What are the differences between server-side and client-side programming?

> I've seen questions (mainly on Stack Overflow), which lack this basic knowledge. The point of this question is to provide good information for those seeking it, and those referencing to it.

In the context of web programming, what are the differences between Server-side programming and Client-side programming? Which languages belong to which, and when do you use each of them?

web-development server-side client-side

edited Jan 12 '15 at 7:49
Community ♦
1

asked Oct 24 '12 at 14:08
Madara Uchiha
3,784 6 18 32

5 Server-side programming is writing code that runs on the server, using languages supported by the server (such as Java, PHP, C#; it is possible to write code that executes on the server-side in JavaScript). Client-side programming is writing code that will run on the client, and is done in languages that can be executed by the browser, such as JavaScript. – FrustratedWithFormsDesigner Oct 24 '12 at 14:13

1 This one is pretty good: programmers.stackexchange.com/questions/138561/… – JeffO Oct 24 '12 at 14:16

7 I think you should include in the question that you refer only to web programming, as in the current form the answers are not complete. For example, server-client communication need not be done in HTTP; client side might not use a browser, etc. – K.Steff Oct 24 '12 at 14:42

@KSteff you are welcome to edit my question to add that. – Madara Uchiha Oct 24 '12 at 17:06

2 When the web was young it was good practice to put most of your logic heavy lifting on the server side (java/c++) and keep browser logic intentionally thin - particularly since browsers back then were not ready for prime time. Now this emphasis has reversed such that browser based tools (Angular.js) are where the bulk of the web application heavy lifting logic now resides (away from the increasingly stripped down server side logic). This has been facilitated by modern industrial strength browsers running extremely fast javascript engines (within an order of magnitude of native code). – Scott Stensland Nov 4 '14 at 16:54

4 Answers

Background

Web development is all about communication. In this case, communication between two (2) parties, over the HTTP protocol:

- The **Server** - This party is responsible for **serving** pages.
- The **Client** - This party *requests* pages from the **Server**, and displays them to the user. In most cases, the client is a **web browser**.
  - The **User** - The user *uses* the **Client** in order to surf the web, fill in forms, watch videos online, etc.

Each side's programming, refers to code which runs at the specific machine, the server's or the client's.

Basic Example

1. The **User** opens his web browser (the **Client**).
2. The **User** browses to http://google.com.
3. The **Client** (on the behalf of the **User**), sends a request to http://google.com (the **Server**), for their home page.
4. The **Server** then acknowledges the request, and replies the client with some meta-data (called *headers*), followed by the page's source.
5. The **Client** then receives the page's source, and *renders* it into a human viewable website.
6. The **User** types Stack Overflow into the search bar, and presses Enter
7. The **Client** submits that data to the **Server**.
8. The **Server** processes that data, and replies with a page matching the search results.
9. The **Client**, once again, renders that page for the **User** to view.

Programming

Server-side Programming

Server-side programming, is the general name for the kinds of programs which are run on the **Server**.

Uses

- Process user input.
- Display pages.
- Structure web applications.
- Interact with permanent storage (SQL, files).

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.

Client-side programming

Much like the server-side, Client-side programming is the name for all of the programs which are run on the **Client**.

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*
- Any language running on a client device that interacts with a remote service is a client-side language.

*HTML and CSS aren't really "programming languages" per-se. They are markup syntax by which the **Client** renders the page for the **User**.

edited Dec 30 '15 at 17:58 community wiki

5 revs, 3 users 92%
Madara Uchiha

7 +1 for a good answer *with* examples of the uses! Just to nitpick: HTML and CSS are not actually programming languages, so they probably shouldn't be compared to "PHP, ASP, and Nearly any language (C++, C#, Java)". ActionScript might be another good example of a client-side language. – FrustratedWithFormsDesigner Oct 24 '12 at 14:25

4 You fail to identify *why* the server is a server and the client is a client. The server is known about by the client, but not the other way around. The server is expected to be executing at all times, there are no client expectations. – Chris McCall Oct 24 '12 at 15:04

3 I would add the fact that a server environment is more controlled. You have no idea what the client is. Also there are security concerns(for both parties) when doing things client side. – stonemetal Oct 24 '12 at 16:28

1 So add it, feel free. – Madara Uchiha Oct 24 '12 at 17:07

I disagree with @ChrisMcCall's definition to a point. There could be exceptions to that rule, such as where a server might rely on a client to process data or provide a service to the server for the server to complete a task. Clients also are sharing an increasing amount of the load for scalability and performance such as in SPAs. These technologies blur that definition. A better definition could be that the end-user and the client are synonymous. It is expected that the end-user exists at the client device, whereas all other nodes would be considered server-side. – RyanJMcGowan Sep 1 '13 at 8:32

In layman's words:

Here I will talk only about web programming.

Client side programming has mostly to do with the user interface, with which the user interacts. In web development it's the browser, in the user's machine, that runs the code, and it's mainly done in **javascript, flash,** etc. This code must run in a variety of browsers.

Its main tasks are:

- validating input (Validation must be done in the server. A redundant validation in the client could be used to avoid server calls when speed is very critical.)
- animation
- manipulating UI elements
- applying styles
- some calculations are done when you don't want the page to refresh so often

The **person in charge** of front end programming **must know**:

- javascript
- css
- HTML
- basic graphic design
- Ajax
- maybe Flash
- some 3rd party javascript libraries like JQuery
- UI design
- information design, etc.

Server side programming has to do with generating dynamic content. It runs on servers. Many of these servers are "headless". Most web pages are not static, they search a database in order to show the user updated personalized information. This sides interacts with the back end, like say, the database.

This programming can be done in a lot of languages:

- PHP
- Java and jsp
- asp
- Perl
- Python
- Ruby on Rails, etc.

This code has to do with:

- Querying the database
- Encode the data into html
- Insert and update information onto the database
- Business rules and calculations

The person in charge of server side programming must know:

- some of the languages mentioned above
- HTML
- SQL,
- linux/unix shell scripting
- OOP
- business rules, etc.

edited Jan 12 '15 at 16:04

answered Oct 24 '12 at 14:36
Tulains Córdova
32.4k 10 75 128

"The person in charge of front end programming must know" Must? I would say that you can perfectly survive with only HTML, CSS, Javascript and Ajax. Saying that server-side programming has to do with generating dynamic content and not saying the same for client side will probably give the wrong intuition... – nbro Feb 29 '16 at 20:52

"Most web pages are not static, they search a database in order to show the user updated personalized information. This sides interacts with the back end, like say, the database." I would rephrase it as: "Pages are generated dynamically by filling the dynamic parts with variable content which is fetched usually from a database. The back-end is already everything related to server-side programming, IMO. – nbro Feb 29 '16 at 20:54

Again, in general, I would say "...a person should know..." and not "must"... – nbro Feb 29 '16 at 20:59

@nbro Why don't you just write your own answer? – Cole Trumbo Nov 9 '16 at 17:46

@ColeTrumbo What's the connection between my comment and yours? Can people critic others' answers in order to improve them? – nbro Nov 9 '16 at 17:56

Other answers have focused on **what** is client-side and server-side programming: what languages are mostly used, what tasks have to be accomplished, etc.

This is absolutely right, but I miss a bit of focus on **what are the differences** between both types of programming, in the context of web programming. Let me try to address that.

Security and permissions

In client-side programming, you do not have access to the full system, because of security concerns. The user does not necessarily trust each and every piece of code that is downloaded from the web and executed on his machine, and this is the main design goal of the client-side environment (the browser and JavaScript engine): to provide an isolated environment where client code can execute but cannot access anything outside the allowed scope.

In server-side programming, it is good practice to also limit the access of each application to the underlying system, but this is much less enforced on you, since in the end, you or your company are in control of that system. This 'isolated cage' design is **not** built-in into the server-side programming tools and languages, but is accomplished through installation setup (using dedicated users with restricted permissions, choosing ports that require or do not require root permissions, etc).

Deployment and platform

In server-side programming, deployment has to happen from outside your code, using some kind of tool (even if it is `make install` or a `git clone`), and this deployment is usually manual — or at least, it is expected to happen in a semi-supervised way. The system (meaning the OS) on which you deploy is usually uniform across a number of machines, but it can be heavily customized to your needs.

In client-side programming, deployment happens from your server-side code, which serves the clients automatically and without supervision. The underlying system (meaning mainly the browser) can be very different across a much larger number of machines. In order to make deployment feasible at all, standards have to be kept, and there is a much stronger trend to a single language and environment.

This is why copying server-side code from one machine to another can take weeks, while client-side code is usually trivial to execute in different machines.

State and secondary effects

(Disclaimer: this is by far the most subjective point of all. Probably there are many wrong aspects to my argumentation. It is just an interesting hypothesis, in my view.)

In server-side programming, state is a much bigger concern, meaning how to retrieve and update data at the request of the user with the possibility of conflicts due to concurrency. Even if most of this complexity is offloaded to a database server, it is the server-side code's responsibility to allow the database to keep its guarantees on data integrity by using its interface correctly (e.g. not use a cache for updates that are never seen by the DB), while it is also a goal of the server-side code not to overload the database with work and keep the user waiting for response.

In client-side programming, presenting the results to the user is a much bigger concern, and this implies secondary effects (mostly printing to the screen). This is not to say that there is no state involved (e.g. cookies), only that the main goal of the code is to actually interface with the user, and this cannot happen without secondary effects.

This is why client-side programming usually requires (at some point) looking at the screen with a demo, to check that all colors and layout are right, while server-side programming can happen almost exclusively in a text-oriented environment, where automated tests check that the logic is still doing what it is supposed to do.

answered Jan 12 '15 at 12:37

logc
1,834 7 16

This is by no means intended to be an accepted answer; rather I offer it as a complementary point (in response to the `when do you use each of them` question) that has yet to be mentioned in the other answers thus far, namely:

Protection of intellectual property

Source code that sits on the client-side (such as in Javascript) is easily readable and / or capable of being reverse-engineered if it has been obfuscated.

Source code that sits on the server-side however can safely protect proprietary algorithms and only return the result; a black-box of sorts.

answered Mar 16 '16 at 19:00

Kosta Kontos
197 1 8

Yeah, but that's not really the most important point, the server is there to server, and the client is there to receive. Some logic is best done on the client (like a shopping cart, you don't let the supermarket keep track of your shopping cart at all time, do you?), and some is best done on the server (getting information from database) – Madara Uchiha Mar 17 '16 at 8:06

@MadaraUchiha, hence the preamble: "This is by no means intended to be an accepted answer; rather I offer it as a complementary point" – Kosta Kontos Jun 9 '16 at 8:38

Shouldn't this be part of another post then? I'm new here, but isn't it preferable to have one complete answer than scattered ones? – Julix Aug 18 '16 at 6:48
