Leading Languages of non-Enterprise Web Applications

I. Introduction

Web development is an ever-changing space and there are many trends come and go. Due to the rapid growth of technology, different changes had taken place. Today, the leading Languages of non-Enterprise web applications are: JavaScript, PHP, Ruby and Python. In this article, we will analyze the pros and cons of these four programming languages for web development and which programming language and web platform to choose for your upcoming online web application.

II. Body

1. Partners become enemies

In the old days, PHP and JavaScript are partners and ruled the Internet together. The partnership was simple. JavaScript handled the client logic in users' browsers, while PHP managed all the server-side tasks on the servers. It was a happy union that supported many great web applications like Wikipedia and Facebook.

Things changed when some clever guys found out that we could get JavaScript running on the server. Suddenly, there was no need to use PHP on the server side, one language can do all the job. With the introduction of Node.js, JavaScript runs everywhere.

But this is not the end of story, PHP still wins in some areas while JavaScript wins in other areas.[1]

a. **Where PHP wins: Deep Code Base**
   The Internet is filled with PHP code, the most popular plat forms for building web sites like WordPress and Drupal are written in PHP. These platforms are open source and have tons of plugins.
b. **Where JavaScript wins: Modern features**
   Node.js plugins are not only newer but also written using modern architectural approaches. They were built by programmers who have a better understanding of how web applications should be organized.
c. **Where PHP wins: Mixing code with content**

PHP can easily mix code with html without using templates. Everything outside of a pair of opening and closing tags is ignored by the PHP parser which allows PHP files to have mixed content.

d. **Where JavaScript wins: Separate concerns**
   If you have to deal chunks of HTML, then consider having a template system to do the job for you. It's cleaner for new programmers to understand and easier to maintain.
e. **Where PHP wins: Simplicity**
   PHP is a relatively simple language, you usually only need to maintain a few variable and basic functions for processing strings and numbers. It's a thin layer that doesn't do much except move the data to the database and back. Much of the work are done by a modern database. PHP is the right tool for a job that is not supposed to be complex.
f. **Where JavaScript wins: Complexity of closure and more**
   Modern JavaScript have new syntax and supports a few useful features like closures. You can do functional programming and pass functions like objects. There is more freedom to do things.
g. **Where PHP wins: Speed of coding**
   Writing PHP for web apps feels faster, there is no compilers nor deployment. You can start your project more quickly.
h. **Where JavaScript wins: Speed of execution**
   Writing JavaScript code is a bit harder, but when it's done, your node.js code can fly. The callback mechanism saves you from juggling the threads. The core is well-built and runs very fast.

2. Other players in web development

Neither Ruby nor Python are in any way restricted to web development. Ruby is a general purpose programming language and Python is much bigger outside web development.

So when we are talking about Ruby and Python for web developments, we are actually talking about frameworks rails and Django.

Ruby on Rails (RoR) is a web framework written in Ruby and is frequently credited with making Ruby "famous". Rails puts strong emphasis on convention-over-configuration and testing. Rails CoC means almost no config files, a predefined directory structure and following naming conventions. There's plenty of magic everywhere: automatic imports, automatically passing controller instance variables to the view, a bunch of things such as template names are inferred automatically and much more. This means a developer only needs to specify unconventional aspects of the application, resulting in cleaner and shorter code. [2]

The most cited argument against Ruby on Rails is that It's "slow". We would agree certainly when compared to the runtime speed of NodeJS or GoLang. Boot speed is another issue, when developing a Rails app, it can be frustrating waiting for the application to boot. Depending on the number of gem dependencies and files, it can take a significant amount of time to start. Also, it can be hard to find good documentation. Particularly for the less popular gems and for libraries which make heavy use of mixins. You'll often end up finding the test suite acts as documentation and you'll rely on this to understand behavior.[3]

Django is a web framework written in Python and was named after the guitarrist Django Reinhardt. Django's motivation lies in the "intensive deadlines of a newsroom and the stringent requirements of the experienced Web developers who wrote it". Django follows explicit is better than implicit (a core Python principle), resulting in code that is very readable even for people that are not familiar with the framework. A project in Django is organized around apps. Each app has its own models, controllers, views and tests and feels like a small project. Django projects are basically a collection of apps, with each app being responsible for a particular subsystem.

Django had its public release a year or two later than Rails and is less popular than Rails, less popular means less libraries and less community supports.[4] Also, someone thinks Django goes too far on emphasizing explicit is better than implicit. They complains that they can't even call a Python function from the template.[5]

III. Conclusion

All of the four languages are great for web developments and can do a great job when used with the right purpose. You can not really go wrong by picking either one of them. My suggestion is always to try all and figure out which one you are most comfortable with.

IV. References

[1] http://www.infoworld.com/article/2866712/php/php-vs-node-js-an-epic-battle-for-developer-mind-share.html

[2] https://bernardopires.com/2014/03/rails-vs-django-an-in-depth-technical-comparison/

[3] https://www.madetech.com/blog/pros-and-cons-of-ruby-on-rails
[4] https://www.quora.com/Why-is-Django-not-as-popular-as-Ruby-on-Rails
[5]http://nando.oui.com.br/2014/04/04/why_i_sort_of_dislike_django.html