**15 Essential HTML5 Interview Questions to Watch Out in 2018**

*HTML and CSS are the foundation of the web. And HTML 5 is the foundation for taking your full stack skills to the next level. Follow through and check 15 most common and essential HTML 5 Interview Questions and Answers to watch out on your developer interview in 2018.*

**Top of Form**

**Bottom of Form**

**Q1: Explain meta tags in HTML**

Topic: **HTML5**  
Difficulty: ⭐

* **Meta tags** always go inside **head tag** of the HTML page
* **Meta tags** is always passed as name/value pairs
* **Meta tags** are not displayed on the page but intended for the browser
* **Meta tags** can contain information about **character encoding**, **description**, **title** of the document etc,

**Example**:

<!DOCTYPE html>

<html>

<head>

<meta name="description" content="I am a web page with description">

<title>Home Page</title>

</head>

<body>

</body>

</html>

**Q2: What is an optional tag?**

Topic: **HTML5**  
Difficulty: ⭐⭐⭐

In HTML, some elements have optional tags. In fact, both the opening and closing tags of some elements may be completely removed from an HTML document, even though the elements themselves are required.

Three required HTML elements whose start and end tags are optional are the html, head, and body elements.

**Q3: Can a web page contain multipleelements? What aboutelements?**

Topic: **HTML5**  
Difficulty: ⭐⭐⭐

Yes to both. The W3 documents state that the tags represent the header(<header>) and footer(<footer>) areas of their nearest ancestor "section". So not only can the page <body> contain a header and a footer, but so can every <article> and <section> element.

**Q4: What is the DOM?**

Topic: **HTML5**  
Difficulty: ⭐⭐⭐

The *DOM (Document Object Model)* is a cross-platform API that treats HTML and XML 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.
* The DOM is constructed progressively in the browser as a page loads, which is why scripts are often placed at the bottom of a page, in the <head>with a defer attribute, or inside a DOMContentLoaded event listener. Scripts that manipulate DOM nodes should be run after the DOM has been constructed to avoid errors.
* 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.

**Q5: Discuss the differences between an HTML specification and a browser’s implementation thereof.**

Topic: **HTML5**  
Difficulty: ⭐⭐⭐

HTML specifications such as HTML5 define a set of rules that a document must adhere to in order to be “valid” according to that specification. In addition, a specification provides instructions on how a browser must interpret and render such a document.

A browser is said to “support” a specification if it handles valid documents according to the rules of the specification. As of yet, no browser supports all aspects of the HTML5 specification (although all of the major browser support most of it), and as a result, it is necessary for the developer to confirm whether the aspect they are making use of will be supported by all of the browsers on which they hope to display their content. This is why cross-browser support continues to be a headache for developers, despite the improved specificiations.

* HTML5 defines some rules to follow for an invalid HTML5 document (i.e., one that contains syntactical errors)
* However, invalid documents may contain anything, so it's impossible for the specification to handle all possibilities comprehensively.
* Thus, many decisions about how to handle malformed documents are left up to the browser.

**Q6: What is HTML5 Web Storage? Explain localStorage and sessionStorage.**

Topic: **HTML5**  
Difficulty: ⭐⭐⭐

With HTML5, web pages can store data locally within the user’s browser. The data is stored in name/value pairs, and a web page can only access data stored by itself.

**Differences between localStorage and sessionStorage regarding lifetime:**

* Data stored through localStorage is permanent: it does not expire and remains stored on the user’s computer until a web app deletes it or the user asks the browser to delete it.
* sessionStorage has the same lifetime as the top-level window or browser tab in which the data got stored. When the tab is permanently closed, any data stored through sessionStorage is deleted.

**Differences between localStorage and sessionStorage regarding storage scope:**

Both forms of storage are scoped to the document origin so that documents with different origins will never share the stored objects.

* sessionStorage is also scoped on a per-window basis. Two browser tabs with documents from the same origin have separate sessionStoragedata.
* Unlike in localStorage, the same scripts from the same origin can't access each other's sessionStorage when opened in different tabs.

**Q7: What's new in HTML 5?**

Topic: **HTML5**  
Difficulty: ⭐⭐⭐

HTML 5 adds a lot of new features to the HTML specification

**New Doctype**

Still using that pesky, impossible-to-memorize XHTML doctype?

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

If so, why? Switch to the new HTML5 doctype. You'll live longer -- as Douglas Quaid might say.

<!DOCTYPE html>

**New Structure**

* <section> - to define sections of pages
* <header> - defines the header of a page
* <footer> - defines the footer of a page
* <nav> - defines the navigation on a page
* <article> - defines the article or primary content on a page
* <aside> - defines extra content like a sidebar on a page
* <figure> - defines images that annotate an article

**New Inline Elements**

These inline elements define some basic concepts and keep them semantically marked up, mostly to do with time:

* <mark> - to indicate content that is marked in some fashion
* <time> - to indicate content that is a time or date
* <meter> - to indicate content that is a fraction of a known range - such as disk usage
* <progress> - to indicate the progress of a task towards completion

**New Form Types**

* <input type="datetime">
* <input type="datetime-local">
* <input type="date">
* <input type="month">
* <input type="week">
* <input type="time">
* <input type="number">
* <input type="range">
* <input type="email">
* <input type="url">

**New Elements**

There are a few exciting new elements in HTML 5:

* <canvas> - an element to give you a drawing space in JavaScript on your Web pages. It can let you add images or graphs to tool tips or just create dynamic graphs on your Web pages, built on the fly.
* <video> - add video to your Web pages with this simple tag.
* <audio> - add sound to your Web pages with this simple tag.

**No More Types for Scripts and Links**

You possibly still add the type attribute to your link and script tags.

<link rel="stylesheet" href="path/to/stylesheet.css" type="text/css" />

<script type="text/javascript" src="path/to/script.js"></script>

This is no longer necessary. It's implied that both of these tags refer to stylesheets and scripts, respectively. As such, we can remove the type attribute all together.

<link rel="stylesheet" href="path/to/stylesheet.css" />

<script src="path/to/script.js"></script>

**Make your content editable**

The new browsers have a nifty new attribute that can be applied to elements, called contenteditable. As the name implies, this allows the user to edit any of the text contained within the element, including its children. There are a variety of uses for something like this, including an app as simple as a to-do list, which also takes advantage of local storage.

<h2> To-Do List </h2>

<ul contenteditable="true">

<li> Break mechanical cab driver. </li>

<li> Drive to abandoned factory

<li> Watch video of self </li>

</ul>

**Attributes**

* require to mention the form field is required
* autofocus puts the cursor on the input field

**Q8: HTML Markup Validity**

Topic: **HTML5**  
Difficulty: ⭐⭐⭐⭐

Consider the following markup. Is it valid? If not, can you explain why?

<figure>

<picture>

<source media="(min-width: 40em)"

srcset="large.jpg 1024w, medium.jpg 640w, small.jpg 320y">

<img src="medium.jpg" alt="London by night">

</picture>

<figcaption>A landscape of London by night</figcaption>

</figure>

**Answer:**

The markup uses the relatively new picture element. The code is all valid apart from the last image specified in the srcset attribute. 320y is not a valid value, and the y should be replaced with a w.

**Q9: What are the building blocks of HTML5?**

Topic: **HTML5**  
Difficulty: ⭐⭐⭐⭐

* **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.

**Q10: What is progressive rendering?**

Topic: **HTML5**  
Difficulty: ⭐⭐⭐⭐

*Progressive rendering* is the name given to techniques used to improve the performance of a webpage (in particular, improve perceived load time) to render content for display as quickly as possible.

It used to be much more prevalent in the days before broadband internet but it is still used in modern development as mobile data connections are becoming increasingly popular (and unreliable)!

Examples of such techniques:

* Lazy loading of images - Images on the page are not loaded all at once. JavaScript will be used to load an image when the user scrolls into the part of the page that displays the image.
* Prioritizing visible content (or above-the-fold rendering) - Include only the minimum CSS/content/scripts necessary for the amount of page that would be rendered in the users browser first to display as quickly as possible, you can then use deferred scripts or listen for the DOMContentLoaded/load event to load in other resources and content.
* Async HTML fragments - Flushing parts of the HTML to the browser as the page is constructed on the back end. More details on the technique can be found [here](http://www.ebaytechblog.com/2014/12/08/async-fragments-rediscovering-progressive-html-rendering-with-marko/).

**Q11: Why to use HTML5 semantic tags?**

Topic: **HTML5**  
Difficulty: ⭐⭐⭐⭐

As their name says, this is for semantic purposes only. It's for improving the automated processing of documents. Automated processing happens more often than you realize - each website ranking from search engines is derived from automated processing of all the website out there.

// machine: okay, this structure looks like it might be a navigation element?

<div class="some-meaningless-class"><ul><li><a href="internal\_link">...</div>

// machine: ah, a navigation element!

<nav class="some-meaningless-class"><ul><li><a>...</nav>

**Q12: What's the difference between Full Standard, Almost Standard and Quirks Mode?**

Topic: **HTML5**  
Difficulty: ⭐⭐⭐⭐

When **web** was **new to the world**, the web pages were written in two versions:

* Netscape Navigator
* Internet Explorer

When **W3C** (World Wide Web Consortium) introduced new standards for web, the browsers introduced **two layout modes** to support **new** and **old web standards**.

Those modes are

* Quirks mode
* Standard mode
* Almost standard mode

**Quirks Mode**

* Browser will emulate **non-standard** behavior in **Netscape Navigator 4** and **Internet Explorer 5**
* This was essential in order to support websites which was built before **web standards**

**Full Standard Mode**

* In **Full Standard** mode, behavior of browser is same as the **HTML & CSS** specification
* Almost **all** modern browser uses **full standard mode** now

To use full standard mode, specify <!DOCTYPE html> in your HTMl document.

<!DOCTYPE html>

<html>

<head>

<title>Document title</title>

</head>

<body>

</body>

</html>

**Almost Standard Mode**

* In **Almost Standard** mode, behavior of browser is same as **full standard mode** but with a very small number of quirks implementation

**Q13: Could you generate a public key in HTML?**

Topic: **HTML5**  
Difficulty: ⭐⭐⭐⭐⭐

The <keygen> element generates a public-private key pair and sends the public key to the server with form submission. The element is expected to be deprecated and does not have broad browser support.

If two parties want to communicate securely, they each need to be able to generate a public-private key pair, and then hare the public key with the other party. The <keygen> is intended to facilitate this within the context of an HTML form. In browsers that implement it (not all do), if the element is included with a form, the browser generates a key-pair locally and sends the public key to the server when the form is submitted. The private key is then stored locally and not shared.

It has been announced that the <keygen> element will be deprecated and removed from the HTML5 standard. This means that alternate methods of key generation will be needed. Naturally this simply means using JavaScript.

**Q14: What is accessibility & ARIA role means in a web application?**

Topic: **HTML5**  
Difficulty: ⭐⭐⭐⭐⭐

Accessibility means letting the user navigate and interact with a website by any means. Let it be a keyboard, a mouse or people with disabilities or impairments.

* **ARIA** stands for **Accessible Rich Internet Application**.
* **ARIA** is a **spec** from **W3C**.
* **ARIA** is created to **improve accessibility** in **web applications**.
* **ARIA** provides a **extra information** to **screen readers** via **HTML attributes**.
* **ARIA** **doesn't affect** how **an element** is **rendered** in a browser.

**Q15: What are Web Components?**

Topic: **HTML5**  
Difficulty: ⭐⭐⭐⭐⭐

* Web components are a set of web platform API.
* Web components allow us to create **custom re-usable widgets** or **component** in a web application.
* Web components are a **part of the browser** and **will work** across **all modern browser**.
* Web components **doesn't require** need **any external libraries** to **work**.

**Features**

* Custom Elements - let write our **own fully featured** DOM elements.
* Shadow DOM - gives the best feature of a **iframe**, **style** and **markup** for the web page.
* HTML Templates - allows to **store** some **markup on the web page** and we can later **clone** and **reuse** them.
* HTML Imports - allow to **import** an **external HTML document**.