**Web Page and Website**

The difference between a website and a web page is that a website is a collection of web pages which are grouped together, and a web page is a smaller part of a larger website usually containing more specific information.

A website is under 1 domain (such as coolwebsite.com). For example if there is a company that owns abccompany.com then this website will have several Webpages like Home, About Us, Contact Us, Testimonials, Products, Services, FAQ’s, and others. All of these pages together make up a Website.

**Static Page and Dynamic Page**

There are basically two main types of website - static and dynamic.

A static site is one that is usually written in plain HTML and what is in the code of the page is what is displayed to the user.

Static Web pages display the exact same information whenever anyone visits it. Static Web pages do not have to be simple plain text. They can feature detailed multimedia design and even videos. However, every visitor to that page will be greeted by the exact same text, multimedia design or video every time he visits the page until you alter that page's source code.

A dynamic site is one that is written using a server-side scripting language such as PHP, ASP, JSP, or Coldfusion.

Dynamic Web pages are capable of producing different content for different visitors from the same source code file. The website can display different content based on what operating system or browser the visitor is using, whether she is using a PC or a mobile device, or even the source that referred the visitor. A dynamic Web page is not necessarily better than a static Web page. The two simply serve different purposes.

|  |  |
| --- | --- |
| Static sites - advantages  Flexibility is the main advantage of a static site - every page can be different if desired, to match the layout to different content, and the designer is free to put in any special effects that a client may ask for in a unique way on different pages. This allows theming - for instance an author may want a different theme for a different book and associated pages or perhaps for a series of books, in order to match the cover designs or the context of the stories.  Cost is generally lower up-front than a dynamic site. | Dynamic sites - advantages  The main advantages of dynamic sites are that by connecting them to databases you can easily pull in information in an organized and structured way to create product pages or categories of related products sorted in a variety of different ways depending on how the user wants to view them.  This ability to connect to a database means that you can also create a content management system - an interface which allows the client to input and manage data via a web-based series of administration pages. That content can be text for their pages and images to go along with the text, or items in their product range with categories, specifications, short and long descriptions, images, etc. In both these cases it can be as simple or as complex as the client requires.  There are little or no ongoing costs unless there is a change in the basic design or an extra capability added. |
| Static sites - disadvantages  The main problem with any static site appears when you wish to update the content. Unless you are conversant with HTML and the design methods used in the site then you have to go back to the designer to have any content changes made. This may be perfectly ok when a new page is required which needs design input, but if all you want to do is change some text then it can be a nuisance for both client and designer.  The second main problem is scalability. If you wish to sell products on your site and you have a lot of them then you may have to construct individual pages for each one, which can take considerable time, effort and cost.  Costs - there are ongoing costs for updating the content. | Dynamic sites - disadvantages  The design of a dynamic site is more fixed than a static one because many of the pages are essentially a template into which data and content is poured to create multiple pages of a similar type. So for instance all your product pages will be essentially the same page layout with different data being displayed. While some customization capability can be built in it is usually quite limited, such a selecting from a set of pre-defined options. Individual layout changes to particular pages are not usually possible.  Costs are higher initially than for a static site, and additional functionality may also cost more, particularly if it's something that wasn't envisaged originally and requires re-writing of the core code or database. |

**Script**

A script is a set of instructions. For Web pages they are instructions either to the Web browser (client-side scripting) or to the server (server-side scripting). Scripts provide change to a Web page.

Most websites make use of both a client side and a server side language. Although there are things both can do, there are some things which can only be done server side, and there are some things which can only be done client side.

Front-end scripting is good for anything that requires user interaction, such as a making the page interactive, displaying or sorting data.

Back-end scripting is good for anything that requires dynamic data to be loaded, such as handling log in, personal information and preferences and provides the specific data which the user wants.

**Client Side and Server Side scripting**

|  |  |
| --- | --- |
| Client Side  Client-side scripts run on the client side, and is done in languages that can be executed by the browser, such as JavaScript  The reason JavaScript is called a client side language is because it runs scripts on your computer after you’ve loaded a web page.  <script>  document.getElementById('hello').innerHTML = 'Hello';  </script> | Server Side scripting  Server-side scripts run on the server, using languages supported by the server such as Java, PHP, C#, etc.  A server side or back-end language runs its scripts before the HTML is loaded, not after.  <h1 id="hello"><?php echo 'Hello'; ?></h1> |
| Uses  •Make interactive webpages.  •Make stuff happen dynamically on the web page.  •Interact with temporary storage, and local storage (Cookies, localStorage).  •Send requests to the server, and retrieve data from it.  •Provide a remote service for client-side applications, such as software registration, content delivery, or remote multi-player gaming.  Example languages  •JavaScript (primarily)  •HTML, CSS – these aren't "programming languages". They are markup syntax by which the Client renders the page for the User.  •Any language running on a client device that interacts with a remote service is a client-side language. | Uses  •Process user input.  •Display pages.  •Structure web applications.  •Interaction with SQL, files, etc.  Example Languages  •PHP  •Python  •ASP.Net in C#, C++, or Visual Basic.  •Nearly any language (C++, C#, Java). These were not designed specifically for the task, but are now often used for application-level web services. |

**Javascript and JQuery**

Javascript

JavaScript is a scripting language that is used inside your web browser.

Unfortunately, JavaScript still has some issues with cross-browser compatibility due to poor JavaScript implementation practices on the part of web browser developers.

function changeBackground(color) {

document.body.style.background = color;

}

onload="changeBackground('red');"

jQuery

jQuery is a library/ framework written in Javascript.

It simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development.

You can code most common JS actions using jQuery with fewer lines of code. jQuery is a Cross-browser, it means that it is compatible with multiple Web browsers.

$('body').css('background', '#ccc');

**AJAX**

AJAX stands for Asynchronous JavaScript and XML. We use AJ to do asynchronous things such updating a page, making actions, etc.

In simpler meaning, AJAX is all about updating parts of a webpage without having to reload the entire thing.

Ajax is the concept of the client calling the server directly to interact with server objects like a database, without a postback involved.

Note: “When you execute something synchronously, you wait for it to finish before moving on to another task. When you execute something asynchronously, you can move on to another task before it finishes.”

**Node.js, Angular.js, React.js**