**HTML**

URL: <https://www.toptal.com/html5/interview-questions>

**Q1).What's the functionality/use of !DOCTYPE?**

A)1 .HTML5:

<!DOCTYPE html>

2. HTML 4.01 Strict, Transitional, Frameset:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">

The DOCTYPE Declaration (DTD or Document Type Declaration) does a couple of things:

a.When performing HTML validation testing on a web page it tells the HTML (HyperText Markup Language) validator which version of (X)HTML standard the web page coding is supposed to comply with. When you validate your web page the HTML validator checks the coding against the applicable standard then reports which portions of the coding do not pass HTML validation (are not compliant).

b.It tells the browser how to render the page in standards compliant mode.

c.All browsers need the doctype. Without the DOCTYPE you are forcing the browsers to render in Quirks Mode.

**Q2) Web Workers**

A)1. The simplest use of workers is for performing a computationally expensive task without interrupting the user interface.

2. Web workers enable multi-threading on the front end by spawning new background threads and running scripts in isolation. As a result, scripts executed by workers need to be contained in separate files. Because web workers execute scripts in isolated threads, scripts do not interfere with the main thread and consequently do not interrupt the UI.

**creating:** let worker = new Worker('fetch.js');

**terminating**: worker.terminate();

**a2)** Web workers at long last bring multi-threading to JavaScript.

A web worker is a script that runs in the background (i.e., in another thread) without the page needing to wait for it to complete.

The user can continue to interact with the page while the web worker runs in the background. Workers utilize thread-like message passing to achieve parallelism.

**Q3) What are some of the key new features in HTML5?**

Key new features of HTML5 include:

1. Improved support for embedding graphics, audio, and video content via the new <canvas>, <audio>, and <video> tags.

2. Extensions to the JavaScript API such as geolocation and drag-and-drop as well for storage and caching.

3. Introduction of “web workers”.

4. Several new semantic tags were also added to complement the structural logic of modern web applications. These include the <main>, <nav>, <article>, <section>, <header>, <footer>, and <aside> tags.

5. New form controls, such as <calendar>, <date>, <time>, <email>, <url>, and <search>.

**Q4) What is the difference between span and div?**

A)

1.The difference is that span gives the output with display: inline and div gives the output with display: block.

2. span is used when we need our elements to be shown in a line, one after the other.

**a2**) div is a block element, span is inline.

This means that to use them semantically, divs should be used to wrap sections of a document, while spans should be used to wrap small portions of text, images, etc.

**For example:**

<div>This a large main division, with <span>a small bit</span> of spanned text!</div>

Note that it is illegal to place a block level element within an inline element, so:

<div>Some <span>text that <div>I want</div> to mark</span> up</div> ...is illegal.)

**Q5) What is the Geolocation API in HTML5?**

A)HTML5’s Geolocation API lets users share their physical location with chosen web sites. JavaScript can capture a user’s latitude and longitude and can send it to the back-end web server to enable location-aware features like finding local businesses or showing their location on a map.

Today, most browsers and mobile devices support the Geolocation API. The Geolocation API works with a new property of the global navigator object.

A Geolocation object can be created as follows:

var geolocation = navigator.geolocation;

The geolocation object is a service object that allows widgets to retrieve information about the geographic location of the user’s device.

**Q6) What’s the difference between the <svg> and <canvas> elements?**

A)

1.The <svg> element is a container for SVG graphics. SVG has several methods for drawing paths, boxes, circles, text, and even bitmap images.

2.SVG is a language for describing 2D graphics, but <canvas> allows you to draw 2D graphics on the fly using JavaScript.

3.SVG is XML-based, which means that every element is available within the SVG DOM. You can attach JavaScript event handlers for an element.

4.In SVG, each drawn shape is remembered as an object. If attributes of an SVG object are changed, the browser can automatically re-render the shape.

5.Canvas is rendered pixel by pixel. In canvas, once the graphic is drawn, it is forgotten by the browser. If its position should be changed, the entire scene needs to be redrawn, including any objects that might have been covered by the graphic.

**Q7) Center a DIV horizontally and vertically.**

A)

HTML:

<div class="content">This works with any content to center data on page</div>

CSS:

.content {

position: absolute;

left: 50%;

top: 50%;

-webkit-transform: translate(-50%, -50%);

transform: translate(-50%, -50%);

}

**Q8) Which JavaScript objects are not accessible to web worker?**

A) Following JavaScript objects are not accessible to web worker:

1. The window object

2. The document object

3. The parent object

**Q9) What is a Manifest file?**

A Manifest file is a simple text file that tells the browser what to cache and what not to cache.

There are three sections of a Manifest file:

1) CACHE MANIFEST - Files listed here are cached after they are downloaded for the first time.

2) NETWORK - Files listed here require a connection to the server, and are never cached.

3) FALLBACK - Files listed here specify fallback pages if a page is inaccessible.

**Q10) What are the new APIs provided by HTML 5 standard?**

A)HTML 5 standard comes with a number of new APIs. Few of it are as follows:

1. Media API 2. Text Track API

3. Application Cache API 4.User Interaction

5. Data Transfer API 6.Command API

7. Constraint Validation API 8. History API

**Q11) What is Web Storage**

The HTML5's web storage feature lets you store some information locally on the user's computer, similar to cookies, but it is faster and much better than cookies. However, web Storage is no more secure than cookies.

The information stored in the web storage isn't sent to the web server as opposed to the cookies where data sent to the server with every request. Also, where cookies let you store a small amount of data (nearly 4KB), the web storage allows you to store up to 5MB of data.

There are two types of web storage, which differ in scope and lifetime:

* **Local storage** — The local storage uses the localStorage object to store data for your entire website, permanently. That means the stored local data will be available on the next day, the next week, or the next year unless you remove it.
* **Session storage** — The session storage uses the sessionStorage object to store data on a temporary basis, for a single window (or tab). The data disappears when session ends i.e. when the user closes that window (or tab).

## The localStorage Object

As stated earlier, the localStorage object stores the data with no expiration date. Each piece of data is stored in a key/value pair. The key identifies the name of the information (like 'first\_name'), and the value is the value associated with that key (say 'Peter').

**Example:**

* <script type="text/javascript">
* // Check if the localStorage object exists
* if(localStorage){
* // Store data
* localStorage.setItem("first\_name", "Peter");
* // Retrieve data
* alert("Hi, " + localStorage.getItem("first\_name"));
* } else{
* alert("Sorry, your browser do not support local storage.");
* }
* </script>

### Example explained:

The above JavaScript code has the following meaning:

* **localStorage.setItem(key, value):** Stores a value associated with a key.
* **localStorage.getItem(key):** Retrieves the value associated with the key.

You can also remove a particular item from the storage by passing the key value to the removeItem()method, like localStorage.removeItem(key).

However, if you want to remove the complete storage use the clear() method, like localStorage.clear(). The clear() method clears all key/value pairs from localStorage at once, so think carefully before you using it.

**Note:** The web storage data (both local and session) will not be available between different browsers, for example the data stored in Firefox browser will not available in Google Chrome, Internet Explorer or other browsers.

## The sessionStorage Object

The sessionStorage object work in the same way as localStorage, except that it stores the data only for one session i.e. the data remains until the user closes that window or tab.

#### Example

* <script type="text/javascript">
* // Check if the sessionStorage object exists
* if(sessionStorage){
* // Store data
* sessionStorage.setItem("last\_name", "Parker");
* // Retrieve data
* alert("Hi, " + localStorage.getItem("first\_name") + " " + sessionStorage.getItem("last\_name"));
* } else{
* alert("Sorry, your browser do not support session storage.");
* }
* </script>

------------------------------------------------\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*----------------------------------------------------

**CSS**

URL: <https://stackoverflow.com/questions/tagged/css?sort=votes>

<http://a4academics.com/interview-questions/79-web/817-css-css3?showall=&limitstart>=

**Q1) What is the difference between visibility:hidden and display:none?**

A)

1. **display:none** means that the tag in question will not appear on the page at all (although you can still interact with it through the dom). There will be no space allocated for it between the other tags.

2. **visibility:hidden** means that unlike display:none, the tag is not visible, but space is allocated for it on the page. The tag is rendered, it just isn't seen on the page.

**For example:**

test | <span style="[style-tag-value]">Appropriate style in this tag</span> | test

Replacing [style-tag-value] with display:none results in:

test | | test

Replacing [style-tag-value] with visibility:hidden results in:

test | | test

**Q2) What is the difference between display: inline and display: inline-block?**

A)https://stackoverflow.com/questions/8969381/what-is-the-difference-between-display-inline-and-display-inline-block

display:block means the element will occupy 100% width, and will start from a new line.

display:inline-block means the element will be inline (does not start from a new line) and will stretch to occupy width of it's own content only.

display: inline; is a display mode to use in a sentence. For instance, if you have a paragraph and want to highlight a single word you do:

<p>

Pellentesque habitant morbi <em>tristique</em> senectus

et netus et malesuada fames ac turpis egestas.

</p>

The <em> element has a display: inline; by default, because this tag is always used in a sentence. The <p> element has a display: block; by default, because it's neither a sentence nor in a sentence, it's a block of sentences.

An element with display: inline; cannot have a height or a width or a vertical margin. An element with display: block; can have a width, height and margin.

If you want to add a height to the <em> element, you need to set this element to display: inline-block;. Now you can add a height to the element and every other block style (the block part of inline-block), but it is placed in a sentence (the inline part of inline-block).

A2)

1.An inline element has no line break before or after it, and it tolerates HTML elements next to it.

2.A block element has some whitespace above and below it and does not tolerate any HTML elements next to it.

3.An inline-block element is placed as an inline element (on the same line as adjacent content), but it behaves as a block element.

**Q3) How to disable a link using only CSS?**

A)

<a href="link.html" class="not-active">Link</a>

.not-active {

pointer-events: none;

cursor: default;

}

**Q4) Position Relative vs Absolute?**

A)1. **Relative** : Relative to it’s current position, but can be moved. Or A RELATIVE positioned element is positioned relative to ITSELF.

2. **Absolute** : An ABSOLUTE positioned element is positioned relative to IT'S CLOSEST POSITIONED PARENT. if one is present, then it works like fixed.....relative to the window.

**Q5) margin vs padding**

A)Margin: is applied to the outside of you element hence effecting how far your element is away from other elements.

Padding: is applied to the inside of your element hence effecting how far your element's content is away from the border.

Also, using margin will not affect your element's dimensions whereas padding will make your elements dimensions (set height + padding) so for example if you have a 100x100px div with a 5 px padding, your div will actually be 105x105px

**Q6) Difference between div id and div class**

A) Ids must be unique where as class can be applied to many things. In CSS, ids look like #elementID and class elements look like .someClass

In general, use id whenever you want to refer to a specific element and class when you have a number of things that are all alike. For instance, common id elements are things like header, footer, sidebar. Common class elements are things like highlight or external-link.

It's a good idea to read up on the cascade and understand the precedence assigned to various selectors: http://www.w3.org/TR/CSS2/cascade.html

The most basic precedence you should understand, however, is that id selectors take precedence over class selectors. If you had this:

<p id="intro" class="foo">Hello!</p>

and:

#intro { color: red }

.foo { color: blue }

The text would be red because the id selector takes precedence over the class selector.

**Q7) span vs p**

A)< span > is used to group inline-elements so that you can style them. Span is Inline. It is non-semantic.

< p > is used for a paragraph which is a block element. Which can be styled also. Paragraph is Block. It is semantic.

**Q8) CSS3 new features**

A)1. CSS Animations and Transitions

2.Calculating Values With calc()

3.Advanced Selectors

4.Border Images

5.Media Queries

6.body {

background-image: url("http://image-gallery.io/mountain-scene.png");

background-size: cover; }

7. flex

.main {

display: flex;

}

8. rem value

p {

font-size: 0.75rem;

}

9. Below are the list of texts added in CSS3 –

Word-wrap

Text-overflow

Word- break

**Q9) Explain opacity in CSS3?**

A)

Opacity is used to hide or show an element in CSS3.

Value – ‘0’ to hide the element and value ‘1’ means showing an element.

Below is the sample for the same –

<p style = “opacity:0”> Hide Text </p>

**Q10) What is the use of z-index in CSS?**

A)

Z-Index is used to avoid the overlapping of the elements.

Default value of z-index is 0 and it will take positive and negative values as well.

**Q11) How do you write a conditional statement in CSS? Give an example.**

A)

Below is the example of writing a conditional statement in CSS –

<style type=”text/css”>

body

{ color: #00BFFF; } </style>

<!—if [ IE 8] >

<style type=”text/css”>

body

{

Background-color: #00FFBF;

}

</style>

<! [end if] -->

**Q12) Explain what elements will match each of the following CSS selectors:**

A)

div, p -> Selects all <div> elements and all <p> elements

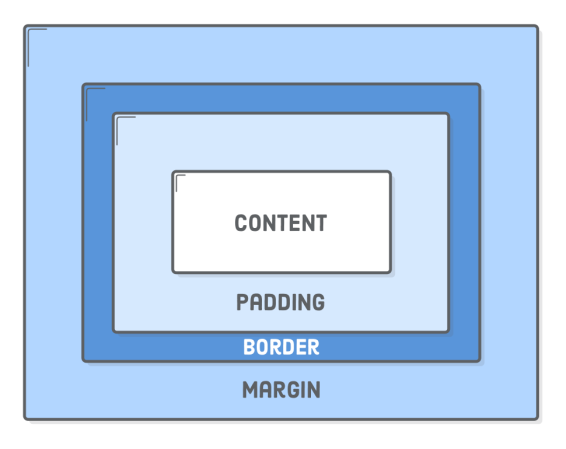
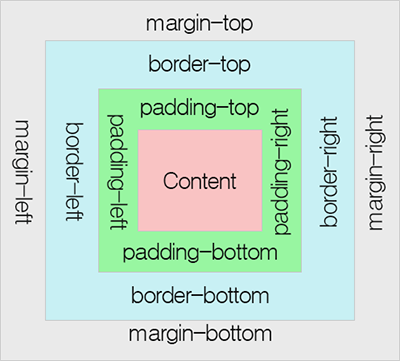
div p ->Selects all <p> elements that are anywhere inside a <div> element

div > p -> Selects all <p> elements where the immediate parent is a <div> element

div + p -> Selects all <p> elements that are placed immediately after a <div> element

div ~ p -> Selects all <p> elements that are anywhere preceded by a <div> element

**Q13) Explain the CSS “box model” and the layout components that it consists of. Provide some usage examples.**

A)

The CSS box model is a rectangular layout paradigm for HTML elements that consists of the following:

Content - The content of the box, where text and images appear

Padding - A transparent area surrounding the content (i.e., the amount of space between the border and the content)

Border - A border surrounding the padding (if any) and content

Margin - A transparent area surrounding the border (i.e., the amount of space between the border and any neighboring elements)

Each of these properties can be specified independently for each side of the element (i.e., top, right, bottom, left) or fewer values can be specified to apply to multiple sides. For example:

/\* top right bottom left \*/

padding: 25px 50px 75px 100px;

/\* same padding on all 4 sides: \*/

padding: 25px;

/\* top/bottom padding 25px; right/left padding 50px \*/

padding: 25px 50px;

/\* top padding 25px; right/left padding 50px; bottom padding 75px \*/

padding: 25px 50px 75px;