

Choosing

# Ruby On Rails

For Your Next Web Development Project



The all-in-one guide for CEOs and  
Product Owners

**IDEA**MOTIVE

*Rails has done more for startups than a whole boatload of Venture Capitalists. Rails has had an incredible impact on the startup ecosystem.*

*Eric Ries. “Lean Startup” author*

Welcome to the Ruby on Rails all-in-one guide from Ideamotive! After our [React Native Development Guide](#), this is another massive piece of content where we take a deep dive into our favourite development frameworks. No more Googling. No more web digging. We give you all the info about Ruby on Rails in one place.

After reading this guide you will know:

- What exactly is Ruby on Rails?
- How does it stack up against other web development frameworks?
- When to use and when not to use Ruby on Rails?
- How can you benefit from it?

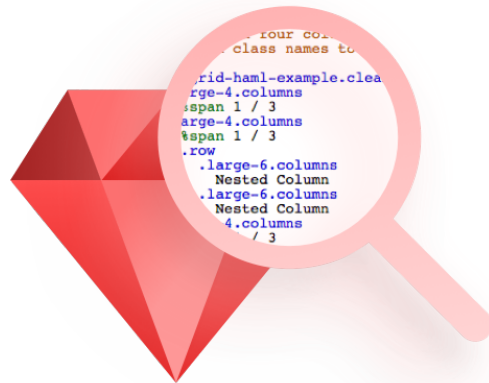
**Enough introduction! The story of Rails begins back in the 1990’s... or is it 2005? Let’s get right into it!**

<p>Why did we create this page?</p>	<p>Who should read our guide?</p>
<p>Blog posts, YouTube tutorials, development forums, workshops, code repositories... There are so many resources to go through when you are looking for comprehensive and reliable information about Ruby on Rails.</p> <p>Who has the time to browse them all? This is your single reliable resource that gathers all Ruby on Rails knowledge in one place. No more searching!</p> <p>We want everybody, regardless of background, to be able to get value from this guide. Which is why we assume no previous knowledge of terminology, and we explain things in detail.</p>	<p>Are you looking for the right technology for your next web development project?</p> <p>Are you a developer who wants to learn Ruby on Rails?</p> <p>If you're a startup, product owner, C-level professional or a marketer, this guide will help you get the hang of Ruby on Rails without reaching the bottom of Google.</p> <p>It also aids developers who want to learn to code in Ruby. Finally, it is simply meant for anyone interested in web app development.</p> <p>The only person who won't find much use for this guide would be a high-level master Ruby on Rails developer. If you're not on that level yet, then you'll definitely be able to learn something!</p>

## Tabel Of Content

---

01. [What is Rails, really?](#)
02. [Ruby on Rails compared to popular alternatives](#)
03. [What is Ruby on Rails used for?](#)
04. [Ultimate list of RoR pros and cons](#)
05. [Ruby on Rails web app development process](#)
06. [How does Ruby on Rails impact your revenue?](#)
07. [The correct way to learn and master Ruby on Rails](#)
08. [Finest Ruby on Rails examples](#)
09. [State of Ruby on Rails for 2019 + future possibilities](#)
10. [Summary - is Ruby on Rails the right choice for you?](#)



# 1. What is Rails, really?

Ruby on Rails (aka Rails aka RoR) is a open-source framework for building web applications. It allows programmers to use readymade solutions to common issues, which means they can save time – and money – during the development process.

[🔗 Check out the official release note for Ruby on Rails 1.0 🔗](#)

Similarly to many of the best software projects in the world, it was first built as a kind of side-project. It was created by [David Heinemeier Hansson](#), founder and CTO of Basecamp. Basecamp was the first true Rails web application.

## What is a web application, and how is it different from a website?

*"[...]a website is defined by its content, while a web application is defined by its interaction with the user. That is, a website can plausibly consist of a static content repository that's dealt out to all visitors, while a web application depends on interaction and requires programmatic user input and data processing.*

*For example, a news site would be a "website", but a spreadsheet or a collaborative calendar would be web "applications". The news site shows essentially the same information to all visitors, while the calendar processes individual data.*

*Practically, most websites with quickly changing content will also rely on a sophisticated programmatic (and/or database) backend, but at least in principle they're only defined by their output. The web application on the other hand is essentially a program that runs remotely, and it depends fundamentally on a processing and a data storage backend."*

Source: [StackOverflow](#)

Rails was extracted from Basecamp in 2003. It took two more years of development led by David and supported by a global community of like-minded developers, and Rails 1.0 was finally released to the public in 2005.

It quickly started growing in popularity as “the open-source web framework that's optimized for programmer happiness and beautiful code”.



**Remember: the role of frameworks is to give software developers readymade elements (code libraries) to save time when building new software.**

This particular framework is a collection of code libraries written in the Ruby programming language. These libraries are readymade, must-have functions for web applications – like forms, buttons, or menus.

You know how, when you write a report or paper, sometimes you just copy paragraphs from other papers and rewrite them in your own words? This is kind of similar to what code libraries are for programmers.

There are several frameworks, for different purposes, for virtually every programming language out there. But Rails is a very special framework, and it makes for a great web application development environment.

It's different than most frameworks. Why? This question is best answered by the creators of the [official guide to Ruby on Rails](#):

*“[Rails] is designed to make programming web applications easier by making assumptions about what every developer needs to get started.[...] Rails is opinionated software.”*

Like most frameworks, Rails provides readymade elements (essentially templates) for building a specific type of software – in this case, web applications.

But, because of its unique design, it also forces programmers to use explicit development practices – “The Rails Way”.

## Two fundamental rules of “The Rails Way”

### **Don't Repeat Yourself:**

"Every piece of knowledge must have a single, unambiguous, authoritative representation within a system." In other words, applications are less buggy when the same information isn't repeated over and over in the codebase.

### **Convention Over Configuration:**

in commercial product development, an MVP working application usually needs to be built as fast as possible. Rails makes it easier by providing default settings that drastically limit the amount of initial configuration.

This is surely inspired by the philosophies that led to the creation of the underlying programming language, Ruby. What is the difference between Ruby and Ruby on Rails?



## Ruby vs Ruby on Rails

**Ruby** is a programming language designed around simplicity and productivity. It can be used to develop almost any type of software, [including](#):

- Games (desktop and mobile)
- Applications (desktop, mobile, web)
- APIs
- Command line tools
- Embedded applications
- Java applications
- Client-side applications
- Test tools for non-Ruby apps

**Ruby on Rails** (RoR, Rails) is a web application framework running on the Ruby programming language.

It provides tools for developers to build web applications faster, and have more fun doing it.

Building in Rails requires developers to learn and adapt to “The Rails Way,” which is a strict set of practices that Rails creators believe to be the best practices for building web applications.



## Interesting facts about Ruby

- Created by Yukihiro “Matz” Matsumoto
- First released to the public in 1995
- Not at all popular until the release of Rails in 2005
- Matz often said that he wanted to “make Ruby natural, not simple”
- Designed to be “[...]more powerful than Perl, and more object-oriented than Python”

Very similarly to Rails, the most important thing about Ruby are the strong convictions of its creator, which influenced every aspect of this programming language. In many ways, Ruby was also designed to maximize programmer happiness.

Another thing that connects Ruby and Rails is that **if you're not able to adapt to the fundamental coding philosophies of these technologies, you won't have much fun using them at all.**

What is special about Ruby?		
<p><b>Everything is an object</b>  <i>Unlike most programming languages, in Ruby, you can give properties and actions to every bit of information, even numbers and symbols.</i></p>	<p><b>Ruby is flexible</b>  <i>It allows developers to freely change / remove / redefine essential parts of the language</i></p>	<p><b>Blocks</b>  <i>Developers can attach blocks to their code (a closure that describes exactly how a method should work)</i></p>
<p><b>Exception handling</b>  <i>Similar to Java or Python, Ruby has features for exception handling, which makes it easy to handle errors</i></p>	<p><b>Independent threading</b>  <i>Multithreading on all platforms on which Ruby runs, regardless of whether the OS supports it</i></p>	



**You can also check out**

---

[First ever video demo of Ruby on Rails](#)

[Official Ruby on Rails website](#)

[The Complete, Original Rails Doctrine](#)

[Rails 1.0. Release Note](#)

[Must-know basics about Ruby](#)



## 2. Ruby on Rails compared to popular alternatives

When you look at the results from the [2018 StackOverflow Developer Survey](#), Ruby is placed somewhere in the middle between the most obscure, and the most commonly used technologies.

**10.1% out of 78,334 developers that responded to the survey use Ruby.**

In the framework category, Ruby on Rails didn't make the cut with the rest of the most popular frameworks. Which isn't surprising, as RoR is not for everyone. But those who know how to use it can do amazing things with it.

Which shows when you look at the need for Ruby on Rails specialists, instead of its popularity among developers. **You'll see that the [demand for Ruby on Rails is big](#).**

**RoR is a mature technology**, and in the software industry, things can be very trend-driven. Rails used to be “the thing” if you wanted to build web applications a few years ago.

But while other technologies may have taken its strong position in popularity rankings, Ruby on Rails remains just as powerful — and in many cases proves itself even more powerful — as the currently popular technologies.

Additionally, RoR has a huge community of developers and supporters who have been working on this technology for years.

However, it's hard to say where exactly this technology stands until we've compared it to popular alternatives. So let's do that!

---

## Ruby on Rails vs PHP web development environment

	Ruby on Rails	PHP
<b>Performance</b>	Can be slow if the app is poorly designed. Great for CPU-intensive tasks.	Pure PHP is faster until you add a framework like Laravel, when the performance drops below RoR levels.
<b>Maintenance</b>	Easy to maintain apps thanks to community and constant improvement.	It can be easy to maintain, as long as a popular web framework with community support is used. Otherwise apps are much harder to maintain.
<b>Scalability</b>	Uses a lot of resources — there's a threshold when it becomes too hard to scale.	Uses less resources, a bit easier to scale.
<b>Support</b>	Supportive global community that might be smaller than the amount of PHP developers, but you definitely won't be left without an answer if you have an issue. Smaller amount of editors and tools available.	Probably the largest amount of support resources out of all programming languages — but a lot of them are outdated. There are a lot of third-party editors and tools for PHP.
<b>Cost</b>	Skills from other languages (like PHP) transfer easily.	Easier to learn. Easier to find developers. Also quite expensive to host apps (in the case of modern apps with powerful functionality).
<b>Development Speed</b>	Faster to build a fully functioning web app. Offers "scaffolding" — it can generate code based on specified parameters.	A bit slower to build a fully functioning web app, especially if you don't use a popular web framework.



## Don't forget!

- RoR is a framework for the programming language Ruby, while PHP is a web-focused programming language with multiple frameworks that might be compared to RoR - like Zend, Codeigniter, Laravel, Symfony2, CakePHP and more.
- As you may remember from the first part of this guide, Rails is opinionated software. Developers choose it because it's easy to set-up and easy to use. When it comes to PHP, many developers prefer to build their own frameworks, rather than use existing ones.



You can also check out

---

[Comparison of Ruby on Rails and PHP.](#)

### Ruby on Rails vs Django (Python framework)

	Ruby on Rails	Django
<b>Performance</b>	Great for commercial startups and modern web apps. Great for CPU-intensive tasks.	Great for academic / scientific uses due to the design of Python.
<b>Maintenance</b>	Easy to maintain.	Some say that updating Django is even easier than RoR.
<b>Scalability</b>	Harder to scale.	Easier to scale.
<b>Support</b>	Large web-development oriented community.	Smaller web-development oriented community.
<b>Cost</b>	A bit harder to learn. Less popular, might be harder / more expensive to find developers. Hosting costs similar to Django.	Easier to learn. Python is more popular among developers. Hosting costs similar to RoR.
<b>Development Speed</b>	Requires much less configuration, speeding up development times.	Greater customization comes with more to configure, which slows down development times.





## Don't forget!

- The biggest difference between Django and RoR is that Django allows for more customization, while Rails makes configuration choices for you to speed up development.
- Python, the language used in Django, is a language very commonly used for academic purposes, whereas the Ruby community — thanks to Rails — is more business-oriented.



**You can also check out**

---

[Detailed comparison of Ruby on Rails and Django](#)

Ruby on Rails vs Node.js + Express (JavaScript)		
	Ruby on Rails	Node.js + Express
<b>Performance</b>	Mainly good for commercial startups and modern web apps. Great for CPU-intensive tasks.	Powerful underlying libraries make for very fast web apps — but it's not suitable for processor-intensive tasks. Great for I/O intensive tasks.
<b>Maintenance</b>	Easy to maintain. Easy database migrations — it's portable across all platforms.	Still new, so there are glitches, and instabilities. But it's still relatively easy to monitor and support.
<b>Scalability</b>	Easy to scale until a threshold — gets much harder at a certain level.	Incomparable scalability — chosen by companies with insane traffic (like Netflix).
<b>Support</b>	Large web-development oriented community.	The community still has to grow, but there are good resources out there.
<b>Cost</b>	Requires a lot of computing resources to run efficiently. Requires highly skilled programmers.	Allows for using the same language (Javascript) on client- and server-side, making it easier to hire devs. Reduces the need for computing resources.
<b>Development Speed</b>	Requires minimum configuration. Opinionated, assumes best practices for building apps. Hands down the fastest way to build an app.	Non-opinionated, requires additional code and configuration before it can match RoR's out-of-the-box features.



## Don't forget!

- Node.js isn't a framework per se — it is an open-source server environment that makes it possible to run JavaScript as the server side of your application.
- The framework in this case is Express, which provides multiple features for web and mobile applications.
- Node.js, as well as Ruby on Rails, can be used with front-end frameworks like React — sometimes it's easier with Node.js, since you only need to know one programming language. But in many cases, learning the framework can be harder than picking up a new language!



### You can also check out

---

[Comparison between RoR and Node.js](#)

[Our own comparison between RoR and Node.js + Express](#)



## 3. What is Ruby on Rails used for?

Rails is great - but it is not a “golden solution” for every type of web project (although the spectrum is pretty wide).

The official “[Getting Started](#)” RoR guide shows you how to build a simple blog as your first application, but it can do much, much more, from custom websites to extremely complicated platforms like GitHub, Shopify, Airbnb, SoundCloud, Zendesk, and Square, to name just a few.

Wondering if your Ruby on Rails is suitable for your project? Read the section below!

## What can I do with Ruby on Rails?

# Startups and SMBs

*Ruby on Rails makes it fast to bootstrap your product and get an MVP running for a small startup.*

*Matthew S, app developer, taken from [G2Crowd RoR reviews](#)*

Ruby on Rails makes it easy and fun to build apps. It allows for open source web application development. And when programmers are happy, they tend to finish their work faster. But that's not the main reason why RoR is great for startups and SMBs.

The main reason is this: thanks to the “Convention over Configuration” principle, Rails is truly one of the top technologies for the types of projects that startups and SMBs often do:

- Prototyping
- Building an MVP
- Projects with short-time-to-market
- Rapid Application Development



## Don't forget!

*Rapid Application Development is the opposite of Waterfall methodologies. It's a project management method designed around fast prototyping, quick iterations, and adapting software to user needs in as little time as possible.*

Another super helpful thing is the unparalleled number of open-source code libraries and projects (known as gems) in the Ruby community. And developers are constantly building new ones!







### What are gems?

In Ruby, a gem is a library that contains a specific piece of functionality as well as any files or assets related to that functionality. Gems can be found for all sorts of common functionality in an application — things like handling money and currency, integrating with credit card processing, tools to make Ruby coding even easier, and more. You could write these things yourself but one big benefit to using a gem instead is that it saves time.

RubyGems is very similar to apt-get, portage, and yum in functionality. [\[source\]](#)

At the time of writing there are 148,845 gems available to download from the popular site [RubyGems](#), which has 128,228 users.

This means that whatever you need in your application, there's probably a gem for it. And if there's a gem for it, then you can finish your app much faster than if you had to build every feature from the ground up.

Popular startups that first built their product with Rails (some have moved on to other technologies)		
		
		

# E-commerce

*“It’s crazy that people are suggesting Shopify has been successful despite Rails. Shopify has been successful BECAUSE OF Rails.”*

*Tobias Lutke, CEO of Shopify. [Source](#)*

What are the most important aspects of an e-commerce web application? To us it’s performance, user experience, and security.



Ruby on Rails has been around for more than 10 years. Over time, it has been perfected by a global community of developers, many of them working for the biggest companies in the tech game. There's no doubt about it — Ruby on Rails is a mature technology.

For e-commerce project managers and developers, this translates into stability — and with stability comes good performance and security. Rails is based on a strong foundation of best software development practices, which means that it's hard to mess up a Rails app.

This is insanely helpful in the context of e-commerce websites. Remember that every second an e-commerce site is down, they're losing customers. So ultimately, mature and stable technology like RoR is the way to go if you don't want to lose money on your e-commerce project.







By default, RoR apps are protected against popular attacks like SQL injection, cross-site scripting and cross-site request forgery. There are simple, but powerful security mechanisms for encryption and cookie signing, and it comes with the basic Rails package.

Rails reduces the need for configuration, but this aspect is really most visible when you're first building your application. In later development stages, when you're working on a living product, Rails gives developers a lot of flexibility in the parts where it matters.

What about user experience? Gems come to the rescue, again. There are ready-made gems for integrating your application with other crucial e-commerce elements:

- Infrastructure providers
- Payment gates
- Email marketing solutions
- Inbound marketing software

This makes it much easier for developers to build websites with all the functionality that customers expect from modern e-commerce sites.

Popular e-commerce web apps built with Rails		
		
		

# Software-as-a-Service (SaaS)

SaaS products are the ultimate web applications. They need to be scalable, fast, and secure. Software-as-a-Service probably beats all other application categories presented here in terms of the amount of connections and operations that need to be made in order to make the end-user happy.

This type of software relies heavily on building safe and fast APIs to allow your server to communicate, and exchange data with various other services. Luckily, **Rails is deeply embedded with REST architecture for building reliable APIs.**








## Don't forget!

*REST is short for 'Representational State Transfer.' It's a style of software architecture that defines certain rules for creating web services. Applications that follow this architecture style are referred to as RESTful.*

Additionally, Rails provides powerful testing and debugging tools, which enable developers to make sure that their applications are working as designed. This, along with all the other advantages, makes Rails a great technology for building **Software-as-a-Service** products.

## Popular SaaS websites built with Rails

 Basecamp	zendesk	 USERVOICE
 slideshare	 INDIEGOGO	 HEROKU

## Don't use Ruby on Rails when...

- **Your product is set to scale for thousands of requests per second from day one** - there is a threshold where scaling RoR products becomes too costly to be reasonable. Luckily, that threshold moves farther away with every new version of Rails, but you'll be better off if you choose a different technology for projects that are built to scale. We are not saying it is impossible to build such an app though — a great example would be [Cookpad](#), built in RoR, which is able to deal with 15k requests per second! [\[source\]](#)
- **You're building an I/O-intensive application** - these types of applications written in Rails have been proven to have poor performance. Ruby on Rails is much better for CPU-intensive applications.
- **You (or your developers) want to build everything yourself** - the Rails community relies on collaboration and open-source gems. While RoR gives developers a lot of flexibility, if you or your team prefer to build everything from the ground up, you'll probably be better off using a different framework, pure Ruby, or a different programming language altogether.

- **You just need a company website** - it would be overkill to use Rails for building a company website that's just an online business card. It's possible, but... just don't do it. Use Wix or just create a static website with HTML and CSS.
- **You're building a blog** - even though the first project in the official Rails learning guide is a simple blog, RoR isn't great for blog-only projects. For setting up your blog, you can check out [Jekyll](#) - great blog site generator written in... Ruby :)



**You can also check out**

[Should you choose Ruby on Rails for your SaaS product Tech Stack?](#)

[Is Ruby on Rails a good choice for your website?](#)

[8 Reasons Why Ruby On Rails Is A Great Fit For Your Next E-commerce Project](#)



## 4. Ultimate list of RoR pros and cons

Time for the good stuff. We've scoured the web in search of strong, factual opinions from developers and project managers who actually used Rails.

Some of them loved it, others not so much! Here's a round-up of what the global software developer community thinks about Ruby on Rails.

## Why do people love Ruby on Rails?

1. The creators and main contributors of Ruby on Rails take security very seriously. They have a whole process, [described on their website in detail](#), for reporting, fixing, and rolling out security patches. Crucial security features come with the basic Rails package, and developers using the latest RoR version are updated about new security issues. [\[source\]](#)
2. Without Rails, Ruby might not have become a popular programming language for web applications. Which is a great thing because Ruby is driven by a unique philosophy, and it makes programmers happy. [\[source\]](#)
3. It's easy to learn - especially if you have experience working with programming languages like Python or PHP. [\[source\]](#)
4. Building an application from the ground up is much faster compared to using other popular technologies.
5. It's a pragmatic language, and pragmatism is visible in the whole culture of Ruby on Rails. Compared to other languages and frameworks, there is very little artificial complexity in RoR. [\[source\]](#)
6. Unlike many other technologies that have been around for 10+ years, Rails is still being updated. Rails is a stable technology that is evolving without any major breaking changes. [\[source\]](#)
7. Rails makes it easy for programmers to switch between projects because the philosophy and software development principles are always the same. [\[source\]](#)



8. The community around Ruby on Rails is a wonderful thing in and of itself. Development of RoR is still managed by its original creator. The amount of programmers using RoR keeps growing. New gems are constantly being added to public repositories. This is very different from other popular frameworks, and it's a huge selling point for this technology. [\[source\]](#)
9. There are easy ways to integrate Rails with popular front-end frameworks, as well as other third-party technologies for web development.
10. Ruby on Rails is free. There are no licensing fees. The base package, all the support resources, as well as endless open-source code libraries — you can download it all and use it without paying a cent.
11. RoR is flexible. As projects grow in complexity, sometimes developers realise that they need to make fundamental changes to the codebase, like changing the database engine. In Rails, things like this are very easy to do, and don't require any changes in the application code. [\[source\]](#)
12. Rails makes it easy to build automated tests, which translates to less time spent debugging in late stages of development. [\[source\]](#)
13. It is a mature technology. Even though the heyday of RoR is over, the community hasn't stopped growing, and Rails is still being improved. Very few technologies have had so much luck for so many years, which goes to show that RoR is simply a great tool. [\[source\]](#)
14. Rails is a tool that works well in various scenarios. It's good for large, modern, public websites, as well as complicated in-house enterprise applications using insane business rules and logic. [\[source\]](#)
15. RoR is readable, easy to maintain, and productive. It enables developers to work on high-level software issues, instead of micromanaging the codebase of their apps. [\[source\]](#)
16. Contained in Ruby on Rails is a whole suite of solutions for various common problems that developers face when building applications. There are a lot of ready-made, tested functionalities that can be

implemented as plugins. Together with gems, this makes development much faster, and much easier.

17. The Ruby on Rails documentation is clear, concise, and makes it easy to find solutions for every problem you might encounter. [\[source\]](#)

---

## Why do people dislike Ruby on Rails?

1. For some developers, Rails is too slow — it may be too heavy in certain uses, but compared to other solutions, it really is one of the fastest technologies out there. The runtime speed is lower compared to certain other technologies. This problem is most visible if a product needs to be substantially scaled in a short period of time. Which doesn't mean that you can't scale Ruby — GitHub is a Ruby app, and it's enormous, and still works amazingly well. [\[source\]](#)
2. Not all hosting companies support Rails. However, this is constantly changing, as most major providers have started hosting Rails apps in recent years.
3. Ruby on Rails enforces programming conventions, and developers need to adapt to them. Not everybody likes this type of approach to development, as many developers have their own set of best practices for building software. [\[source\]](#)
4. If developers aren't careful when designing their apps, Rails apps can choke on multi-threading, causing poor performance and increased infrastructure costs. [\[source\]](#)
5. It's easier to find developers with experience in Java, PHP, or Python.

6. There are many gems, developers might spend a lot of time verifying the actual value of a gem, and whether it fits their project or not. But there is also a group of reliable, tested gems that everybody likes to use. [\[source\]](#)



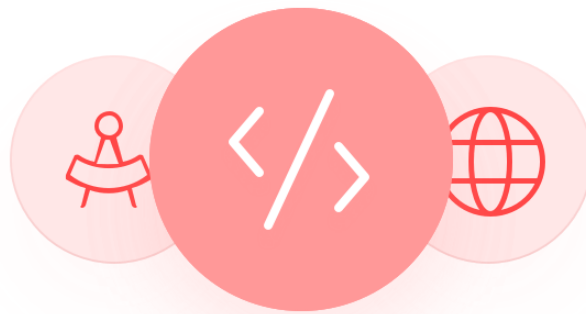
### You can also check out

---

[Thread on Reddit about pros and cons of Ruby on Rails development](#)

[What are the advantages and disadvantages of Ruby? \[Quora Thread\]](#)

[Why is Ruby on Rails A Pitch Perfect Back End Technology?](#)



## 5. Ruby on Rails web app development process

Every developer, and by extension, every product team, has their own development process. Do you know how to build a web app? How to develop a web application with Ruby web development?

RoR is a great web app development framework. When you use a technology like Ruby on Rails, which forces programmers to use a set of strict conventions, there are still many aspects of the process that you can customize to your own liking. Ruby on Rails web development is a pleasure for programmers.

Because of that, we don't want to get too deep into the process of building web applications with Ruby on Rails.

Instead, without dwelling on technical details and nuances, we want to show you the revolutionary project management methods that go along with the philosophy and conventions of Rails.

## Rapid Application Development

*Project managers and programmers are always looking for a way to build products faster. RAD (Rapid Application Development) is one of the many methods that helps them do that.*

*It was developed in 1991 by an IT consultant from the UK named James Martin.*



## The basic idea behind RAD is less planning, more prototyping.

The global software industry was already moving at a breakneck pace back then. Planning was problematic because of the speed with which the market was changing.

It turned out that long-term planning and execution in the traditional Waterfall style was far from the best way to build commercially successful software products.

RAD became popular because it enabled developers and managers to be flexible, and adapt to the market.

## There are four basic stages to Rapid Application Development:

### 1. Initial Plan

The first stage is mainly about clear communication between the client, the development team, and all managers involved. It should also include a good deal of research and strategic analysis.

The goal of this stage is to reach definitive product requirements that are understood and approved by everybody involved in development.

### 2. UX Design

This step involves taking the finalized requirements from stage 1, and creating simple prototypes of necessary functionalities.

It's not full-on coding yet. The main goal of this stage is to get client (user) approval for basic User Experience models that are based on product requirements and – if necessary – change or redefine the requirements that can't be met.

### 3. Development

This is when the fun begins. When the UX models of the app are approved, it's time to build the product.

Traditionally, programmers would take years to build everything according to the initial plan. The problem is that when they were finished, the market usually didn't need to product anymore, or the client realised that they wanted a completely different product.

In order to avoid this, RAD forces the product development team to constantly communicate with the client (or future app users) in order to collect feedback,

notify them about development progress, and catch destructive design issues early on.

## 4. Implementation

In the last stage, the product is tested and implemented. This is also time for creating necessary tutorials to make it easy for users to learn the new application.

### Agile Development

In 2001, 10 years after the creation of RAD, a new, radical idea was born in a ski-lodge in Utah (or so the legends say).

It was during a meeting of 14 prominent figures from the tech industry, who were noticing that developers far and wide have started to switch to a new, more flexible, more iterative development process.

**The result of this meeting was the [Agile Manifesto](#) – a short document that is a collection of general philosophies and conventions for a new, better way to build software.**



The manifesto is short enough that we can quote the whole thing:

*“We are uncovering better ways of developing software by doing it and helping others do it.*

*Through this work we have come to value:*

*Individuals and interactions over processes and tools*

*Working software over comprehensive documentation*

*Customer collaboration over contract negotiation*

*Responding to change over following a plan*

*That is, while there is value in the items on the right, we value the items on the left more.”*

There are also [12 detailed principles](#) built upon these four core values.

This manifesto enclosed the key fundamentals of new project management methodologies that, similarly to RAD, relied on communication and enabled developers to change products along with changing user needs.

If you think Agile is just Scrum, think again! There are a lot of **Agile** methods, including:

- [FDD](#)
- [Extreme Programming](#)
- [DSDM](#)
- [Crystal](#)
- [Lean](#)
- [Scrum](#)
- [Kanban](#)

They're all different in certain aspects, but they all share the key ingredients of Agile project development. What are those ingredients?

**1. Product vision** - defines goals for the end-product

**2. Product roadmap** - a strategic outline of what needs to be done to achieve goals

**3. Product backlog** - a list of specific tasks that need to be completed to finish the product

**4. Release plan** - dates and deadlines for rolling out the application

**5. Sprint backlog** - every task, linked with necessary additional information, that is supposed to be completed during one sprint

**6. Increment** - new features that are finished and presented to the client (user) at the end of each sprint

Generally speaking, Agile is about **focusing on great communication** (which translates to many strategic meetings / calls with a pre-planned agenda), and

the idea that development should be divided into small, for instance weekly, **timeboxes** (sprints).



**You can also check out**

---

[Agile Methodology detailed overview](#)

[Ideal Web Application development process](#)

[Rapid Application Development in practice](#)



## 6. Economy of Ruby on Rails

Do you know the easiest way to annoy a professional software developer? Ask them for an application cost estimate! (queue the laugh track)

Some of you will know exactly why, but if you're not sure why that joke works — check out this enlightening short essay by Michael Wolfe, a serial entrepreneur and founder of over five commercially successful startups:



[Michael Wolfe's answer to "Why are software development task estimations regularly off by a factor of 2-3?"](#)

In all seriousness, estimating the costs of building a web application is trickier the less information you have. It's hard to say exactly how many work hours and how much money is necessary if you don't know anything about the project except the elevator pitch.

But you can get a general idea of the potential cost if you look at **average production** and **maintenance costs** for common types of projects.

## Average production and maintenance costs for Rails web apps

It's hard to find any solid data, or universal framework for estimating development costs. We have to make do with what we were able to find.

How much does Ruby development cost? For a general idea, here's a very simple reference guide of how much certain projects cost on average. Here it is:

- **\$5,000 to \$15,000** - you'll be able to build a simple website, a single landing page with basic interactive elements, or add a feature to your existing website.
- **\$15,000 to \$60,000** - you'll be able to build a large interactive website, e-commerce site that could generate \$1,000,000 in revenue, add a large feature to your existing website, or build a **very basic MVP of your web / mobile application**.
- **\$60,000 to \$200,000** - you'll be able to build a top-class website with a custom CMS, an e-commerce website that could generate \$10,000,000 in revenue, a web application that automates enterprise processes, or a **more advanced MVP web / mobile application**.
- **\$200,000 to \$500,000** - you'll be able to build software add-ons for legacy enterprise infrastructures, create the foundation of a custom IT infrastructure for a

medium-sized business, or an **MVP web / mobile application that includes all required features**

- **\$500,000 and more** - you'll be able to build complicated web applications that require a lot of integration, build and integrate a complete and custom software solution for your company, or a **full web / mobile application**

## Does Ruby on Rails give you any cost advantages during development?

Yes it does. Ruby on Rails gives certain points of leverage when it comes to managing your production budget.

**Time efficiency** - Ruby on Rails contains many ready-made plugins and modules, which allow developers not to waste time on writing the code from scratch. On average, Ruby on Rails developers build applications [30–40 percent faster](#) than teams using other programming languages and frameworks. Every hour less that developers work means more money in your budget.

**No licencing costs** - Since Ruby on Rails is an open source, distributed under the MIT license, you will not be paying for usage of the framework.

**Community support** - RoR has a huge community of developers who continue to create and publish new **gems**. These are also open sourced and free to use. Active community also means that if you have a problem with your code, most likely there is someone on Reddit/Slack/Discord ready to help you out.

**Examples of ready-to-implement plugins in RoR - Ideamotive**

## Team selection!

Database management  
Input data validation  
Internationalization  
View generation (templating engine)  
Application routing  
Email sending  
File uploading  
Application security  
Automatic tests  
Caching (so app can even load faster!)  
Working with dynamic browser content (modern javascript support)  
Dynamic client updates (websockets)  
WYSIWYG text editors  
Basic crash reporting (e.g. to email)

## So much for production, but how much does it cost to maintain RoR applications?

The basic answer is “it depends.” Depends on how big your application is, how complex, how many people are working on it, how many users it has, and endless other factors.

But there are certain basic costs that will appear in virtually all Rails web app projects. These include:

- **Continuous integration services** - an invaluable tool for adding new changes and features to existing applications. CI services mainly provide automated testing which ensures that new code doesn't collide with the existing code base. The average cost of these services ranges from around \$300 to \$600 per year.

- **Hosting and asset storage** - not every hosting provider used to allow Ruby on Rails applications in the past, but nowadays, most major providers have no problems with it. You can read more about the most popular RoR hosting providers on our blog.
- **Development** - to keep your app working, you will need to keep paying developers throughout the whole lifecycle of your product. Bugs will always happen, and somebody needs to fix them.
- **Updates** - when a new version of Rails is released, updating your app will take some time but it is a must. If you continue using outdated Rails in your app, then it's no longer secure and vulnerable to attacks. It is good to keep somebody on a team who will be taking care of it or keep track on RoR updates so you don't miss any.
- **Code repository** - modern web applications rely on cloud-stored code repositories that contain all executable and configuration files. The biggest player in the code repo game is [Github](#) (which is built on Ruby on Rails!), where costs range from \$9 to \$21 per user / per month. The runner-up in the market, [GitLab](#) is also built on Ruby!
- **Third party integrations** - these costs appear if you need to connect your application with a ready-made premium service for processing payments / email outreach / security, or anything else you need, but don't have the budget to develop a custom solution. There are gems for fast and easy integration with plenty of services, like Stripe for instance.
- **Analytics** - if you want to collect some basic data, you are good to go with free to use Google Analytics. But for more sophisticated data collection, like heatmaps or recording users' sessions, you may pay extra (the most popular solution, [Hotjar](#), is free for small projects, but if you want to collect data on a scale, you'll have to go with paid version).
- **Crash monitoring** - especially important when it comes to external apps. A third-party solution which allows to track the critical errors so you can



react quicker when they appear. One great example is [Rollbar](#), which allows you to set up the email alerts.

## So is it cheaper to build and maintain applications in Ruby on Rails compared to other technologies?

We do not say that Ruby on Rails will get you more money.

**It won't.**

You still need to spend, covering the cost of your team (developers, project managers, designers, and so on), assets and maintaining the app after it's done.

But Ruby on Rails will certainly help you find cost optimizing opportunities when it comes to building your next web product. The great amount on ready-to-use gems and premade solutions is one of the most important reasons why you should go with Ruby. Of course, the scale of the gain heavily depends on the complexity of the project and the amount of innovative solutions you would like to implement in the project. For some of them, you will need to build on your own.

But with Ruby on Rails, you will never have to reinvent the wheel.

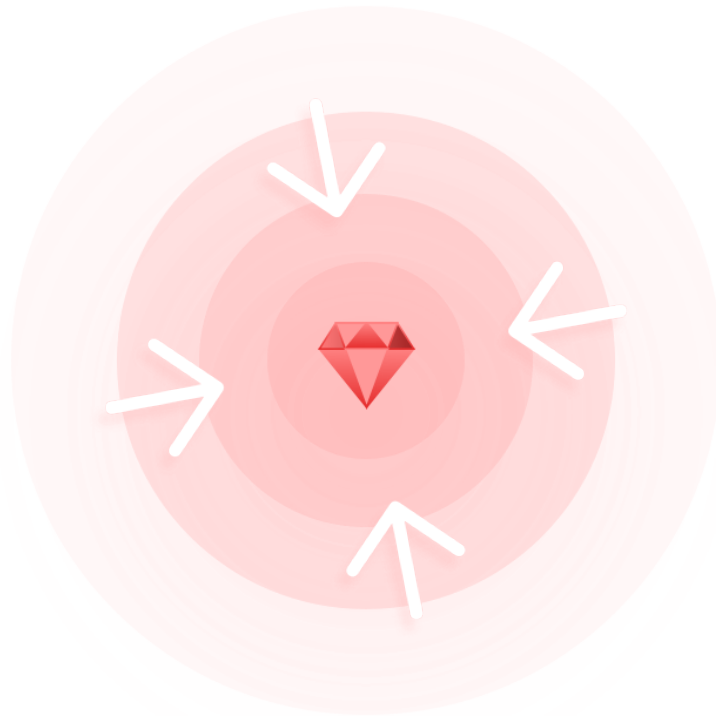
And the RoR community already invented a solid amount of different wheels.



**You can also check out**

---

[This thread on Quora about advantages of Ruby on Rails](#)  
[Ruby on Rails: Powerful and Cost-Effective Tool for Web Development](#)



## **7. The correct way to learn and master Ruby on Rails**

There are many paths to mastering the art and philosophy of the Ruby on Rails framework, and Ruby on Rails development.

For professional developers who already have experience building applications in Ruby, Python or PHP, adapting to Rails should be quite easy. After getting to know Ruby, going into Rails should mostly be a matter of:

- studying and practicing Ruby (for those switching from a different language),
- learning the Rails conventions,
- getting accustomed to the Ruby package manager as well as gems,
- becoming a part of the Rails community,
- practicing, practicing, and even more practicing.

However, if you're not a master programmer who can switch between languages and frameworks without blinking an eye, or you're at the very beginning of your professional software development journey, here's everything you need to become a skilled Rails developer.

## Foundations – for beginners

Before you get into Rails, you need to learn the rudimentary technologies of modern web development, including:

- HTML + CSS (for organizing and styling content on websites and web applications)
- Version control system like GIT (for managing the codebase and efficient teamwork between developers)

Contrary to some popular beliefs, knowledge of JavaScript and Database Management **is not necessary for learning Ruby on Rails**. Although it would certainly help ;)

And how can you learn these things? One of the best resources for beginner programmers is FreeCodeCamp. It's an interactive in-browser coding course. You sign-up, you instantly start coding, and the curriculum takes you through

everything you need to know to become a professional developer — including, but not limited to, everything listed above.

What's more, at the end of the course, you will receive a certificate, along with your first professional projects to quickly build up your portfolio. Did I mention that it's all for free? Oh, and there's an insanely helpful community there as well. If you're just starting out with a goal to build web applications for a living, there really is no better way to do it than FreeCodeCamp:



### **Beginner programmers: learn the foundations of modern web development at [FreeCodeCamp](#)**

If you've got the money for it, you can also go for paid courses like [Codecademy](#) and literally thousands of similar courses around the web. Most of them pale in comparison to what you get at FreeCodeCamp for free, but they might offer additional resources to expand your knowledge.

The most important thing here is to take your time. Building modern web applications is not rocket science, but it's also not gathering potatoes in the field. There's a reason why developer salaries are so hugely disproportionate to other professions — it takes a lot of hard work to learn the necessary skills.

You'll increase your chances if you take it slow. Be patient, and don't expect to start earning the big bucks that programmers do within a few months. It will take you at least a year to master the foundations of web development if you're coming into this with no prior experience or knowledge.

After you've mastered the basics, you can start thinking about getting into Ruby on Rails.

## Getting into Ruby

Is it possible for someone with basic web development skills to setup the Ruby on Rails environment and build an application? It definitely is! Is it a good way to learn? Definitely NOT!

Sorry, but you need to learn Ruby first. There's no way around it. You don't need to master it before you start using Rails, but you do need to get an understanding of how it works, and get some practice time with building different things using this language.

Here are some resources that will help you:

- [TryRuby](#) - an interactive environment where you can play around with the language
- [Official 20-minute tutorial](#) - once you install Ruby, you can go through the official tutorial, which will show you a bit of what can be done
- [Ruby Documentation \(+ list of the best learning materials\)](#) - right on the Ruby website, in the documentation section, you'll also find all the learning material you might need to learn, practice, and master Ruby
- [Learn Ruby the Hard Way](#) - also listed in the Ruby documentation, but we wanted to single it out here because it's one of the best courses that comes with an extra bit of value. It doesn't just teach you, it also helps you organize your learning process in a way that thoroughly prepares you for professional Ruby development. Probably the best Ruby on Rails book.
- [CodeWars](#) - you might want to visit this site once you start getting the hang of Ruby. It's a platform that offers a whole lot of coding challenges that you can solve in-browser. Solving real problems is the best way to practice and expand your skills! Plus, once you finish a challenge, you also get to see how other people solved it. There's almost never just one way to solve a problem and reading code written by other developers is another great way to learn.

## Entering Ruby on Rails

With a strong foundation of basic web development skills, plus an understanding of, and experience in using Ruby, you'll be ready to enter the world of Ruby on Rails. Which is a great world to be in, looking at the average Ruby on Rails developer salary. Is Ruby on Rails worth learning? Very much so!

One of the best courses is the [Ruby on Rails tutorial by Michael Hartl](#). It's a pretty long course for building Rails applications, which in this case is a good thing because you don't want to rush your learning. Finishing this course will enable you to start building applications in Rails, but there's much more to learn!

**You need to learn about all the different elements that make up Rails (like `l18n` and `ActiveJob`) as well as get accustomed to using gems. These resources will help you do so:**

- [Official Ruby on Rails guides](#) - learn all about the various ingredients of Rails, the best site to learn Ruby on Rails
- [RubyGems](#) - browse gems, learn about building and using Ruby gems
- RoR blogs and communities like:
  - [Official Rails blog](#)
  - [Reddit for Rubyists](#)
  - [RubyFlow](#)
  - [RubyLand](#)
  - [Ruby on Rails 5by5 podcast](#)
- [RubyTapas](#) + [RailsCasts](#) + [Confreaks TV](#) - additional educational materials in the form of screencasts

Another thing you might want to do is browse Rails applications on GitHub or other code repositories, and analyze how they're made. Here are a few links to get you started:

- <https://github.com/octobox/octobox>
- <https://github.com/helpyio/helpy>
- <https://github.com/opf/openproject>
- <https://github.com/discourse/discourse>
- <https://gitlab.com/gitlab-org/gitlab-ce/>

As a final note — don't try to go through all of the educational materials, or read all of the blogs on RoR. Once you're able to start confidently building Rails applications, you should focus on practicing as much as you can (actually writing code), and use educational materials and the community to get unstuck whenever you're not sure of the code you're writing.

For more information about securing your dream job as a Ruby on Rails developer, please read [this article](#).

You can also check our [job openings](#). We are always looking for great talents to join our team!

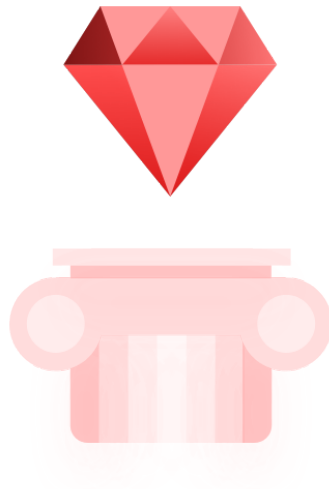


### You can also check out

---

[24 Ruby on Rails experts to follow](#)

[How To Find Your Dream Developing Job At A Ruby On Rails Agency?](#)



## 8. Finest Ruby on Rails examples

If you're wondering which companies use Ruby on Rails, and which popular sites are actually Ruby on Rails websites, look no further – here are some of our favorite popular Ruby on Rails websites.





If you work in web and application development — you know GitHub. It is the most popular software hosting service with the gigantic open-source library.

The service is now used by 1.8 million businesses, start-ups, and software development companies. It is also supported by the amazing community of 28 million software developers who have contributed over 85 millions of repositories so far.

The Github frontend went through many changes, and throughout the year, Github devs used different JS libraries (Pjax, jQuery, and more) now focusing solely on [Vanilla JS](#).

But the back-end is still pure Ruby.



Shopify is an e-commerce software platform with over 600,000 online retailers. It provides them with an easy to configure framework to do business on the internet and sell online. The platform allows its clients to create their own shopping experience thanks to tons of easily customizable themes.

Shopify is probably one of the most successful Ruby on Rails-based tech companies worldwide. They have been scaling massively within the framework. Simon Eskildsen from Shopify boasted about being able to address 80,000 requests per second. They also use the Liquid templating language for the front end and Turbograft, their hard fork of Turbolinks.



You probably know this one. It's one of the most popular online coding bootcamps, offering courses on Ruby



A very valuable tool, which was made evident by the 2012 [\\$1.2 Billion acquisition](#) by Microsoft. It is a social networking toolset for the modern

along with 11 other programming languages.

It was founded in 2001, and it has had 45,000,000 users so far. The [tech behind Codecademy](#) is written using Ruby on Rails along with Go and Javascript, and it's hosted on Amazon.

The company has raised \$47.5 million in [four funding rounds](#), the latest one was closed in 2016.

enterprise, which makes it easier for employees to collaborate in real time regardless of where they're located, which department they work in, or what type of applications they're using.

Yammer offers apps for web, desktop, and mobile, and it can be integrated with other enterprise software. It was originally founded in 2008, and it has grown to 7 million users. It uses Rails, Java and Node.js in [its tech stack](#).



Many adventures may not have happened if it weren't for this app, and the immense value that it has introduced to the worldwide community of travellers. It is one of the [world's greatest startup stories](#), where the hard work of the founders paid off immensely, and resulted in the disruption of a whole industry.

Founded in 2008, it grew to become one of the biggest marketplaces for renting and booking rooms and apartments for tourists to live in while they explore a new city. There are over 4,500,000 available listings from 65,000 cities in 191 countries on Airbnb. All of this powered, [among](#)



Most developers will know Heroku, one of the most popular cloud application platforms. Now Salesforce-owned (since 2011), it was initially founded in July 2007.

With support for multiple programming languages, including Ruby, Java, Node.js, Python, PHP and Scala, it gives developers a very straightforward way to get their products to market quickly. It reduces a lot of the typical problems that come with deploying, scaling, and managing applications.

It is built on a relatively [tight tech stack](#), written in Ruby on Rails along with Go and Erlang. Before it was acquired by Salesforce, the team had managed to

[other technologies](#), by Ruby on Rails, Java and Javascript.

With 4,500,000 listings in over 65,000 cities in 191 countries, Airbnb offers the widest variety of unique spaces for everyone, at any price point around the globe. [The project has raised](#) \$4.4 billion in 12 rounds, the last one closed in 2017.

raise \$13 million in three [rounds of funding](#) in just three years.



Founded in 2010, Fiverr is a unique platform for listing and buying freelance services of all types — from typical things like copywriting, to weird things like ordering videos with a custom message recorded by a guy who dresses like Jesus.

It lists over 3,000,000 services across 100 various categories, from people in 196 countries. It has a place among the 400 top websites in the world according to Alexa.com. The [tech stack](#) consists of a lot of technologies, and Ruby on Rails is one of them.

The company has raised \$111 million in five [rounds of funding](#).



If you're not a photographer, you probably don't know it — but trust me, it's a big one for the industry.

It's the world's leading community and toolset for aspiring & professional photographers to publish their work, get recognized, and get paid for it through a network of exclusive distribution partners. The site provides a way for photographers to build a portfolio, as well as tools to better understand trends and one's own photography.

It was founded in 2009, and is now host to over 15,000,000 users from 195 countries, helping them make a living from their craft. The project has raised

	<p>\$22.3 million in seven <a href="#">rounds of funding</a>, the latest one closed in 2015.</p> <p>This web application uses Rails in combination with Node.js, Python, and Go, among <a href="#">other technologies</a>.</p>
<div data-bbox="292 804 663 891" data-label="Image"> </div> <p>Available on mobile devices as well as in browsers, Scribd is an app that opens up access to a huge collection of books, audiobooks, documents, press articles, and other types of literary content. It is the biggest community-driven online library of its kind.</p> <p>It was founded in 2005, and it grew to over 100 million active users monthly. The project raised \$47.8 million in six <a href="#">funding rounds</a>, the latest closed in 2015.</p> <p>The <a href="#">tech stack</a> consists of Ruby on Rails, along with React and Backbone.js, among other technologies.</p>	<div data-bbox="948 770 1203 929" data-label="Image"> </div> <p>This platform is the world's leader in streaming for gamers. It powers the biggest events in the esports scene, and also lends its infrastructure to provide live and on-demand content in cooperation with media, publishers, and game creators.</p> <p>It is the main place where gamers go to stream and watch others not just play, but also lead their daily lives, go to the gym, and much more. The content on this platform has evolved a lot since its creation in 2007.</p> <p>Twitch has raised \$35 million in two <a href="#">rounds of funding</a>, the latest one closed in 2013. It was acquired by Amazon in 2014 for \$970 million. The <a href="#">tech stack</a> here is a mix of C++, Ruby on Rails, Go, and Ember.js, among other technologies.</p>



Founded in 2008, IndieGoGo has grown to become one of the most popular crowdfunding platforms in the world.

The company has raised \$56.5 million in five [rounds of funding](#), the latest closed in 2014. It relies a lot on Rails in its [tech stack](#), with AngularJS, Less, and multiple other technologies.



AngelList is a startup that relies mainly on Ruby and doesn't have many other programming languages in [their tech stack](#) except a bit of Javascript.

Founded in 2010, it is a kind of social platform that connects angel investors and potential employees with startups. It has raised \$26.2 million in eight [rounds of funding](#), the latest one closed in 2015.

## GROUPON

Groupon is a website that offers discounts and gift cards from local companies as well as enterprises. It's a niche type of e-commerce company, and since its creation in 2008, it has grown to become one of the biggest sites of its type.

The [tech stack](#) consists of Ruby on Rails along with Go, JRuby, Java, Clojure,

## KICKSTARTER

In our opinion, this is one of the best Ruby on Rails company websites. This amazing platform allows you to seek funding for your project or support others people's ideas.

Almost 150,000 projects were successfully funded via Kickstarter so far, including Oculus Rift and Wasteland 2. We have also seen much weirder projects, e.g. combat kitchenware (a

Scala and several other technologies. It is available in 48 countries. Groupon has raised \$1.4 billion in seven [rounds of funding](#), the latest closed in 2016.

frying pan attached to a sword) or digging a hole without any reason whatsoever.

Kickstarter is one of our favorites RoR-based websites in terms of UX/UI design with a user-friendly interface and intuitive user path. The service itself was built with the Ruby on Rails framework and Perl language with the main libraries being jQuery, React, and History.js.



### You can also check out

---

[40 examples of web applications built on Ruby on Rails](#)  
[Best Fintech Companies Powered By Ruby on Rails](#)



## 9. State of Ruby on Rails for 2019 + future possibilities

Fourteen years have passed since the release of the first version of the Ruby on Rails Web Development framework. In the internet world, it's a whole era in which a product can rise in popularity and then slowly fade away.

But it didn't happen to Ruby on Rails. It is now a truly mature technology, actively developed by a team of dedicated creators and supported by a great community. The huge amount of work put into the framework resulted in Ruby on Rails being one of the best thought-through backend technologies available on the market.

The long history of Ruby on Rails development gives a set of advantages:

1. It provides a lot of security – years of RoR development made it possible to find a great amount of breaches, bugs, errors to fix.
2. A very stable API – the Ruby on Rails updating process doesn't require developers to change the code for the app. With less stable frameworks, devs often need to implement a series of fixes in the code, which can take days and even weeks. Paired with the stability of RoR itself it provides an enormous business advantage!
3. Changes in the framework are evolutionary, not revolutionary. Since Rails 3.0 (when Rails and Merb were merged), it is a stable framework and the updates don't bring breaking changes that “flip the table.”

What's more important is that Ruby on Rails core team is still working on the updates for the framework (more on this in the next section). If you want to take a close look at what exactly is being done with Ruby on Rails at the moment, you can go ahead and analyze the [Rails Github page](#), where you'll find all the info you need, down to every single change that was made to RoR across the years.

## Ruby on Rails in Numbers

What demonstrates the popularity even better are the numbers:

1. There are currently [over 3,500 GitHub contributors](#) to the ROR's source code.
2. The framework has been [downloaded over 130 million](#) times up until now.
3. In the [Stack Overflow 2018 survey](#), Ruby ranked 13th in the most popular programming languages ranking. It's hard, however, to call 13 an unlucky number in this case. In fact, comparing the data to the [Stack Overflow 2017 survey](#), it seems that Ruby has become even more popular since last year.
4. [The TIOBE index](#) highlights Ruby as one of the 20 most popular technologies used around the world.
5. According to [BuiltWith.com](#), there are currently around one [million live websites using Ruby on Rails](#). The framework is popular in all major countries



known for its great devs, like the United States, Canada, and Germany.

6. Stackshare data shows that there are more than 2,500 open RoR positions in almost 500 companies.

## Ruby on Rails Development in 2019

The year 2018 welcomed a new major update to the framework – the 5.2.0 final version. It keeps the system up-to-date with such things as improved cloud file uploading, enhanced encryption, and much more. By the end of 2018, we have also witnessed the Ruby 2.6.0 release. It brought another set of improvements: new module (RubyVM::AbstractSyntaxTree), new features and a series of performance improvements. But the most important was probably the initial implementation of a JIT (Just-In-Time) compiler which works in a slightly different way. Recently, we also welcomed the Ruby 2.6.2 release, which brought minor fixes and changes.

**The biggest update coming in 2019 will be the Rails 6.0 version.** It will be released on the 30th of April during RailsConf2019.

Rails 6.0 is a major update. It will be packed with many features that both smaller and bigger applications will benefit from, like Action Mailbox, Action Text, Parallel Testing, and Action Cable Testing. Some of them can be already tested – you can find more info about it on the [official Ruby on Rails website](#).

**For now - we wait. But we will update this section as soon as new versions of Rails are released!**

### **Ruby on Rails - what will the future bring?**

It is hard to predict the future in the long run when it comes to development frameworks. But the future looks bright. Apart from this year's updates, the Ruby on Rails core team has a long distance plan: they are heading towards the so-called 3×3 goal – it is planned to become times faster than it is right now until 2020.

The Ruby community is important as well – people will continuously create new gems and support the product itself. Most of the new frameworks can't count on this kind of support from its users.

Let's also not forget the huge amount of apps already written in Ruby on Rails. They will be developed and maintained as well, which will create ongoing demand for new solutions, updates and Ruby on Rails specialists on the market.

**Overall, Rails is still going strong, and the demand for Ruby on Rails developers is here to stay.** It's one of the most popular technologies for custom web application development. As a Ruby on Rails development company, we can proudly say that it is our favorite technology.

And for the non-believers who'd rather look at popularity numbers instead of digging into the subject matter, we have an opinion from Avi Flombaum, Organizer of NYC on Rails. It was originally written in answer to a [2016 Quora question](#) about the decline of Rails, but it continues to be relevant:

*Why is Ruby on Rails on the decline? Why do you say it's slow?*

*Twilio is a large Rails application and it just went public.*

*DollarShaveClub is a large Rails application and it just got acquired for a billion dollars.*

*Shopify went public 6–8 months ago and it's a large Rails application. The rumors of Ruby on Rails death have been greatly exaggerated. What are you going to do as you try to chase trends forever?*



**You can also check out**

---

[State Of Ruby On Rails Web Development At The Beginning Of 2019](#)

[Rails 6.0.0 Beta1 Released! What's New?](#)



## **10. Summary – Is Ruby on Rails the right choice for you?**

We can't tell you for sure whether you should, or shouldn't use Ruby on Rails in your product's tech stack. There are too many variables to consider, and it depends on the unique context of your project.

**If you are a startup**, there are plenty of areas where Ruby on Rails can come in handy. If you just start building your product, Ruby makes it fast to bootstrap and get an MVP running.

**If you are a product owner** of an existing product, especially if it's ecommerce or SaaS, you may consider switching to Ruby as well. It gives you great performance, provides amazing user experience and high levels of security.

**If you're a developer** looking to sharpen your skills and increase your earning potential, then learning Ruby, along with Rails should definitely be on your to-do list.

In any other case, if you need additional data to make an informed decision as to whether Ruby on Rails is the right technology for you, your team, or your product, then we have summarized the most important information from this guide, right here in the list below. Hope this helps!

- Ruby on Rails (*aka* Rails *aka* RoR) is an open-source **framework for building web applications**, written in the Ruby programming language. It was first created as part of Basecamp's codebase, extracted from it in 2003, and released to the public in 2005.
- As confirmed by the global RoR community, it is a technology **optimized for programmer happiness and beautiful code**. It is one of the fastest technologies for building a quick MVP, making it perfect for projects with a tight timeline.
- **Rails is opinionated software**. It makes assumptions about what you need to build a web application, does a lot of the basic things for you, and allows you to focus on the high-level design of your application. It requires programmers to adopt the "Rails Way" of software development.
- **Ruby on Rails is a mature technology with a large, and very active global community**. Developers keep building new open-source gems, which allow others to build products even faster.
- **Rails is perfect for startups and SMBs** (speed and ease of development), **e-commerce projects** (security, maturity, stability), and **SaaS products** (RESTful architecture, and ease of integration with other services).
- Rails goes hand-in-hand with modern project management methodologies like Scrum, and Rapid Application Development and other Agile methodologies.
- Rails provides a cost advantage in the early stages of product development because it's an open-source technology with no licensing costs, and the huge community of

RoR developers continues to create and publish new gems that others can use for free.

- The correct A-to-Z way to become a master of Ruby on Rails is to first learn the basics of web development (HTML, CSS, JavaScript, Git, etc.), then the programming language Ruby, and only then learn Ruby on Rails.
- Rails is used in tech stacks of industry-leading companies like: Basecamp, Airbnb, GitHub, Soundcloud, Codecademy, Yammer, Heroku, Fiverr, 500px, Scribd, Twitch, Indiegogo, Angellist, and Groupon.
- The Rails community has maintained a steady number of contributions to the codebase for over 10 years, and there are thousands of open job positions that require Ruby on Rails, as well as thousands of existing Rails applications that need maintenance.

## Conclusion

**Ruby on Rails one of the best frameworks out there. It has a long history of development and it has been helping companies of all sizes. It is supported by an amazing community and thanks to that, we see a bright future for it.**

We hope that this guide has expanded your knowledge about Ruby on Rails. Truly it is one of the most important technologies in web development, and it has helped countless startups produce beautiful, functional, modern applications. Maybe it will help you too?

## Wow! You reached the end!

Thank you for reading our guide. Hopefully, at this point, your understanding of Ruby on Rails is much deeper and maybe you even start feeling the love towards this framework (as we do!).

We did our best to cover all the important areas connected to Ruby on Rails. But we know that with the growth of the framework, this page will need to grow as well. That's why you can count on us to constantly update and develop the content so it is always up to date and provides the highest quality of information possible.

In the meantime, you can spread the word about Ruby on Rails by sharing this guide.

*Ideamotive Team*

### SHARE THIS GUIDE IN SOCIAL MEDIA

