accessed document on the screen.

Fig. 14.11. Browser architecture

14.4.1. Web Documents

The web documents may be classified into three categories as shown in figure 14.12

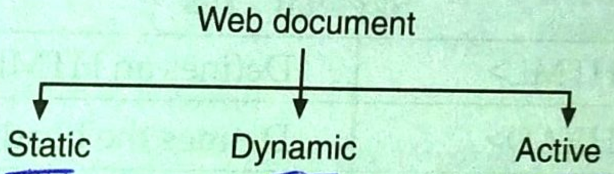


Fig. 14.12. Categories of web documents.

Static Documents

The contents of static documents are fixed. They are created and stored in a server. If required the client can get a copy of static document. The contents of the static document are determined when it is created. The contents cannot changed when the static document is being used. It is possible to change the contents of static document at the server but the user cannot change them.

14.4.2 Hypertext Markup Language (HTML)

HTML is a language used for creating the web pages. It uses certain marks to format the text. For example, if a part of text is required to be *boldface* then we can use the beginning and ending bold face tags (marks) in the text as shown below :

<B> – Beginning of boldface

</B> – End of boldface

Here, <B> and </B> are the instructions for the browser. The browser will make the part of the text between these tags bold. HTML lets the user to use only ASCII characters for the main text as well as for formatting instructions. Hence, every computer can receive the whole document as an ASCII document. The formatting instructions are used by the browser to format the data.

14.5. STRUCTURE OF A WEB PAGE

Basically, a web page may be divided into following two parts :

1. Head
2. Body

1. **Head :** This is the first page of a web page. The head carries the title of the web page and some other parameters that the browser will use.

2. **Body :** The body carries actual contents of a web page. It includes the text and tags. Text is the actual information carrying matter and the tags decide the appearance of the document.

14.5.1 HTML Tags

Tags are the marks which are embedded into the text. A tag is always enclosed in two signs (< and >). They usually come in pairs as beginning and ending tags. A tag will have a list of attributes. Each attribute is followed by an equal to sign and value associated with the attribute. The general format of a tag is shown in figure 14.13.

<Tag Name Attribute = Value Attribute = Value ... > </Tag Name >

(a) Beginning tag **(b) Ending tag**

Fig. 14.13.

14.5.2. Some Common HTML Tags

Table 14.1 shows some common HTML tags.

Table 14.1. Common Tags.

Beginning Tag	Ending Tag	Meaning
Skeletal Tags		
<HTML>	</HTML>	Defines an HTML document
<HEAD>	</HEAD>	Defines the head of the document
<BODY>	</BODY>	Defines the body of the document
Title and Header Tags		
<TITLE>	</TITLE>	Defines the title of the document
<Hn>	</Hn>	Defines different headers (n is an integer)
Text Formatting Tags		
<B>	</B>	Boldface
<I>	</I>	Italic
<U>	</U>	Underlined
_		Subscript
[	]	Superscript
Data Flow Tags		
<CENTER>	</CENTER>	Centered
<BR>		Line break
List Tags		
<OL>	</OL>	Ordered list
<UL>	</UL>	Unordered list
<LI>	</LL>	An item in a list

	Image Tags	
<IMG>		Defines an image
	Hyperlink Tags	
<A>	</A>	Defines an address (hyperlink)
	Executable Contents	
<APPLET>	</APPLET>	The document is an object

14.6. DYNAMIC DOCUMENT

The dynamic documents are not present in a predefined format, like static documents. A dynamic document is created by a web server whenever a browser requests for the document. Let us consider figure 14.14 to understand how a dynamic document is created and passed on to the client. First the client sends a request to the web server. After receiving this request, the web server will execute an application program to create a dynamic document. The server returns the dynamic document as a response of the request to the client.

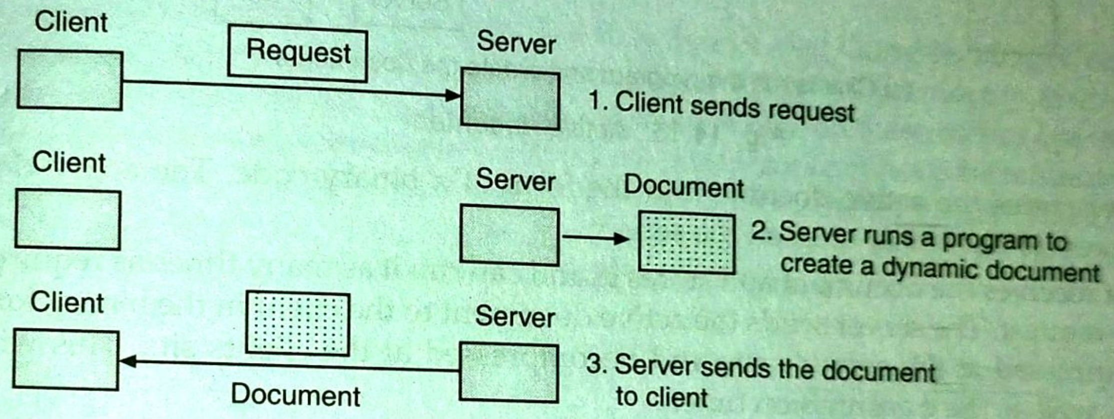

Client
Request
Server
1. Client sends request
Client
Server
Document
2. Server runs a program to create a dynamic document
Client
Server
Document
3. Server sends the document to client

Fig. 14.14. Dynamic document.

The contents of a dynamic document will be different corresponding to every request. A simple example of a dynamic document is to get time and data from the server. A server follows the following steps given below to handle dynamic documents :

(i) The server checks the URL in order to find if it has defined a dynamic document.

(ii) If the URL has defined the dynamic document, then the server executes the program.

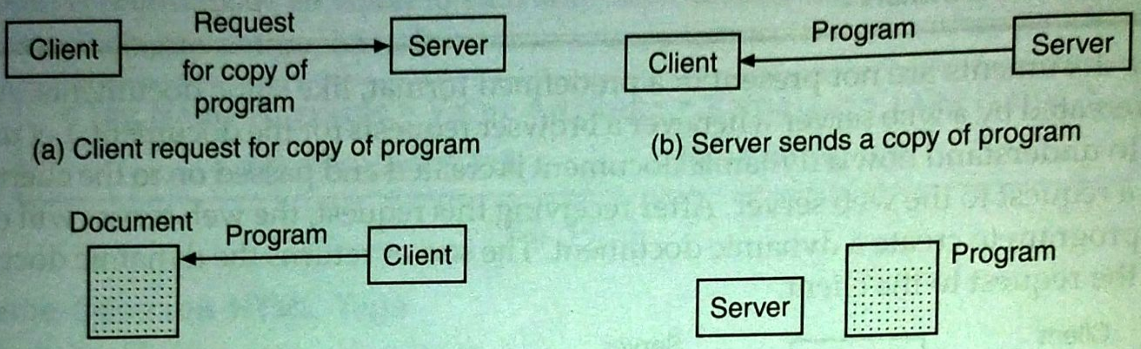
(iii) The output of this program is the dynamic document. It is returned back to the client.

CGI Program

It is a code which is written in one of the languages that supports CGI (such as C, C++, etc.).

14.6.2. Active Documents

Active documents can be defined as the program, that is needed to be run at the client side. The examples of active documents are the programs creating animated graphics on the screen or the one which helps interaction with the user. Let us consider figure 14.15. It shows that whenever a browser requests for an active document, the server will send a copy of document in the form of byte code. The active document will then be run at the browser (client) site.



(a) Client request for copy of program

(b) Server sends a copy of program

(c) Clients run the program and create the documents

Fig. 14.15. Active document

The server stores the active document in the form of a binary code. The active document is stored on the server but it is not run on the server.

The client receives the document and stores it, and can run it as many times as required without repeating the request. The server sends the active document to the client in the binary form. Hence, it can be compressed at the server's site and decompressed at the clients site. This will save the bandwidth as well as the transmission time.

Steps in Creation of an Active Document

Let us consider figure 14.15 to understand the creation, compilation and execution of an active document.

(i) At the server, a program is written in source code and stored in a file.

(ii) Then, the program is compiled and binary code is created and stored in a file at the server's site.

(iii) A client (browser) requests for a copy of program as shown in figure 14.15 (a). This program is transported from the server to the client in the compressed form.

(iv) The client converts the received program from binary code into executable code using its own software.

(v) The client runs the program to create a result which can include animation or interaction with the user.