# The Best Web Programming Languages Course

## Programming Languages: An Introduction

A programming language is a formal constructed language designed to communicate instructions to a machine, particularly a computer. Programming languages can be used to create systems to control the behavior of a machine or to express algorithms.

Since the invention of computers, hundreds of programming languages have been designed and made available to the whole world, and more are being constantly created.

Programming languages are different in a set of aspects that include the two main components of a programming language, the semantics and the syntax. For your own understanding, the syntax is the form or type, while the semantics are the meaning of that type or form.

## Programming Languages Characteristics

Every language is different but there is some point most of them come together to the following three main traits:

• Abstractions - abstraction is a technique for managing complexity of computer systems. Most of the languages have particular rules that help define data structures and also manipulate the manner in which commands and instructions are executed.

Through abstraction, the programmer can achieve more in less steps and add extra levels of functionality according to his/her needs while hiding unnecessary (in the viewpoint of the programmer) operations the underlying hardware uses to make things work and represent data.

• Function and Target - We use web programming languages in various ways and in order to achieve various functionality. Thus, the function that a language is used for, or the target goal we aim to complete with it, does define a language and classifies them into several groups like client-side, server-side, etc

• Expressive Power - Yet another classification of programming languages can be made considering the computations that they can express. This is an important aspect to be analyzed before starting an actual project.

## What to consider when choosing a programming language?

Choosing a programming language to learn or apply is not always an easy process and often does not depend on personal preferences, rather than what is going to be worked on, or even what is strictly required by higher level staff members like the software architect. The following list introduces some of the most important factors you should consider when choosing a web programming language:

• ***Community*** - Although most languages nowadays have a detailed documentation on their official sites, programmers often get stuck in such problems that require more than the basics in order to be solved. A wide community supporting forums on a language is of a great help and will save you a lot of time as well as unnecessary lines of coding. If you're among the few working with a language, chances are, you're on your own, and no one can really help you on specific problems you might encounter.

• ***Security*** - Teams and individuals work hard to deliver secure ways for you to protect what you are building from harmful attacks, but don't forget hackers also work hard to break down systems of their interest. Although languages get refined every year or so, choosing the actual most secure language (meaning the one which

supports more security features than others) is the way to making sure your work is not going to be in the rubbish bin anytime soon (well, not always, but at least, you have to try)

• ***Future Stability*** **-** The field of web development is changing fast, and what users need constantly define new ways/processes/ tools that developers need to achieve their aims. Needless to say, programming languages are also on the famous process of rising and fading respectively for some and others. It is a good consideration to work with a programming language that you (or others) think that will still be popular after at least 5 years (so you don't get to change what you've previously worked on every 1,2 or 3 years). A good example of a stable language is Javascript, which seems to have a long way to go into the history of web programming languages.

• ***Integration*** **-** Because companies/individuals that create programming languages also create tools and other technologies, it is common that some languages will work better using the tools created by the very same people who invented the language. For example, .NET works best if combined with other Microsoft technology, PHP and Apache are also a great match etc.

• ***Tools & Libraries*** **-** Programming languages are only the core, and there is a lot to be planned and researched in order to come up with what is required. At this point, tools and libraries provide predefined code (frameworks) to help you get more done and in a rather optimized way. Libraries do stand for a lot of complex functions that otherwise would be a pain in the ass and still even if you did pass the done phase, it would probably going to break down on some cases you hadn't thought about.

Languages, usually, have their respective supporting libraries and tools that can be used to enhance productivity and efficiency. However, it is not always true that the more libraries are available to a language, the better this language is.

**HTML & CSS: The Fundamentals of Web Development**

HTML and CSS are the most important technologies you have to know in web development. Even though the easiest ones, they provide the basic structure for the static pages of your website. HTML is a markup language made of elements expressed in tags and various attributes they can have. On the other hand, CSS stands for Cascading Style Sheets and provides the necessary styling for pages. Both of them are pretty easy to start learning, and they give a beginner a taste of what is it like to program.

HTML and CSS are not programming languages, but they do have certain rules and syntaxes, which is common trait for other programming languages out there.

**Easy to Learn, Difficult to Master**

Seems like a joke, but seriously, think about that. Most projects require that you design a website's static pages just like a psd template they found on the internet. This is where the whole thing enters a new level. It is easy to start on your own and build whatever it comes to your mind, but when requirements are mandatory, things get harder. While HTML is basically of the same "difficulty", CSS seems to be messing everything up and wasting you hours on little details that need to be on the right position, showing properly on smaller screen devices and stuff like that. Not to mention vendor-prefixes that make your work look different on different browsers. However, don't get too scared. Frameworks are here to help in a lot of aspects.

**CSS Frameworks**

CSS frameworks are the most common thing you'll ever encounter on websites you or others build. It means less code by you, and getting more done properly. While there are a lot of frameworks that have already made it

to potential customers (that is, well-known websites) the most popular ones usually have more features and provide better stability. One of the world's most popular front-end frameworks is Bootstrap which makes development in the front-end faster and easier. And it is really easy to learn thanks to the great documentation online. Another awesome framework for CSS is Foundation by Zurb which also provides quite a lot of features and tools.

**What's Next?**

One thing is for sure, HTML and CSS are going to be here for a way too much time. No website can function without these two core technologies, that provide the core markup for pages. Both are widely supported in forums and discussion boards on various famous sites on the internet and that is what you will need from time to time, a way to get things working when you're stuck.

Having a good knowledge in HTML5 and CSS3 (referred to as the latest versions) will most probably serve you for a brighter path on other langauges, but it might also be getting you a job, like working on HTML and CSS as a designer, especially when you're coding the pages you've designed. Everybody will appreciate that.

**Javascript**

Javascript is used to make website interactive. But what makes that so cool and awesome? Javascript is not only helping developers build great front-ends in combination with HTML and CSS, but it is also found on mobile applications, and even on the back-end in the form of Node.js. The growth of Javascript came because of its great user experience features across platforms. There is not a required development environment that needs to be set up for beginners to start coding and testing, you can get started right from your browser, which is nice for everyone trying to learn it. Javascript is not a single purpose langauge.

It is so scalable and flexible that can be used for a lot of tasks like animation effects, logical algorithms, and even databases.

**Complexity: It's Everywhere**

With great features, comes great complexity. While it is easy for most to learn the basics (which is true for almost every programming language), it is rather complex trying to achieve more advanced stuff. One way of looking at it is browsers. Your very same Javascript code may not behave the same way in different browser environments, and figuring out why your code works in Chrome but not in Firefox or other browsers may be boring or even frustrating sometimes.

Javascript is a dynamically untyped languages, which basically means the same thing can mean something different depending on the context. As projects grow larger, it may be hard to maintain and debug when errors occur (which is not that rare). It takes experience to learn the tricks to automated testing and other techniques used to avoid bugs.

**Javascript Frameworks**

There are many Javascript frameworks built by both professionals and individuals, and most of them, are powerful tools for web developers to use in their projects. The following ones, are the most popular and useful frameworks that programmers use everyday.

***jQuery*** **-** Now a decade old, jQuery is one of the plugins that need no introduction. It is the most used Javascript framework in the world and hardly any app on the web ignores it. It is responsible for making cross-browser sites a reality and offers a ton of other features like:

- DOM Traversal
- Event Handling
- Animations
- AJAX

***AngularJS*** **-** Developers are more and more using AngularJS to build and maintain complex web applications. AngularJS is also widely used to create Single Page Application in a rather clean and maintainable way. The following are the most important core features of it.

- Data-Binding - which is the automatic synchronization of data between the model and view components.
- Scope - objects that refer tot he model. They serve as a middleware between controller and view.
- Controller - Javascript functions that are bound to a particular scope
- Directives - markers on DOm elements, used to create custom HTMl tags that serve as new, custom widgets.
- Routing - the concept of switching view.
- MVC (Model - View - Controller) - MVC is a design pattern for dividing an application into several different parts, each with it's own unique responsibilities.
- Deep Linking - allows you to encode the state of application in the URL so that it can later be bookmarked. That actually means the application can be restoren from the URL to the same state.
- Dependency Injection - AngularJs has a built-in dependency injection subsystem that helps the developer by making the application easier to develop, understant and test.

***React*** **-** this nice open-source Javascript library provides a view for data rendered as HTML. Views are typically rendered using components that contain additional components specified as HTML tags. Some of the main uses of it include:

• Just the UI - A lot of people use React as the View in Model-View-Controller. Since React makes no assumptions about the rest of the technology stack, it is easy to try it out on smaller features in an existing project.

• Virtual DOM - React abstracts away the DOM from you, giving a simpler programming model and way better performance.

• Data Flow - React implements one-way reactive data flow which reduces boilerplate and is easier to reason about than traditional data binding.

***Backbone*** **-** yet another Javascript framework with RESTful JSON interface and based on the MVC design pattern. Backbone helps you make code modular.

***Ember*** **-** Also a MVC framework, which includes a templating and view engine that automatically updates when data changes, like Angular, Backbone and React. It presents the concept of web components that let programmers extend HTML with their own tags.

**The Future of Javascript**

Javascript is one of the most popular web programming languages in the world. It will continue to stay relevant to the web for quite a long time. Web applications are increasingly being created and used on both desktop and mobile, and with the ongoing innovation on Javascript, it will only get better and more useful for a rising community of its users all around the globe. Being such a poweful language, support is a must, and people seem to be more interested to both replying and learning from forums and dicussion boards. Jobs for Javascript

developers do pay quite a lot, with an average annual salary of around $90k and up to $125k for full-stack developers of Javascript.

**Python**

Python is such a general purpose language that is used in many applications and domains like web development, education, desktop GUI, software development etc. It sands for handling integration tasks, being versatile and extensible (the Python Packages include thousands of third party modules for python). In web development, Python is mostly found on the back end side with frameworks such as Django and Pyramid, micro-frameworks like Flask and Bottle and advanced content management systems such as Plone.

**Python Features**

Python did not become so popular by chance. There stand several reasons (features) why this langauge made it to a lot of applications and users all around the world. The following introduce the most important ones:

• Easy to Learn - Nobody would prefer a language that is hard to learn (at least the basics). Python is a minimalistic language.

It makes programmers focus on the problem and not on the language because of the pseudo-code it has that feels like reading English. The syntax is so simple that most people get surprised when being introduced to it for the first time.

• Open Source - To be clear, open source is not just about being free. It means people can freely distribute copies of it, access the source code and most importantly, make changes to it for use in personal projects. The language is constantly being improved by a wide community of developers.

• High-Level Language - The higher the language level, the less you have to worry about how the machine understands it. For example, if in C you have to think of memory allocation, you don't have to in Python. Low level details are avoided in a lot of aspects in Python.

• Portability - Being open source comes with great advantages, and one of them is bringing the language to a whole bunch of platforms like Linux,Windows, Macintosh, FreeBSD etc. If you are careful enough to avoid system dependent code, than your programs will work seamlessly in all platforms.

• Object Oriented - Both procedure-oriented programming (which is building a program around procedures/functions that are pieces of reusable code) and object-oriented programming are supported in Python. They are powerful tools in the hands of developers and come in really simple ways of implementation.

• Extensibility - Python takes future growth of implementation seriously. You can code a part of the program in C or C++ for performance reasons and use it in your Python program. Python is also embeddable. You can embed it within your C/C++ programs to give them "scripting" capabilities for your program's users.

• Interpreted - Python instructions are executed directly, without previously being compiled into machine-language instructions. In other words, it does not need compilation to binary code, and you can run the program directly from the source code.

Some of the world's most popular sites use Python to make their services available to every user. Some of these websites include:

• Google - the most popular search engine uses Python to handle the traffic and computing needs of the search engine and its connected apps.

• Youtube - Python is used widely in Youtube to power all the features we love. Everytime you watch a video on Youtube, you are actually executing a bunch of Python code.

• Dropbox - uses a combination of wxPython and PyObjC on the Mac to deliver service bound to the disk and network.

• Quora - Adam d'Anggelo, answering to the choice of Python for Quora development, says it has only two drawbacks, speed and typechecking. He says Python was found to be fast enough for what they needed it, while they worked hard to reinvent some technology to solve typechecking.

• Survey Monkey - yet another service that started the application from the beginning to write it in Python, which was a rather promising language for testing and deploying new features.

Python was designed for web servers that handle a large amount of traffic, that is why it is chosen by industry leaders. It helps do more with fewer lines of readable code.

**Ruby**

Designed and developer by Yukihiro Matsumoto in the mid 1990s in Japan, Ruby was the perfect language for Yukihiro, as it was:

- Truly Object Oriented
- Syntactically Simple
- Had Iterators and Closures
- Exception Handling
- Garbage Collection

Ruby is a decent performer, but really fast hardware shall be used to make use of the good performance. Since hitting version 2.0, introducing a lot of new features and stability improvements.

**Ruby on Rails**

Ruby on Rails is the most powerful Ruby web framework out there. Just like in other langauges, you should learn good Ruby (at least the basic stuff) in order to understand and work on frameworks. Ruby on Rails supports the best technologies, that save you a ton of time tring to achieve using alternative or fragmented services. But there is a lot more to that, and the following gives you a better idea why you should learn Ruby on Rails.

• MVC architecture - Ruby on Rails is based on the MVC design pattern that enables data to be separated from presentation.

• Community - Ruby and Rails are both open source, and have a thriving community all around the world that is willing to help you, which is great for new developers.

• Debugging - Rails provides detailed error logs, making it easier than ever before to debug applications.

• Libraries - a host of libraries is available that do a great job in simplifying coding of common programming tasks like form validations, sessions management etc.

• Database Access Library - Ruby on Rails includes Active Record which simplifies data handling in databases by automatically mapping tables to classes and rows to objects.

Ruby is similar to Python. Just like Python, it was designed to make programming more productive by emphasizing short and simple code that is consistent and flexible.

The main difference between the two is in the language or syntax they use. In Python, there is only one right way to program stuff, which is obviously efficient and fast. In Ruby, there are several ways to do the same thing, and some may be faster than others.

Regarding frameworks, Ruby on Rails is a very common open-source web framework that enables developers to create dynamic websites in a quick and productive way.

**ASP**

ASP is short for Active Server Pages, and obviously, it is a server-side scripting language. ASP is developed by Microsoft, and it is primary thought to work under IIS (Internet Information Server) on Windows NT Server. ASP enables the server to deliver dynamic, database driven content to the client with minimal effort.

Similar to other back-end languages, servers first read the ASP file to see if there are tasks that need to be executed. Only when the server is done with the execution part, the result is delivered to the client. The client only sees the result of the server's work. That means when you try to view the source code by the "View Source" common menu in web browsers, you only the rendered HTML and CSS, but not the actual instructions that made that markup look like that.

But why should developers learn ASP?

• Language Independence ASP is defined as a scripting engine which helps developers develop in virtually any language. There are endless modules for other languages like Perl and Python that extend the engine implementation capabilities.

• Ease of Use You can use ASP on your HTML pages just by embedding it like this: <% and %>

• Hosting There is a wide support by hosting companies and prices aren't high. There are more than 800 hosting companies currently supporting ASP with prices under $10.

• Extensibility COM components make ASP a limitless language. There might now be a way to send e-mail using standard ASP functions but there are a lot of component that will help you achieve this.

• Short Learning Curve It is easy to learn ASP if you're already a programmer or even if you only know basic HTML. The learning curve is short and not so difficult.

• Huge Community There is a wide community of professional ASP developers ready to answer your questions all over the internet. So, you will never get stuck in some sort of trouble.

**Perl**

Perl is a programming language mainly designed for text processing. It was developed by Larry Wall. It runs on a variety of platforms like Windows, Mac OS and Linux. The language can also be considered a general-purpose language used for a wide range of tasks including system administration, web development, network programming and GUI development. Perl is not officially an acronym but few people used it as **Practical Extraction and Report Language**.

Perl supports both procedural and object-oriented programming and has an impressive collection of third-party modules. Perl 5 added some new features that support complex data structures, first-class functions, which

means, closures as values, and an object-oriented programming model. These features include references, packages, class-based method dispatch, and compiler directives. Another major additional feature introduced with Perl 5 was the ability to package code as reusable modules. According to Larry Wall "The whole intent of Perl 5's module system was to encourage the growth of Perl culture rather than the Perl core."

All versions of Perl do automatic data-typing and automatic memory-management. The interpreter knows the type and storage requirements of every data object in the program; it allocates and frees storage for them as necessary using reference counting.

Perl derives from the C programming language and to a lesser extent from sed, awk, the Unix shell, and many other tools and languages.

**Go**

The Go programming language is an open source project (programming language) created by Google in 2007. Go is recognizably in the tradition of C, but it makes changes to improve conciseness, simplicity, and safety.

Go is expressive, concise, clean, and efficient. Its concurrency mechanisms make it easy to write programs that get the most out of multicore and networked machines, while its novel type system enables flexible and modular program construction.

Some other advantaged of the Go languages are:

• Compiles very quickly

• Supports concurrecy at the language level

• Functions are first-class objects in Go

• Strings and maps are built into the language

• Go has garbage collection

**Java**

Even though it seems like Java is mainly the desktop language, it is used quite extensively to create web application. This is done using the Java Enterprise Edition. According to W3Tech, Java is the server-language of choice for large-scale websites with a high volume of traffic. Java Servlets, JSP and WebObjects are some of the solutions on the server side which use Java. An important reason why Java is used among sites with such workload is the speed it offers. Let's have a brief look at Java's most used technologies for the web.

• ***Java Servlet API*** - The Java Servlet API enables you to define HTTP-specific classes. A servlet class extends the actual capabilities of servers that host applications that are accessed by way of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend applications hosted by web servers. For example, you might use a servlet to get the text input from an online form and print it back to the screen in an HTML page and format, or you might use a different servlet to write the data to a file or database instead. A servlet runs on the server side - without an application GUI or HTML user interface (UI) of its own.

• ***JavaServer Pages Technology*** - JavaServer Pages (JSP) technology provides a simplified, quick way to create dynamic web content. JSP technology enables rapid development of web-based applications that are server and platform independent. JSP technology lets you add snippets of servlet code directly into a text-based document.

Typically, a JSP page is a text-based document that contains two types of text: Static data, which can be expressed in any text-based format, such as HTML,

Wireless Markup Language (WML), or XML JSP technology elements, which determine how the page constructs dynamic content

• ***Java Message Service API*** **-** Messaging is a method of communication between software components or applications. A messaging system is a peer-to-peer facility. In other words, a messaging client can send messages to and receive messages from any other client. Each client connects to a messaging agent that provides facilities for creating, sending, receiving, and reading messages. By combining Java technology with enterprise messaging, the Java Message Service (JMS) API provides a powerful tool for solving enterprise computing problems.

• ***Java API for XML Processing*** **-** The Java API for XML Processing (JAXP), part of the Java SE platform, supports the processing of XML documents using the Document Object Model (DOM), the Simple API for XML (SAX), and Extensible Stylesheet Language Transformations (XSLT). JAXP enables applications to parse and transform XML documents independent of a particular XML-processing implementation.

• ***Java Persistence API*** - The Java Persistence. API is a Java technology standards-based solution for persistence. Persistence uses an object-relational mapping approach to bridge the gap between an object-oriented model and a relational database. Java technology persistence consists of three areas:

– The Java Persistence API

– The query language

– Object-relational mapping metadata