Como esse curso funcionará

▼ Introdução

Este curso é sobre ensinar Ruby. Uma tarefa que leva dezenas de linhas de código para ser realizada em Java ou uma centena em C pode levar apenas algumas em Ruby, graças ao Ruby pré-empacotar tantas funções úteis em métodos fáceis de usar e convenientes.

Ruby tem uma arma secreta que faz dela o amor dos desenvolvedores web: o framework **Ruby on Rails**. O Rails foi otimizado para escrever código mais rápido e com menos dores de cabeça. Uma iteração mais rápida significa que o produto final tem mais probabilidade de atender às necessidades dos clientes, típico de startups.

▼ Aprendendo Ruby antes de Ruby on Rails

O Rails é um framework construído exclusivamente em Ruby, e cada pedaço de código nele é escrito em Ruby. Quando algo no seu projeto quebra, é melhor você ser capaz de depurá-lo, o que significa ter um forte entendimento de Ruby, como ele funciona e como são suas mensagens de erro.

▼ + Infos

⚙️ Tools

os preceitos abaixo são para o SO Linux

▼ Runtime

caso você já possua um arquivo hello.rb, navegue até seu diretório e o execute com:

```
ruby hello.rb
```

para imprimir a versão do programa:

```
ruby --version
```

ou é possível executá-lo sem armazená-lo em um arquivo:

```
ruby -e 'puts 123'
```

▼ Ruby Interativo

Outra ferramenta que vem com o Ruby é o `irb`(Interactive Ruby Shell), e ele é outro tipo de shell, no entanto ele espera que você digite códigos em ruby em vez de comandos do sistema, exe.:

```
$ irb
> puts "Hellow world!"
Hellow world!
=> nil
>
```

Por enquanto não sabemos nada sobre o retorno dessa declaração “nil”, você pode sair digitando exit ou apertando ctrl + d.

🍽️Preparativos

▼ Rubi elegante

A linguagem Ruby tem convenções estilísticas específicas que tornam a leitura escrita de código Ruby mais fácil. São elas:

1. A função de tabulação do seu editor deve ser definida para 2 espaços;
2. Comentários são feitos com #;
3. Ao definir ou inicializar um **método**, **variável** ou a **arquivo**, você deve sempre usar a formatação `snake_case`;

   ```
   def this_is_a_great_method
       # ...
   end
   ```

4. Constantes são declaradas com todas as letras maiúsculas:

```
FOUR = 'four'
```

5. Ao trabalhar com do/end, prefira `{ }` quando toda a expressão couber em uma linha;

```
[1, 2, 3].each { |i| do_some_stuff }
```

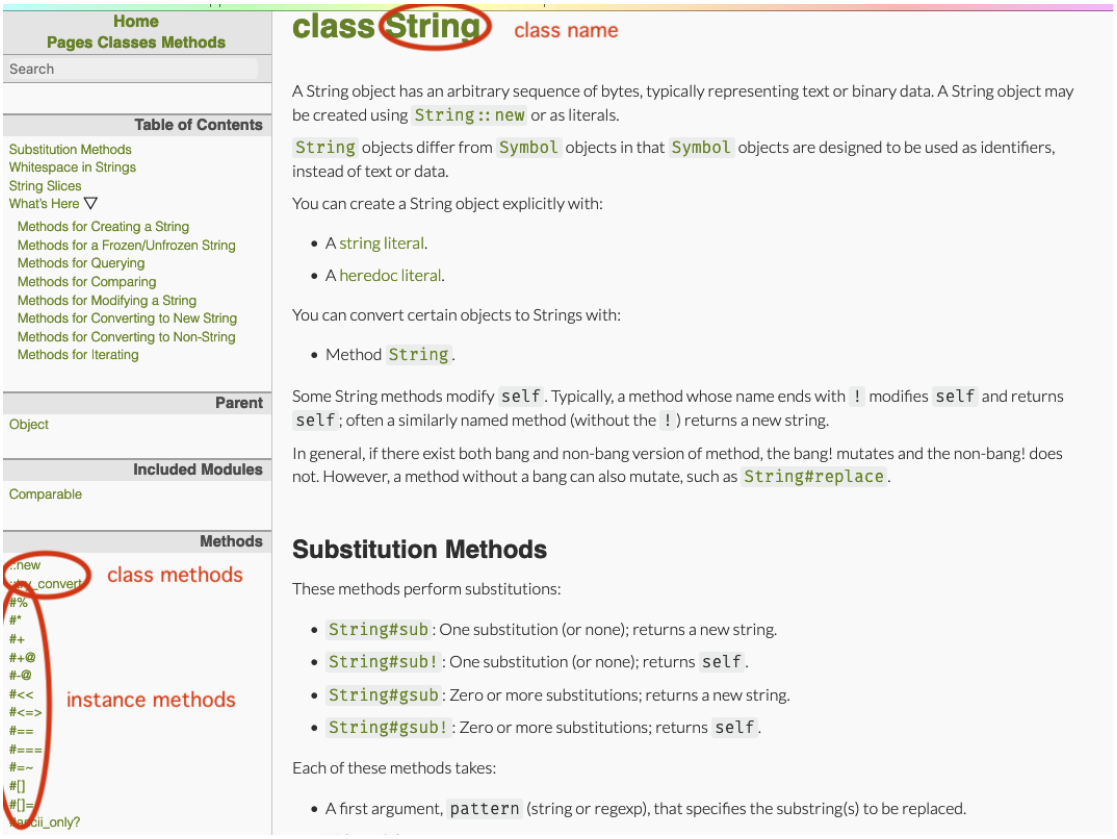
6. Para declarar um nome de classe é necessário utilizar a formatação `PascalCase:`

```
class MyFirstClass
end
```

▼ Lendo Documentação

É necessário estudar a biblioteca do Ruby, não necessariamente se debruçar sobre a lib. Mas sim estabelecer uma rotina de frequenta-lá, é importante saber que a maioria dos Devs, se referem a documentação como **API**(”Você deu uma olhada na API do Array?”), que significa Application Programming Interface.

Afim de ler melhor a documentação do Ruby, a imagem abaixo captura os pontos de focos com 3 círculos.



Nome da classe ou nome do módulo

O primeiro círculo no top, vemos a palavra “String”. Esta é a **Classe** ou **Módulo**. Caso veja alguma classe sendo referenciada com um :: símbolo, como este Enconding::Converter. O símbolo :: é usado para definir um **namespace**, que é apenas uma maneira de agrupar classes em Ruby e diferenciar de outras classes de mesmo nome.

Métodos

Os métodos são listados com :: ou #. Métodos denotados por :: são considerados *métodos de classe* e os # são considerados *métodos de instância*. Convenção **APENAS** para ler a documentação.

Exemplo: Métodos de instância vs métodos de classe

A grande lição que se tira dessa documentação é que os **Public Instance Methods** (#) podem ser aplicados a qualquer instância da classe. Podemos olhar a string “world wide web” e aplicar diretamente métodos de instância a ela. Por exemplo o método de instância #split que pode ser encontrado na barra lateral dos Methods é um método de instância, e podemos chamar esse método em qualquer string diretamente:

```
irb :001 > "world wide web".split
=> ["world", "wide", "web"]
```

Além desses métodos, existem também dois métodos de classe: ::new e ::try_convert. Métodos de classe públicos são chamados diretamente da classe, no nosso exemplo a classe é a String.

```
irb :001 > b = String.new("blue")
=> "blue"
irb :002 > String.try_convert("red")
=> "red"
```

Resumimos em poucos passos como se deve extrair informações da documentação do Ruby para se tornar fluente em Ruby.

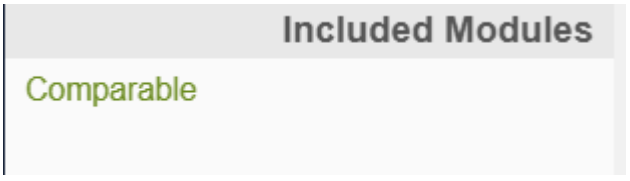
Pai

Em ruby, cada classe é subclasse de uma classe “mãe”. A classe que estamos olhando também tem acesso a métodos - tanto instâncias quanto classes - documentadas na classe mãe. E para isso existe outra documentação rsrs.

Neste exemplo, a subclasse String é filho da classe Object.

Módulos incluídos

Módulos incluídos significam que para aquela classe, suas funcionalidades estão incluídas. Para o método String, existe o módulo Comparable:



Isso significa que podemos fazer algo assim:

```
irb 001 > "cat".between?("ant", "zebra")
=> true
```

O método `between?` não está listado em String, mas mesmo assim pode ser chamado devido ao módulo incluído `Comparable`. Se verificarmos o módulo `Comparable` encontraremos o método `between?`

Recursos suplementares

Por a documentação oficial ser um pouco deficiente, aqui ficaram algumas opções de sites não oficiais sobre a documentação do ruby que possuem aspecto mais atual e são em tese mais simples de usar:

1. documentação não oficial
2. rubydoc.info/stdlib/core/index

▼ O que são “gemas” do rubi?

Há dois lados principais para este termo. O primeiro lado se refere a uma coleção de arquivos Ruby, ou biblioteca Ruby, que executa uma determinada tarefa. Outro lado se refere ao sistema de publicação que está por trás da organização, listagem e publicação dessas bibliotecas, ou gems.

O código em uma gem é como pacotes pré-empacotados de códigos escrito por alguém que resolveu um problema útil. Isso significa que você gastará tempo com recursos úteis ao invés reinventar a roda.

Como usar?

```
gem install <gem name>
```

onde `<gem name>` é o nome da gema real que você irá instalar.

Instalar gems é bom para uso único, mas com o tempo, você perderá o controle de quais gems pertencem a qual projeto. Conforme iremos progredindo seremos apresentado ao Gemfile que oferece uma solução simples para organizar gems.

Depurando código Ruby com Pry

Pry é uma biblioteca alternativa ao irb, como instalar:

```
gem install pry
```

Comando que, assim como o irb, irá abrir uma nova sessão.

Use pry para depuração

```
require "pry"

a = [1,2,3]
a << 4
binding.pry #execution will pause here
puts a
```

Isso significa que quando o programa chegar em binding.pry ele abrirá uma nova sessão em vez de passar para a próxima linha de código. Assim é possível brincar com as variáveis e objetos para ver por que as coisas não funcionam. Ao terminar a depuração digitando ctrl + d é possível sair da sessão.