1. Are the HTML tags and elements the same thing?

No. HTML elements are defined by a starting tag, closing tag and may contain some content.

For example : <h1>Heading 1</h1> is a HTML element but just <h1> is a starting tag and </h1> is a closing tag

1. What are void elements in HTML?

HTML elements which do not have closing tags or do not need to be closed are Void elements. For Example <br />, <img />, <hr />, etc.

1. What are HTML Entities?

In HTML some characters are reserved like ‘<’, ‘>’, ‘/’, etc. To use these characters in our webpage we need to use the character entities called HTML Entities.

|  |  |
| --- | --- |
|  |  |
| < | &lt; |
| > | &gt; |
| & | &amp; |
| ‘ | &apos; |
| “ | &quot; |
| Space | &nbsp; |

1. How to optimize website assets loading?

* **CDN hosting** - A CDN or content delivery network is geographically distributed servers to help reduce latency.
* **File compression** - This is a method that helps to reduce the size of an asset to reduce the data transfer
* **File concatenation** - This reduces the number of HTTP calls
* Minify scripts - This reduces the overall file size of js and CSS files
* Parallel downloads - Hosting assets in multiple subdomains can help to bypass the download limit of 6 assets per domain of all modern browsers. This can be configured but most general users never modify these settings.
* Lazy Loading - Instead of loading all the assets at once, the non-critical assets can be loaded on a need basis.

1. What is Doctype & what are the different type of doctypes?

It is a special tag that specifies the document type and version of HTML that a web page follows.

<!DOCTYPE html> at the beginning of an HTML document, it means that the page is written using HTML5 standards.

* Strict Doctype
* Transitional Doctype
* Frameset Doctype

1. How can we indicate character set being used by HTML Document?

We can use <meta > tag.

For Example: <meta characterset= “UTF-8”>

1. Can we display a web page inside a web page or Is nesting of webpages possible?

Yes, we can display a web page inside another web page. HTML provides a tag <iframe> using which we can achieve this functionality.

Example : <iframe src= “url of the web page to embed” />

1. How is Cell Padding different from Cell Spacing?

Cell Spacing is the space or gap between two consecutive cells. Whereas, Cell Padding is the space or gap between the text/content of the cell and the edge/ border of the cell.

1. What is the difference between “display: none” and “visibility: hidden”, when used as attributes to the HTML element.

When we use the attribute “visibility: hidden” for an HTML element then that element will be hidden from the webpage but still takes up space. Whereas, if we use the “display: none” attribute for an HTML element then the element will be hidden, and also it won’t take up any space on the webpage.

1. Difference Between ID & Class attribute.

ID Attribute

* The id attribute is used to provide a unique identifier for a single HTML element on a page.
* An id value should be unique within the entire HTML document. No two elements should have the same id.

Class Attribute

* The class attribute is used to assign one or more class names to an HTML element.
* Multiple elements can share the same class, and the same class can be applied to multiple elements.

1. Significance of Head & Body Tag.

Head

* This tag contains Meta Data about web page.
* <Link>, <meta>, <Style>, <Script> are contained in Head tag.

Body

* Contains body of HTML Document.

1. How can we include audio or video in a webpage?

HTML5 provides two tags: <audio> and <video> tags using which we can add the audio or video directly in the webpage.

1. Semantic & Non Semantic Element?

Semantic elements in HTML are tags that carry *meaningful information about the structure* and content of a web page. Eg. <header>, <main>, <section>, <nav>, <footer>, etc.

non-semantic elements are *used for layout, formatting,* and other purposes, without providing inherent meaning or structure to the content they enclose. Eg.<div>, <span>, <hr>, <br>, etc.

1. What is difference between Block and inline elements?

Block Element

A block-level element always starts on a new line, and the browsers automatically add some space (a margin) before and after the element.

Inline Element

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

1. What is the difference between link tag <link> and anchor tag <a>?

The <link> tag links external resources, such as CSS stylesheets, to an HTML document. The <a> tag creates links to other pages or resources within the same document.

1. What is an image map?

An image map allows us to define clickable regions on an image, each of which corresponds to a different hyperlink or action.

1. How to create a new HTML Element?

<script>document.createElement(“myElement”)</script>

<myElement>Hello!!</myElement>

1. Can we create multicolor text on web page?

<font color= “color-name”></font>

1. How to make a picture the background image of web page?

<body background= “image.gif”>

1. What is SVG?

Used for creating two-dimensional vector graphics.

1. Media Type Supported by HTML

* Audio

<audio controls>  
 <source src="audio.mp3" type="audio/mpeg">

Your browser does not support the audio element.

</audio>

* Video

<video controls>  
 <source src="video.mp4" type="video/mp4">

Your browser does not support the video element.

</video>

1. Different API’s introduced in HTML5.

Canvas API

Web Audio API

Web Storage API

IndexedDB API

File API

Geolocation API

WebSockets API

Web Workers API

Drag and Drop API

Notifications API

WebRTC (Web Real-Time Communication) API

Fullscreen API

History API

Service Workers API

1. SVG (Scalable Vector Images)

JPEG, PNG, GIF are Bitmap Images.

SVG is vector Image

1. How to use SVG in HTML?

1. <img src = “filename.svg” width = “200px” height = “200px”/>

2. <svg width="100" height="100">

<circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />

</svg>

3. Using css background

1. What is ViewPort and ViewBox in HTML?

Viewport is the visible area of the SVG image.  
The viewBox attribute defines the position and dimension of an SVG viewport. The value of the viewBox attribute is a list of four numbers: min-x, min-y, width and height.

<svg viewBox="0 0 100 100" >

1. SVG Line

<svg height="210" width="500">

<line x1="0" y1="0" x2="200" y2="200" stroke="blue" stroke-width="12"/>

</svg>

The x1 attribute defines the start of the line on the x-axis

The y1 attribute defines the start of the line on the y-axis

The x2 attribute defines the end of the line on the x-axis

The y2 attribute defines the end of the line on the y-axis

Stroke is used to provide color to line.

1. SVG Rectangle  
   <svg width="400" height="180">

<rect x="50" y="20" width="150" height="150" fill="black" stroke = "pink" stroke-width = "5" rx="20" ry="20"/>

</svg>

The x attribute defines the left position of the rectangle (e.g. x="50" places the rectangle 50 px from the left margin)

The y attribute defines the top position of the rectangle (e.g. y="20" places the rectangle 20 px from the top margin)

The rx and the ry attributes rounds the corners of the rectangle.

1. SVG Circle

<svg height="100" width="100">

<circle cx="50" cy="50" r="40" stroke="black" stroke-width="3" fill="red" />

</svg>

The cx and cy attributes define the x and y coordinates of the center of the circle. If cx and cy are omitted, the circle's center is set to (0,0)

The r attribute defines the radius of the circle

1. Ellipse SVG

<svg height="140" width="500">  
 <ellipse cx="200" cy="80" rx="100" ry="50" fill: “yellow” stroke = “purple” stroke-width= “2” />

</svg1>

The cx attribute defines the x coordinate of the center of the ellipse

The cy attribute defines the y coordinate of the center of the ellipse

The rx attribute defines the horizontal radius

The ry attribute defines the vertical radius

1. Polygon SVG

<svg height="210" width="500">  
 <polygon points="100,10 40,198 190,78 10,78 160,198" fill = "lime" stroke = "RED" stroke-width = "5" fill-rule = "evenodd"/>

</svg>

<article> Defines independent, self-contained content

<aside> Defines content aside from the page content

<details> Defines additional details that the user can view or hide

<figcaption> Defines a caption for a <figure> element

<figure> Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.

<footer> Defines a footer for a document or section

<header> Specifies a header for a document or section

<main> Specifies the main content of a document

<mark> Defines marked/highlighted text

<nav> Defines navigation links

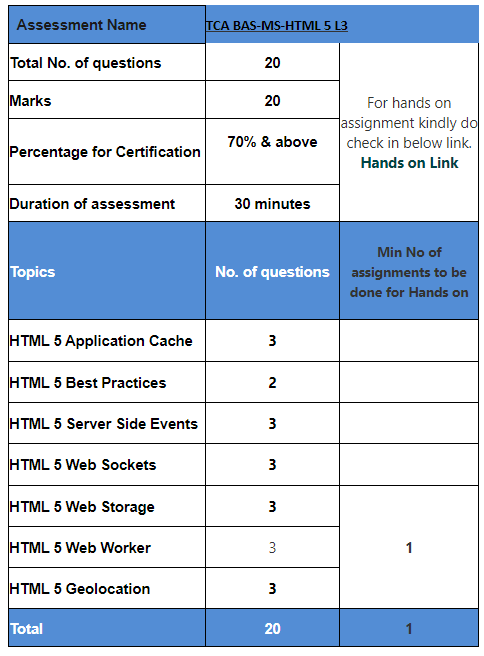
<section> Defines a section in a document

<summary> Defines a visible heading for a <details> element

<time> Defines a date/time

1. Drag and Drop

Events in Drag: dragstart, dragend, dragover, dragenter, dragleave, drop.



HTML 5 Application Cache

HTML5 Application Cache (AppCache) was a feature introduced in HTML5 to allow web applications to be accessed offline. It provided a way for developers to specify resources that browsers should cache and use when the application is offline.

How It Worked

To use AppCache, developers created a manifest file that listed the resources (HTML, CSS, JavaScript, images, etc.) that should be cached. The browser would then cache these resources and use them when the application was accessed without an internet connection.

A typical manifest file looked like this:

CACHE MANIFEST

# Version 1.1

# The following resources will be cached

CACHE:

index.html

styles.css

script.js

logo.png

# Resources listed here will always be fetched from the network

NETWORK:

api/data.json

# Fallback content if the user is offline

FALLBACK:

/ /offline.html

**Explanation**

* **CACHE MANIFEST**: This line indicates the start of a manifest file. The browser uses it to identify the file as a manifest.
* **# Version 1.1**: This is a comment. Comments in manifest files are used to version the cache. Changing this version number forces the browser to update the cache with new files.
* **CACHE**: This section lists the files to be cached for offline use. The browser will download and cache these files when the application is first loaded.
  + index.html: The main HTML file of the application.
  + styles.css: The CSS file for styling the application.
  + script.js: A JavaScript file containing the application's logic.
  + logo.png: An image file used by the application.
* **NETWORK**: This section specifies which resources should never be cached and should always be fetched from the network.
  + api/data.json: A JSON file fetched from the network. This might contain dynamic data that should always be up-to-date.
* **FALLBACK**: This section specifies a fallback resource for offline access. If a user tries to access a URL while offline, the application will serve the specified fallback file.
  + / /offline.html: If the user tries to access any URL (denoted by /) and is offline, serve offline.html instead.

**Important Considerations**

* **Order Matters**: Resources in the manifest should be correctly ordered to ensure the necessary files are available when offline.
* **Updating the Cache**: To force an update, you must change the manifest file, such as by updating the version comment.
* **HTTP Headers**: The manifest file should be served with the correct MIME type (text/cache-manifest) to ensure browsers recognize it.

Advantages

* Offline Access: Allowed applications to work without an internet connection.
* Performance: Reduced load times by serving cached resources.

Disadvantages

* Complexity: Required careful management to ensure the cache was updated correctly.
* Obsolescence: Updates to cached resources were not always immediate, leading to outdated content.
* Limited Control: Developers had limited control over caching behavior.

Deprecation

AppCache was deprecated because it had several limitations and complexities that made it difficult to use effectively. It was also not well-suited for modern web applications that required more dynamic caching capabilities. As a result, it was replaced by the more flexible and robust Service Workers.

Service Workers

Service Workers provide more advanced features for handling caching and offline access, including:

Network Interception: Ability to intercept and handle network requests programmatically.

Fine-Grained Caching Control: Developers can define caching strategies for different resources.

Background Synchronization: Support for background sync and push notifications.

Due to these enhancements, Service Workers are the preferred method for managing offline capabilities and caching in modern web applications. If you're developing new applications, it's recommended to use Service Workers instead of AppCache.

HTML 5 Best Practices

HTML 5 Server Side Events

<https://www.w3schools.com/html/html5_serversentevents.asp>

<https://html5doctor.com/server-sent-events/>

A server-sent event is when a web page automatically gets updates from a server.

This was also possible before, but the web page would have to ask if any updates were available. With server-sent events, the updates come automatically.

The “EventSource” object is used to receive server-sent event notifications

HTML 5 Web Sockets

[https://websockets.spec.whatwg.org//](https://websockets.spec.whatwg.org/)

<https://www.tutorialspoint.com/html/html_websocket.htm>

Web sockets are persistent connections. Persistent means that whenever the connection is open it will be in open state until and unless we close it.

Web Socket implementation should be implemented both on the client side as well on the server side. On server side web socket can be implemented using node php, etc.

WebSockets allow bidirectional communication which means both client and server can send data to each other independently and simultaneously.

Syntax

**var Socket = new WebSocket(url, [protocol] );**

Here first argument is ur which specifies the URL to which to connect. The second argument is protocol and is optional. If present, specifies the protocol that the server must support for the connection to be successful.

ReadyState Attributes of WebSocket

“readyState” represents the state of the connection.

If Socket.readyState has a value of

* 0 indicates that the connection has not yet been established.
* 1 indicates that the connection is established and communication is possible.
* 2 indicates that the connection is going through the closing handshake.
* 3 indicates that the connection has been closed or could not be opened.

WebSocket Methods

Following are the methods associated with WebSocket object. Assuming we created Socket object as mentioned above −

|  |  |
| --- | --- |
| **Method** | **Description** |
| **Socket.send()** | The send(data) method transmits data using the connection. |
| **Socket.close()** | The close() method would be used to terminate any existing connection. |

WebSocket Events

Following are the events associated with WebSocket object.

|  |  |  |
| --- | --- | --- |
| **Event** | **Values & Event Handler** | **Values & Description** |
| open | Socket.onopen | This event occurs when socket connection is established. |
| message | Socket.onmessage | This event occurs when client receives data from server. |
| error | Socket.onerror | This event occurs when there is any error in communication. |
| close | Socket.onclose | This event occurs when connection is closed. |

HTML 5 Web Storage

<https://www.w3schools.com/html/html5_webstorage.asp>

With web storage, web applications can store data locally within the user's browser.

Before HTML5, application data had to be stored in cookies, included in every server request. Web storage is more secure, and large amounts of data can be stored locally, without affecting website performance.

Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.

Web storage is per origin (per domain and protocol). All pages, from one origin, can store and access the same data.

HTML web storage provides two objects for storing data on the client:

window.localStorage - stores data with no expiration date. The data will not be deleted when the browser is closed,

window.sessionStorage - stores data for one session (data is lost when the browser tab is closed)

Before using web storage, check browser support for localStorage and sessionStorage:

if (typeof(Storage) !== "undefined") {

  // Code for localStorage/sessionStorage.

} else {

  // Sorry! No Web Storage support..

}

HTML 5 Web Worker

<https://www.w3schools.com/html/html5_webworkers.asp>

<https://learn.microsoft.com/en-us/previous-versions/msdn10/hh549259(v=msdn.10)>

<https://www.tutorialspoint.com/html/html_web_workers_api.htm>

A web worker is a JavaScript that runs in the background, independently of other scripts, without affecting the performance of the page. We can continue to do whatever you want: clicking, selecting things, etc., while the web worker runs in the background.

Before creating a web worker, check whether the user's browser supports it:

if (typeof(Worker) !== "undefined") {

  // Yes! Web worker support!

  // Some code.....

} else {

  // Sorry! No Web Worker support..

}

Create a Web Worker Object

The following lines check if the worker already exists, if not - it creates a new web worker object and runs the code in "demo\_workers.js":

if (typeof(w) == "undefined") {

  w = new Worker("demo\_workers.js");

}

Then we can send and receive messages from the web worker.

Add an "onmessage" event listener to the web worker.

w.onmessage = function(event){

  document.getElementById("result").innerHTML = event.data;

};

When the web worker posts a message, the code within the event listener is executed. The data from the web worker is stored in event.data.

Terminate a Web Worker

When a web worker object is created, it will continue to listen for messages (even after the external script is finished) until it is terminated.

To terminate a web worker, and free browser/computer resources, use the terminate() method:

w.terminate();

Reuse the Web Worker

If you set the worker variable to undefined, after it has been terminated, you can reuse the code

 w = undefined;

Full Web Worker Example Code

We have already seen the Worker code in the .js file. Below is the code for the HTML page:

Example

<!DOCTYPE html>

<html>

<body>

<p>Count numbers: <output id="result"></output></p>

<button onclick="startWorker()">Start Worker</button>

<button onclick="stopWorker()">Stop Worker</button>

<script>

var w;

function startWorker() {

  if (typeof(Worker) !== "undefined") {

    if (typeof(w) == "undefined") {

      w = new Worker("demo\_workers.js");

    }

    w.onmessage = function(event) {

      document.getElementById("result").innerHTML = event.data;

    };

  } else {

    document.getElementById("result").innerHTML = "Sorry! No Web Worker support.";

  }

}

function stopWorker() {

  w.terminate();

  w = undefined;

}

</script>

</body>

</html>

Since web workers are in external files, they do not have access to the following JavaScript objects:

* The window object
* The document object
* The parent object

HTML 5 Geolocation

<https://www.w3schools.com/html/html5_geolocation.asp>

The HTML Geolocation API is used to get the geographical position of a user.

The getCurrentPosition() method is used to return the user's position.

The Geolocation object also has other interesting methods:

* watchPosition() - Returns the current position of the user and continues to return the updated position as the user moves (like the GPS in a car).
* clearWatch() - Stops the watchPosition() method.