and JavaScript. It's used to install and manage third-party libraries and tools.

Testing Frameworks:

- Jest: A JavaScript testing framework developed by Facebook.
- Mocha: A feature-rich JavaScript test framework for Node.js and the browser.
- Chai: An assertion library that works well with Mocha and other testing frameworks.

Linters:

- ESLint: A static code analysis tool for identifying and fixing common programming errors and enforcing coding standards.

Debugger:

- Chrome DevTools: Built into the Chrome browser, it provides a set of web developer tools for debugging JavaScript.

Web Frameworks:

- React: A JavaScript library for building user interfaces.
- Angular: A TypeScript-based web application framework developed by Google.
- Vue.js: A progressive JavaScript framework for building user interfaces.

REST API Clients:

- Postman: A popular API testing and development tool.

Variable

variables are used to store and manipulate data

Variable Declaration:

- We can declare a variable using the ***var, let, or const*** keyword.
  Syntax
  keyword(let,var,const) varaibleName;

`Let:` Variables declared with let are block-scoped, meaning they are only visible within the block (or statement) where they are defined.

`Var:` Variables declared with var are function-scoped, meaning they are visible throughout the entire function in which they are declared

`const` must be initialized during declaration, and once assigned, the value cannot be changed. It creates a constant reference to the value.

Variable Naming Conventions:

- Rules for naming variables. Use meaningful names that describe the purpose of the variable. Variables are case-sensitive.

Global and Local Scope(variable Scope)/Local and Global variable

- Variables declared outside of any function or block have global scope, while those declared within a function or block have local scope.
- Variables declared with var are function-scoped, while variables declared with let and const are block-scoped. It's generally recommended to use let and const to avoid unexpected behavior.

  -Variables declared outside of any function or block have global scope(can access by other function or class), while those declared within a function or block have local scope(only accessible with in same function)

Data Type

Data Types: Determined What type of data variable can hold such as 1 2 3(Integer) , “HI”(String)

JavaScript has several data types that variables can hold, including:

1.Primitive Data Types:

- These are the basic building blocks of data in JavaScript. They include:
  - Number: Represents numeric values.
  - String: Represents textual data.
  - Boolean: Represents true or false values.
  - Null: Represents the intentional absence of any object value.
  - Undefined: Represents uninitialized variables.

Eg

```
let numberVar = 42;         // Number-> non-decimal  number such as 1,2 ,3 4,
let stringVar = "Hello";    // String-> set of character Hello is set of character H, e,l,l,o
let booleanVar = true;      // Boolean -> Either true or false
let arrayVar = [1, 2, 3];    // Array -> can hold more than one value
let objectVar = { key: 'value' };  // Object -> store key and ites value eg “Name” :Ram  “Name is key and its value is Ram
```

Data Type

2.Object Data Type:

- Objects are complex data types that can hold key-value pairs. They can represent more complex structures and behaviors.

```
let person = {
 /*Left hand sight :Key : Right Hand Sight Value* /
         firstName: "John",
         lastName: "Doe",
        age: 30,
         isStudent: false
};
```

3.Array Data Type

- Arrays are special kinds of objects that store ordered collections of values
- `let numbers = [1, 2, 3, 4, 5];`
- `let fruits = ["apple", "banana", "orange"];`
- ` Array Value are access with Index and index start with 0 which mean   number[0] = 1;number[1] = 2;number[2] = 3;`
-

Data Type

Function Data Type:

- In JavaScript, functions are first-class citizens, which means they can be assigned to variables, passed as arguments to other functions, and returned as values from other functions.

```
function add(x, y) {
    return x + y;
}

let result = add(5, 10); // Calls the function and assigns the result to 'result'
```

JavaScript Identifiers

All JavaScript variables must be identified with unique names.These unique names are called identifiers.Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

The ***general rules(Naming Convention)*** for constructing names for variables (unique identifiers) are:

- Names can contain letters, digits, underscores, and dollar signs.
- Names must begin with a letter.
- Names can also begin with $ and _ (but we will not use it in this tutorial).
- Names are case sensitive (y and Y are different variables).
- Reserved words (like JavaScript keywords) cannot be used as names.

Operators

Coming Soon… You can go with more if you want it now

More

DOM (Document Object Model)

The Document Object Model (DOM) is a programming interface that represents the structure of a document as a tree of objects. The document can be an HTML or XML document, and the DOM provides a way for programs to manipulate the structure, style, and content of these documents.

The DOM represents the document as a tree of nodes, where each node corresponds to an element, attribute, or piece of text in the document. The topmost node is the "document" node, and it has child nodes representing the HTML or XML elements in the document.

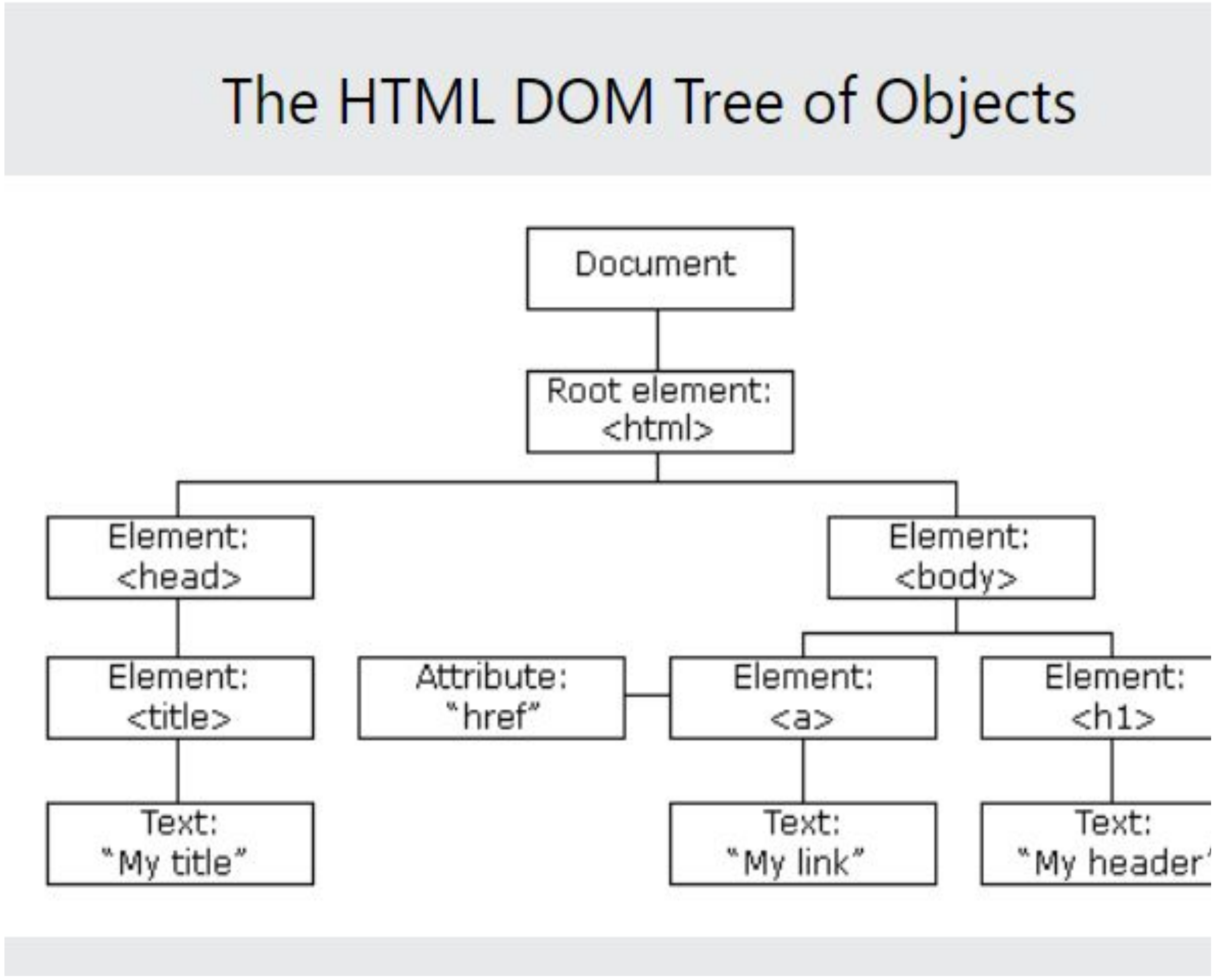
JavaScript can be used to interact with the DOM, allowing developers to dynamically update and modify the content and structure of a web page. For example, you can use JavaScript to add, remove, or modify HTML elements, change styles, handle events, and more.

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:

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



Examples

```
var myDocument    document

// Access an element by its ID
var myElement = document getElementById "myElementId"

// Change the text content of the element
myElement.textContent    "Hello, World!"
```

In this example, document refers to the document object, and getElementById is a method of the document object that allows you to access an HTML element by its ID. The textContent property is then used to change the text content of the element.

Methods for selecting elements based on different criteria

Document Object Model (DOM) provides a variety of methods that allow you to interact with and manipulate web documents DOM method is used to select the first element that matches a specified selector within the document. Such as ID ,Name Or Css Selector

1. **getElementById:**
   - This method selects an element by its ID.

```
var element = document.getElementById('myElementId');
```

2. **getElementsByClassName:**
   - This method selects a collection of elements with the specified class name.

```
var elements = document.getElementsByClassName('example');
```

3. **getElementsByTagName:**
   - This method selects a collection of elements with the specified tag name.

```
var paragraphs = document.getElementsByTagName('p');
```

Methods for selecting elements based on different criteria

4. **getElementsByName:**

- This method selects a collection of elements with the specified name attribute.

```
var inputs = document.getElementsByName('myInputName');
```

6. **querySelector** and **querySelectorAll** on specific elements:

- You can also use querySelector and querySelectorAll on specific elements to search for elements within a particular context.

```
var paragraphInsideDiv = document.querySelector('div.example p');

var allParagraphsInsideDiv = document.querySelectorAll('div.example p
```

Methods for selecting elements based on different criteria

7. **createElement:**
   - Creates a new HTML element.

```
var newElement = document.createElement('div');
```

8. **appendChild:**
   - Appends a child node to a parent node.

```
parentElement.appendChild(newElement);
```

9. **removeChild:**
   - Removes a child node from its parent node.

```
parentElement.removeChild(childElement);
```

Methods for selecting elements based on different criteria

10. **setAttribute**:

- Sets the value of an attribute on the specified element.

```
element.setAttribute('class', 'newClass');
```

11. **addEventListener**:

- Attaches an event handler to the specified element for a specific event.

```
element.addEventListener('click', function() {
 console.log('Element clicked!');
});
```

Queryselector

querySelector : method is used to select the first element that matches a specified CSS selector within the document. It is part of the Document Object Model (DOM) API and provides a convenient way to access and manipulate DOM elements based on their CSS selectors.

An application programming interface (API) is a way for two or more computer programs or components to communicate with each other. It is a type of software interface, offering a service to other pieces of software

Examples

```
// Select the first element with the class "example"
var element = document.querySelector('.example');

// Do something with the selected element
element.style.color = 'red';

// Select the first paragraph element
var paragraph = document.querySelector('p');

// Select the element with the ID "myElement"
var myElement = document.querySelector('#myElement');

// Select the first element with the attribute "data-custom"
var customElement = document.querySelector('[data-custom]');

Note selecting Tags/Element Form Html Code
```

Examples

```
var elements = document.querySelectorAll('.example');

// Loop through the NodeList and do something with each element
elements.forEach(function(element) {
  element.style.fontSize = '16px';
});
```

Adding DOM ELEMENT

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Add DOM Element and CSS</title>
  <style>
    /* Define a CSS rule for the new element */
    .newElement {
      background-color: lightgreen;
      padding: 10px;
      margin-top: 10px;
    }
  </style>
</head>
<body>
<!--- Script — – ->
</html>
```

```
<script>
  // Create a new div element
  var newElement = document.createElement('div');
  // Set the text content of the new element
  newElement.textContent = 'New Element';
  // Add a class to the new element
  newElement.classList.add('newElement');
  // Apply additional styles using the style property
  newElement.style.fontSize = '18px';
  newElement.style.fontWeight = 'bold';
  // Append the new element to the body
  document.body.appendChild(newElement);
 </script>
```

Applying Inline Css In JS

```
<script>
  // Create a new div element
  var newElement = document.createElement('div');
  // Set the text content of the new element
  newElement.textContent = 'New Element';
  // Apply inline CSS styles using the style property
  newElement.style.backgroundColor = 'lightgreen';
  newElement.style.padding = '10px';
  newElement.style.marginTop = '10px';
  newElement.style.fontSize = '18px';
  newElement.style.fontWeight = 'bold';
  // Append the new element to the body
  document.body.appendChild(newElement);
 </script>
```

Why Queryselector

The choice between querySelector and other methods like getElementById, getElementsByClassName, or getElementsByTagName often depends on your specific use case and the flexibility you need in selecting elements from the DOM. Here are some reasons why you might prefer querySelector over other methods:

1. Versatility:

- querySelector is more versatile than other methods because it allows you to use any valid CSS selector. This flexibility is particularly useful when you need to select elements based on complex criteria, such as combinations of classes, attributes, or nested structures.

var element = document.querySelector('div.example > p');

2. Single Selection:

- querySelector is designed for selecting a single element. If you are sure that only one element matches your criteria, it provides a concise and direct way to retrieve that element without having to deal with arrays or NodeLists.

var element = document.querySelector('#myElementId');

3. Readability:

- Using CSS selectors can often make your code more readable and expressive. It allows you to specify the selection criteria in a way that is similar to how you would describe it in a stylesheet.

var element = document.querySelector('.example');

DOM Events

Events are things that happen in the system you are programming, which the system tells you about so your code can react to them.

For example, if the user clicks a button on a webpage, you might want to react to that action by displaying an information box. In this article, we discuss some important concepts surrounding events, and look at how they work in browsers. This won't be an exhaustive study; just what you need to know at this stage. MORE

Mouse Events:

- click: Triggered when a mouse button is clicked.
- dblclick: Triggered when a mouse button is double-clicked.
- mousedown: Triggered when a mouse button is pressed.
- mouseup: Triggered when a mouse button is released.
- mousemove: Triggered when the mouse pointer moves.
- mouseover: Triggered when the mouse pointer enters an element.
- mouseout: Triggered when the mouse pointer leaves an element.

DOM Events

Keyboard Events:

- keydown: Triggered when a key on the keyboard is pressed down.
- keyup: Triggered when a key on the keyboard is released.
- keypress: Triggered when a key that produces a character value is pressed.

Form Events:

- submit: Triggered when a form is submitted.
- change: Triggered when the value of an input element changes (e.g., text input or select box).
- input: Similar to change, triggered when the value of an input element changes.
- focus: Triggered when an element receives focus.
- blur: Triggered when an element loses focus.

Window Events:

- load: Triggered when the page finishes loading.
- resize: Triggered when the browser window is resized.
- scroll: Triggered when the document is scrolled.

DOM Events

Touch Events:

- touchstart: Triggered when a touch point is placed on the touch surface.
- touchmove: Triggered when a touch point is moved along the touch surface.
- touchend: Triggered when a touch point is removed from the touch surface.

Other Events

- Drag And Drop Event
- Media => Play Pause Event
- Custom Events(Using CustomEvent Constructor)

MORE

Trigger Function With HTML Event

```
<!-- HTML button that triggers the function on click -->
 <button id="myButton" onclick="myFunction()">Click me</button>

 <script>
   // JavaScript function to be called on button click
   function myFunction() {
     alert('Button clicked! Function executed.');
   }
 </script>
```

Trigger Function With JS Event

```
<!-- HTML element that triggers the event -->
 <button id="myButton">Click me</button>
 <script>
   // Function to be triggered
   function myFunction() {
     alert('Button clicked! Function executed.');
   }
   // Get a reference to the button element by its ID
   var myButton = document.getElementById('myButton');
   // Add an event listener to the button element
   myButton.addEventListener('click', myFunction);
 </script>
```

JQuery And Ajax

jQuery is a fast and lightweight JavaScript library that simplifies many common tasks in web development, such as DOM manipulation, event handling, and AJAX requests. Below, I'll provide some common use cases of jQuery along with examples:

Key features of jQuery include:

1. **DOM Manipulation:** jQuery provides a simplified syntax for selecting and manipulating HTML elements in the Document Object Model (DOM). It allows developers to easily update, add, or remove elements on a webpage.
2. **Event Handling**: jQuery simplifies the process of handling events like clicks, mouseovers, and keypresses. It provides a consistent and easy-to-use interface for binding and unbinding event handlers
3. **Ajax**: jQuery simplifies the process of making asynchronous HTTP requests using the AJAX technique. This allows developers to retrieve data from a server without requiring a full page refresh.
4. **Animation**: jQuery includes built-in methods for creating simple animations and effects, making it easier to add dynamic and visually appealing elements to a webpage
5. **Cross-browser Compatibility**: jQuery abstracts many browser-specific complexities, making it easier to write code that works consistently across different web browsers.

Library

👉 A collection of pre-written code that can be used to add functionality to a program.

OOP

👉 A programming paradigm based on objects, classes, and inheritance.

HOW TO CONNECT JQUERY

To use jQuery on a webpage, you typically include the jQuery library in your HTML file, either by downloading it and hosting it locally or by linking to a hosted version. Once included, you can use jQuery syntax to perform various tasks with less code compared to plain JavaScript

In the example, $(document).ready() ensures that the jQuery code runs after the HTML document has been fully loaded

```
<!DOCTYPE html>
<html lang="en">
<head>
  <!-- Include jQuery library -->
  <script
src="https://code.jquery.com/jquery-3.6.0.min.js">
</script>
</head>
<body>
  <!-- Your HTML content here -->

  <!-- Example: Use jQuery to hide an element with
ID "myElement" -->
  <script>
    $(document).ready(function(){
      $("#myElement").hide();
    });
  </script>
</body>
</html>
```

DOM MANIPULATION WITH JQUERY

Selecting Elements:

- Use the ***$()*** function to select elements using CSS-style
- selectors.

```
// Select element with ID "myElement"
var myElement = $("#myElement");

// Select all paragraphs
var allParagraphs = $("p");

// Select all elements with class "myClass"
var elementsWithClass = $(".myClass");
```

Modifying Content:

- Use methods like `text()`, `html()`, and `val()` to get or set the content of elements.

```
// Get the text content of an element
var textContent = $("#myElement").text();

// Set the HTML content of an element
$("#myElement").html("<p>New
content</p>");

// Get or set the value of an input field
var inputValue = $("#myInput").val();
```

DOM MANIPULATION WITH JQUERY

Changing CSS:

- Use the `css()` method to get or set CSS properties of elements.

```
// Change the background color of an element
$("#myElement").css("background-color", "blue");

// Get the current font-size of an element
var fontSize = $("#myElement").css("font-size");
```

Adding or Removing Elements:

- Use methods like **`append()`**, **`prepend()`**, **`after()`**, and **`remove()`** to add or remove elements.

```
// Append a new paragraph inside an element
$("#myElement").append("<p>New paragraph</p>");
```

```
// Remove an element
$("#toBeRemoved").remove();
```

Event Handling:

- Use methods like **`click()`**, **`on()`**, and **`bind()`** to handle events on elements.

```
// Add a click event handler to an element
$("#myButton").click(function(){
 alert("Button clicked!");
});

// Use the on() method for more flexibility
$("#myElement").on("mouseenter", function(){
 console.log("Mouse entered!");

});
```

Basic Animation Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Simple jQuery Animation</title>
  <style>
    body {
      margin: 0;
      padding: 0;
      display: flex;
      align-items: center;
      justify-content: center;
      height: 100vh;
      background-color: #f0f0f0;
    }
    #animatedBox {
      width: 100px;
      height: 100px;
      background-color: #3498db;
      display: none;
      position: relative;
      cursor: pointer;
```

```
</head>
<body>
  <div id="animatedBox"></div>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <script>
    $(document).ready(function () {
      // Fade in the box when the page loads
      $("#animatedBox").fadeIn(1000);

      // Add a click event handler for the box
      $("#animatedBox").click(function () {
        // Move the box to the right
        $(this).animate({
          left: '+=50px'
        }, 500);
      });
    });
  </script>
</body></html>
```

AJAX

AJAX (Asynchronous JavaScript and XML) is a technique used in web development to create dynamic and interactive user interfaces. It allows web pages to retrieve and send data to a server asynchronously, without requiring a full page reload. This enables the development of more responsive and interactive web applications.

Key components of AJAX include:

1. Asynchronous Operation: The "A" in AJAX stands for asynchronous, meaning that operations can be performed in the background without blocking the main thread of the application. This allows other tasks to continue while data is being fetched or sent.
2. XMLHttpRequest Object: AJAX relies on the XMLHttpRequest object, which is a browser API that enables communication with a server using HTTP or HTTPS. While the name suggests XML, modern AJAX requests often deal with other data formats like JSON.
3. Data Formats: While XML was traditionally associated with AJAX, modern applications often use JSON (JavaScript Object Notation) as the preferred data format due to its simplicity and ease of use with JavaScript. However, other formats like HTML, plain text, or XML can also be used.
4. Callback Functions: AJAX operations are asynchronous, meaning that the response from the server may not be immediately available. Callback functions are used to handle the server's response once it's received. These functions are specified in the AJAX request and are triggered upon successful completion or failure.
5. Cross-Origin Requests: AJAX requests are subject to the same-origin policy, which restricts requests to the same domain as the web page making the request. To overcome this limitation, techniques like Cross-Origin Resource Sharing (CORS) or JSONP (JSON with Padding) can be used.

common callback functions used in AJAX requests:

Callback functions are functions that are passed as arguments to other functions and are executed later, after the asynchronous operation finishes

1. ***`success`*:** This callback function is triggered when the AJAX request is successful. It takes the server's response as an argument. You can perform actions like updating the UI or processing the data within this function.

```
$.ajax({
 url: 'https://api.example.com/data',
 method: 'GET',
 success: function(data) {
 // Handle the successful response
 console.log(data);
 },
 // Other settings...

    });
```

common callback functions used in AJAX requests:

2. error: This callback function is triggered if there is an error during the AJAX request. It provides information about the error, such as the XMLHttpRequest object, the status, and the error message.

```
$.ajax({
 url: 'https://api.example.com/data',
 method: 'GET',
 error: function(xhr, status, error) {
 // Handle errors
 console.error(status + ': ' + error);
 },
 // Other settings...
});
```

common callback functions used in AJAX requests:

3.complete: This callback function is triggered regardless of whether the request was successful or resulted in an error. It's executed after the success or error callback functions.

```
$.ajax({
 url: 'https://api.example.com/data',
 method: 'GET',
 complete: function() {
 // This will be executed after success or error
 console.log('Request complete');
 },
 // Other settings...
});
```

By using these callback functions, you can control the flow of your code based on the outcome of the asynchronous AJAX operation. This allows you to update the UI, handle errors, or perform other actions in response to the server's response.

Basic example of an AJAX request using jQuery:

```
$.ajax({
  url: 'https://api.example.com/data',
  method: 'GET',
  dataType: 'json',
  success: function(data) {
    // Handle the successful response
    console.log(data);
  },
  error: function(xhr, status, error) {
    // Handle errors
    console.error(status + ': ' + error);
  }
});
```

In this example, the $.ajax() function is used to make an asynchronous GET request to a server endpoint ('https://api.example.com/data'). The success callback function is executed when the request is successful, and the server's response is passed as the data parameter. The error callback function is executed if there is an issue with the request.

What Is Graphics

Graphics Designing

Graphic design is a creative and visual communication discipline that involves the use of visual elements, such as images, typography, color, and layout, to convey a message or idea. Graphic designers use various tools and techniques to create visual content for a wide range of media, including print, digital, and multimedia platforms.

Principles

Graphic design principles are fundamental guidelines that help designers create visually appealing and effective designs. These principles provide a framework for arranging and organizing visual elements to convey a clear message or achieve a specific goal. Here are some key graphic design principles:

Balance:

- Balance refers to the distribution of visual weight in a design. There are two types of balance:
  - Symmetrical Balance: Elements are evenly distributed on either side of a central axis.
  - Asymmetrical Balance: Visual weight is distributed unevenly but is still balanced through careful placement of elements.

Contrast:

- Contrast involves highlighting the differences between elements to create visual interest and emphasize important aspects. This can include differences in color, size, shape, or texture.

Emphasis (Dominance):

- Emphasis is used to create a focal point in a design. This is where the viewer's eye is drawn first. It can be achieved through size, color, contrast, or positioning.

Unity (Harmony):

- Unity creates a sense of cohesion and completeness in a design. It ensures that all elements work together to convey a unified message. Consistent use of colors, fonts, and styles contributes to unity.

Principles

Repetition (Consistency):

- Repetition involves using consistent visual elements throughout a design. This helps establish a sense of rhythm and reinforces the overall theme. Repetition can include consistent colors, fonts, shapes, or patterns.

Proximity (Grouping):

- Proximity refers to the arrangement of related elements close to each other. Grouping related items helps convey relationships and makes the design more organized and easier to understand.

Hierarchy:

- Hierarchy involves organizing elements to indicate their importance. This helps guide the viewer's eye through the design, emphasizing key information. Size, color, and placement contribute to hierarchy.

Alignment:

- Alignment ensures that elements are visually connected and creates a sense of order. Proper alignment helps establish a clean and organized appearance. Elements can be aligned along edges or centered on a page.

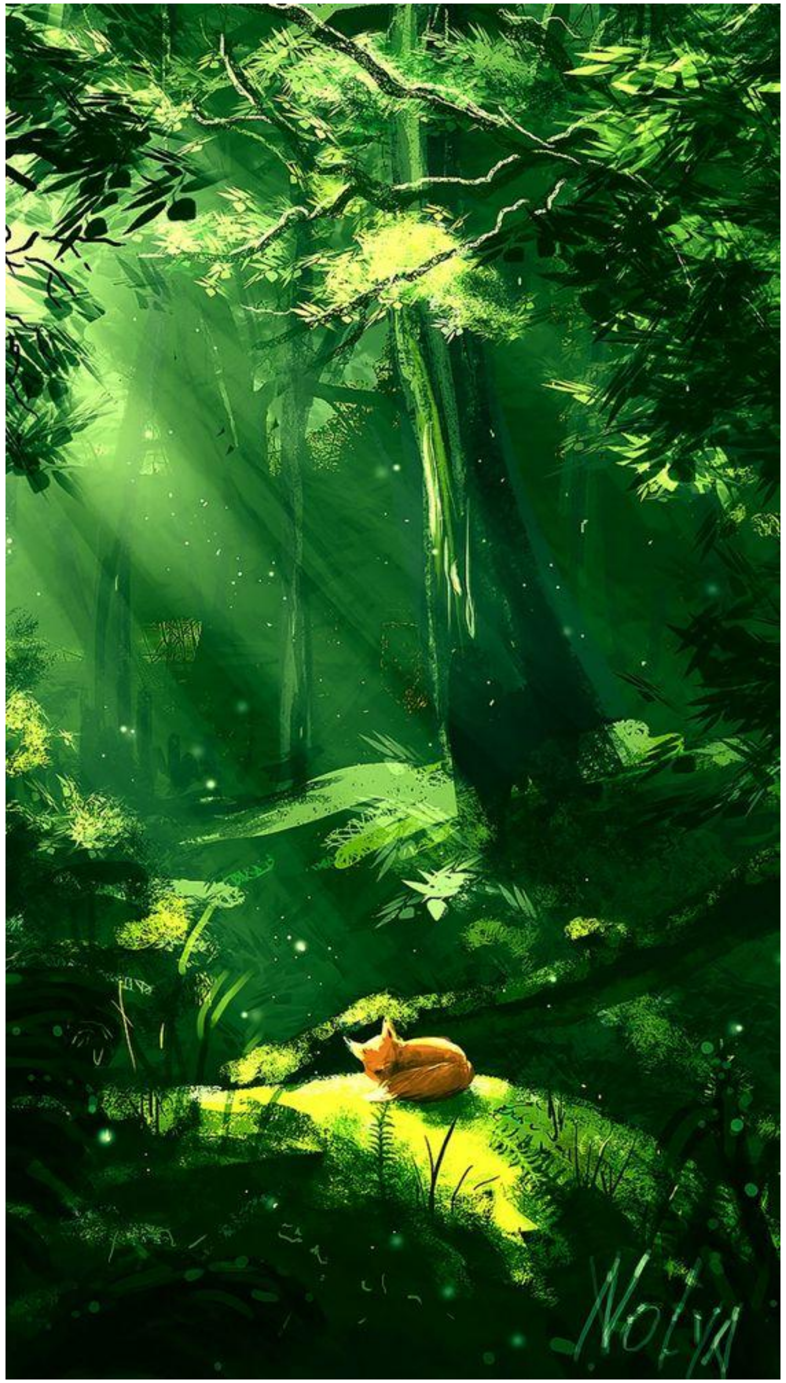
White Space (Negative Space):

- White space is the empty space around and between elements in a design. It helps reduce clutter, improve readability, and create a sense of balance. White space is a valuable design element in itself.

Typography:

- Typography involves the selection and arrangement of fonts and typefaces. Consistent and appropriate typography contributes to readability and reinforces the overall design style.

These principles are not strict rules but rather guidelines that designers use to make informed decisions. Effective graphic design often involves a combination of these principles to create visually pleasing and impactful designs that effectively communicate the intended message.



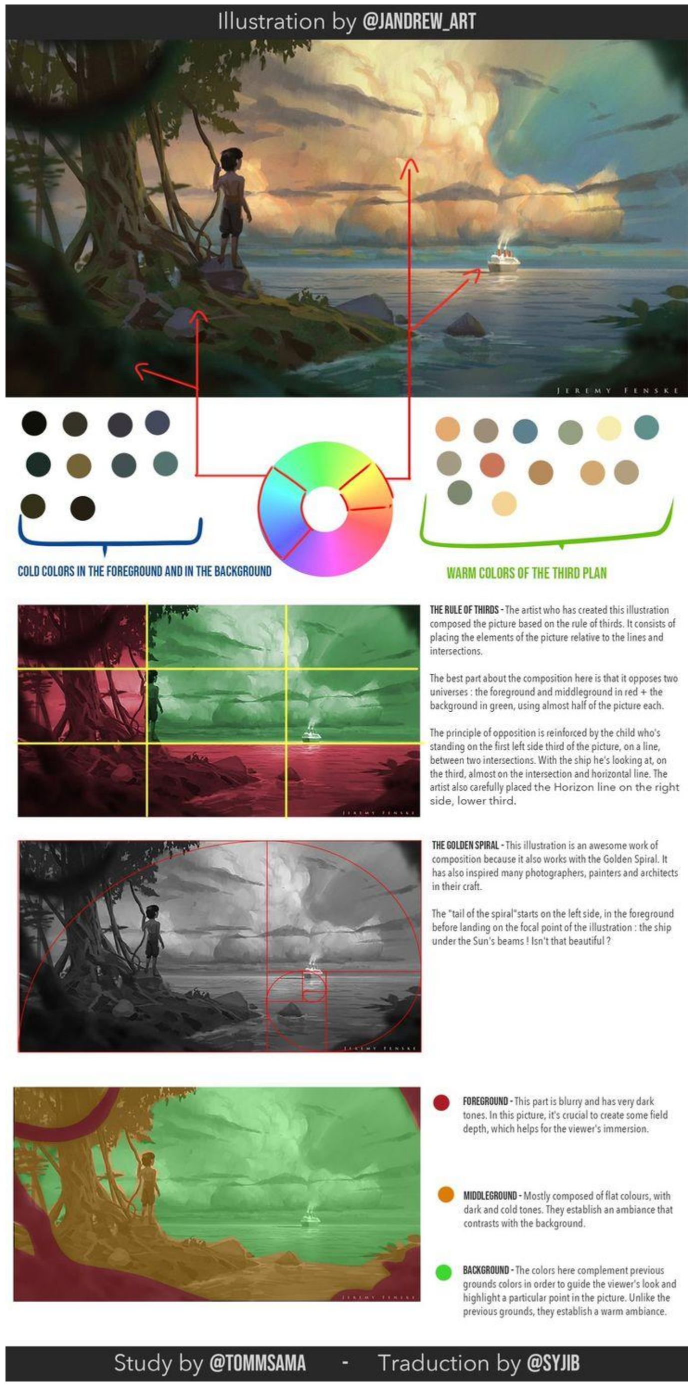


Illustration by @JANDREW_ART

JEREMY FENSKE
COLD COLORS IN THE FOREGROUND AND IN THE BACKGROUND
WARM COLORS OF THE THIRD PLAN

THE RULE OF THIRDS - The artist who has created this illustration composed the picture based on the rule of thirds. It consists of placing the elements of the picture relative to the lines and intersections.

The best part about the composition here is that it opposes two universes : the foreground and middleground in red + the background in green, using almost half of the picture each.

The principle of opposition is reinforced by the child who's standing on the first left side third of the picture, on a line, between two intersections. With the ship he's looking at, on the third, almost on the intersection and horizontal line. The artist also carefully placed the Horizon line on the right side, lower third.

THE GOLDEN SPIRAL - This illustration is an awesome work of composition because it also works with the Golden Spiral. It has also inspired many photographers, painters and architects in their craft.

The "tail of the spiral"starts on the left side, in the foreground before landing on the focal point of the illustration : the ship under the Sun's beams ! Isn't that beautiful ?

🟥 **FOREGROUND** - This part is blurry and has very dark tones. In this picture, it's crucial to create some field depth, which helps for the viewer's immersion.

🟧 **MIDDLEGROUND** - Mostly composed of flat colours, with dark and cold tones. They establish an ambiance that contrasts with the background.

🟩 **BACKGROUND** - The colors here complement previous grounds colors in order to guide the viewer's look and highlight a particular point in the picture. Unlike the previous grounds, they establish a warm ambiance.

Study by @TOMMSAMA - Traduction by @SYJIB

Type Of Image/Image Formats

In graphic design, there are several types of images, each serving different purposes and suitable for various contexts. Here are some common types of images:

Raster Images (Bitmap Images):

- Description: Composed of pixels, where each pixel contains information about color.
- Examples: JPEG, PNG, GIF, BMP.
- Suitable For: Photographs, detailed illustrations, complex graphics.
- Characteristics: Resolution-dependent, may lose quality when resized.

Vector Images:

- Description: Composed of mathematical paths and shapes rather than pixels.
- Examples: SVG (Scalable Vector Graphics), AI (Adobe Illustrator), EPS.
- Suitable For: Logos, icons, illustrations, scalable graphics.
- Characteristics: Resolution-independent, maintains quality at any size.

JPEG (Joint Photographic Experts Group):

- Description: Lossy compression format suitable for photographs and images with gradients.
- Suitable For: Photographs, web images.
- Characteristics: Supports millions of colors, variable compression levels, loss of quality with compression.

PNG (Portable Network Graphics):

- Description: Lossless compression format suitable for images with transparency.
- Suitable For: Web graphics, logos, images requiring transparency.
- Characteristics: Supports millions of colors, lossless compression, transparency.

Type Of Image/Image Formats

GIF (Graphics Interchange Format):

- Description: Lossless compression format often used for simple animations and images with limited colors.
- Suitable For: Simple animations, logos, images with few colors.
- Characteristics: Supports transparency, limited to 256 colors, animation support.

TIFF (Tagged Image File Format):

- Description: Lossless or lossy compression format suitable for high-quality images.
- Suitable For: Professional photography, print, archival.
- Characteristics: Supports multiple layers, various compression options, high-quality images.

BMP (Bitmap):

- Description: Uncompressed raster image format.
- Suitable For: Simple graphics, icons.
- Characteristics: Large file sizes, no compression, supports various color depths.

SVG (Scalable Vector Graphics):

- Description: XML-based vector image format.
- Suitable For: Scalable graphics, logos, icons.
- Characteristics: Editable in text editors, resolution-independent, supports interactivity.

RAW Images:

- Description: Unprocessed image data directly from the camera's sensor.
- Suitable For: Professional photography, editing.
- Characteristics: High-quality, retains more detail, requires processing.

Pixels:

- A pixel (short for "picture element") is the smallest unit of a digital image. It is a tiny square or dot that represents a single point of color in an image. The combination of pixels creates the overall visual representation of an image. Images are made up of millions of pixels, each having its own color value.

Lossy Compression:

- Lossy compression is a method of reducing the file size of an image or other multimedia files by discarding some of the data. During the compression process, certain details deemed less essential are removed or approximated, resulting in a smaller file size. The term "lossy" comes from the fact that some data is lost during compression, and subsequent decompression may not perfectly recreate the original.
- Examples of Lossy Compression Formats:
    - JPEG (Joint Photographic Experts Group) is a common lossy compression format used for photographs and images where some loss of quality is acceptable.

Lossless Compression:

- Lossless compression is a method of reducing the file size of an image or other data without sacrificing any data quality. In lossless compression, the original data can be perfectly reconstructed from the compressed version. This compression method is typically used when preserving the highest possible quality of the image is essential.
- Examples of Lossless Compression Formats:
    - PNG (Portable Network Graphics) is a common lossless compression format often used for images with transparency or when preserving image quality is crucial.
    - GIF (Graphics Interchange Format) is another lossless format used for simple graphics and animations.

Vector And Raster Graphics

Vector Image

They are resolution-independent and can be scaled to any size without losing quality.

Characteristics:

- Composed of points, lines, curves, and shapes.
- Scalable without loss of quality.
- Ideal for logos, icons, and illustrations.
- Common formats: SVG (Scalable Vector Graphics), AI (Adobe Illustrator), EPS.

Raster Images (Bitmap Images):

- also known as bitmap images, are made up of a grid of pixels. Each pixel contains information about color, and the overall image is composed of these individual pixels.
- *Characteristics:*
    - Composed of pixels.
    - Resolution-dependent (may lose quality when resized).
    - Ideal for photographs and detailed graphics.
    - Common formats: JPEG, PNG, GIF, BMP.

Comparison

- Scalability:
  - Vector: Scalable to any size without loss of quality.
  - Raster: Resolution-dependent, may lose quality when scaled up.
- File Size:
  - Vector: Generally smaller file sizes, especially for simpler graphics.
  - Raster: File size can be larger, especially for high-resolution images.
- Editing:
  - Vector: Easily editable, can be manipulated with precision.
  - Raster: Editing is more challenging, especially for complex edits, as changes affect individual pixels.
- Use Cases:
  - Vector: Ideal for logos, icons, illustrations, and any graphics that need to be resized frequently.
  - Raster: Ideal for photographs, detailed images, and graphics that do not need frequent resizing.
- Common Formats:
  - Vector: SVG, AI, EPS.
  - Raster: JPEG, PNG, GIF, BMP.

Color Palette

A color palette is a carefully selected set of colors chosen for a specific design, project, or artwork. It serves as a visual foundation, guiding the color choices within a cohesive and harmonious range. The use of a color palette is essential in design for several Reason

Consistency:

- A color palette ensures consistency in the visual identity of a design. By defining a set of colors, designers establish a unified look and feel throughout the project.

Branding:

- For businesses and brands, a color palette is a crucial element of branding. Consistent use of specific colors helps create brand recognition and reinforces the brand's personality and values.

Visual Hierarchy:

- A well-designed color palette aids in creating a visual hierarchy within a layout. Different colors can be used to emphasize or de-emphasize elements, guiding the viewer's attention to key information.

Emotional Impact:

- Colors evoke emotions and moods. By strategically selecting colors in a palette, designers can influence the emotional response of the audience. For example, warm colors may create a sense of energy and excitement, while cool colors may convey calmness and serenity.

Color Palette

Coherence and Harmony:

- A color palette ensures that all colors within a design work harmoniously together. This harmony contributes to a visually pleasing and professional appearance.

Accessibility:

- Designers need to consider accessibility for all users, including those with visual impairments. A thoughtfully chosen color palette helps create designs that are accessible and easily distinguishable.

Cultural and Contextual Considerations:

- Colors carry cultural and contextual meanings. A well-designed color palette takes into account the cultural associations of colors and aligns with the intended message or purpose of the design.

Versatility:

- A carefully curated color palette is versatile and can be adapted to various design elements and contexts. It provides flexibility while maintaining a cohesive visual identity.

Efficiency:

- Designing with a color palette enhances efficiency in the creative process. Designers can focus on making informed decisions within the predefined color range, streamlining the design workflow.

Print and Production Consistency:

- In print and production processes, having a consistent color palette helps ensure that the final output matches the designer's intentions. This is crucial for maintaining brand consistency across different mediums.

Adobe Color Wheel

Types of color Palette

Monochromatic Palette:

- Characteristics:
  - Consists of variations in lightness and saturation of a single color.
  - Provides a clean and elegant look.
- Applications:
  - Suitable for creating a calm and unified visual experience.
  - Often used in minimalist designs.

Analogous Palette:

- Characteristics:
  - Uses colors that are next to each other on the color wheel.
  - Creates a sense of harmony and cohesion.
- Applications:
  - Ideal for designs where a smooth transition between colors is desired.
  - Commonly used for nature-inspired themes.

Types of color Palette

Complementary Palette:

- Characteristics:
  - Uses colors that are opposite each other on the color wheel.
  - Creates a high contrast and vibrant look.
- Applications:
  - Adds excitement and energy to a design.
  - Use sparingly to avoid overwhelming the viewer.

Triadic Palette:

- Characteristics:
  - Uses colors evenly spaced around the color wheel.
  - Offers a balance between contrast and harmony.
- Applications:
  - Provides a dynamic and visually appealing composition.
  - Useful for creating vibrant and balanced designs.

Types of color Palette

Tetradic Palette (Double-Complementary):

- Characteristics:
  - Uses two pairs of complementary colors.
  - Offers a wide range of color choices.
- Applications:
  - Allows for diverse color combinations.
  - Suitable for complex designs with multiple elements.

Split-Complementary Palette:

- Characteristics:
  - Uses a base color and the two colors adjacent to its complementary color.
  - Offers a balance between contrast and harmony.
- Applications:
  - Creates a visually interesting composition with less tension than a complementary palette.
  - Suitable for designs that require contrast without being too overpowering.

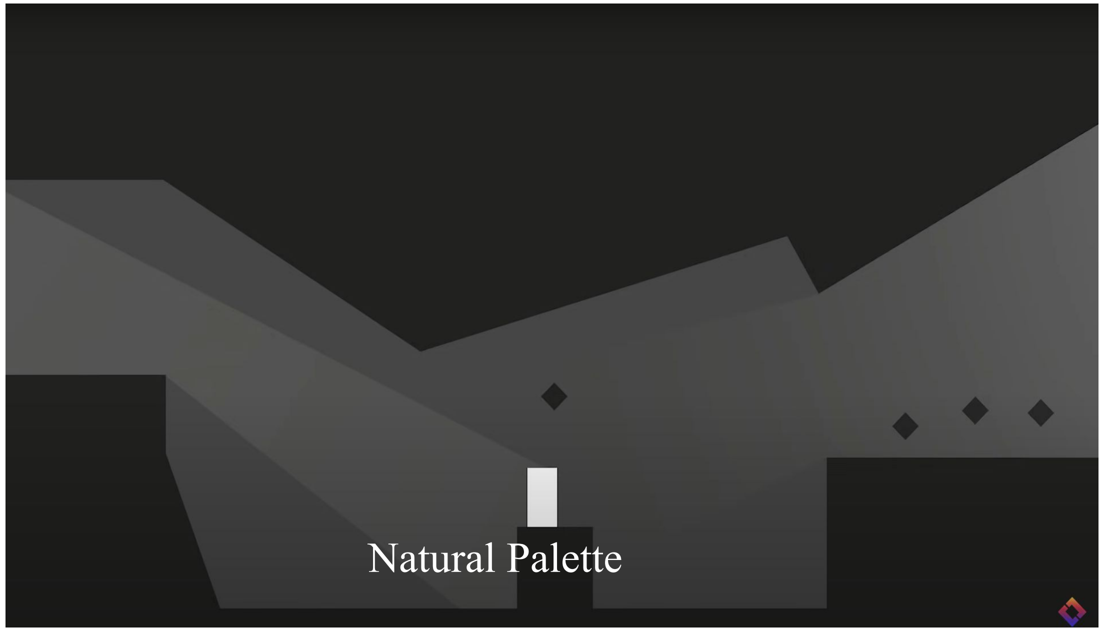
Types of color Palette

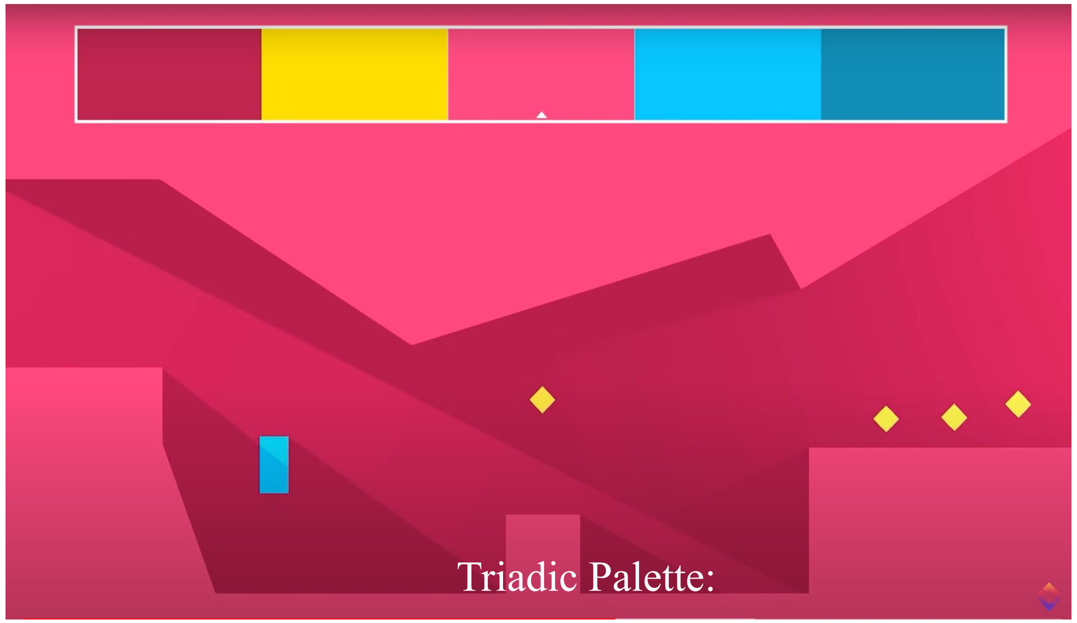
Square Palette (Rectangle or Tetradic):

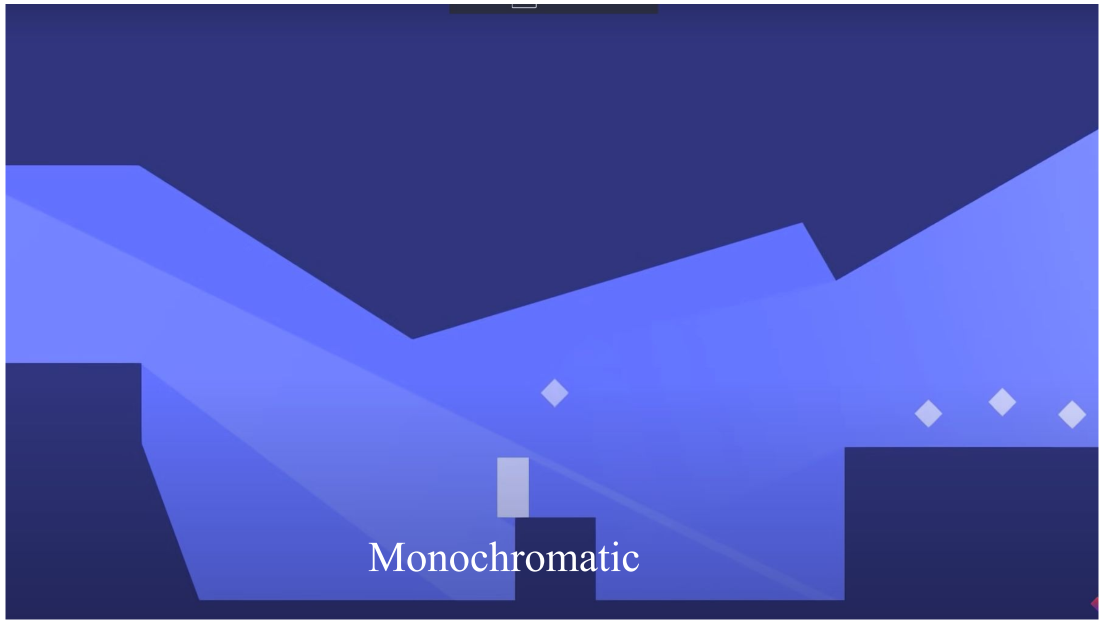
- Characteristics:
    - Uses four colors spaced evenly around the color wheel.
    - Provides a rich and diverse color scheme.
- Applications:
    - Ideal for designs that require multiple color variations.
    - Offers flexibility in creating visually engaging compositions.

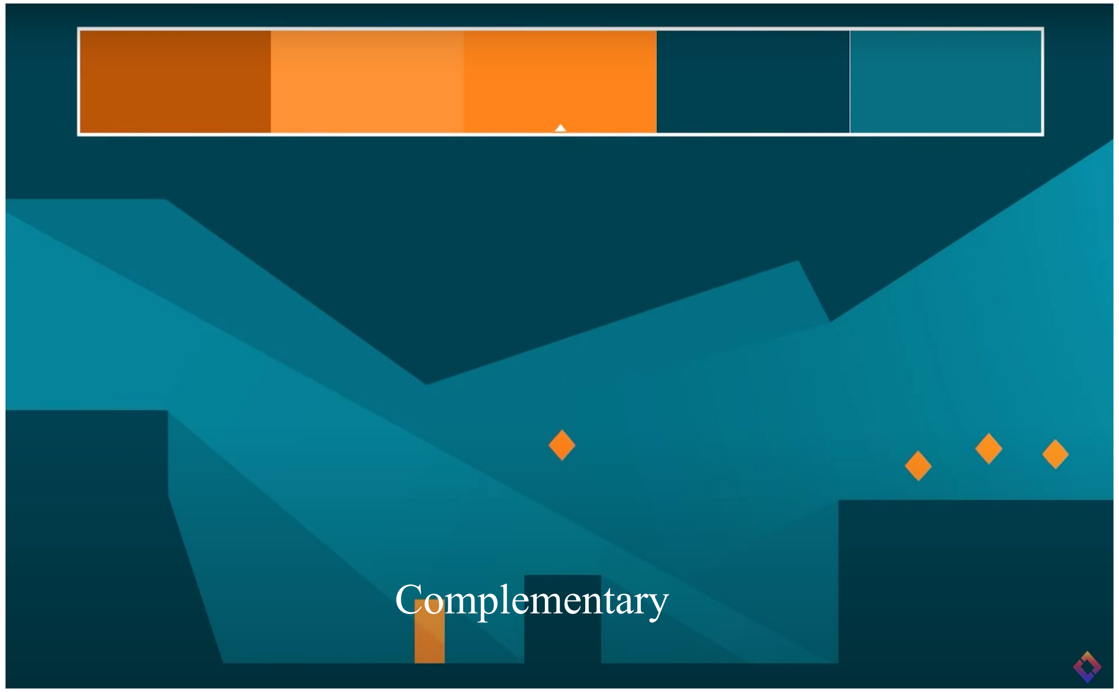
Neutral Palette:

- Characteristics:
    - Utilizes neutral colors such as grays, blacks, and whites.
    - Creates a subtle and sophisticated appearance.
- Applications:
    - Ideal for conveying a sense of elegance and simplicity.
    - Commonly used as a backdrop for highlighting other colors.


Natural Palette


Triadic Palette:


Monochromatic


Complementary

Photoshop

Photoshop is a popular and powerful graphics editing software developed by Adobe. It's widely used for a variety of tasks, including photo editing, graphic design, digital painting, and more. Here are some key features and tasks you can perform in Photoshop:

Basic Editing:

- Crop and resize images.
- Adjust brightness, contrast, and saturation.
- Correct color balance and levels.

Layers:

- Work with layers to organize and manipulate elements independently.
- Apply blending modes to control how layers interact.

Selections:

- Use various selection tools (lasso, magic wand, etc.) to isolate parts of an image.
- Refine selections with tools like the Refine Edge or Select and Mask.

Text Editing:

- Add and format text with a variety of fonts and styles.
- Apply text effects like shadows, gradients, and warp text.

Filters and Effects:

- Apply artistic filters and effects to enhance or transform images.
- Use smart filters for non-destructive editing.

Photoshop

Retouching:

- Remove imperfections using tools like the Healing Brush, Clone Stamp, and Spot Healing Brush.
- Dodge and burn to adjust exposure locally.

Drawing and Painting:

- Create digital illustrations using brushes and painting tools.
- Use the Pen tool for precise paths and shapes.

3D Modeling:

- Photoshop has basic 3D modeling capabilities.

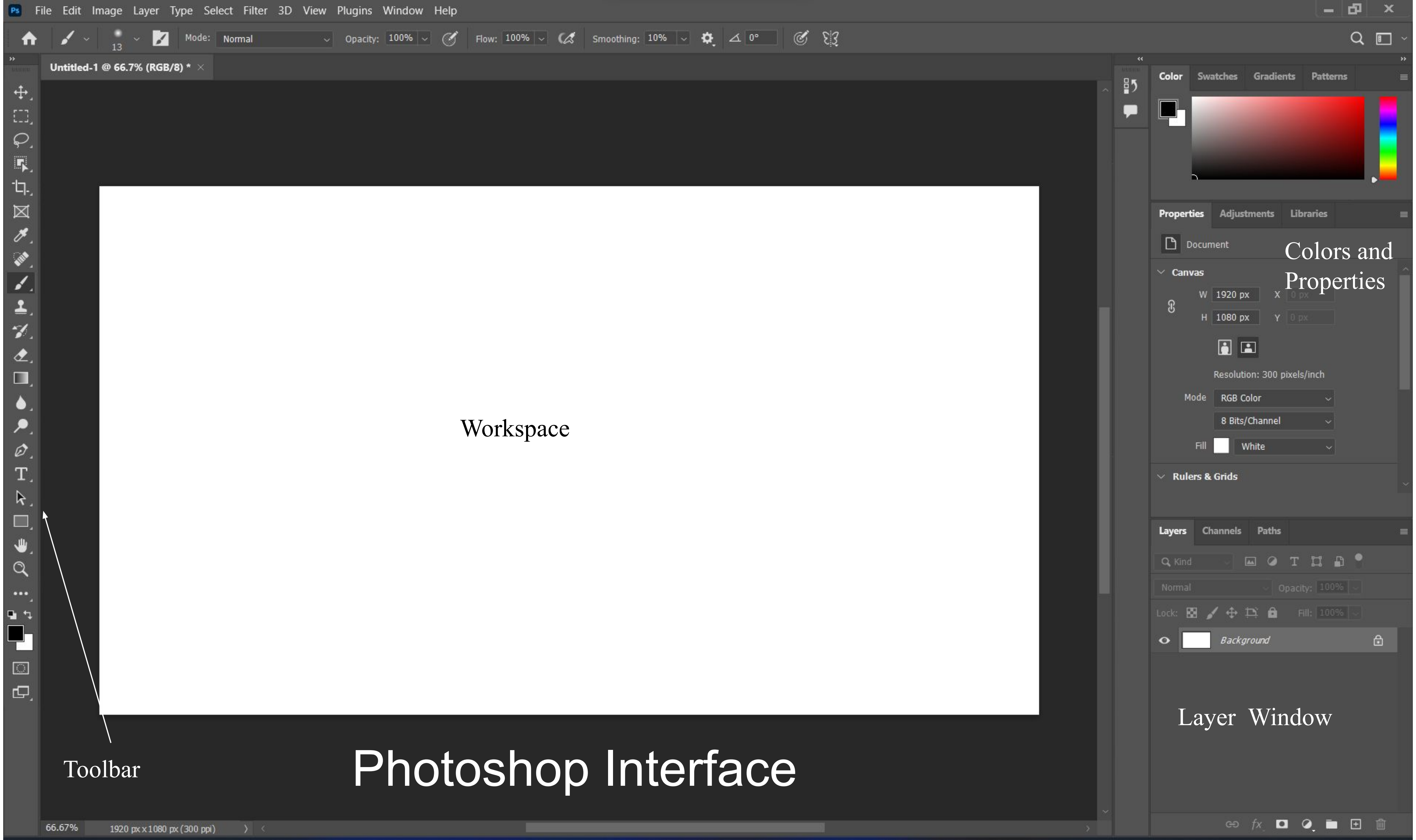
Automation:

- Record and play back actions for repetitive tasks.
- Use batch processing to apply actions to multiple files.

Integration:

- Seamless integration with other Adobe products like Illustrator and InDesign.

Remember, Photoshop can be complex, so it's a good idea to explore tutorials and practice regularly to become more proficient. Adobe offers a variety of learning resources, and there are numerous tutorials available online to help you master Photoshop's features.



Photoshop Interface

Photoshop file format

Adobe Photoshop uses several file formats to save and store different types of information. The primary file format for saving edited images is the PSD (Photoshop Document) format. Here are some key file formats associated with Adobe Photoshop:

PSD (Photoshop Document):
- The native file format of Adobe Photoshop.
- Supports layers, masks, transparency, and other advanced features.
- Preserves all editing capabilities, allowing you to continue working on the file with all its layers intact.

PSB (Photoshop Big):
- Similar to PSD but designed for larger files that exceed the PSD file size limit.
- PSB is used for high-resolution images or images with a large number of layers.

JPEG (Joint Photographic Experts Group):
- A widely used compressed image format.
- Lossy compression, which means some image data is discarded to reduce file size.
- Suitable for web and general use but not ideal for images that require extensive editing due to the loss of data.

PNG (Portable Network Graphics):
- A lossless image format that supports transparency.
- Suitable for images that require a transparent background.
- Commonly used for web graphics.

Photoshop file format

GIF (Graphics Interchange Format):

- Supports simple animations and transparency.
- Limited to 256 colors, making it suitable for simple graphics and animations.
- Widely used for web graphics.

TIFF (Tagged Image File Format):

- A versatile and widely supported file format.
- Supports both lossless and lossy compression.
- Suitable for high-quality printing and professional photography.

PDF (Portable Document Format):

- Photoshop can save files as PDFs, preserving layers and other editing features.
- Useful for creating documents with both text and images.

RAW:

- Photoshop supports various camera RAW formats.
- RAW files contain unprocessed image data directly from a digital camera's sensor.
- Useful for advanced editing and adjustments.

When saving a file in Photoshop, you can choose from these formats based on your intended use and the features you want to preserve. Keep in mind that using the native PSD format is recommended during the editing process to retain all layers and editing capabilities. Once your editing is complete, you can save or export to a different format depending on your needs.

Canvas Size:

- Definition: Canvas size refers to the dimensions (width and height) of the working area in Photoshop.
- Adjustment: You can change the canvas size without resampling the image content, which means you're adjusting the size of the workspace, but not the actual image pixels.
- Use Cases: Useful when you want to add more space around your image or crop part of it without changing the actual image size.

Image Size:

- Definition: Image size refers to the dimensions (width and height) of the actual image in pixels or other units.
- Adjustment: Changing the image size may involve resampling, where Photoshop adds or removes pixels to fit the new dimensions.
- Use Cases: You might change the image size when preparing it for different output purposes, like printing or web display. Be cautious about excessive upsizing, as it can result in a loss of image quality.

Resolution:

- Definition: Resolution is the number of pixels per unit of measurement (e.g., pixels per inch - PPI or pixels per centimeter - PPC).
- Adjustment: Changing the resolution affects the density of pixels in an image. Higher resolution generally means more detail but may not affect the image's dimensions on the screen.
- Use Cases: Important for print work, where a higher resolution is generally required for better print quality. For web or screen use, a lower resolution is often sufficient.

Undo And History Panel

The "Undo" command and the "History" panel are essential features that allow you to correct mistakes, step backward through your actions, and review the sequence of changes made to your document. Here's an explanation of each:

Undo Command:

- Shortcut: Ctrl + Z (Windows) or Command + Z (Mac)
- Function: The Undo command reverses the last action you performed. You can use it multiple times to step back through a series of recent changes.
- Limitations: The number of undo levels is configurable in Photoshop preferences, but excessive undoing may result in loss of earlier history states.

History Panel:

- Access: Window > History (or press Alt + F9)
- Function: The History panel provides a chronological list of actions and changes made during your Photoshop session. Each item in the history panel is a snapshot of your document at a particular state.
- Navigation: You can click on any history state to revert your document to that specific point. This allows you to jump back and forth between different stages of your editing process.
- Limitations: The number of history states is also configurable in Photoshop preferences. Be mindful that too many history states can consume a significant amount of RAM.

How To Use History Panel

How to Use Undo and the History Panel:

- Undo (Ctrl + Z / Command + Z):
  - Press the shortcut to undo the last action.
  - Press it multiple times to continue stepping back through your recent actions.
- History Panel:
  - Open the History panel from the Window menu.
  - Click on any state in the history panel to revert your document to that specific point.
  - You can also use the "Step Backward" and "Step Forward" buttons in the History panel to navigate through the history states.

Tips:

- Snapshots: In the History panel, you can create snapshots of your document at specific points. This allows you to return to those points even after performing other actions.
- History Brush: You can use the History Brush tool in combination with the History panel to selectively paint or erase certain areas based on the history states.

These features are crucial for maintaining flexibility and correcting errors during your Photoshop workflow.

Saving In Photoshop

Saving a document in Adobe Photoshop involves different options depending on your needs. Here are the main ways to save a document:

Save (Ctrl + S / Command + S):

- Use this command to save the current document to the same file and location.
- If you're saving for the first time, Photoshop will prompt you to choose a location, file name, and format (usually PSD).

Save As (Ctrl + Shift + S / Command + Shift + S):

- Use this command to save a copy of the document or to save it with a new name or in a different location.
- Allows you to choose the file format and adjust settings.

Export As (Alt + Shift + Ctrl + W / Option + Shift + Command + W):

- Use this option to export your document in various formats suitable for web use.
- Provides options for file type, quality, and other settings.

Export > Quick Export as PNG or JPG:

- Found in the "Export" submenu, this allows you to quickly export your document as either a PNG or JPG file.

Always consider the requirements of the project and the platforms where the image will be used when choosing the file format and settings.

Saving In Photoshop

.

Remember that if you are working with layers, transparency, and other advanced features, it's often a good idea to save your work as a PSD (Photoshop Document) to preserve all the editable information. For final output, you might save or export copies in formats like JPEG, PNG, or others, depending on your intended use (print, web, etc.).

Photoshop Tools

Adobe Photoshop comes equipped with a diverse range of tools that cater to various aspects of image editing and manipulation. Here is an overview of some essential tools available in Photoshop:

Selection Tools:

- Marquee Tools (M): Make rectangular or elliptical selections.
- Lasso Tools (L): Draw freeform or polygonal selections.
- Magic Wand (W): Selects pixels with similar colors.

Crop and Slice Tools:

- Crop Tool (C): Cuts out a selected area or adjusts the canvas size.
- Slice Tool (C): Divides an image into sections for web optimization.

Painting and Drawing Tools:

- Brush Tool (B): Paints with a selected foreground color.
- Pencil Tool (B): Draws hard-edged lines.
- Eraser Tool (E): Erases parts of an image.
- Paint Bucket (G): Fills an area with the selected foreground color.

Photoshop Tools

Retouching Tools:

- Clone Stamp (S): Copies one area of an image to another.
- Healing Brush (J): Fixes imperfections while preserving texture.
- Spot Healing Brush (J): Quickly removes blemishes.

Text Tools:

- Horizontal Type Tool (T): Adds horizontal text.
- Vertical Type Tool (T): Adds vertical text.
- Text Mask Tools (T): Creates a text mask.

Transform Tools:

- Move Tool (V): Moves and arranges layers.
- Rotate Tool (R): Rotates the canvas or a selected layer.
- Scale Tool (S): Resizes the canvas or a selected layer.

Gradient and Fill Tools:

- Gradient Tool (G): Creates smooth transitions between colors.
- Paint Bucket (G): Fills an area with a color or pattern.

Photoshop Tools

Navigation Tools:

- Hand Tool (H): Moves the image within the window.
- Zoom Tool (Z): Zooms in or out of the image.

Eyedropper and Color Tools:

- Eyedropper Tool (I): Samples a color from the image.
- Color Sampler Tool (I): Measures color values.
- Foreground and Background Color (D): Resets to default black and white, or allows setting custom colors.

3D Tools:

- 3D Object Tools: Used for working with 3D models in Photoshop.

Layers

In Adobe Photoshop, layers are fundamental to the image editing process. Layers allow you to separate and organize different elements within a document, making it easier to manipulate and control specific parts of an image independently. Here's an overview of key concepts related to layers in Photoshop:

Layer Basics:

- Background Layer: When you create a new document, it usually starts with a background layer. This layer is locked by default but can be converted to a regular layer for more flexibility.

Creating and Managing Layers:

- New Layer: You can create a new layer by clicking on the "New Layer" button at the bottom of the Layers panel.
- Duplicate Layer: Duplicates the selected layer, creating an identical copy.
- Delete Layer: Removes the selected layer from the document.
- Layer Visibility: The eye icon next to each layer controls its visibility. Clicking it toggles the layer on and off.

Layer Types:

- Text Layers: Created when you add text using the Text tool. Allows for easy text editing.
- Shape Layers: Created when you draw shapes or use the Shape tools. Vector-based layers.
- Adjustment Layers: Allow you to make non-destructive adjustments to the layers beneath them without changing the original pixels.
- Smart Objects: Layers that contain raster or vector images, preserving their original quality. Useful for scaling without loss of detail.

Layers

Layer Order:

- Layer Stacking Order: The order of layers in the Layers panel determines how they overlap in the image. The top layer appears in front of the layers below.

Layer Styles:

- Layer Styles: Add effects like drop shadows, glows, and strokes to layers.
- Blending Modes: Control how a layer blends with the layers beneath it.

Grouping Layers:

- Group Layers: Combine multiple layers into a group to organize and manage them collectively.

Opacity and Fill:

- Opacity: Adjusts the transparency of a layer.
- Fill: Adjusts the transparency of the layer content but not the layer effects.

Linking Layers:

- Link Layers: Links multiple layers together, so they move or transform simultaneously.

Layer Masks:

- Layer Masks: Allow you to hide or reveal portions of a layer without permanently deleting pixels.

Adjustment Layers:

- Adjustment Layers: Make color and tonal adjustments that affect all the layers below them without altering the original pixel data.

Transform

In Adobe Photoshop, "transform" refers to the ability to modify the size, orientation, and shape of a selected element within an image. The Transform command allows you to make changes to a layer, selection, or path. There are various transformation options available, providing flexibility and control over the appearance of elements in your document. The primary transformations include:

Scale:

- Description: Changes the size of the selected element.
- How to Use: Activate the Transform command (Ctrl + T / Command + T), then drag the corner handles while holding down the Shift key to maintain proportions.

Rotate:

- Description: Rotates the selected element around a specified point.
- How to Use: Activate the Transform command, move the cursor outside the selection, and click and drag to rotate. Holding down the Shift key constraints rotation to 15-degree increments.

Skew:

- Description: Shifts the selected element horizontally or vertically.
- How to Use: Activate the Transform command, right-click, and choose "Skew." Drag the corner handles to skew the selection.

Distort:

- Description: Allows you to pull the corners of a selection in different directions independently.
- How to Use: Activate the Transform command, right-click, and choose "Distort." Drag the corner handles to distort the selection.

Transform

Perspective:

- Description: Adjusts the perspective of a selection as if viewing it from an angle.
- How to Use: Activate the Transform command, right-click, and choose "Perspective." Drag the corner handles to adjust the perspective.

Warp:

- Description: Distorts the selected element based on a grid.
- How to Use: Activate the Transform command, right-click, and choose "Warp." Adjust the control points on the grid to warp the selection.

Free Transform:

- Description: Combines multiple transformations into one operation, allowing you to scale, rotate, skew, and more.
- How to Use: Activate the Free Transform command (Ctrl + T / Command + T), then use the corner handles and right-click for additional options.

Filters

In Adobe Photoshop, filters are powerful tools that allow you to apply various effects and adjustments to images, altering their appearance or adding creative elements. Filters can be applied to entire images or specific layers, providing a wide range of possibilities for enhancing or transforming your designs. Here are some common types of filters and their applications:

1. Filter Categories:

- Filter Gallery: A collection of artistic and texture filters that can be applied to images.
- Blur Filters: Soften or blur images for creative or corrective purposes.
- Sharpen Filters: Enhance the clarity and sharpness of images.
- Noise Filters: Add or reduce noise in images to achieve specific effects.
- Distort Filters: Warp or transform images in various ways.
- Pixelate Filters: Create pixelated or mosaic effects.
- Stylize Filters: Apply artistic and stylized effects.
- Render Filters: Generate complex textures, patterns, or 3D effects.
- Adjustment Filters: Make tonal adjustments to images.

Filters

2. Applying Filters:

- Select the Layer: Before applying a filter, select the layer or area of the image you want to modify.
- Filter Menu: Go to "Filter" in the top menu to access the various filter categories.
- Filter Gallery: Located within the Filter menu, it provides a visual interface for applying multiple filters at once.

****3. Smart Filters:**

- Description: Smart Filters are non-destructive filters that can be applied to Smart Objects, allowing for adjustments even after the filter is applied.
- Benefits: Maintain flexibility and editing capabilities without permanently altering the original image.

****4. Filter Effects Examples:**

- Gaussian Blur: Softens an image or creates a shallow depth of field.
- Motion Blur: Adds a directional blur, simulating motion.
- Posterize: Reduces the number of color tones, creating a poster-like effect.
- Oil Paint: Applies an oil painting effect to the image.
- Emboss: Creates a raised or engraved appearance by emphasizing edges.
- Lens Flare: Simulates the appearance of lens flare from light sources.

Filters

****5. Custom Filters and Plugins:**

- Third-Party Plugins: Photoshop supports third-party filters and plugins, expanding the range of available effects.
- Filter Forge, Nik Collection, etc.: Examples of popular third-party filter sets.

****6. Filter Masks:**

- Description: You can use layer masks with filters to control where the filter effect is applied within an image.
- Application: Useful for selectively applying filters to specific areas.

****7. Filter Presets:**

- Description: Some filters come with presets that offer predefined settings for achieving specific looks.
- Customization: After applying a preset, you can further customize the filter settings.

****8. Filter Effects in Video:**

- Description: Filters can also be applied to video layers in Photoshop, allowing for dynamic effects in video projects.
- Timeline Panel: Use the timeline panel for video editing and filter application.

Filters in Photoshop provide a wide array of creative possibilities, from subtle enhancements to dramatic transformations. Experimenting with different filters and their settings can lead to unique and visually appealing results in your designs and images.