# Server-side Programming

Most large-scale websites use server-side code to dynamically display different data when needed, generally pulled out of a database stored on a server and sent to the client to be displayed via some code (e.g. HTML and JavaScript).

Perhaps the most significant benefit of server-side code is that it allows you to tailor website content for individual users. Dynamic sites can highlight content that is more relevant based on user preferences and habits. It can also make sites easier to use by storing personal preferences and information — for example reusing stored credit card details to streamline subsequent payments.

It can even allow interaction with users of the site, sending notifications and updates via email or through other channels. All of these capabilities enable much deeper engagement with users.

In the modern world of web development, learning about server-side development is highly recommended.

## What is server-side website programming?

Web browsers communicate with [web servers](https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server) using the **H**yper**T**ext **T**ransfer **P**rotocol ([HTTP](https://developer.mozilla.org/en-US/docs/Glossary/HTTP)). When you click a link on a web page, submit a form, or run a search, an **HTTP request** is sent from your browser to the target server.

The request includes a URL identifying the affected resource, a method that defines the required action (for example to get, delete, or post the resource), and may include additional information encoded in URL parameters (the field-value pairs sent via a [query string](https://en.wikipedia.org/wiki/Query_string)), as POST data (data sent by the [HTTP POST method](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST)), or in associated [cookies](https://developer.mozilla.org/en-US/docs/Glossary/Cookie).

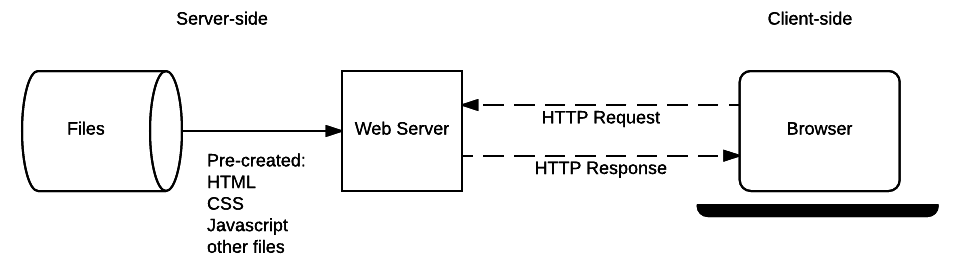
Web servers wait for client request messages, process them when they arrive, and reply to the web browser with an **HTTP response** message. The response contains a status line indicating whether or not the request succeeded (e.g. "HTTP/1.1 200 OK" for success).

The body of a successful response to a request would contain the requested resource (e.g. a new HTML page, or an image, etc...), which could then be displayed by the web browser.

## Static sites

The diagram below shows a basic web server architecture for a *static site* (a static site is one that returns the same hard-coded content from the server whenever a particular resource is requested). When a user wants to navigate to a page, the browser sends an HTTP "GET" request specifying its URL.

The server retrieves the requested document from its file system and returns an HTTP response containing the document and a [success status](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#Successful_responses) (usually 200 OK). If the file cannot be retrieved for some reason, an error status is returned (see [client error responses](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#Client_error_responses) and [server error responses](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#Server_error_responses)).



## Dynamic sites

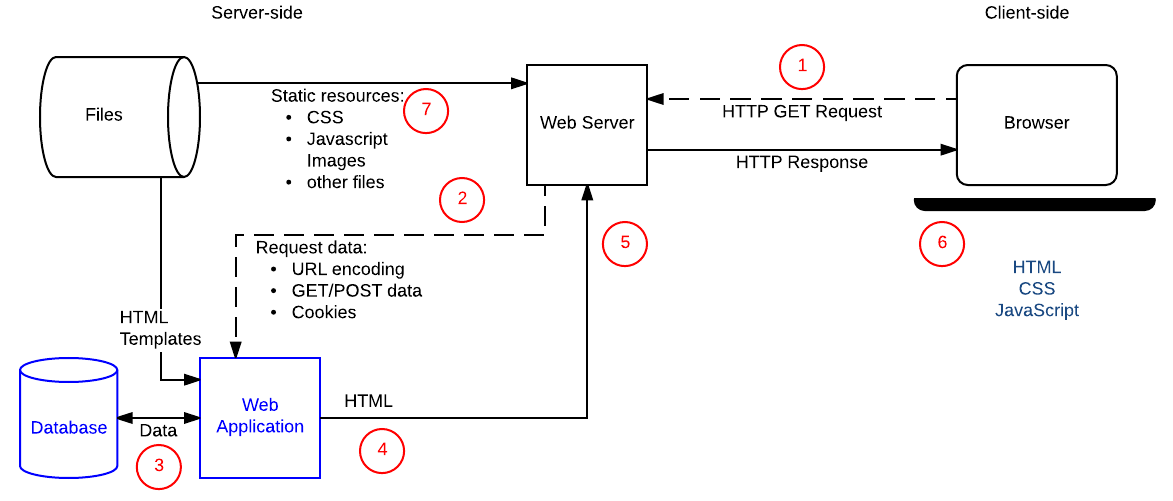
A dynamic website is one where some of the response content is generated dynamically, only when needed. On a dynamic website HTML pages are normally created by inserting data from a database into placeholders in HTML templates (this is a much more efficient way of storing large amounts of content than using static websites).

A dynamic site can return different data for a URL based on information provided by the user or stored preferences and can perform other operations as part of returning a response (e.g. sending notifications).

Most of the code to support a dynamic website must run on the server. Creating this code is known as "**server-side programming**" (or sometimes "**back-end scripting**").

The diagram below shows a simple architecture for a dynamic website. As in the previous diagram, browsers send HTTP requests to the server, then the server processes the requests and returns appropriate HTTP responses.

Requests for static resources are handled in the same way as for static sites (static resources are any files that don't change —typically: CSS, JavaScript, Images, pre-created PDF files etc).



Requests for dynamic resources are instead forwarded (2) to server-side code (shown in the diagram as a Web Application). For "dynamic requests" the server interprets the request, reads required information from the database (3), combines the retrieved data with HTML templates (4), and sends back a response containing the generated HTML (5,6).

Source: <https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks>