Auditor wala answer:
The most significant aspect to be performed is the comparison with the competitors in the market are as follows:
1. Accessibility: Site must be friendly to search engine spiders (crawlers).
2. URL Check: URL should be clean, short and descriptive. Descriptive means keyword rich. It should be simple and easy for users to understand.
3. Sitemaps file and robot.txt file verification: Google Webmaster Tools is required to verify the robot.txt file.
4. Page load time: Long load time of pages may slow down crawling and indexing of the site.
5. Image alt tags: Image should have relevant keyword rich image alt attributes text. It help in improving overall context of the page.
6. Code quality: Poor coding have undesirable impact. Code quality also get checked.
7. Title tag review: Title tag of each page should be unique and descriptive. Brand name of organization should not be included in URL if it is there then it should be placed in the end of the title tag. Title tag should be less than 70 characters.
8. Content review: The content of site should be reviewed for use of header tag and quality content. The percentage of pages on the site with title content should be less in comparison with total number of pages on the site.

XML sitemaps
XML Sitemaps are a useful and in some cases essential tool for your website. In particular, if you have reason to believe that the site is not fully indexed, an XMLSitemap can help you increase the number of indexed pages. As sites grow in size, the value of XML Sitemap files tends to increase dramatically, as additional traffic flows to the newly included URLs.

Google, Yahoo!, and Microsoft all support a protocol known as XML Sitemaps. Google first announced it in 2005, and then Yahoo! and Microsoft agreed to support the protocol in 2006. Using the Sitemaps protocol you can supply the search engines with a list of all the URLs you would like them to crawl and index. Adding a URL to a Sitemap file does not guarantee that a URL will be crawled or indexed. However, it can result in pages that are not otherwise discovered or indexed by the search engine getting crawled and indexed. In addition, Sitemaps appear to help pages that have been relegated to Google's supplemental index make their way into the main index. This program is a complement to, not a replacement for, the search engines' normal, linkbased crawl.

Types of sitemaps are:
● Index Sitemap (how many URL sitemaps a website have)
● URL Sitemap (contain final information of URLs on webpage)
XML sitemaps are further divided into 3 different categories.
● Sitemaps for webpages (commonly known as xml sitemap in community)
● Image sitemaps (details of images and their URLs on website)
● Video sitemaps (what webpages have videos embedded in them and their details)
So, final tree of sitemaps categorization we have is;
● XML Sitemap
○ Index Sitemap
○ URL Sitemap
■ Sitemaps for Webpages

- Sitemaps for Images
- Sitemaps for videos

The benefits of Sitemaps include the following:

- For the pages the search engines already know about through their regular spidering, they use the metadata you supply, such as the last date the content was modified (lastmod date) and the frequency at which the page is changed (changefreq), to improve how they crawl your site.
- For the pages they don’t know about, they use the additional URLs you supply to increase their crawl coverage.
- For URLs that may have duplicates, the engines can use the XML Sitemaps data to help choose a canonical version.
- Verification/registration of XML Sitemaps may indicate positive trust/authority signals.

The crawling/inclusion benefits of Sitemaps may have second order positive effects, such as improved rankings or greater internal link popularity.

Define DOM.Explain in node tree for HTML document and list level of DOM.

The Document Object Model (DOM) is an object oriented representation of an HTML or XML document. The structure of an HTML and XML document is hierarchical, so the DOM structure resembles that of a tree. DOM provides an API to access and modify this tree of objects. The DOM API is specified in a language independent manner by the W3C, and mappings are available for most programming languages. The DOM API is used in a number of situations, like on the server side in a Java program to perform some manipulation on an XML document. For the scope of this and the next chapter, we will look in more details at the DOM in the context of the browser and see how with DOM API you can dynamically modify the HTML or XHTML page.

In order to look at the structure of the DOM, consider this short, yet complete and valid

XHTML page: Example :

```
<?xml version=”1.0” encoding=”ISO88591”?>
<!DOCTYPE html PUBLIC “// W3C//DTD XHTML 1.0 Strict//EN”
“http://www.w3.org/TR/xhtml1/DTD/xhtml1strict. dtd”>
<html xmlns=”http://www.w3.org/1999/xhtml”>
<head>
<title>DOM Example</title>
</head>
<body>
<h1>Page title</h1>
<p> Some <i>very</i> unimportant text.
</p> </body>
</html>
```

Levels of DOM:

The DOM has three levels of support; each level corresponds to subsequent recommendations by the W3C:

- Level 1 was first released in 1998, when browsers were already providing a DOMlike API. It was released in part as an effort to define a common API that would be implemented by web browsers. Adoption has been slow, but today it is generally accepted that DOM level 1 is supported by all the mainstream browsers.

● Level 2 was first released in 2000, and most of the API is supported by all the mainstream browsers, except Internet Explorer which implements a smaller but still significant subset of the specification. If you are developing an application targeting the browsers deployed today, using level 2 recommendations as a reference is your best bet.

● Level 3 is still a work in progress. Starting with level 2, the DOM specification has been split into a number of documents: one specification for core components, one for HTML specific aspects of the API, one dealing with events, and so on. Only a subset of those has been published by the W3C as recommendations, while others are still works in progress. Most browsers only support a minimal subset of level 3, and Internet Explorer does not support level 3 at all. For those reasons, at this point most web developers consider that it is still too early to use the DOM level 3.

Ajax wala answer:

Ajax isn't a technology. It's really several technologies, each flourishing in its own right, coming together in powerful new ways. Ajax incorporates:

○ standards based presentation using XHTML and CSS;
○ dynamic display and interaction using the Document Object Model;
○ data interchange and manipulation using XML and XSLT;
○ asynchronous data retrieval using XMLHttpRequest;
○ and JavaScript binding everything together.

● The classic web application model works like this: Most user actions in the interface trigger an HTTP request back to a web server. The server does some processing — retrieving data, crunching numbers, talking to various legacy systems — and then returns an HTML page to the client.

How Ajax works

● An Ajax application eliminates the startstopstartstop nature of interaction on the Web by introducing an intermediary.

● Ajax engine between the user and the server. It seems like adding a layer to the application would make it less responsive, but the opposite is true.

● Instead of loading a webpage, at the start of the session, the browser loads an Ajax engine written in JavaScript and usually tucked away in a hidden frame . This engine is responsible for both rendering the interface the user sees and communicating with the server on the user's behalf.

● The Ajax engine allows the user's interaction with the application to happen asynchronously independent of communication with the server . So the user is never staring at a blank browser window and an hourglass icon, waiting around for the server to do something.

● Every user action that normally would generate an HTTP request takes the form of a JavaScript call to the Ajax engine instead. Any response to a user action that doesn't require a trip back to the server such as simple data validation, editing data in memory, and even some navigation the engine handles on its own.

● If the engine needs something from the server in order to respond if it's submitting data for processing, loading additional interface code, or retrieving new data the engine makes those requests asynchronously, usually using XML, without stalling a user's interaction with the application.

AJAX is based on internet standards, and uses a combination of:

- **XMLHttpRequest object (to retrieve data from a web server)**
- **JavaScript/DOM (to display/use the data)**

The XMLHttpRequest object is used to exchange data with a server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page. XMLHttpRequest is an API that provides client functionality for transferring data between a client and a server. It provides an easy way to retrieve data from a URL without having to do a full page refresh. This enables a Web page to update just a part of the page without disrupting what the user is doing. XMLHttpRequest is used heavily in AJAX programming. XMLHttpRequest was originally designed by Microsoft and adopted by Mozilla, Apple, and Google. Despite its name, XMLHttpRequest can be used to retrieve any type of data, not just XML, and it supports protocols other than HTTP (including file and ftp).

Advantages of Web Services

1. Interoperability: Web services are accessible over network and runs on HTTP/SOAP protocol and uses XML/JSON to transport data, hence it can be developed in any programming language. Web service can be written in java programming and client can be PHP and vice versa.
2. Reusability: One web service can be used by many client applications at the same time.
3.Loose Coupling: Web services client code is totally independent with server code, so we have achieved loose coupling in our application.
4. Easy to deploy and integrate, just like web applications. 5. Multiple service versions can be running at same time.

Different types of Web Services

There are two types of web services:

SOAP Web Services: Runs on SOAP protocol and uses XML technology for sending data.

Restful Web Services: It’s an architectural style and runs on HTTP/HTTPS protocol almost all the time. REST is a stateless clientserver architecture where web services are resources and can be identified by their URIs. Client applications can use HTTP GET/POST methods to invoke Restful web services.

REST Web Services

- REST is the acronym for REpresentational State Transfer. REST is an architectural style for developing applications that can be accessed over the network.
- REST is a stateless clientserver architecture where web services are resources and can be identified by their URIs. Client applications can use HTTP GET/POST methods to invoke Restful web services. REST doesn’t specify any specific protocol to use, but in almost all cases it’s used over HTTP/HTTPS.
- When compared to SOAP web services, these are lightweight and doesn’t follow any standard. We can use XML, JSON, text or any other type of data for request and response.

Advantages of REST web services

- Learning curve is easy since it works on HTTP protocol
- Supports multiple technologies for data transfer such as text, xml, json, image etc.
- No contract defined between server and client, so loosely coupled implementation.
- REST is a lightweight protocol
- REST methods can be tested easily over browser.