Ruby as we know was created by Yukihiro Matsumoto in 1995 with the goal of making a programming language that was more natural to understand. David Heinemeier Hansson created Ruby on Rails in 2003 with the three basic principles in mind: the Ruby Programming Language, Model-View-Controller Architecture, and Programmer Happiness.

Since Rails extends the Ruby language, it is a dynamically-typed, strongly-typed, compiled, object-oriented programming language. It is also considered a multi-paradigm language, specifically being Object-Oriented, Procedural, and Functional. Ruby has methods and classes and uses objects and attributes so it’s Object-Oriented at its core but it also implements concepts from other paradigm. It is considered Procedural because it can create and call methods outside of classes. It is also considered Functional because it can use lamdas to pass a method into another method.

The goal of Ruby on Rails is to simply Web Development. It combines Ruby with HTML, CSS, and JavaScript to create a web application that runs on a web server. The structure of this language is summed up a concept known as MVC, which is Model-View-Controller. The model layer is where classes are defined for the data the application will use and store. The view layer is mainly to create the HTML that will be displayed on the browser. Lastly, the controller ties everything together. It passes data from the model to the view and then the view returns the HTML to the controller and the controller sends it the browser.

The biggest asset for Ruby and Rails is the RubyGems library which carries over 100,000 gems that can help you produce or implement almost any feature you might desire. For example, one of the guides I used to download Rails suggested the use of a gem called Bulma. This gem helped to simplify the HTML and CSS I needed to design my website. So I didn’t have spend as much time looking up how to do basic things in HTML and CSS.

With all the simplification benefits of using Rails, there are some disadvantages depending on your project goals. One complaint is Ruby is not as efficient for websites that require a lot of simultaneous activity. A second complaint from some programmers is that many of the complex operations that are occurring are hidden behind simple directives. So, if you are wanting to customize your application a little deeper, it may be difficult to know where certain behaviors are coming from.

The biggest lesson I learned while using Rails is that there is no perfect way to develop with this language. It’s best to just jimp right because every tutorial you’ll find will have different ways that the same task is implemented. Additionally, the more you continue to work with the language, the more it begins to make sense. At first, I didn’t quite grasp the MVC concept but as I began to flesh out the project, how each element works together began to make more sense.