# **Disclaimer:**

COPYRIGHT NOTICE

©All content, materials, and intellectual property contained within this document, including but not limited to text, images, diagrams, graphics, code, and any other original works, are protected by copyright laws and international treaties. This intellectual property is the sole and exclusive property of SRIKANTH TEKUMUDI

Unauthorized use, reproduction, distribution, modification, or transmission of any portion of this document, in any form or by any means, electronic or mechanical, without the prior written permission of SRIKANTH TEKUMUDI, is strictly prohibited and may result in severe civil and criminal penalties.

By accessing, viewing, or using this document, you acknowledge and agree to abide by all applicable copyright laws and the terms and conditions outlined herein. This copyright notice serves as a legal warning and reminder that the unauthorized use of this intellectual property is strictly prohibited and will be vigorously enforced.

Thank you for respecting the intellectual property rights of SRIKANTH TEKUMUDI

# 

# **1)differences Between HTML and HTML5**

1)HTML5 has audio and video support

2)Canvas,SVG and other graphic are supported

int HTML5

3)HTML5 more mobile friendly

4)Doctype declarations is simple

5)Allowing Drag and Drop effects

6)Allows JS to run in background.Possible due to Js Web Worker API in HTML5

7)New elements like nav,header,aside,article,main,<section>

# **2)What are new tags in media elements in HTML5?**

<audio> - This tag is used to embed audio content into a web page.

<video> - This tag is used to embed video content into a web page.

<source> - This tag is used to specify the URL of the media file and its type.

<track> - This tag is used to provide timed text tracks for video and audio elements.

<audio> and <video> attributes - New attributes were introduced for <audio> and <video> tags to specify controls, autoplay, loop, and other features.

These new tags and attributes have made it easier to add and manipulate media elements in web pages without relying on third-party plugins or tools.

# **3)Drag and Drop?**

Implementing drag and drop functionality in HTML5 is relatively simple. Here are the basic steps:

Add the draggable attribute to the element you want to make draggable. For example, if you want to make an image draggable, you can add the attribute to the <img> tag like this:

<img src="image.jpg" draggable="true">

Add event listeners for the dragstart and dragover events. The dragstart event fires when the user starts dragging an element, and the dragover event fires when the user is dragging an element over a drop target.

document.addEventListener("dragstart", function(event) {

// code to handle dragstart event

});

document.addEventListener("dragover", function(event) {

// code to handle dragover event

});

Specify the drop target by adding event listeners for the dragenter and drop events. The dragenter event fires when the user drags an element over a drop target, and the drop event fires when the user drops the element on the target.

document.addEventListener("dragenter", function(event) {

// code to handle dragenter event

});

document.addEventListener("drop", function(event) {

// code to handle drop event

});

In the event handlers, you can use the dataTransfer object to access the data being dragged and dropped. For example, you can use the setData() method to set the data to be transferred and the getData() method to retrieve it.

document.addEventListener("dragstart", function(event) {

event.dataTransfer.setData("text/plain", "Hello World!");

});

document.addEventListener("drop", function(event) {

var data = event.dataTransfer.getData("text/plain");

console.log(data); // outputs "Hello World!"

});

That's it! With these basic steps, you can implement drag and drop functionality in your web applications.

# **4)What are the new Form input elements in HTML5?**

<input type="email">: This input element is used for email addresses and can validate that the entered text is in the correct email format.

<input type="url">: This input element is used for URLs and can validate that the entered text is in the correct URL format.

<input type="tel">: This input element is used for telephone numbers and can validate that the entered text is in the correct phone number format.

<input type="search">: This input element is used for search queries and can provide a search-specific keyboard on mobile devices.

<input type="color">: This input element is used for selecting colors and displays a color picker.

<input type="range">: This input element is used for selecting a value within a range, and can display a slider control.

<input type="date">, <input type="time">, and <input type="datetime-local">: These input elements are used for selecting dates, times, or both, depending on the type.

<input type="number">: This input element is used for entering numeric values and can display a numeric keypad on mobile devices.

<input type="file">: This input element is used for uploading files from the user's device.

# **5)Explain the concept of web storage of HTML5?**

Web storage is a feature introduced in HTML5 that allows web applications to store data locally in a user's browser. There are two types of web storage: localStorage and sessionStorage.

localStorage allows developers to store data without an expiration date. This means that the data will persist even after the browser is closed and reopened. The stored data can be accessed by any page within the same domain.

sessionStorage, on the other hand, allows developers to store data for a single session. This means that the data will be available only while the browser is open and the session is active. Once the session ends or the browser is closed, the data is cleared.

Both localStorage and sessionStorage provide a simple key-value storage mechanism using the setItem(), getItem(), and removeItem() methods of the Storage object.

# **6)Explain Geolocation API in HTML5**

The Geolocation API is a feature introduced in HTML5 that allows web applications to retrieve a user's geographical location information, such as latitude and longitude, directly from the browser. This information can be used to provide location-based services, such as finding nearby restaurants or displaying maps.

The Geolocation API works by accessing various sources of location information, such as GPS, Wi-Fi, and IP address. The browser prompts the user for permission to access their location, and once the user grants permission, the browser determines the location using the available sources.

The Geolocation API provides a simple API for developers to access the users location information, using the navigator.geolocation object. The getCurrentPosition() method is used to retrieve the user's current location, while the watchPosition() method is used to continuously monitor the user's location and provide updates

# **7)*List page structure element of HTML5***

HTML5 introduced several new structural elements that can be used to create well-organized and accessible web pages. Some of the main page structure elements in HTML5 are:

<header>: Used to define the header section of a web page.

<nav>: Used to define a block of navigation links.

<main>: Used to define the main content of a web page.

<article>: Used to define a self-contained article or piece of content on a web page.

<aside>: Used to define a block of content that is related to the main content but not necessarily part of it.

<section>: Used to define a section of a web page.

<footer>: Used to define the footer section of a web page.

<h1> to <h6>: Used to define headings on a web page.

<figure> and <figcaption>: Used to define a figure and its caption.

# **8)What is Application Cache in HTML5 and why is it used?**

Application Cache, also known as AppCache, is a feature in HTML5 that allows web developers to create web applications that can be accessed offline. It allows users to continue using a web application even when they are not connected to the internet.

AppCache works by allowing developers to specify a list of files that should be cached locally by the user's browser. These files can include HTML, CSS, JavaScript, images, and other resources needed for the application to run. When a user accesses the web application while online, their browser will download and cache these files. When the user goes offline, their browser will use the cached files to continue running the application.

AppCache is used to improve the user experience of web applications by allowing them to be used offline. This is especially useful for mobile applications where internet connectivity may be spotty or slow. By using AppCache, developers can ensure that their application remains accessible and functional, regardless of the user's internet connection.

Advantages:

Offline Browsing

Speed

# **9)What is the difference between HTML elements and tags**

An HTML tag is a piece of code that specifies a particular type of HTML element. Tags are used to define the structure and content of an HTML document. Tags are surrounded by angle brackets, such as <html>, <body>, and <p>. They usually come in pairs, with an opening tag and a closing tag that surround the content of the element, like this: <p>This is a paragraph.</p>

So, to summarize, a tag is the syntax used to define an HTML element, while an element refers to the entire content between the opening and closing tags, including the tags themselves and any content in between.

# **10) What are tags and attributes in HTML?**

HTML attributes, on the other hand, provide additional information about HTML elements. They are added to the opening tag of an element and are used to modify the element's behavior or appearance. For example, the href attribute is used to specify the URL of a hyperlink, and the src attribute is used to specify the URL of an image. Here is an example of an anchor tag (<a>) with an href attribute:

HTML tags are used to define the structure and content of an HTML document. They are surrounded by angle brackets, such as <html>, <body>, and <p>. HTML tags usually come in pairs, with an opening tag and a closing tag that surround the content of the element. For example, a paragraph element can be defined using the <p> tag as follows

# **11) Inline and block elements in HTML5?**

Inline elements are displayed within a line of text and do not start on a new line.

Examples of inline elements include <span>, <a>, <img>, <strong>, and <em>.

Inline elements can be nested inside block elements, but not other inline elements.

Width and Height: Inline elements adjust their width and height automatically to fit their content, so specifying these properties may have no effect.

Margin and Padding: Setting margin or padding on inline elements may not have the expected results because they won't create a block around the content. Instead, they will add space around the element, affecting the layout of other elements on the same line.

\*\*Block elements\*\*

Block elements start on a new line and take up the full width available.

Examples of block elements include <div>, <h1> to <h6>, <p>, <ul>, <ol>, and <li>.

Block elements can contain other block elements and/or inline elements.

Here's an example of the difference between inline and block elements:

difference between canvas and SVG?

50. Explain new input types provided by HTML5 for forms?

Explain the layout of HTML

# **12)what is box modal**

box modal is essentially a box that wraps around every HTML element.

it consists of margin,border,padding,,content

The box model is a fundamental concept in CSS that describes how elements are

laid out on a web page. In the box model, every HTML element is considered a rectangular box that contains the content, padding, border, and margin.

The box model can be broken down into four parts:

Content: This is the actual content of the HTML element, such as text, images, or other HTML elements.

Padding: This is the space between the content and the border. It is typically used to create space around the content and improve readability.

Border: This is the line that surrounds the padding and content. It is used to define the boundary of the element and can be styled with different colors, widths, and styles.

Margin: This is the space outside the border. It is typically used to create space between elements and improve the overall layout of the page.

All of these parts together determine the total size of the HTML element. The width and height properties of an element refer to the total size, including content, padding, and border, but not margin.

It's important to understand the box model when creating layouts with CSS because the size and positioning of elements depend on it. For example, if you set the width of an element to 100px and add 10px of padding and a 2px border, the total width of the element will be 124px (100px + 10px + 10px + 2px + 2px). This can affect how the element interacts with other elements on the page and how it responds to different screen sizes and devices.

# **13) how position:static,relative,sticky,absolute works**

# position:static||relative||fixed||absolute||sticky

# position:static #by default

# # static positioned elements are not affected by top,left,right,bottom

# # it always position in it's normal flow

# position:relative

# # positioned relative to its normal flow

# # Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

# position:fixed

# #is positioned relative to viewport which means it always stays at same place even if page scrolled

# position:absolute

# # is positioned relatives to nearest positioned ancestor

# if no positioned elements it positioned,it uses document body

# position:sticky

# # positioned based on user scroll position

# z-index property tells stack order of element

# **14)What is box-sizing?**

box-sizing is a CSS property that determines how the width and height of an HTML element are calculated. The box-sizing property has two possible values:

by default it is content-box

content-box: This is the default value. The width and height of an element are calculated based on the content inside the element, and any padding or border added to the element is added to the width and height.

border-box: The width and height of an element are calculated based on the content inside the element, and any padding or border added to the element is included in the width and height. This means that the padding and border do not increase the overall size of the element, and any specified width or height will include the padding and border

# The CSS box-sizing property allows us to include the padding and border in an element's total width and height.

# By default width calculated like this

# width + padding + border = actual width of an element

# height + padding + border = actual height of an element

# box-sizing: border-box;

# **15)what are the new features in CSS3**

# box-shadow,animations,transitions,attribute selectors

# css3 attribute selectors:

# a[href^='Sri']

# a[href$='Sri']

# a[href\*='Sri']

# pseudo classes:

# visited,hover,n-child,

# :not()

# CSS3 colors:

# previous hexadecimal and rgb()

# but now

# RGBA(r,g,b,alpha)

# alpha between 0-1

# Rounded Corners:

# border-radius:5px;

# Shadow:

# dropshadows,textshadow

# box-shadow:x y blur spread color,

# text-shadow:

# multiple background images:

# background-image:he

# url(firstImage.jpg),

# url(secondImage.gif),

# url(thirdImage.png);

# **16) what is difference between pseudo classes and pseudo elements:**

Pseudo-classes: Select and style an element based on its state or interaction with the user, such as :hover, :active, :focus, :visited, :nth-child(), etc.

Pseudo-elements: Select and style a specific part of an element, such as ::before, ::after, ::first-letter, ::first-line, etc

# **17) What are the values of white-space?**

# white-space:nowrap

# white-space:normal

# white-space:pre

# white-space:pre-wrap

# **18)what is difference between width:auto and width:100%**

The width:auto property in CSS sets the width of an element to be determined automatically by the browser based on the content inside it. This means that the element will expand or shrink to fit its content.

On the other hand, the width:100% property sets the width of an element to be equal to its parent element's width. This means that the element will take up 100% of the available horizontal space within its parent container.

The key difference between these two properties is that width:auto adjusts the width based on the content inside the element, while width:100% adjusts the width based on the size of the parent element.

In general, if you want an element to adjust its width dynamically based on its content, you should use width:auto. If you want an element to take up the full width of its parent container, you should use width:100%.

# **19)What is the purpose of the z-index?**

it helps to specify stack order of positioned elements that may overlap another

z-index default is 0 can take any positive or negative numbers

The element is positioned: z-index only works on positioned elements, that is, elements with a position property of absolute, fixed, relative, or sticky

# **20) How would you optimize the loading speed of a webpage using HTML and CSS techniques?**

answer this

To optimize the loading speed of a webpage using HTML and CSS techniques, you can implement the following practices:

1. Minimize and compress your CSS and HTML: Remove unnecessary whitespace, comments, and code, and combine multiple CSS or JavaScript files into a single file. Minification reduces file size, leading to faster loading times. Additionally, consider using gzip compression to further reduce the file sizes.

2. Use asynchronous loading: Load CSS and JavaScript files asynchronously to prevent blocking the rendering of the page. Place JavaScript files at the end of the body or use the `async` or `defer` attributes to ensure they don't delay the rendering process.

3. Optimize images: Compress and optimize images to reduce their file size. Use appropriate image formats (e.g., JPEG for photos, PNG for graphics with transparency) and consider using responsive images with the `srcset` and `sizes` attributes to serve different sizes based on the device's capabilities.

4. Lazy loading: Implement lazy loading for images and videos. This technique loads content as the user scrolls, rather than all at once, improving initial page load time. There are libraries and plugins available to assist with lazy loading implementation.

5. Minimize HTTP requests: Reduce the number of HTTP requests by combining multiple CSS files into one and using CSS sprites for icons or small images. This reduces the round trips required to fetch resources, improving overall loading speed.

6. Use efficient selectors: Optimize CSS selectors to target elements more efficiently. Avoid using overly complex selectors that require extensive DOM traversal, as they can slow down rendering. Use classes and IDs whenever possible to improve performance.

7. Inline critical CSS: Inline the critical CSS directly into the HTML file or use techniques like server-side rendering to deliver the critical styles inline. This allows the browser to render the page faster, even before external stylesheets are loaded.

8. Utilize caching: Set appropriate cache headers to enable browser caching of static resources. This allows returning visitors to load the page faster by retrieving cached files instead of making new requests.

9. Implement a Content Delivery Network (CDN): Use a CDN to distribute your website's static assets across multiple servers geographically closer to your users. This reduces latency and improves loading speed.

10. Use CSS animations and transitions judiciously: Animations and transitions can enhance user experience, but excessive use can negatively impact performance. Be mindful of the number and complexity of animations on your page.