**What is metadata in HTML5 and how is it specified?**

Metadata is data that describes other data, providing additional information about an HTML document. Its purpose is to help browsers, search engines and other web applications better interpret a document. Examples of metadata include the description, author, and keywords.

The <meta> tag is used to define metadata about an HTML document. <meta> tags are always enclosed within the <head> of the HTML document.

### What are the new tags for multimedia in HTML5?

Unlike previous versions, HTML5 allows developers to create multimedia objects without the need for additional plugins. The new tags that facilitate this are:

* **<audio>** - used to embed audio content
* **<video>** - used to embed video content
* **<embed>** - used to embed content from an external source
* **<source>** - used to embed multiple media resources
* **<track>** - used to specify text tracks (such as subtitles) for audio and video content

### How can HTML5 be used for data storage purposes?

While HTML5 doesn’t include data storage functionality as part of its native specification, it’s possible to store data through the integration of data-related APIs. These can be used with HTML5 documents to manipulate client-side databases with SQL.

The two HTML5 APIs for data storage and querying are:

* **IndexedDB API**, a standard maintained by the World Wide Web Consortium
* **Web SQL Database**, now considered deprecated

### How is the Geolocation API implemented in HTML5?

The HTML5 Geolocation API uses the device’s GPS, WiFi or mobile signal to triangulate the user’s latitude and longitude coordinates. The user must give their permission before geolocation services can be used on their device.

From the developer’s point of view, the Geolocation API is implemented firstly by calling the geolocation.navigator object. Three methods are then used by the API to handle location data:

1. **getCurrentPosition** identifies the device’s current location and returns a position object with data
2. **watchPosition** returns a value whenever the device location changes
3. **clearWatch** cancels the previous watchPosition call

### When should <div> tags be used in HTML5?

HTML5 introduces a number of new tags (such as <aside>, <footer>, and <header>) that take the place that <div> has traditionally occupied in previous HTML versions.

However, <div> should still be used in HTML5 when no other semantically appropriate element is available. Generally, this will be for styling purposes, such as wrapping the web page or introducing the opening paragraph.

### What types of graphics are supported by HTML5?

:

Unlike previous versions, HTML5 offers inbuilt graphics features. The two types of graphics supported by HTML5 are:

* **SVG**(Scalable Vector Graphics), used to create vector-based graphics, such as diagrams and icons
* **Canvas**, used to draw graphics, such as shapes

### What are image maps in HTML5?

:

Image maps are images with several clickable areas that link to different web pages. They use the <map> and <area> tags. Coordinates are specified to segment the image into different areas and then relevant links are applied.

### What are some of the most important APIs in HTML5?

:

HTML5 supports a wide range of APIs. Some of the most popular ones include:

* **Geolocation API**, used to identify the user’s location
* **Web Speech API**, which provides speech recognition functionality
* **Clipboard API**, which provides copy, cut, and paste functionality
* **History API**, which provides access to the browser navigation history
* **Web Notifications API**, used to send web-based notifications to users

### How can a copyright symbol be added to the webpage using HTML5?

The ‘&copy’ code is used to generate the copyright symbol (©). As a best practice, it should be accompanied by a year and the name of the organization.

To include copyright information in the footer of a web page, the code would look like this:

<footer> <small>&copy; Copyright Date, Organization Name</small> </footer>

### What are tags and attributes in HTML?

Tags are the primary component of the HTML that defines how the content will be structured/ formatted, whereas Attributes are used along with the HTML tags to define the characteristics of the element. For example, <p align=” center”>Interview questions</p>, in this the ‘align’ is the attribute using which we will align the paragraph to show in the center of the view.

### . In how many ways can we position an HTML element? Or what are the permissible values of the position attribute?

There are mainly 7 values of position attribute that can be used to position an HTML element:

1. **static**: Default value. Here the element is positioned according to the normal flow of the document.
2. **absolute**: Here the element is positioned relative to its parent element. The final position is determined by the values of left, right, top, bottom.
3. **fixed**: This is similar to absolute except here the elements are positioned relative to the <html> element.
4. **relative**: Here the element is positioned according to the normal flow of the document and positioned relative to its original/ normal position.
5. **initial**: This resets the property to its default value.
6. **inherit**: Here the element inherits or takes the property of its parent.

### 22. In how many ways you can display HTML elements?

1. **inline**: Using this we can display any block-level element as an inline element. The height and width attribute values of the element will not affect.
2. **block**: using this, we can display any inline element as a block-level element.
3. **inline-block**: This property is similar to inline, except by using the display as inline-block, we can actually format the element using height and width values.
4. **flex**: It displays the container and element as a flexible structure. It follows flexbox property.
5. **inline-flex**: It displays the flex container as an inline element while its content follows the flexbox properties.
6. **grid**: It displays the HTML elements as a grid container.
7. **none**: Using this property we can hide the HTML element.

### Difference between link tag <link> and anchor tag <a>?

The anchor tag <a> is used to create a hyperlink to another webpage or to a certain part of the webpage and these links are clickable, whereas, link tag <link> defines a link between a document and an external resource and these are not clickable.

### What is the difference between <figure> tag and <img> tag?

The <figure> tag specifies the self-contained content, like diagrams, images, code snippets, etc. <figure> tag is used to semantically organize the contents of an image like image, image caption, etc., whereas the <img> tag is used to embed the picture in the HTML5 document.

### Explain CSS position property?

* **Absolute:**To place an element exactly where you want to place it. absolute position is actually set relative to the element's parent. if no parent is available then the relative place to the page itself (it will default all the way back up to the element).
* **Relative:**"Relative to itself". Setting position: relative; on an element and no other positioning attributes, it will no effect on its positioning. It allows the use of z-index on the element and it limits the scope of absolutely positioned child elements. Any child element will be absolutely positioned within that block.
* **Fixed:**The element is positioned relative to the viewport or the browser window itself. viewport doesn't change if you scroll and hence the fixed element will stay right in the same position.
* **Static:**Static default for every single page element. The only reason you would ever set an element to position: static is to forcefully remove some positioning that got applied to an element outside of your control.
* **Sticky:**Sticky positioning is a hybrid of relative and fixed positioning. The element is treated as relative positioned until it crosses a specified threshold, at which point it is treated as fixed positioned.

### Different Box Sizing Property?

The box-sizing CSS property sets how the total width and height of an element are calculated.

* **Content-box:** The default width and height values apply to the element's content only. The padding and border are added to the outside of the box.
* **Padding-box:** Width and height values apply to the element's content and its padding. The border is added to the outside of the box. Currently, only Firefox supports the padding-box value.
* **Border-box:** Width and height values apply to the content, padding, and border.

### Can you name the four types of @media properties?

The four types of @media properties are:

1. All → It’s the default property. Used for all media-type devices.
2. Screen → Used for computer screen, mobile screen.
3. Print → Used for printers.
4. Speech → Used for screen readers.

### 29. What is the grid system?

CSS Grid Layout is the most powerful layout system available in CSS. It is said to be a 2-dimensional system, meaning it can handle both columns and rows, unlike flexbox which is largely a 1-dimensional system.

### What are the different ways to hide the element using CSS?

* Using display property(**display: none**). It’s not available for screen readers. The element will not exist in the DOM if display: none is used.
* Using visibility property(**visibility: hidden**), will take up the space of the element. It will be available to screen reader users. The element will actually be present in the DOM, but not shown on the screen.
* Using position property (**position: absolute**). Make it available outside the screen.

### What does the :root pseudo-class refer to?

The :root selector allows you to target the highest-level “parent” element in the DOM, or document tree. It is defined in the CSS Selectors Level 3 specification.

### Difference between CSS grid vs flexbox?

* CSS Grid Layout is a two-dimensional system, meaning it can handle both columns and rows. Grid layout is intended for larger-scale layouts which aren’t linear in design.
* Flexbox is largely a one-dimensional system (either in a column or a row). Flexbox layout is most appropriate to the components of an application.

### What does \* { box-sizing: border-box; } do? What are its advantages?

* It makes every element in the document include the padding and border in the element’s inner dimension for the height and width computation.
* In box-sizing: border-box, The height of an element is now calculated by the content's height + vertical padding + vertical border width.
* The width of an element is now calculated by the content's width + horizontal padding + horizontal border width.
* Java script

### Explain what a callback function is and provide a simple example JUNIOR

A **callback** the function is a function that is passed to another function as an argument and is executed after some operation has been completed. Below is an example of a simple callback function that logs to the console after some operations have been completed.

### Explain different methods of iteration in Javascript JUNIOR

For objects:

* **for** loops - **for (var property in obj) { console.log(property); }**. However, this will also iterate through its inherited properties, and you will add a **obj.hasOwnProperty(property)** check before using it.
* **Object.keys()** - **Object.keys(obj).forEach(function (property) { ... })**. **Object.keys()** is a static method that will list all enumerable properties of the object that you pass it.
* **Object.getOwnPropertyNames()** - **Object.getOwnPropertyNames(obj).forEach(function (property) { ... })**. **Object.getOwnPropertyNames()** is a static method that will list all enumerable and non-enumerable properties of the object that you pass it.

For arrays:

* **for** loops - **for (var i = 0; i < arr.length; i++)**. The common pitfall here is that **var** is in the function scope and not the block scope and most of the time you would want a block-scoped iterator variable. ES2015 introduces **let** which has block scope and it is recommended to use that instead. So this becomes: **for (let i = 0; i < arr.length; i++)**.
* **forEach** - **arr.forEach(function (el, index) { ... })**. This construct can be more convenient at times because you do not have to use them **index** if all you need is an array of elements. There are also the **every** and **some** methods that will allow you to terminate the iteration early.

### Explain what is Hoisting in Javascript SENIOR

**Hoisting** is the concept in which Javascript, by default, moves all declarations to the top of the current scope. As such, a variable can be used before it has been declared.

Note that Javascript only hoists declarations and not initializations.

* Explain what is Hoisting in Javascript SENIOR
* Hoisting is the concept in which Javascript, by default, moves all declarations to the top of the current scope. As such, a variable can be used before it has been declared.
* Note that Javascript only hoists declarations and not initializations.