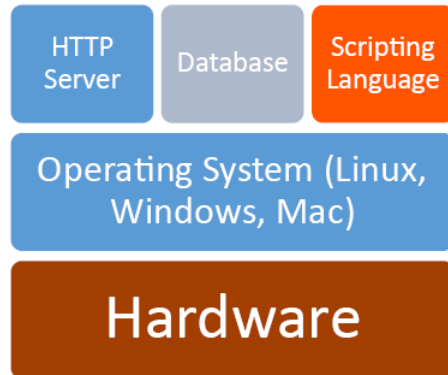


## 3.1: Server-Side and Client-Side Scripting

### Server-Side and Client-Side Scripting



#### PHP

Created in 1994 at the hands of Rasmus Lerdorf, PHP began as a set of CGI scripts developed to track views of his resume online. Rasmus continued adding scripts to his collection so he could do more with his websites. Over time, some friends began to use it as well. By June of 1995, enough of a framework was in place that Rasmus decided to make PHP public. As others embraced it, and began to submit their own work, PHP grew. By version 3 it was decided that the time had come for a more professional name. In homage to its original name of Personal Home Page, the PHP acronym was kept, but was changed to a recursive representation of “hypertext preprocessor.” PHP was now an independent language, with object oriented capabilities, high extensibility, and had a growing following.

As the community grew, the core team of Rasmus, Andi Gutmans and Zeev Suraski continued their work. Gutmans and Suraski rewrote the core of the engine, and dubbed version 4 Zend, a blend of Gutmans and Suraski’s first names. Now with dozens of developers and even more contributors, PHP has grown to version 5, and is installed on tens of millions of servers around the world. It continues to rank as one of the top ten web development languages.

With strong semblance to languages like C++ and Perl, the goal was to create a language that allowed fast development of dynamic pages. It is a server-side language, which means it runs on the server before anything is sent to the user’s computer. This is in contrast to client-side languages, where the code is sent to the user’s computer to be processed locally with languages like JavaScript.

Some advantages to server-side languages are that the code is hidden from the user, and secures what is taking place in the background. It also reduces the work load that the user’s computer is burdened with. This, however, also means the server must be powerful enough to support the number of users requesting pages, as it must bear the brunt of the computation.

PHP is a parsing engine, which means it examines the php file, performs any php related tasks it finds, and passes the result to the web server. This makes it an interpreted language, as the output and script are run on demand, as opposed to a compiled language where the code is transformed and saved into a runnable form.

#### JavaScript

JavaScript is a client-side script, meaning the browser processes the code instead of the web server. Client-side scripts are commonly used when we want to validate data before sending it to the web server, adjusting the interface in response to user feedback, and for implementing other advanced features. Since JavaScript is part of the browser, it can be run without a web server present. If the computer is slow or busy, the performance of our code may be reduced. If JavaScript is disabled (less of a concern today than just a few years ago) then our script will not run. This being said, this is less of an issue now, and JavaScript can reduce the number of communications to a server, reducing transmission time and improving performance.

JavaScript will be our client-side scripting example. As JavaScript can be handled within the browser, we can capitalize on it to validate user data, react to user actions that affect appearance, and interact with the user’s computer, without requiring the involvement of their internet connection or our server. Like CSS, JavaScript is not a fully formed language that can stand on its

own. Like PHP and Java, JavaScript is an object oriented language that is multi-platform. Unlike Java (but still like PHP) it is a loosely, typed language.

There is a common misconception that Java and JavaScript are the same thing. Review some of the larger differences between them below if you are already familiar with Java.

Table 3.1.1 Java vs JavaScript

JavaScript	Java
Object-oriented. No distinction between types of objects. Inheritance is through the prototype mechanism, and properties and methods can be added to any object dynamically.	Class-based. Objects are divided into classes and instances with all inheritance through the class hierarchy. Classes and instances cannot have properties or methods added dynamically.
Variable data types are not declared (dynamic typing).	Variable data types must be declared (static typing).
Cannot automatically write to hard disk.	Cannot automatically write to hard disk.

Writing JavaScript code is a lot like writing PHP. Both languages use many of the same concepts and can look very similar in terms of code. Since we will have covered many of the foundational concepts JavaScript uses in the PHP section, we will focus on differences between JavaScript and PHP. We will look at examples that highlight how JavaScript can be integrated with other languages, like responding to event driven actions to modify our pages in real time. Bear in mind that the power of the language can be used to perform many of the same tasks we have examined already in PHP.

A best practice for using JavaScript in your site is to create your entire site without it, and then add it where it can improve the user experience (a process called progressive enhancement). This will help ensure that your site will still operate (albeit maybe not as attractively) if JavaScript is not present.

---

3.1: Server-Side and Client-Side Scripting is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

- [3.1: Server-Side and Client-Side Scripting](#) by [Michael Mendez](#) has no license indicated.