

learning-zone / html-interview-questions Public

HTML Interview Questions (vHTML5)

[🔗 learning-zone.github.io/html-interview-questions/](https://learning-zone.github.io/html-interview-questions/)

 275 stars  133 forks

 Star

 Watch ▾

 Code

 Issues

 Pull requests

 Actions

 Projects

 Wiki

 Security

 Insights

 master ▾

...



learning-zone Update README.md ...

✓ on Jul 15, 2021

 387

[View code](#)

 README.md

HTML5 Interview Questions

Click  if you like the project. Pull Request are highly appreciated.

Table of Contents

- *HTML5 Events List*
- *HTML5 Tags*

Sl.No	Questions
01.	What is difference between HTML and XHTML?
02.	What are the building blocks of HTML5?
03.	What are the semantic tags available in html5?
04.	Why you would like to use semantic tag?
05.	What does a <DOCTYPE html> do?

Sl.No	Questions
06.	What happens when DOCTYPE is not given?
07.	What are the new form elements in HTML5?
08.	How many new form elements are introduced in html5?
09.	What is difference between span tag and div tag?
10.	What are optional closing tag?
11.	What is a self closing tag?
12.	Explain the difference between block elements and inline elements?
13.	What are semantic and non-semantic elements?
14.	What is the purpose of main element?
15.	Define semantic markup. What are the semantic meanings for <section>, <article>, <aside>, <nav>, <header>, <footer> and when/how should each be used in structuring html markup?
16.	When should you use section , div or article ?
17.	Can a web page contain multiple <header> elements? What about <footer> elements?
18.	What is Character Encoding?
19.	What is the purpose of meta tags?
20.	What does async and defer refer in script tag? Describe the difference between <script> , <script async> and <script defer>
21.	Describe the difference between a cookie , sessionStorage and localStorage ?
22.	Does localStorage throw error after reaches maximum limits?
23.	Why to use IndexedDB instead of WebSQL in HTML5?
24.	Explain Application Cache in HTML5. OR What is a manifest file in HTML?
25.	What is the purpose of cache busting and how can you achieve it?
26.	What ARIA and screenreaders are, and how to make a website accessible?
27.	What is the purpose of the alt attribute on images?

Sl.No	Questions
28.	What does <code>enctype='multipart/form-data'</code> mean?
29.	What is difference between Select and Datalist?
30.	Explain some of the pros and cons for CSS animations versus JavaScript animations?
31.	What does CORS stand for and what issue does it address?
32.	Can you describe the difference between progressive enhancement and graceful degradation?
33.	What is the DOM? How does the DOM work?
34.	How does the browser rendering engine work?
35.	What is the difference between standards mode and quirks mode?
36.	What is Critical Rendering Path?
37.	What are the Benefits of Server Side Rendering (SSR) Over Client Side Rendering (CSR)?
38.	Name 3 ways to decrease page load?
39.	Ways to improve website performance
40.	Comparison of browsers engines like Chrome, Firefox, Internet explorer, Safari?
41.	What does the lang attribute in html do?
42.	What is desktop first and mobile first design approach?
43.	What are data- attributes good for?
44.	Explain the difference between layout, painting and compositing?
45.	Explain about HTML Layout Engines used by browsers?
46.	How to make page responsive?
47.	Does the following trigger http request at the time of page load?
48.	List the API available in HTML5?
49.	How geolocation api works in html5?

Sl.No	Questions
50.	What is the use of WebSocket API?
51.	Explain about HTML Canvas?
53.	What is difference between SVG and Canvas?
54.	Explain Drag and Drop in HTML5?
55.	Explain Microdata in HTML5?
56.	What are the HTML tags which deprecated in HTML5?
57.	How you can Use Modernizr in HTML5?
58.	What is progressive rendering?
59.	What is an iframe and how it works?
60.	Explain the use of rel="nofollow", rel="noreferrer", rel="noopener" attribute?
61.	How can you highlight text in HTML?
62.	How can I get indexed better by search engines?
63.	What is the difference between an "attribute" and a "property" in HTML?
64.	How do you set IE compatibility mode?
65.	What is an optional tag?
66.	What is an HTML preprocessor? Have you used different HTML templating languages before?
67.	How do you change the direction of html text?
68.	When is it appropriate to use the small element?
79.	How do you serve a page with content in multiple languages?
70.	What is the difference between <section> and <div> ?
71.	Discuss the differences between an HTML specification and a browser's implementation thereof.
72.	Why you would use a srcset attribute in an image tag? Explain the process the browser used when evaluating the content of this attribute.
73.	What is accessibility & ARIA role means in a web application?

Q. What is difference between HTML and XHTML?

The Extensible Hypertext Markup Language, or XHTML, has two important notes for front end developers.

1. It needs to be well formed, meaning all elements need to be closed and nested correctly or you will return errors.
2. Since it is more strict than HTML it requires less pre-processing by the browser, which may improve your sites performance.

[↑ back to top](#)

Q. What are the building blocks of HTML5?

- **Semantics:** allowing you to describe more precisely what your content is.
- **Connectivity:** allowing you to communicate with the server in new and innovative ways.
- **Offline and storage:** allowing webpages to store data on the client-side locally and operate offline more efficiently.
- **Multimedia:** making video and audio first-class citizens in the Open Web.
- **2D/3D graphics and effects:** allowing a much more diverse range of presentation options.
- **Performance and integration:** providing greater speed optimization and better usage of computer hardware.
- **Device access:** allowing for the usage of various input and output devices.
- **Styling:** letting authors write more sophisticated themes.

[↑ back to top](#)

Q. What are the semantic tags available in html5?

HTML5 semantic tags define the function and the category of your text, simplifying the work for browsers and search engines, as well as developers.

HTML5 offers new semantic elements to define different parts of a web page:

- <article>
- <aside>

- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>

Syntax:

```
<header></header>
<section>
    <article>
        <figure>
            <img>
            <figcaption></figcaption>
        </figure>
    </article>
</section>
<footer></footer>
```

[↑ back to top](#)

Q. Why you would like to use semantic tag?

- Search Engine Optimization, accessibility, repurposing, light code.
- Many visually impaired person rely on browser speech and semantic tag helps to interpret page content clearly.
- Search engine needs to understand page content to rank and semantic tag helps.
- Semantic code aids accessibility. Specially, many people whose eyes are not good rely on speech browsers to read pages to them. These programs cannot interpret pages very well unless they are clearly explained.
- Help Search engines to better understand pages. Search engine need to understand what your content is about when rank you properly on search engines. Semantic code

tends to improve your placement on search engines, as it is easier for the "search engine spiders" to understand.

- It's easier to read and edit, which saves time and money during maintenance.

[↑ back to top](#)

Q. What does a <DOCTYPE html> do?

A DOCTYPE is always associated to a DTD (Document Type Definition). A DTD defines how documents of a certain type should be structured (i.e. a button can contain a span but not a div), whereas a DOCTYPE declares what DTD a document supposedly respects (i.e. this document respects the HTML DTD). For webpages, the DOCTYPE declaration is required. It is used to tell user agents what version of the HTML specifications your document respects.

Once a user agent has recognized a correct DOCTYPE, it will trigger the no-quirks mode matching this DOCTYPE for reading the document. If a user agent doesn't recognize a correct DOCTYPE, it will trigger the quirks mode .

[↑ back to top](#)

Q. What happens when DOCTYPE is not given?

The web page is rendered in quirks mode. The web browser engines use quirks mode to support older browsers which do not follow the W3C specifications. In quirks mode CSS class and id names are case insensitive. In standards mode they are case sensitive.

[↑ back to top](#)

Q. What are the new form elements in HTML5?

There are five new form elements in the HTML5 forms specification: <datalist> , <output> , <progress> , and <meter> .

1. Datalist Tag

Allows to attach a list of suggestions to a text input element. As soon as the user begins to type in the text field, the list of suggestions appears and the user can choose from the suggestions with the mouse.

```
<p>Enter your favorite browser name:</p>
<input type="text" list="browsers" name="favorite_browser">
<datalist id="browsers">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Internet Explorer">
  <option value="Opera">
  <option value="Safari">
</datalist>
```

Live demo

2. Meter Tag

Indicates a numeric value that falls within a range. The tag supports a number of attributes: **value**: If you don't specify a value, the first numeric value inside the `<meter></meter>` pair becomes the value.

- **max**: The maximum possible value of the item.
- **min**: The minimum possible value of the item.
- **high**: If the value can be defined as a range, this is the high end of the range.
- **low**: If the value can be defined as a range, this is the low end of that range.
- **optimum**: The optimal value of the element.

```
<p>Disk Usage: <meter value="0.2">20%</meter></p>

<p>Total Score: <meter value="6" min="0" max="10">6 out of 10</meter></p>

<p>Pollution Level: <meter low="60" high="80" max="100" value="85">Very High</meter>
```

Live demo

3. Output Tag

It indicates a section of the page that can be modified by a script (usually JavaScript).

```
<form oninput="result.value=parseInt(a.value)+parseInt(b.value)">
  <input type="range" id="a" value="50" > +
  <input type="number" id="b" value="100" > =
  <output name="result" for="a b"></output>
</form>
```

[Live demo](#)

4. Progress Tag

Indicates how much of a task has been completed (often marked as a percentage). It is expected to be modified through JavaScript code.

```
<p>Progress: <progress id="bar" value="0" max="100"><span>0</span>%</progress></p>

<script type="text/javascript">
    var i = 0;
    var progressBar = document.getElementById("bar");

    function countNumbers() {
        if(i < 100) {
            i = i + 1;
            progressBar.value = i;
            // For browsers that don't support progress tag
            progressBar.getElementsByName("span")[0].textContent = i;
        }

        // Wait for sometime before running this script again
        setTimeout("countNumbers()", 100);
    }
    countNumbers();
</script>
```

[Live demo](#)

[↑ back to top](#)

Q. How many new form elements are introduced in html5?

Sl.No	Element	Description
01.	color	Gives the end user a native color picker to choose a color.
02.	date	Offers a datepicker.
03.	datetime	An element to choose both date and time.

Sl.No	Element	Description
04.	datetime-local	An element to choose both date and time, with local settings support.
05.	email	A field for entering e-mail address(es).
06.	month	Choose a full month.
07.	number	Picking a number.
08.	range	Offers a slider to set to a certain value/position.
09.	search	A field for search queries.
10.	tel	Choosing a telephone number.
11.	time	Input a certain time.
12.	url	Entering a URL.
13.	week	Picking a specific week.

Example:

```

<input type="color" value="#b97a57">

<input type="date" value="2020-06-08">

<input type="datetime" value="2020-06-09T20:35:34.32">

<input type="datetime-local" value="2020-06-09T22:41">

<input type="email" value="robert@robertnyman.com">

<input type="month" value="2020-06">

<input type="number" value="4">

<input type="range" value="15">

<!-- Note: If not set, default attribute values are min="0", max="100", step="1". -->

<input type="search" value="[Any search text]">

<input type="tel" value="[Any numeric value]">

<!-- Note: Most web browsers seem to let through any value at this time. -->

```

```
<input type="time" value="22:38">  
  
<input type="url" value="https://www.google.com/">  
  
<!-- Note: requires a protocol like http://, ftp:// etc in the beginning. -->  
  
<input type="week" value="2020-W24">
```



[↑ back to top](#)

Q. *What is difference between span tag and div tag?*

The primary difference between div and span tag is their default behavior. By default, a `<div>` is a **block-level-element** and a `` is an **inline element**.

- `<div>` is a block level element which means it will render it on its own line with a width of a 100% of the parent element.
- `` is an inline element which means it will render on the same line as the previous element, if it is also an inline element, and its width will be determined by its content.

```
<div>Demo Text, with <span>some other</span> text.</div>
```

[↑ back to top](#)

Q. *What are optional closing tag?*

`<p>`, ``, `<td>`, `<tr>`, `<th>`, `<html>`, `<body>` , etc. don't have to provide end tag. Whenever browser hits a new tag it automatically ends the previous tag.

[↑ back to top](#)

Q. *What is a self closing tag?*

In HTML5 it is not strictly necessary to close certain HTML tags. The tags that aren't required to have specific closing tags are called "self closing" tags.

An example of a self closing tag is something like a line break (`
`) or the meta tag (`<meta>`). This means that the following are both acceptable:

```
<meta charset="UTF-8">
...
<meta charset="UTF-8" />
```

[↑ back to top](#)

Q. Explain the difference between *block elements* and *inline elements*?

- block elements `<h1>`, `<p>`, ``, ``, ``,
- inline elements ``, `<a>`, ``, `<i>`, ``

[↑ back to top](#)

Q. What are semantic and non-semantic elements?

- **Semantic elements:** clearly describes its meaning to both the browser and the developer. For example: `<form>` , `<table>` , `<article>` , `<aside>` , `<details>` , `<figcaption>` , `<figure>` , `<footer>` , `<header>` , `<main>` , `<mark>` , `<nav>` , `<section>` , `<summary>` , `<time>` clearly defines its content.
- **Non-semantic elements:** `<div>` and `` tells nothing about its content.

[↑ back to top](#)

Q. What is the purpose of *main* element?

The HTML `<main>` element represents the dominant content of the `<body>` of a document. The main content area consists of content that is directly related to or expands upon the central topic of a document, or the central functionality of an application.

```
<main role="main">
  <p>Geckos are a group of usually small, usually nocturnal lizards.
  They are found on every continent except Australia.</p>
  <p>Many species of gecko have adhesive toe pads which enable them to climb walls
</main>
```



Note: A document mustn't have more than one `<main>` element that doesn't have the `hidden` attribute specified.

[↑ back to top](#)

Q. Define semantic markup. What are the semantic meanings for <section>, <article>, <aside>, <nav>, <header>, <footer> and when/how should each be used in structuring html markup?

- <header> is used to contain introductory and navigational information about a section of the page. This can include the section heading, the author's name, time and date of publication, table of contents, or other navigational information.
- <article> is meant to house a self-contained composition that can logically be independently recreated outside of the page without losing its meaning. Individual blog posts or news stories are good examples.
- <section> is a flexible container for holding content that shares a common informational theme or purpose.
- <footer> is used to hold information that should appear at the end of a section of content and contain additional information about the section. Author's name, copyright information, and related links are typical examples of such content.

[↑ back to top](#)

Q. When should you use section, div or article?

- <section> , group of content inside is related to a single theme, and should appear as an entry in an outline of the page. It's a chunk of related content, like a subsection of a long article, a major part of the page (eg the news section on the homepage), or a page in a webapp's tabbed interface. A section normally has a heading (title) and maybe a footer too.
- <article> , represents a complete, or self-contained, composition in a document, page, application, or site and that is, in principle, independently distributable or reusable, e.g. in syndication. This could be a forum post, a magazine or newspaper article, a blog entry, a user-submitted comment, an interactive widget or gadget, or any other independent item of content.
- <div> , on the other hand, does not convey any meaning, aside from any found in its class, lang and title attributes.

[↑ back to top](#)

Q. Can a web page contain multiple <header> elements? What about <footer> elements?

Yes, header elements can be used multiple times in documents. A `<header>` tag must be present for all articles, sections, and pages, although a `<footer>` tag is not necessary.

From W3C standards

A header element is intended to usually contain the section's heading (an `h1-h6` element), but this is not required. The header element can also be used to wrap a set of contents, a search form, or any relevant logos.



The footer element represents a footer for its nearest ancestor sectioning content element. A footer typically contains information about its section such as who wrote the document, copyright data, and the like.

[↑ back to top](#)

Q. What is a physical tag and logical tag in HTML?

ToDo

[↑ back to top](#)

Q. What is Character Encoding?

Character encoding is a method of converting bytes into characters. To validate or display an HTML document properly, a program must choose a proper character encoding. This is specified in the tag:

```
<meta charset="utf-8"/>
```

- **UTF-8:** A Unicode Translation Format that comes in 8-bit units that is, it comes in bytes. A character in UTF8 can be from 1 to 4 bytes long, making UTF8 variable width.

Q. What is the purpose of meta tags?

The META elements can be used to include name/value pairs describing properties of the HTML document, such as author, expiry date, a list of keywords, document author etc.

```
<!DOCTYPE html>
<html>
  <head>
    <!--Recommended Meta Tags-->
    <meta charset="utf-8">
    <meta name="language" content="english">
    <meta http-equiv="content-type" content="text/html">
    <meta name="author" content="Author Name">
    <meta name="designer" content="Designer Name">
    <meta name="publisher" content="Publisher Name">
    <meta name="no-email-collection" content="name@email.com">
    <meta http-equiv="X-UA-Compatible" content="IE=edge"/>

    <!--Search Engine Optimization Meta Tags-->
    <meta name="description" content="Project Description">
    <meta name="keywords" content="Software Engineer,Product Manager,Project Man">
    <meta name="robots" content="index,follow">
    <meta name="revisit-after" content="7 days">
    <meta name="distribution" content="web">
    <meta name="robots" content="noodp">

    <!--Optional Meta Tags-->
    <meta name="distribution" content="web">
    <meta name="web_author" content="">
    <meta name="rating" content="">
    <meta name="subject" content="Personal">
    <meta name="title" content=" - Official Website.">
    <meta name="copyright" content="Copyright 2020">
    <meta name="reply-to" content="">
    <meta name="abstract" content="">
    <meta name="city" content="Bangalore">
    <meta name="country" content="INDIA">
    <meta name="distribution" content="">
    <meta name="classification" content="">

    <!--Meta Tags for HTML pages on Mobile-->
    <meta name="format-detection" content="telephone=yes"/>
    <meta name="HandheldFriendly" content="true"/>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    <meta name="apple-mobile-web-app-capable" content="yes" />
```

```
<!--http-equiv Tags-->
<meta http-equiv="Content-Style-Type" content="text/css">
<meta http-equiv="Content-Script-Type" content="text/javascript">

<title>HTML5 Meta Tags</title>
</head>
<body>
    ...
</body>
</html>
```



↑ back to top

Q. What does `async` and `defer` refer in `script` tag? Describe the difference between `<script>`, `<script async>` and `<script defer>`

- **Async:** Downloads the script file during HTML parsing and will pause the HTML parser to execute it when it has finished downloading.
- **Defer:** Defer downloads the script file during HTML parsing and will only execute it after the HTML parser has completed. Not all browsers support this.

The `async` attribute is used to indicate to the browser that the script file can be executed asynchronously. The HTML parser does not need to pause at the point it reaches the script tag to fetch and execute, the execution can happen whenever the script becomes ready after being fetched in parallel with the document parsing.

The `defer` attribute tells the browser to only execute the script file once the HTML document has been fully parsed.

Example

```
<!-- Without async or defer, browser will run your script immediately, before render
<script src="myscript.js"></script>
```

```
<!-- With async (asynchronous), browser will continue to load the HTML page and rend
<!-- Google Analytics is usually added like this -->
<script async src="https://google-analytics.com/analytics.js"></script>
```

```
<!-- With defer, browser will run your script when the page finished parsing. (not n
<script defer src="myscript.js"></script>
```



[⬆ back to top](#)

Q. *Describe the difference between a cookie, sessionStorage and localStorage ?*

- cookie : A text file saved on the users computer to store and retrieve data
- sessionStorage : Is memory space in a browser to save temporary data until the window or tab is closed.
- localStorage : Like cookie, where data can be saved and retrieved after browser sessions, but stored in memory like sessionStorage. Data is stored as plain key value pairs and can be stored as Json objects.

	cookie	localStorage	sessionStorage
Initiator	Client or server. Server can use Set-Cookie header	Client	Client
Expiry	Manually set	Forever	On tab close
Persistent across browser sessions	Depends on whether expiration is set	Yes	No
Sent to server with every HTTP request	Cookies are automatically being sent via Cookie header	No	No
Capacity (per domain)	4kb	5MB	5MB
Accessibility	Any window	Any window	Same tab

Note: If the user decides to clear browsing data via whatever mechanism provided by the browser, this will clear out any cookie , LocalStorage , or sessionStorage stored. It's important to keep this in mind when designing for local persistance, especially when comparing to alternatives such as server side storing in a database or similar (which of course will persist despite user actions).

[↑ back to top](#)

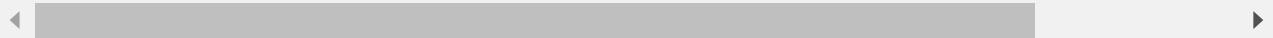
Q. Does localStorage throw error after reaches maximum limits?

Yes

Example:

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>HTML5 localStorage</title>
    </head>
    <body>
        <script type="text/javascript">
            try{
                if(window.localStorage){ // Check if the localStorage object exists

                    var result = "";
                    var characters  = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
                    var charactersLength = characters.length;
                    for(var i = 0; i < 10000; i++){
                        result += characters.charAt(Math.floor(Math.random() * character));
                        localStorage.setItem("key"+i, result);
                    }
                } else {
                    alert("Sorry, your browser do not support localStorage.");
                }
            } catch(e) {
                console.log('Exception: '+e);
            }
        </script>
    </body>
</html>
```



Output

```
Exception: QuotaExceededError: Failed to execute 'setItem' on 'Storage':
Setting the value of 'key3230' exceeded the quota.
```

[↑ back to top](#)

Q. Why to use IndexedDB instead of WebSQL in HTML5?

1. WebSQL

It is an API that is only supported in Chrome and Safari (and Android and iOS by extension). It provides an asynchronous, transactional interface to SQLite. Since 2010, it has been deprecated in favor of IndexedDB.

Advantages

- Supported on major mobile browsers (Android Browser, Mobile Safari, Opera Mobile) as well as several desktop browsers (Chrome, Safari, Opera).
- Good performance generally, being an asynchronous API. Database interaction won't lock up the user interface. (Synchronous API is also available for WebWorkers.)
- Good search performance, since data can be indexed according to search keys.
- Robust, since it supports a transactional database model.
- Easier to maintain integrity of data, due to rigid data structure.

Disadvantages

- Deprecated. Will not be supported on IE or Firefox, and will probably be phased out from the other browsers at some stage.
- Steep learning curve, requiring knowledge of relational databases and SQL.
- Suffers from object-relational impedance mismatch.
- Diminishes agility, as database schema must be defined upfront, with all records in a table matching the same structure.

2. IndexedDB

It is the successor to both LocalStorage and WebSQL, designed to replace them as the "one true" browser database. It exposes an asynchronous API that supposedly avoids blocking the DOM, but as we'll see below, it doesn't necessarily live up to the hype. Browser support is extremely spotty, with only Chrome and Firefox having fully usable implementations.

Advantages

- Good performance generally, being an asynchronous API. Database interaction won't lock up the user interface. (Synchronous API is also available for WebWorkers.)
- Good search performance, since data can be indexed according to search keys.
- Supports versioning.

- Robust, since it supports a transactional database model.
- Fairly easy learning curve, due to a simple data model.
- Decent browser support: Chrome, Firefox, mobile FF, IE10.

Disadvantages

- Very complex API resulting in large amounts of nested callbacks.

[↑ back to top](#)

Q. Explain Application Cache in HTML5. OR What is a manifest file in HTML?

HTML5 provides an application caching mechanism that lets web-based applications run offline. Developers can use the Application Cache (AppCache) interface to specify resources that the browser should cache and make available to offline users. Applications that are cached load and work correctly even if users click the refresh button when they are offline.

Using an application cache gives an application the following benefits:

- **Offline browsing:** users can navigate a site even when they are offline.
- **Speed:** cached resources are local, and therefore load faster.
- **Reduced server load:** the browser only downloads resources that have changed from the server.

Syntax:

```
<html manifest="example.appcache">
  ...
</html>
```

Live demo

*Note: Using the application caching feature described here is at this point highly discouraged; it's in the process of being removed from the Web platform. Use **Service Workers** instead. In fact as of Firefox 44, when AppCache is used to provide offline support for a page a warning message is now displayed in the console advising developers to use Service workers instead (bug 1204581).*

[↑ back to top](#)

Q. What is the purpose of cache busting and how can you achieve it?

Browsers have a cache to temporarily store files on websites so they don't need to be re-downloaded again when switching between pages or reloading the same page. The server is set up to send headers that tell the browser to store the file for a given amount of time. This greatly increases website speed and preserves bandwidth.

However, it can cause problems when the website has been changed by developers because the user's cache still references old files. This can either leave them with old functionality or break a website if the cached CSS and JavaScript files are referencing elements that no longer exist, have moved or have been renamed.

Cache busting is the process of forcing the browser to download the new files. This is done by naming the file something different to the old file.

A common technique to force the browser to re-download the file is to append a query string to the end of the file.

```
<!-- src="js/script.js" => src="js/script.js?v=2" -->
<script src="js/script.js?v=2"></script>
```

The browser considers it a different file but prevents the need to change the file name.

[↑ back to top](#)

Q. What ARIA and screenreaders are, and how to make a website accessible?

Screen readers are software programs that provide assistive technologies that allow people with disabilities (such as no sight, sound or mouse-ing ability) to use web applications. You can make your sites more accessible by following ARIA standards such as semantic HTML, alt attributes and using [role=button] in the expected ways

[↑ back to top](#)

Q. How to use data- attribute in html5

To Do

[↑ back to top](#)

Q. What is the purpose of the alt attribute on images?

The alt attribute provides alternative information for an image if a user cannot view it. The alt attribute should be used to describe any images except those which only serve a decorative purposes, in which case it should be left empty.

```

```

[↑ back to top](#)

Q. What does enctype='multipart/form-data' mean?

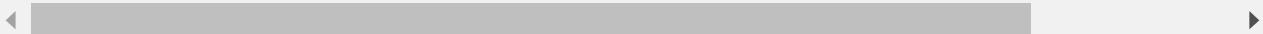
The enctype attribute specifies how the form-data should be encoded when submitting it to the server.

Example: 01

```
<form action="fileupload.php" method="post" enctype="multipart/form-data">
    <p>Please select the file you would like to upload.</p>
    <input type="file" name="upload">
    <br>
    <input type="submit" value="Upload File">
</form>
```

Example: 02

```
<form action="/urlencoded?token=A87412B" method="POST" enctype="application/x-www-fc
    <input type="text" name="username" value="" />
    <input type="text" name="password" value="" />
    <input type="submit" value="Submit" />
</form>
```



Example: 03

```
<form action="action.do" method="get" enctype="text/plain">
    Name: <input type="text" name="name" />
    Phone: <input type="number" name="phone" />
    <input type="submit" value="Submit" />
</form>
```

Sl.No	Value	Description
01.	application/x-www-form-urlencoded	Default. All characters are encoded before sent (spaces are converted to "+" symbols, and special characters are converted to ASCII HEX values)
02.	multipart/form-data	No characters are encoded. This value is required when you are using forms that have a file upload control
03.	text/plain	Spaces are converted to "+" symbols, but no special characters are encoded

[↑ back to top](#)

Q. What is difference between Select and Datalist?

For the select element, the user is required to select one of the options you've given. For the datalist element, it is suggested that the user select one of the options you've given, but he can actually enter anything he wants in the input.

Select

```
<select name="browser">
  <option value="firefox">Firefox</option>
  <option value="ie">IE</option>
  <option value="chrome">Chrome</option>
  <option value="opera">Opera</option>
  <option value="safari">Safari</option>
</select>
```

Datalist

```
<input type="text" list="browsers">
<datalist id="browsers">
  <option value="Firefox">
  <option value="IE">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
```

[↑ back to top](#)

Q. Explain some of the pros and cons for CSS animations versus JavaScript animations?

Regarding optimization and responsiveness the debate bounces back and forth but, the concept is:

- CSS animations allows the browser to choose where the animation processing is done, CPU or the GPU. (Central or Graphics Processing Unit)
- That said, adding many layers to a document will eventually have a performance hit.
- JS animation means more code for the user to download and for the developer to maintain.
- Applying multiple animation types on an element is harder with CSS since all transforming power is in one property transform
- CSS animations being declarative are not programmable therefore limited in capability.

[↑ back to top](#)

Q. What does CORS stand for and what issue does it address?

Cross-Origin Resource Sharing (CORS) is a W3C spec that allows cross-domain communication from the browser. By building on top of the XMLHttpRequest object, CORS allows developers to work with the same idioms as same-domain requests. CORS gives web servers cross-domain access controls, which enable secure cross-domain data transfers.

[↑ back to top](#)

Q. Can you describe the difference between progressive enhancement and graceful degradation?

- Graceful degradation is when you initially serve the best possible user experience, with all modern functionality, but use feature detection to "gracefully degrade" parts of your application with a fallback or polyfill.

- Progressive enhancement ensures a page works at the lowest expected abilities of browsers. So if you have a JavaScript web application that enhances a persons ability to send information to a database with features like ajax – at the very least you need to provide the ability for a person to send that same information without JavaScript enabled. In this case a simple form with full-page refresh will do what you need.

[↑ back to top](#)

Q. What is the DOM? How does the DOM work?

The DOM (Document Object Model) is a cross-platform API that treats HTML documents as a tree structure consisting of nodes. These nodes (such as elements and text nodes) are objects that can be programmatically manipulated and any visible changes made to them are reflected live in the document. In a browser, this API is available to JavaScript where DOM nodes can be manipulated to change their styles, contents, placement in the document, or interacted with through event listeners.

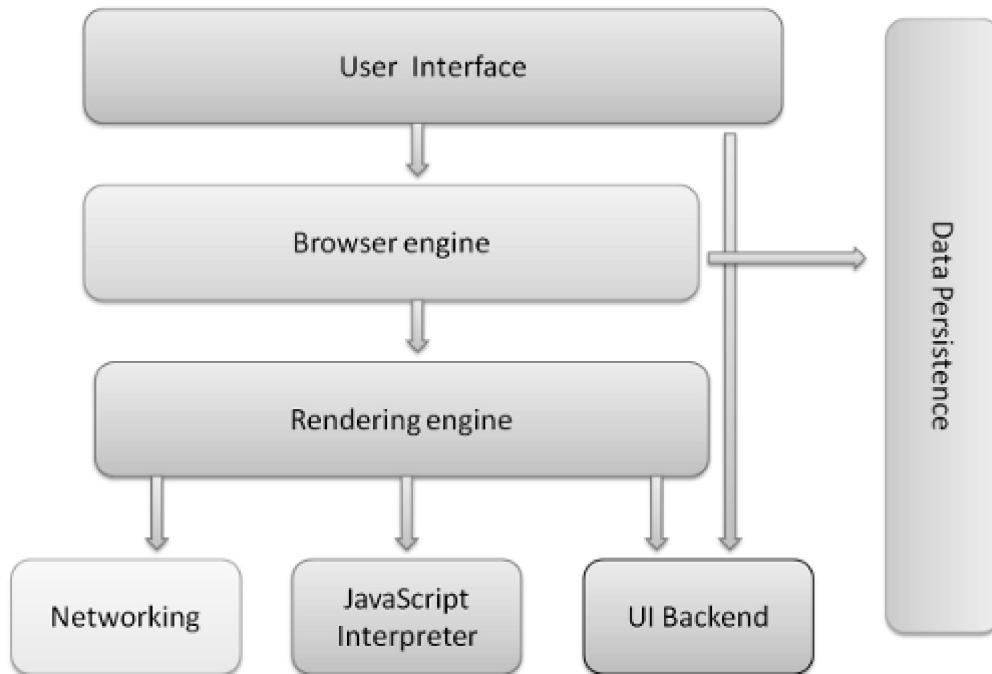
- The DOM was designed to be independent of any particular programming language, making the structural representation of the document available from a single, consistent API.
- `document.getElementById()` and `document.querySelector()` are common functions for selecting DOM nodes.
- Setting the `innerHTML` property to a new value runs the string through the HTML parser, offering an easy way to append dynamic HTML content to a node.

[↑ back to top](#)

Q. How does the browser rendering engine work?

In order to render content the browser has to go through a series of steps:

- Document Object Model(DOM)
- CSS object model(CSSOM)
- Render Tree
- Layout
- Paint



[↑ back to top](#)

Q. What is the difference between standards mode and quirks mode?

In **Quirks mode**, layout emulates nonstandard behavior in Navigator 4 and Internet Explorer 5. This is essential in order to support websites that were built before the widespread adoption of web standards. In **Standards mode**, the behavior is described by the HTML and CSS specifications.

For HTML documents, browsers use a `<!DOCTYPE html>` in the beginning of the document to decide whether to handle it in quirks mode or standards mode.

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset=UTF-8>
    <title>Hello World!</title>
  </head>
  <body>
  </body>
</html>
  
```

[↑ back to top](#)

Q. What is Critical Rendering Path?

- Constructing the DOM Tree
- Constructing the CSSOM Tree
- Running JavaScript - parser blocking resource
- Creating the Render Tree
- Generating the Layout
- Painting

[↑ back to top](#)

Q. What are the Benefits of Server Side Rendering (SSR) Over Client Side Rendering (CSR)?

- We are using server side rendering for two reasons:
 - performance benefit for our customers
 - Consistent SEO performance
- The main difference is that for SSR your server's response to the browser is the HTML of your page that is ready to be rendered, while for CSR the browser gets a pretty empty document with links to your javascript. That means for SSR your browser will start rendering the HTML from your server without having to wait for all the JavaScript to be downloaded and executed.
- for SSR, the user can start viewing the page while all of that is happening. For the CSR world, you need to wait for all of the above to happen and then have the virtual dom moved to the browser dom for the page to be viewable.

[↑ back to top](#)

Q. Name 3 ways to decrease page load?

1. LocalStorage
2. Caching resources
3. DNS-prefetch (sample below)
4. Keep resources on a CDN

[↑ back to top](#)

Q. Ways to improve website performance

- Minimize HTTP Requests
 - Sites are mainly slow because of too many (or too large) HTTP requests. We can eliminate unnecessary request;
 - combined files: js to a file, css to a file
 - CSS sprites: CSS Sprites are the preferred method for reducing the number of image requests. Combine your background images into a single image and use the CSS background-image and background-position properties to display the desired image segment.
- Use a Content Delivery Network CDN
 - A CDN is essentially many optimized servers around the world that deliver web content to users based on their geographic location. This means big performance improvements for site users. Because, say, if a person accessing your site in India, they will be retrieving web content from a server nearby
- Optimize Images:
 - image sizes make a huge difference to site speed. The larger content/images, the slower the site. we could:
 - Changing the resolution: reducing the “quality” of the image (and thereby the file size)
 - Compressing the picture: increasing the efficiency of image data storage
 - Cropping the picture: when cropping, you are cutting out unneeded areas and thus making the image smaller in size
- Put Scripts at the Bottom:
 - Javascript files can load after the rest of your page. The simplest solution is to place your external Javascript files at the bottom of your page, just before the close of your body tag. Now more of your site can load before your scripts. Another method that allows even more control is to use the defer or async attributes when placing external .js files on your site.
 - Async tags load the scripts while the rest of the page loads, but this means scripts can be loaded out of order. Basically, lighter files load first. This might be fine for some scripts, but can be disastrous for others.
 - The defer attribute loads your scripts after your content has finished loading. It also runs the scripts in order. Just make sure your scripts run so late without breaking your site.

- Add an Expires or a Cache-Control Header
 - Web page designs are getting richer and richer, which means more scripts, stylesheets, images, and Flash in the page. A first-time visitor to your page may have to make several HTTP requests, but by using the Expires header you make those components cacheable. This avoids unnecessary HTTP requests on subsequent page views. Expires headers are most often used with images, but they should be used on all components including scripts, stylesheets, and Flash components.
- Gzip Components
 - Compression reduces response times by reducing the size of the HTTP response. Gzipping generally reduces the response size by about 70%.
- Put Stylesheets at the Top:
 - This is because putting stylesheets in the HEAD allows the page to render progressively.
- Avoid CSS Expressions
- Use GET for AJAX Requests:
 - Ajax is that it provides instantaneous feedback to the user because it requests information asynchronously from the backend web server
- Make JavaScript and CSS External:
 - Using external files in the real world generally produces faster pages because the JavaScript and CSS files are cached by the browser. JavaScript and CSS that are inlined in HTML documents get downloaded every time the HTML document is requested. This reduces the number of HTTP requests that are needed, but increases the size of the HTML document. On the other hand, if the JavaScript and CSS are in external files cached by the browser, the size of the HTML document is reduced without increasing the number of HTTP requests.
- Use get to ajax request:
 - POST is implemented in the browsers as a two-step process: sending the headers first, then sending data. So it's best to use GET, which only takes one TCP packet to send (unless you have a lot of cookies).
- No 404s:

- HTTP requests are expensive so making an HTTP request and getting a useless response (i.e. 404 Not Found) is totally unnecessary and will slow down the user experience without any benefit.
- Reduce Cookie Size:
 - HTTP cookies are used for a variety of reasons such as authentication and personalization. Information about cookies is exchanged in the HTTP headers between web servers and browsers. It's important to keep the size of cookies as low as possible to minimize the impact on the user's response time.
- Reduce DNS Lookups
- Minify JavaScript and CSS
- Avoid Redirects
- Remove Duplicate Scripts
- Configure Etags
- Make Ajax Cacheable
- Post-load Components
- Preload Components
- Reduce the Number of DOM Elements
- Minimize the Number of iframes
- Minimize DOM Access
- Optimize CSS Sprites
- Don't Scale Images in HTML
- Make favicon.ico Small and Cacheable
- Avoid Empty Image src

[↑ back to top](#)

Q. Comparison of browsers engines like Chrome, Firefox, Internet explorer, Safari?

- Chrome:
 - Layout rendering engine **Webkit**.
 - JavaScript engine **V8**
- Firefox:
 - Layout rendering engine **Gecko**.
 - JavaScript engine **Spider monkey**
- Internet explorer:
 - Layout rendering engine **Trident**.
 - JavaScript engine **Chakra**
- Safari:
 - Layout rendering engine **Webkit**.
 - JavaScript engine JavascriptCore i.e **Nitro**

[↑ back to top](#)

Q. What does the lang attribute in html do?

- Helps in styling pages by using them in css `:lang()` pseudo class Spelling and grammar checkers Languade detection by search engines

[↑ back to top](#)

Q. What is desktop first and mobile first design approach?

- Desktop first : General selectors and styles designed to make the site look good on DESKTOP screens defined globally. But they affect all devices, and must be overridden by max-width media queries targeting minimum screen size
- Mobile First : General selectors and styles designed to make the site look good on small MOBILE screens go here. But they affect all devices, and must be overridden by min-width media queries targeting maximum scrren size

In desktop first approach the media queries will be written with respect to max-width whereas in mobile first approach media queries will be written with respect to min-width

[↑ back to top](#)

Q. *What are data- attributes good for?*

- The HTML5 data attribute lets you assign custom data to an element. When we want to store more information/data about the element when no suitable HTML5 element or attribute exists

[↑ back to top](#)

Q. *Explain the difference between layout, painting and compositing?*



- **JavaScript:** Typically JavaScript is used to handle work that will result in visual changes, whether it's jQuery's animate function, sorting a data set, or adding DOM elements to the page. It doesn't have to be JavaScript that triggers a visual change, though: CSS Animations, Transitions, and the Web Animations API are also commonly used.
- **Style calculations:** This is the process of figuring out which CSS rules apply to which elements based on matching selectors, for example, .headline or .nav > .nav__item. From there, once rules are known, they are applied and the final styles for each element are calculated.
- **Layout:** Once the browser knows which rules apply to an element it can begin to calculate how much space it takes up and where it is on screen. The web's layout model means that one element can affect others, for example the width of the `<body>` element typically affects its children's widths and so on all the way up and down the tree, so the process can be quite involved for the browser.
- **Paint:** Painting is the process of filling in pixels. It involves drawing out text, colors, images, borders, and shadows, essentially every visual part of the elements. The drawing is typically done onto multiple surfaces, often called layers.

- **Compositing:** Since the parts of the page were drawn into potentially multiple layers they need to be drawn to the screen in the correct order so that the page renders correctly. This is especially important for elements that overlap another, since a mistake could result in one element appearing over the top of another incorrectly.

[↑ back to top](#)

Q. Explain about HTML Layout Engines used by browsers?

Engine	Status	Embedded in
WebKit	Active	Safari browser, plus all browsers hosted on the iOS App Store
Blink	Active	Google Chrome and all other Chromium-based browsers like Opera and Microsoft Edge
Gecko	Active	Firefox browser and Thunderbird email client, plus forks like SeaMonkey and Waterfox
KHTML	Discontinued	Konqueror browser
Presto	Discontinued	formerly in the Opera browser
EdgeHTML	Discontinued	formerly in the Microsoft Edge browser
Trident	Discontinued	Internet Explorer browser and Microsoft Outlook email client

[↑ back to top](#)

Q. How to make page responsive?

Responsive Web Design is about using HTML and CSS to automatically resize, hide, shrink, or enlarge, a website, to make it look good on all devices (desktops, tablets, and phones).

1. Setting the viewport

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

2. Responsive Images

If the CSS width property is set to 100%, the image will be responsive and scale up and down

```

```

3. Show different Images depending on Browser Width

The HTML `<picture>` element allows you to define different images for different browser window sizes.

```
<picture>
  <source srcset="img_small.jpg" media="(max-width: 600px)">
  <source srcset="img_large.jpg" media="(max-width: 1500px)">
  <source srcset="img.jpg">
  
</picture>
```

4. Responsive Text Size

The text size can be set with a "vw" unit, which means the "viewport width". That way the text size will follow the size of the browser window.

```
<h1 style="font-size:10vw">Hello World</h1>
```

5. Media Queries

Using media queries you can define completely different styles for different browser sizes.

```
/* Use a media query to add a breakpoint at 800px: */
@media screen and (max-width: 800px) {
  .left, .main, .right {
    width: 100%; /* The width is 100%, when the viewport is 800px or smaller */
  }
}
```

[↑ back to top](#)

Q. Does the following trigger http request at the time of page load?

```


<div style="display: none;">
    
</div>
```

- Yes

[↑ back to top](#)

Q. *List the API available in HTML5?*

1. High Resolution Time API

The High Resolution Time API provides the current time in sub-millisecond resolution and such that it is not subject to system clock skew or adjustments.

It exposes only one method, that belongs to the `window.performance` object, called `now()`. It returns a `DOMHighResTimeStamp` representing the current time in milliseconds. The timestamp is very accurate, with precision to a thousandth of a millisecond, allowing for accurate tests of the performance of our code.

```
var time = performance.now();
```

2. User Timing API

It allows us to accurately measure and report the performance of a section of JavaScript code. It deals with two main concepts: mark and measure. The former represents an instant (timestamp), while the latter represents the time elapsed between two marks.

```
performance.mark("startFoo");
// A time consuming function
foo();
performance.mark("endFoo");

performance.measure("durationFoo", "startFoo", "endFoo");
```

3. Network Information API

This API belongs to the `connection` property of the `window.navigator` object. It exposes two read-only properties: `bandwidth` and `metered`. The former is a number representing an estimation of the current bandwidth, while the latter is a Boolean whose value is true if the user's connection is subject to limitation and bandwidth usage, and false otherwise.

Sl.No	API	Description
01.	<code>navigator.connection.type</code>	Network Type
02.	<code>navigator.connection.downlink</code>	Effective bandwidth estimate (downlink)
03.	<code>navigator.connection.rtt</code>	Effective round-trip time estimate (rtt)
04.	<code>navigator.connection.downlinkMax</code>	Upper bound on the downlink speed of the first network hop (downlinkMax)
05.	<code>navigator.connection.effectiveType</code>	Effective connection type
06.	<code>navigator.connection.saveData</code>	True if the user has requested a reduced data usage mode from the user agent (saveData)

4.) Vibration API

It exposes only one method, `vibrate()`, that belongs to the `window.navigator` object. This method accepts one parameter specifying the duration of the vibration in milliseconds. The parameter can be either an integer or an array of integers. In the second case, it's interpreted as alternating vibration times and pauses.

```
// Vibrate once for 2 seconds
navigator.vibrate(2000);
```

5.) Battery Status API

The Battery Status API exposes four properties (`charging` , `chargingTime` , `dischargingTime` , and `level`) and four events. The properties specify if the battery is in charge, the seconds remaining until the battery is fully charged, the seconds remaining until the battery is fully discharged, and the current level of the battery. These properties belongs to the `battery` property of the `window.navigator` object.

```
// Retrieves the percentage of the current level of the device's battery
var percentageLevel = navigator.battery.level * 100;
```

6.) Page Visibility API

The Page Visibility API enables us to determine the current visibility state of the page. The Page Visibility API is especially useful for saving resources and improving performance by letting a page avoid performing unnecessary tasks when the document isn't visible.

```
//document.hidden retuns true if page is not visible.
console.log('Page Visibility: '+document.hidden);
```

7.) Fullscreen API

The Fullscreen API provides a way to request fullscreen display from the user, and exit this mode when desired. This API exposes two methods, `requestFullscreen()` and `exitFullscreen()`, allowing us to request an element to become fullscreen and to exit fullscreen.

```
document.addEventListener("keypress", function(e) {
    if (e.keyCode === 13) { // Enter Key
        toggleFullScreen();
    }
}, false);

function toggleFullScreen() {
    if (!document.fullscreenElement) {
        document.documentElement.requestFullscreen();
    } else {
        if (document.exitFullscreen) {
            document.exitFullscreen();
        }
    }
}
```

Live demo

[↑ back to top](#)

Q. How geolocation api works in html5?

The Geolocation API allows the user to provide their location to web applications if they so desire. For privacy reasons, the user is asked for permission to report location information.

The Geolocation API is published through the `navigator.geolocation` object.

```
if ("geolocation" in navigator) {  
    /* geolocation is available */  
} else {  
    /* geolocation IS NOT available */  
}
```

Example

```
<!DOCTYPE html>  
<html>  
    <head>  
        <title>Geolocation</title>  
    </head>  
    <body>  
        <p><button onclick="geoFindMe()">Show my location</button></p>  
        <div id="out"></div>  
    </body>  
  
<script type="text/javascript">  
/**  
     * The Geolocation API allows the user to provide their location to web applications  
     * if they so desire. For privacy reasons, the user is asked for permission to  
     * report location information.  
*/  
function geoFindMe() {  
    var output = document.getElementById("out");  
  
    if (!navigator.geolocation){  
        output.innerHTML = "<p>Geolocation is not supported by your browser</p>";  
        return;  
    }  
  
    function success(position) {  
        var latitude = position.coords.latitude;  
        var longitude = position.coords.longitude;  
  
        output.innerHTML = '<p>Latitude is ' + latitude + '° <br>Longitude is ' + longitude + '</p>';  
        var img = new Image();  
        img.src = "https://maps.googleapis.com/maps/api/staticmap?center=" + latitude + "," + longitude + "&zoom=12&size=400x300";  
        output.appendChild(img);  
    }  
    navigator.geolocation.getCurrentPosition(success);  
}  
</script>
```

```

        }

        function error() {
            output.innerHTML = "Unable to retrieve your location";
        }

        output.innerHTML = "<p>Locating...</p>";

        navigator.geolocation.getCurrentPosition(success, error); //function to get
    }
</script>
</html>

```



Live demo

[↑ back to top](#)

Q. *What is the use of WebSocket API?*

The **WebSocket API** is an advanced technology that makes it possible to open a two-way interactive communication session between the user's browser and a server. With this API, you can send messages to a server and receive event-driven responses without having to poll the server for a reply.

Interfaces

Sl.No	API	Description
01.	WebSocket	The primary interface for connecting to a WebSocket server and then sending and receiving data on the connection.
02.	CloseEvent	The event sent by the WebSocket object when the connection closes.
03.	MessageEvent	The event sent by the WebSocket object when a message is received from the server.

Example

```

// Create WebSocket connection.
const socket = new WebSocket('ws://localhost:8080/');

// Connection opened
socket.addEventListener('open', function(event) {

```

```

        socket.send('Hello Server!');
    });

    // Listen for messages
    socket.addEventListener('message', function(event) {
        console.log('Message from server ', event.data);
    });

```

[↑ back to top](#)

Q. Explain about HTML Canvas?

canvas is an HTML element which can be used to draw graphics via JavaScript. This can, for instance, be used to draw graphs, combine photos, or create animations.

Colors, Styles, and Shadows

Property	Description
fillStyle	Sets or returns the color, gradient, or pattern used to fill the drawing
strokeStyle	Sets or returns the color, gradient, or pattern used for strokes
shadowColor	Sets or returns the color to use for shadows
shadowBlur	Sets or returns the blur level for shadows
shadowOffsetX	Sets or returns the horizontal distance of the shadow from the shape
shadowOffsetY	Sets or returns the vertical distance of the shadow from the shape

Line Styles

Property	Description
lineCap	Sets or returns the style of the end caps for a line
lineJoin	Sets or returns the type of corner created, when two lines meet
lineWidth	Sets or returns the current line width
miterLimit	Sets or returns the maximum miter length

Rectangles

Method	Description
rect()	Creates a rectangle
fillRect()	Draws a "filled" rectangle
strokeRect()	Draws a rectangle (no fill)
clearRect()	Clears the specified pixels within a given rectangle

Paths

Method	Description
fill()	Fills the current drawing (path)
stroke()	Actually draws the path you have defined
beginPath()	Begins a path, or resets the current path
moveTo()	Moves the path to the specified point in the canvas, without creating a line
closePath()	Creates a path from the current point back to the starting point
lineTo()	Adds a new point and creates a line to that point from the last specified point in the canvas
clip()	Clips a region of any shape and size from the original canvas
arc()	Creates an arc/curve (used to create circles, or parts of circles)
arcTo()	Creates an arc/curve between two tangents

Transformations

Method	Description
scale()	Scales the current drawing bigger or smaller
rotate()	Rotates the current drawing
translate()	Remaps the (0,0) position on the canvas
transform()	Replaces the current transformation matrix for the drawing
setTransform()	Resets the current transform to the identity matrix. Then runs transform()

Text

Property	Description
font	Sets or returns the current font properties for text content
textAlign	Sets or returns the current alignment for text content
textBaseline	Sets or returns the current text baseline used when drawing text
fillText()	Draws "filled" text on the canvas
strokeText()	Draws text on the canvas (no fill)
measureText()	Returns an object that contains the width of the specified text

```

<!DOCTYPE html>
<html>
  <head>
    <title>HTML5 Canvas</title>
  </head>
  <body>

    <div><b>Text</b></div>
    <canvas id="text" width="200" height="100" ></canvas><br/><br/>

    <div><b>Straight Line</b></div>
    <canvas id="line" width="300" height="0" style="border: 1px solid #333;"></canvas><br/><br/>

    <div><b>Rectangle</b></div>
    <canvas id="rectangle" width="300" height="200" style="border: 1px solid #999;"></canvas><br/><br/>

    <div><b>Rectangle with Style</b></div>
    <canvas id="rect">This browser does not support Canvas!</canvas><br/><br/>

    <div><b>Circle</b></div>
    <canvas id="circle">This browser does not support Canvas!</canvas><br/><br/>

<script type="text/javascript">
  // Text with style
  var canvas = document.getElementById('text');
  var context = canvas.getContext('2d');
  context.font = '20pt Calibri';

```

```
context.fillStyle = 'blue';
context.fillText('Hello World!', 50, 50);

// Straight Line
var canvas = document.getElementById("line");
var context = canvas.getContext("2d");
context.moveTo(50, 150);
context.lineTo(250, 50);
context.stroke();

// Rectange with style
var canvas = document.getElementById("rect");
var context = canvas.getContext("2d");
context.fillStyle = "#FF0000";
context.fillRect(0, 0, 300, 200);

// Circle
var canvas = document.getElementById("circle");
var context = canvas.getContext("2d");
context.beginPath();
context.arc(95, 50, 40, 0, 2 * Math.PI);
context.stroke();
</script>
</body>
</html>
```



Live demo

[↑ back to top](#)

Q. What is difference between SVG and Canvas?

SVG

The Scalable Vector Graphics (SVG) is an XML-based image format that is used to define two-dimensional vector based graphics for the web. Unlike raster image (e.g. .jpg, .gif, .png, etc.), a vector image can be scaled up or down to any extent without losing the image quality.

There are following advantages of using SVG over other image formats like JPEG, GIF, PNG, etc.

- SVG images can be searched, indexed, scripted, and compressed.

- SVG images can be created and modified using JavaScript in real time.
- SVG images can be printed with high quality at any resolution.
- SVG content can be animated using the built-in animation elements.
- SVG images can contain hyperlinks to other documents.

Example:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      #svgelem {
        position: relative;
        left: 50%;
        -webkit-transform: translateX(-20%);
        -ms-transform: translateX(-20%);
        transform: translateX(-20%);
      }
    </style>
    <title>HTML5 SVG</title>
  </head>
  <body>
    <h2 align="center">HTML5 SVG Circle</h2>
    <svg id="svgelem" height="200" xmlns="http://www.w3.org/2000/svg">
      <circle id="bluecircle" cx="60" cy="60" r="50" fill="blue" />
    </svg>
  </body>
</html>
```

Live demo

Canvas

Canvas is a HTML element used to draw graphics on a web page. It is a bitmap with an "immediate mode" graphics application programming interface (API) for drawing on it. The `<canvas>` element is only a container for graphics. In order to draw the graphics, you are supposed to use a script. Canvas has several strategies when it comes to drawing paths, boxes, circles, text & adding images.

Example:

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

```

<title>HTML5 Canvas Tag</title>
</head>
<body>
    <canvas id="newCanvas" width="200" height="100" style="border:1px solid #000000">
        <script>
            var c = document.getElementById('newCanvas');
            var ctx = c.getContext('2d');
            ctx.fillStyle = '#7cce2b';
            ctx.fillRect(0,0,300,100);
        </script>
    </body>
</html>

```



Live demo

Differences between SVG and Canvas

SVG	Canvas
Vector based (composed of shapes)	Raster based (composed of pixels)
Multiple graphical elements, which become the part of the page's DOM tree	Single element similar to <code>div</code> in behavior. Canvas diagram can be saved to PNG or JPG format
Modified through script and CSS	Modified through script only
Good text rendering capabilities	Poor text rendering capabilities
Give better performance with smaller number of objects or larger surface, or both	Give better performance with larger number of objects or smaller surface, or both
Better scalability. Can be printed with high quality at any resolution. Pixelation does not occur	Poor scalability. Not suitable for printing on higher resolution. Pixelation may occur

[↑ back to top](#)

Q. Explain Drag and Drop in HTML5?

HTML5 drag-and-drop uses the DOM event model and drag events inherited from mouse events. A typical drag operation begins when a user selects a draggable element, drags the element to a droppable element, and then releases the dragged element.

Event	Description
Drag	It fires every time when the mouse is moved while the object is being dragged.
Dragstart	It is a very initial stage. It fires when the user starts dragging object.
Dragenter	It fires when the user moves his/her mouse cursor over the target element.
Dragover	This event is fired when the mouse moves over an element.
Dragleave	This event is fired when the mouse leaves an element.
Drop	Drop It fires at the end of the drag operation.
Dragend	It fires when user releases the mouse button to complete the drag operation.

Example

```
<!DOCTYPE HTML>
<html>
    <head>
        <script>
            function allowDrop(ev) {
                ev.preventDefault();
            }

            function drag(ev) {
                ev.dataTransfer.setData("text", ev.target.id);
            }

            function drop(ev) {
                ev.preventDefault();
                var data = ev.dataTransfer.getData("text");
                ev.target.appendChild(document.getElementById(data));
            }
        </script>
    </head>
    <body>
        <div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
        
    </body>
</html>
```



[Live demo](#)[↑ back to top](#)

Q. Explain Microdata in HTML5?

Microdata is a standardized way to provide additional semantics in web pages. Microdata lets you define your own customized elements and start embedding custom properties in your web pages. At a high level, microdata consists of a group of name-value pairs.

The groups are called **items**, and each name-value pair is a **property**. Items and properties are represented by regular elements. Search engines benefit greatly from direct access to this structured data because it allows search engines to understand the information on web pages and provide more relevant results to users.

At a high level, microdata consists of a group of name-value pairs

- **itemscope**:- To create an item
- **itemprop**:- To add a property to an item

Example

```
<div itemscope>
  <p>My name is <span itemprop="name">Elizabeth</span>.</p>
</div>

<div itemscope>
  <p>My name is <span itemprop="name">Daniel</span>.</p>
</div>
```

[↑ back to top](#)

Q. What are the HTML tags which deprecated in HTML5?

Deprecated Tags

The following elements are not available in HTML5 anymore and their function is better handled by CSS.

Sl.No	Tags (Elements)	Description
01.	<acronym>	Defines an acronym

Sl.No	Tags (Elements)	Description
02.	<applet>	Defines an applet
03.	<basefont>	Defines an base font for the page.
04.	<big>	Defines big text
05.	<center>	Defines centered text
06.	<dir>	Defines a directory list
07.		Defines text font, size, and color
08.	<frame>	Defines a frame
08.	<frameset>	Defines a set of frames
10.	<isindex>	Defines a single-line input field
11.	<noframes>	Defines a noframe section
12.	<s>	Defines strikethrough text
13.	<strike>	Defines strikethrough text
14.	<tt>	Defines teletype text
15.	<u>	Defines underlined text

Deprecated Attributes

Removed Attributes	From the Elements
rev	link, a
charset	link and a
shape	a
coords	a
longdesc	img and iframe.
target	link
nohref	area
profile	head

Removed Attributes	From the Elements
version	html
name	img
scheme	meta
archive	object
classid	object
codebase	object
codetype	object
declare	object
standby	object
valuetype	param
type	param
axis	td and t
abbr	td and t
scope	td
align	caption, iframe, img, input, object, legend, table, hr, div, h1, h2, h3, h4, h5, h6, p, col, colgroup, tbody, td, tfoot, th, thead and tr.
alink	body
link	body
vlink	body
text	body
background	body
bgcolor	table, tr, td, th and body.
border	table and object.
cellpadding	table
cellspacing	table

Removed Attributes	From the Elements
char	col, colgroup, tbody, td, tfoot, th, thead and tr.
charoff	col, colgroup, tbody, td, tfoot, th, thead and tr.
clear	br
compact	dl, menu, ol and ul.
frame	table
compact	dl, menu, ol and ul.
frame	table
frameborder	iframe
hspace	img and object.
vspace	img and object.
marginheight	iframe
marginwidth	iframe
noshade	hr
nowrap	td and th
rules	table
scrolling	iframe
size	hr
type	li, ol and ul.
valign	col, colgroup, tbody, td, tfoot, th, thead and tr
width	hr, table, td, th, col, colgroup and pre.

[↑ back to top](#)

Q. How you can Use Modernizr in HTML5?

Modernizr is a JavaScript library that detects which HTML5 and CSS3 features visitor's browser supports. In detecting feature support, it allows developers to test for some of the new technologies and then provide fallbacks for browsers that do not support them. This is called **feature detection** and is much more efficient than browser sniffing.

Using Modernizr with CSS

By default, Modernizr sets classes for all of your tests on the root element (`<html>` for websites). This means adding the class for each feature when it is supported, and adding it with a `no-` prefix when it is not (e.g. `.feature` or `.no-feature`).

```
.no-cssgradients .header {  
    background: url("images/glossybutton.png");  
}  
  
.cssgradients .header {  
    background-image: linear-gradient(cornflowerblue, rebeccapurple);  
}
```

Using Modernizr with JavaScript

Modernizr keeps track of the results of all of its feature detections via the `Modernizr` object.

```
if (Modernizr.canvas) {  
    alert("This browser supports HTML5 canvas!");  
} else {  
    alert("no canvas :(");  
}
```

[↑ back to top](#)

Q. *What is progressive rendering?*

Progressive Rendering is the technique of sequentially rendering portions of a webpage in the server and streaming it to the client in parts without waiting for the whole page to rendered.

It implies that once the important material is produced on the server, you may stream it to the client without having to wait for non-critical stuff to be displayed. It combines the advantages of both CSR (Client Side Rendering) and SSR (Server Side Rendering) (Server Side Rendering).

Client Side Rendering

Client Side Rendering (CSR) is a technique in which content is rendered in the browser using JavaScript. Instead of getting all the content from the HTML file itself, the server sends HTML with an empty body and script tags that contain links to JavaScript bundles that the browser will use to render the content.

Typical page load behaviour in CSR —

- Browser requests the server for HTML
- Server sends HTML with script tags in head and no content in body
- Browser parses the HTML and makes http requests to load the scripts
- Once the scripts are loaded, the browser parses them and makes API requests and loads all the content asynchronously

Since the all the content starts loading only after loading the initial JavaScript, it takes a longer time to show any content on the page. If the user is on a slow network, the content is blocked for an even longer time due to lower bandwidth and higher latency.

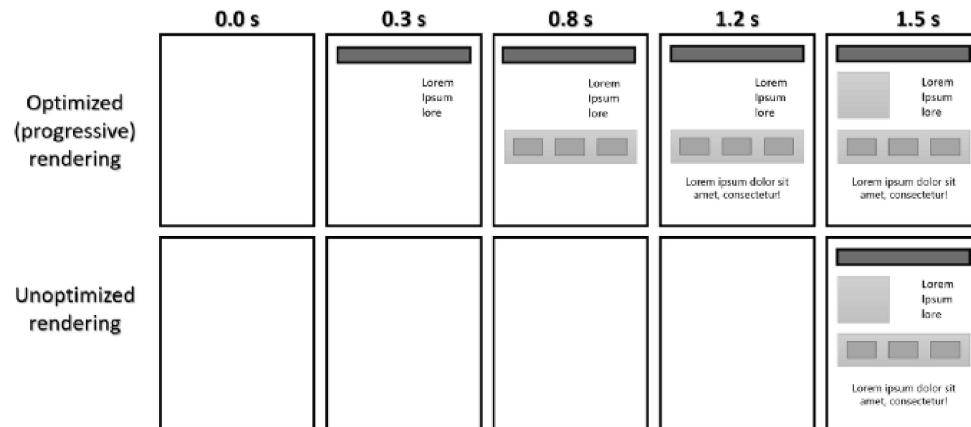
Server Side Rendering

When rendering on the server side, the HTML is rendered on the server and sent to the client. The content that we need to display on the screen becomes available immediately after the HTML is parsed; hence, primary rendering of content is faster than CSR.

Typical page load behaviour in SSR —

- Browser requests the server for HTML.
- Server makes API requests (usually co-located) and renders the content in the server.
- Once the page is ready, the server sends it to the browser.
- The browser loads and parses the HTML and paints the content on the screen without waiting for the JavaScript bundle(s) to load.
- Once the JavaScript bundle(s) are loaded, the browser hydrates interactivity to DOM elements, which is usually attaching event handlers and other interactive behaviours.

Since the APIs are usually co-located with the server, the content is loaded super fast (faster than CSR) and the HTML is sent to the browser. Initial JavaScript load doesn't block content load as the HTML sent by the server already has the content.



[⬆ back to top](#)

Q. What is an iframe and how it works?

The `<iframe>` HTML element represents a nested browsing context, embedding another HTML page into the current one. Each embedded browsing context has its own **session history** and **document**. The browsing context that embeds the others is called the parent browsing context. The topmost browsing context — the one with no parent — is usually the browser window, represented by the **Window** object.

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML5 iframe</title>
  </head>
  <style type="text/css">
    iframe {
      border: 1px solid #333;
      width: 50%;
    }
    .output {
      background: #eee;
    }
  </style>
  <body>
    <p>The Inline iFrame Example</p>
    <iframe id="inlineFrameId"
      title="Inline iFrame Example">
    </iframe>
  </body>
</html>
```

```

width="300"
height="200"
src="https://www.openstreetmap.org/export/embed.html?bbox=-0.00401794910430908
Sorry your browser does not support inline frames.
</iframe>
</body>
</html>

```



Live demo

The Iframe Tag Attributes

Attribute	Description
allow	indicates what features the iframe is allowed to use (e.g. fullscreen, camera, autoplay)
allowfullscreen	grants or denies permission for the iframe to appear in full-screen mode
height	sets the height of the iframe (if not specified, the default height is 150 pixels)
loading	sets lazy loading or eager loading for the iframe
referrerpolicy	sets what referrer information should be sent in the request for the iframe
src	the address of the resource included in the iframe
width	sets the width of the iframe (if not specified, the default width is 300 pixels)

Note: Because each browsing context is a complete document environment, every <iframe> in a page requires increased memory and other computing resources.

[↑ back to top](#)

Q. Explain the use of rel="nofollow", rel="noreferrer", rel="noopener" attribute?

1. rel="nofollow"

When `rel="nofollow"` tag is used, it instruct the search engines not to pass any PageRank from one page to the other. It does not allow it to pass the authority to the specific website. The main advantage of using this attribute is to control the spam attack.

There may be times, when you do not have control over what people publish on your websites, for example some blog comments and some kind of forum posting.

Example

```
<a href="https://www.website.com" rel="nofollow">Link to yoursite.com</a>
```

2. `rel="noreferrer"`

Noreferrer is related to analytics and tracking. The referrer value shows the previous page where a user came from. By using the noreferrer attribute on a link, you are preventing other pages from seeing that traffic came from a click on your link.

Example

```
<a href="https://www.website.com" rel="noreferrer">Link to yoursite.com</a>
```

3. `rel="noopener"`

It prevents the new page from being able to access the `window.opener` property and will make it run in a separate process.noopener tag works as a security fix which prevents malicious links to take control over an opened tab.

Example

```
<a href="https://www.website.com" target="_blank" rel="noopener">Link to yoursite.co
```



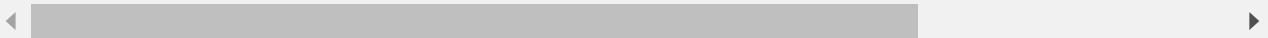
↑ back to top

Q. How can you highlight text in HTML?

The `<mark>` HTML element represents text which is marked or highlighted for reference or notation purposes, due to the marked passage's relevance or importance in the enclosing context.

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Highlight text in HTML</title>
  </head>
  <body>
    <p>Search results for "salamander":</p>
    <hr>
    <p>Several species of <mark>salamander</mark> inhabit the temperate rainforest c
      <p>Most <mark>salamander</mark>s are nocturnal, and hunt for insects, worms, and
    </body>
</html>
```



Live demo

[⬆ back to top](#)

Q. How can I get indexed better by search engines?

HTML tags are used to influence the way our pages appear in search results. With the help of certain tags, we can turn regular search snippets into rich snippets, and maybe even into featured snippets. And, as our search snippets get more advanced, they are able to secure better **Search Engine Results Pages (SERP)** positions and attract more traffic.

Here are all the HTML tags that still matter:

1. Title tag

Title tags are used by search engines to determine the subject of a page and display it in SERP. As a rule of thumb, titles that are under 60 characters long will fit on most screens. In HTML, a title tag looks like this:

```
<title>Your Title Goes Here</title>
```

2. Meta description tag

Meta description is a short paragraph of text used to describe your page in search results. The function of meta description is similar to the title. It provides a little more detail about your page and it helps users decide whether to visit your page or not. In HTML, a meta description tag looks like this:

```
<meta name="description" content="Your description goes here">
```

3. Heading tags

Headings (H1-H6) are used to split your page into sections or chapters. Each heading is like a small title within the page. In HTML, a heading looks like this:

```
<h1>Your heading goes here</h1>
```

4. Image alt attribute

The `alt` text attribute is a part of an image tag, and it provides a description for an image. Alt text plays a major role in image optimization. It makes your images accessible both to search engines (by telling them what a particular image means) and to people (by displaying an alternative text in case a particular image cannot be loaded or by helping screen readers convey images). In HTML it may look like this:

```

```

5. Open Graph tags

Open Graph (OG) tags are placed in the `<head>` section of a page and allow any webpage to become a rich object in social networks. OG tags let you control how the information about your page is represented when shared via social channels. This possibility may help you enhance the performance of your links on social media, thus driving more click-throughs and increasing conversions. In HTML, it can look like this:

```
<meta name="og:title" property="og:title" content="Your Open Graph Title Goes Here">
```



6. Robots tag

A robots tag is an element in the HTML of a page that informs search engines which pages on your site should be indexed and which should not. Its functions are similar to robots.txt, but robots.txt gives suggestions. Whereas robots tags give instructions. In HTML, it can look like this:

```
<meta name="robots" content="index, follow">
```

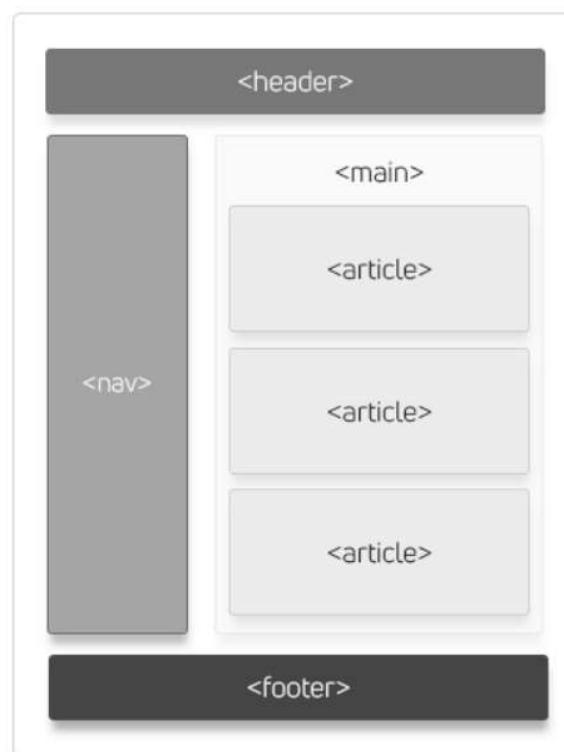
7. Canonical tag

A canonical tag is a way of telling search engines that a specific URL represents the master copy of a page. Using the canonical tag prevents problems caused by identical or "duplicate" content appearing on multiple URLs. Practically speaking, the canonical tag tells search engines which version of a URL you want to appear in search results. In HTML, it may look like this:

```
<link href="URL" rel="canonical">
```

8. HTML5 semantic tags

One of the most important features of HTML5 is its semantics tags. Semantic tags refers to syntax that makes the HTML more comprehensible by better defining the different sections and layout of web pages. It makes web pages more informative and adaptable, allowing browsers and search engines to better interpret content. For example, instead of using `<div id="header"></div>` you can use a `<header></header>` tag.



[↑ back to top](#)

Q. What is the difference between an "attribute" and a "property" in HTML?

Attributes are defined by HTML. Properties are accessed from DOM (Document Object Model) nodes.

Example

```
<input id="inputId" type="text" value="Hello World!" />
```

The **value** property reflects the current text-content inside the input box, whereas the **value** attribute contains the initial text-content of the **value** attribute from the HTML source code

Difference between HTML attributes and DOM properties:

Attribute	Property
Attributes are defined by HTML.	Properties are defined by the DOM.
The value of an attribute is constant.	The value of a property is variable.