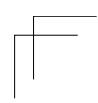
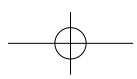
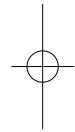
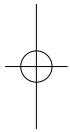
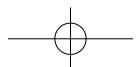


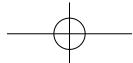
目次

1. マンガでわかるRuby 誕生の契機	3
1.1. 2年前からあった要望の声	4
1.2. Rubyエンジニア youchanとの出会い	4
1.3. コラボレーションは足し算ではなく掛け算	6
2. Kame Remocon制作記	8
2.1. Kame Remoconの開発方針	9
2.2. Opalについて	11
2.3. Kame RemoconとdRuby	12
2.4. コンソールからのリモートコントロール	14
2.5. 魔方陣投稿サイト	15
3. CSS組版	18
3.1. Vivliostyleのハマりどころ	19
4. あとがき	22
4.1. 著者紹介	25



2

マンガでわかるRuby 誕生の契機



2019年4月の技術書典6にて『マンガでわかるRuby①～基本・再帰関数編～』を頒布しました。この本は、Rubyエンジニアのyouchanさんと、『わかばちゃんと学ぶ』シリーズ著者の渕川あいによるコラボレーションで実現しました。どのようなきっかけで、どのように本の方針が定まつていったのでしょうか？

2年前からあった要望の声

実は2017年から、[わかばちゃんのRuby本を希望する声](#)がありました。

わかばちゃんはJavaScriptとjQuery、RubyとRails、PHPとWordPressがあつたら読みたいな

2017年4月30日 北上ユキカさん @feilice_labb のツイートより

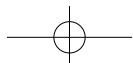
マンガでわかるRubyは絶対欲しい。…つか、これはこの前直に、あいさんに欲しい旨伝えていた奴だ。 2018年10月10日 ちいさいあおかみさん @siu_long のツイートより

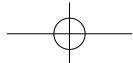
『わかばちゃんと学ぶGoogleアナリティクス』や『マンガでわかるDocker』を描きながらも、頭の中にはふわふわと「マンガでわかるRubyもいつか実現させたいな」という気持ちが漂っていました。 youchanさんに出会ったのはそんな折でした。

Rubyエンジニア youchanとの出会い

youchanさんと初めてお会いしたのは、[もくもく執筆会 #techbook_meetup](#) だったと記憶しています。2018年10月に行われた技術書典5に向けて、youchanさんが新刊『猫と森羅と日本語とRuby』を執筆する際、表紙イラストと挿絵を依頼してくださったのが始まりです。

技術書典5の打ち上げにて、イベントの熱気もそのままにお酒を酌み交わしていたところ、話が弾み「今回は挿絵だけの依頼だったけど、次回は共著を書いてみたいね」「それならマンガでわかるRubyはどうかな？」——かくして、私たちは技術書典6に向けて始動したのです。





Rails Girlsに参加した人のnext stepになるような本を書きたい

その2ヶ月後、youchanさんが Rails Girls Japan Advent Calendar 2018 に記事を投稿しました。記事より一部を抜粋します。

こんにちは。Rails GirlsではTokyoの5回目のオーガナイザーをしました。よう(@youchan)といいます。Rails Girlsに参加した皆さん、どうでしたでしょうか？楽しかったですか？こう質問すれば、大抵の参加者からは楽しかったー！という返事が返ってきます。そうなんですよ。Rails Girlsは参加者の満足度の高いイベントとしても知られています。それでは、Rails Girlsに参加したあとにRailsで何かつくりましたか？（中略）Rails Girlsのカリキュラムでは拍子抜けするくらい簡単にWebアプリケーションが作れたと思います。カスタマイズも簡単！それはRuby on Railsというフレームワークがとても良く出来ていることの証左でもあります。実はRails Girlsのカリキュラムの中にはほとんどプログラミングという要素が出てこないのです。Rails Girlsはプログラミング教室ではないのでこれでよいのですが、私は皆さんにもっとプログラミングの面白さを知ってもらえたなら嬉しいなと思っています。そこで、Rails Girlsを終えたくらいの人に向けて楽しくプログラミングを体験してもらえる教材をつくりたいと思いました。私の最近の趣味は技術書の同人誌を書くことです。技術書典という技術書の同人誌即売会があります。この技術書典に出会って同人誌を書くようになりました。これまで自分が書きたいことがあってそれを書いてきたのですが、もっと読者について考えて、読者に寄り添った本を書きたいなと思うようになりました。Rails Girlsに関わってきた中で、Rails Girlsの卒業生に向けて、もっとプログラミングの楽しさを伝えられる本が書けたらよいなと思いました。そんな中、湊川あいさんと知り合い、彼女の書く本(マンガ)が私の望みにとても適っていると思い、一緒に「マンガでわかるRuby」を書きましょうと提案しました。湊川さんに快諾していただき「マンガでわかるRuby」の企画が始まりました。いま二人で企画を進めているところです。

[Rails Girlsに参加した人のnext stepになるような本を書こうと思っています - Qiita より](#)

youchanさんのこの記事により、youchanさんが目指すビジョンが私と近いことがハッキリとわかりましたし、それと同時にマンガでわかるRubyが進むべき方向性が定まったように思います。

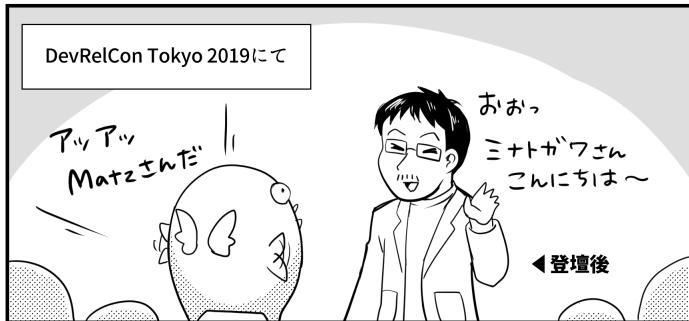
コラボレーションは足し算ではなく掛け算

マンガでわかるRubyは、私、湊川あい1人では実現できなかった本です。前職ではPHP畠の人間でしたし、Rubyの作法や現場での勘所も持ち合わせていません。youchanさんがいなければマンガでわかるRuby専用のgem「Kame Remocon」もこの世には存在しなかつたでしょう。

小学生の頃の湊川は意固地な性格でした。ほとんどのことは1人で出来てしまう上、完璧主義だったので「みんなで協力しましょう」なんて言葉は「よく噛んで食べましょう」ぐらい意味のない言葉だと思っていました。たとえば学校の班活動。自分1人が発表内容を必死に模造紙にまとめている中、遊んでいる子、寝ている子、文句だけ言う子……。みなさんも経験があると思います。自分がいくら頑張っても、それは班全体の評価になってしまいます。なんて不平等なのでしょうか。「共同作業しても損するだけだ、人の手を借りたら負けだ」一いつしかそんな固定概念が出来上がっていました。

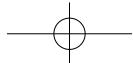
コラボレーションの素晴らしさに気付いたのは、社会人になってからです。私が幸運だったのは2つ。とびっきり仕事ができる社員たちに囲まれたことと、その中でも上司が私の得意分野を認めてくれたことです。戦略立案が得意な上司と、手を動かしてWebコンテンツを作るのが得意な私との共同作業は、困難にぶつかりながらも次々と成果を上げていきました。コラボレーションとは、単に一緒に作業することではなく、それぞれの得意な部分を生かし合って、足し算ではなく掛け算を作ることだったのです！1人の労働力が10だとします。単に一緒に作業するだけなら $10 + 10 = 20$ 。ところがうまくコラボレーションすると $10 \times 10 = 100$ の価値のものが生まれるわけです。

その会社を退職しフリーランスになった今も、基本的には1人で動くものの、必要に応じてコラボレーションをしています。1人で作りきることが目的ではない。本来の目的は、よりわかりやすく・正確で・読んでいて楽しい解説作ることのはず。意固地になるのではなく、自分ができること・できないことを素直に受け入れ、尊敬できる相手とコラボレーションする。「人の力を借りるのは負けなんかじゃない」そう気付いたことで、実現できるとの可能性がグンと広がったように思います。



Matzさんご快諾ありがとうございました！改めて感謝申し上げます！(湊川)

Kame Remocon制作記



さて、以上のような経緯で「マンガでわかるRuby」の制作を始めたわけですが、問題はテーマの一つであるタートルグラフィックスです。既存のRubyから使えるタートルグラフィックスはいくつかあったのですが、gemになっていてインストールしやすいとか、グラフィックスライブラリに依存しているとそっちのインストール方法とかも煩雑でないほうがいいとか、最後は魔方陣投稿サイトを作って投稿できるようにするといいよね、だったらWebベースのものがいいかもとかいろいろ理由あって……本当は単に作りたかっただけなのですが……タートルグラフィックスのgemをつくることにしました。マンガのなかで亀を操作するリモコンなんだという説明を取り入れたので、その名もKame Remoconと名付けました！

Kame Remoconの開発方針

開発方針というほど大袈裟なものではありませんが、どういう構成にしてやるかということ検討しました。そのときに考慮したことは次の通りです。

ブラウザに描画すること

これは投稿サイトを作るということからというのもあるのですが、始めからOpalを使いたかったからです。

Opalというのは、RubyからJavaScriptへソースコードを変換するコンパイラです。詳しくは次節で解説します。

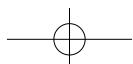
ブラウザからでもコンソールからでも実行できること

ブラウザから実行できるというのは、つまりOpalでブラウザ側のRubyで実行するということです。これは、投稿サイトを考えたときにRubyのプログラムをサーバーサイドで実行するようになるとセキュリティ上考慮しなければならないことが増えるためです。ブラウザで実行すれば投稿者は自身の環境しかプログラムから操作できないことになります。

またコンソールからでも実行できるようにしたのは、.rbファイルとして保存したり、irbから呼び出せるようにするためにです。これはRubyの入門としても成立させるために必要なことでした。これらは同じインターフェースで呼び出せるように考慮しました。

なるべくシンプルなインターフェースにする

初心者向けなのでなるべく簡単につかえるようにしたいですね。



たとえば、レシーバーが無くても亀を操作するメソッドを呼びだせるようにしたりです。レシーバーを明示しなければならないとすると、メソッドを定義する場合にメソッドにレシーバーも渡す必要がでてきたりします。

```
def draw_xxx(kame)
    kame.forward 100
    ...
end
```

と書くより

```
def draw_xxx
    forward 100
    ...
end
```

と書けたほうがシンプルですよね。

以上のような方針に則りKame Remoconの実装していました。Kame Remoconのソースコードのなかにはこれらの方針を実現するためのさまざまな工夫が施してあります。この本ではそのエッセンスを解説したいと思います。

ソースコードは以下のリポジトリにあります。

<https://github.com/youshan/kame-remocon>

Opalについて

OpalはRubyのソースコードをJavaScriptのソースコードに変換するコンパイラです。 Rubyで書いたプログラムをJavaScriptに変換してブラウザ上で実行できます。

Rubyにはeval()というメソッドがあって、Rubyのプログラムを文字列として与えて実行します。たとえば次のようなプログラムです。

hello.rb

```
world = "world"
eval("puts 'Hello, #{world}.'")
```

このプログラムはOpalでも実行できます。Opalで実行するときは、opal-parserをrequireして以下ののようにする必要があります。

hello_opal.rb

```
require "opal-parser"
world = "world"
eval("puts 'Hello, #{world}.'")
```

以下のコマンドで実行できます。

```
opal hello_opal.rb
```

eval()メソッドを使うことでブラウザ上でもRubyのプログラムを実行できることがわかりました。

Opalでブラウザ上にアプリケーションを作る方法は拙著の「Pragmatic Opal」にさまざまなテクニックと共に解説しています。ご興味のある方は是非手に取ってみてくださいね。

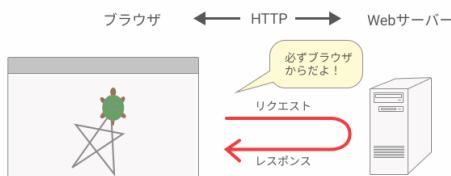


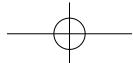
<http://www.impressrd.jp/news/180316/NP>

Kame RemoconとdRuby

Kame Remoconはコンソールから操作することも出来るようにデザインされています。実際に亀が動くのはブラウザ上ですがコンソール側のRubyのプログラムを実行して亀を動かすことも出来ます。Remoconという名前はここから来ています。

Remocon機能を実現するにはブラウザで実行されているOpalのプログラムとコンソール側のRubyのプログラムが通信する必要があります。通常、ブラウザとWebサーバーの通信というとHTTPを使った通信になります。HTTPはブラウザからWebサーバーヘリクエストしてWebサーバーがレスポンスを返すというリクエスト-レスポンス方式です。Kame Remoconの場合はコンソールからブラウザへの方向の通信になりますのでHTTPでは出来ません(そもそもWebサーバーってどこにあるんだっけ?)。





そこでWebSocketという技術が使われます。WebSocketはWebサーバーを起点としてクライアントにメッセージを送ることもできます。(このようなメッセージングをプッシュ型と呼んだりします。)

Kame RemoconではdRubyというライブラリをつかってWebSocketの通信を行ないます。dRubyはRubyの分散オブジェクトのライブラリでMRIの標準添付ライブラリです。つまり、Rubyは追加のインストールなしに分散オブジェクトが使えるということです。分散オブジェクトというのは、プロセスやコンピューター間でオブジェクト同士が通信するための技術です。dRubyではコンピューター同士の通信を表向きはRubyのオブジェクトのメソッド呼び出しという形で見せるので、Rubyのプログラムから扱うことが容易になります。また、通信方式は隠蔽されていて、dRubyでは通信方式を変更する仕組みも用意されています。

このdRubyの通信方式をWebSocketに変更すればブラウザとWebサーバーの間の通信をdRubyで行なうことができます。そのためのgemが、`drb-websocket`と`opal-drb`です。それぞれ以下のようなライブラリです。

drb-websocket

MRIのdRubyに通信方式にWebSocketを追加します。

<https://github.com/youshan/druby-websocket>

opal-drb

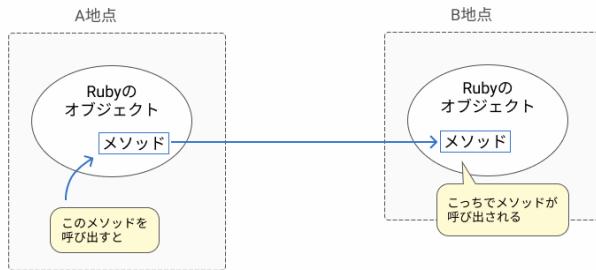
OpalにdRubyを実装したものです。

<https://github.com/youshan/opal-drb>

コンソールからのリモートコントロール

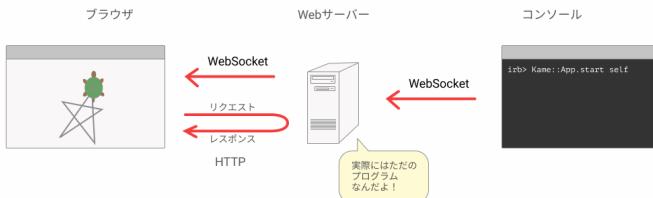
Kame RemoconはKame::App.startというメソッドで開始しますが、このときにWebサーバーを起動します。(Webサーバーがどこにあるという疑問が解けましたね。) WebサーバーとブラウザはdRubyを使ってWebSocketで通信するのでしたね。

dRubyはRubyのオブジェクトを介してネットワークの中で離れた場所にあるコンピューターやプロセスとやり取りします。離れているところにある二つのオブジェクトは透過的に利用できます。A地点にあるオブジェクトのメソッド呼び出しはB地点にあるオブジェクトに伝わってメソッド呼び出しがして実行されます。



Kame RemoconはdRubyのこのような特性を利用してブラウザとコンソールとの間でWebSocketを使った通信を行うだけでなく、ブラウザとコンソールで同じインターフェースで亀の操作を行えます。

サーバーはブラウザとの通信を確立すると、dRubyのオブジェクトを作つてコンソール側のプログラムに渡します。(このオブジェクトをリモートオブジェクトと呼ぶことにします。) コンソール側のプログラムはこのリモートオブジェクトを介してブラウザにある亀の操作を行います。



魔方陣投稿サイト

Kame Remoconで皆さんのが作ったオリジナルの魔方陣を投稿して自慢大会で出来たら楽しむなと思って、魔方陣の投稿サイトを作りました。

残念ながら「マンガでわかるRuby」の頒布に間に合いませんでした。(「マンガでわかるRuby」は技術書典6で頒布しました。) その後、1ヶ月ほど遅れて投稿サイトを公開することができました。

投稿サイトでは、Kame Remoconの持つ機能の他にユーザー管理や投稿する魔方陣の管理をしたりする機能があります。こういうWebアプリケーションの機能を実現するにはデータベースを利用します。データベースに保存するデータを管理したり、それを画面に表示したり、Webアプリケーションを構成する要素は沢山あります多くのWebアプリケーションには共通点があります。そのような共通点を集めて開発を容易にするのがRuby on Railsのようなフレームワークです。

Opalにもそのようなフレームワークがあります。Opalを使う場合はブラウザの画面を描くプログラムとサーバーでデータベースを管理するプログラムをRubyという共通の言語で記述できます。ブラウザでもサーバー側でも同じ言語で記述できるという特性はプログラミングをする上で利点があります。そのような利点を生かしたプログラミングスタイルをIsomorphic Programming(アイソモーフィックプログラミング)と言います。

魔方陣投稿サイトはIsomorphic Programmingのフレームワークとしてmeniliteを使っています。また画面を描くためのフレームワークとしてhyaliteを使っています。これらのフレームワークの使いかたは前述の「Pragmatic Opal」で解説されています。

menilite

<https://github.com/youshan/menilite>

hyalite

<https://github.com/youshan/hyalite>

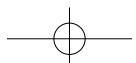
魔方陣投稿サイト

<http://kame-remocon.youshan.org>

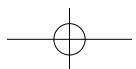
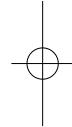
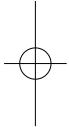
ソースコード => <https://github.com/youshan/magic-turtle>



「何でもかんでも簡略化して易しくすればいい」は間違い。大事なのは「それが読者の学びに繋がるかどうか」。読者のことを考えるとはどういうことか、改めて実感した出来事でした(湊川)



CSS組版



Vivliostyleのハマりどころ

改ページしたい

本を作る中で、ページの途中で強制的に改ページしたいときがあると思います。
Re:VIEWだと \clearpage と書きますが、Vivliostyleだと [page break] と書きます。

[page break] と書いておくとpage-breakというCSSクラスに変換されます。

```
.page-break {  
    page-break-after: always;  
}
```

ところがこの改ページがどうも効いていないようでした。そこで色々と試してみた結果、page-break-after: always; を追記することで、無事解決しました。before属性だけでなくafter属性でも効かせる必要があるみたいです。このbefore・after両方でガチガチに設定するとうまくいくパターンは他にも何回か遭遇しました。普段Webページを作るときは意識しないことですが、CSS組版では性質上ページ区切りが発生しますから、こういうことが発生するのかなと思いました。「うまくいかないときはbefore・afterガチガチ」、覚えておくとどこかで役立つかもしれません。

```
.page-break {  
    page-break-before: always;  
    page-break-after: always; /* 追加 */  
}
```

トンボを消したい

印刷会社さんが冊子を作ってくれるとき、裁断の目安になる目安線「トンボ」。印刷データには必要でも、電子版のPDFには不要です。このトンボをVivliostyleで消すにはどうすればいいのでしょうか？

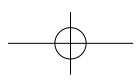
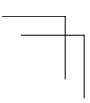
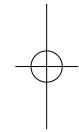
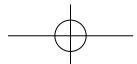
これが意外と簡単で、既存のCSSの@pageセレクタから marks: crop cross; という記述を消すだけで、綺麗にトンボだけ消えてくれました。

```
@page {  
    size: A5;  
    bleed: 3mm;  
    /* marks: crop cross; */  
    margin: 6mm;  
}
```

ページまわりの余白・塗り足し領域を指定したい

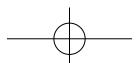
前述の@pageセレクタでは、見ての通りページ設定を命令することができます。ページまわりの余白はどのように指定するのでしょうか？なんと、おなじみの「margin属性」を使います。Webの仕事をしている人にとっては見慣れた属性だと思います。感覚的でわかりやすいですね。一方、初めて見る属性もあります。「bleed属性」……これは一体何でしょうか？bleedというのは「塗り足し領域」のことです。ここでは塗り足し領域の幅を指定できます。断裁したときのズレで、塗られていない白い部分が残ってしまうとダサくなってしまうので、ページの外側にはみ出して色を塗る塗り足し領域が必要です。日本の印刷業界では3mmとするのが標準的なのでbleed: 3mm;と指定しています。

『マンガでわかるRuby 1』は上記のCSSを使ったのですが、margin: 6mm;だと余白が少なめで、紙面全体がやや窮屈に感じました。今回はmargin: 8mm;にしてみたのですが、印刷の仕上がりのほどはいかがでしょうか？

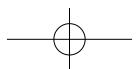
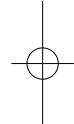
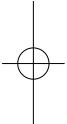


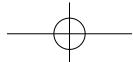
21





あとがき





youchan

ようです。「マンガでわかるRuby 番外編」いかがでしたほうか?そして「マンガでわかるRuby①」はもう手に取っていただけましたでしょうか?

この番外編ではタートルグラフィックスライブラリの「Kame Remocon」の解説記事を書かせていただきました。

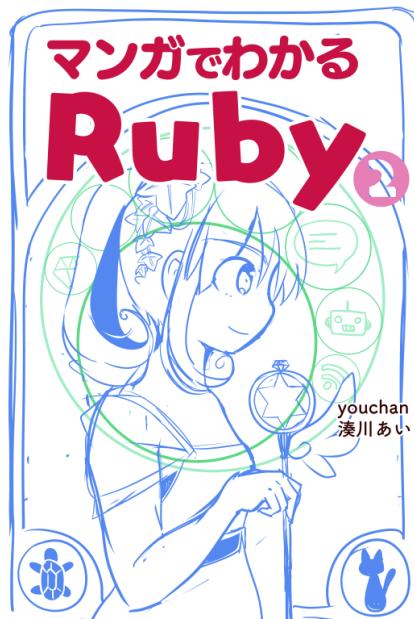
「マンガでわかるRuby①」の対象読者にはちょっと高度な内容だったかも知れませんね。でも成長というのは少し背伸びした先にあると思いません?逆に「マンガでわかるRuby①」ではもの足りなかった人には少しは興味を持ってもらえたかもしれませんね。

湊川さんとのコラボレート作品はこれで2冊目になりました。湊川さんはいつもぱぱっとまんがを描き上げて、ものすごい勢いで本を完成させていきます。私はいつも湊川さんをお待たせしてばかりで大変心苦しくもありますが、辛抱強く待ってくれる湊川さんのおかげで2冊目も無事書き上げることができました。「相方」湊川あいさんにこの場を借りて感謝を伝えたいと思います。ありがとうございました。

湊川あい

こんにちは!湊川(みなとがわ)あいです。蒸し暑い日が続きますがみなさんいかがお過ごしでしょうか。湊川はエアコン除湿モードフル稼働に加え、チョコミントをキメながら頑張っています。2019年7月27日にめでたく初開催となった技術書博覧会、略して技書博(ぎしょはく)。「せっかくイベントに出るなら新刊を出したいね」ということでyouchanさんとこの本を作るに至りました。

そしてそして、弊サークルは2019年9月22日開催の技術書典7にも無事当選いたしました!次回新刊『マンガでわかるRuby②』のマンガ制作も進めていきたいと思います。お楽しみに!



制作中の『マンガでわかるRuby②』表紙ラフ

そういうば最近、インタビューでわかばちゃんの名前の由来を聞かれます。実は、2014年にnote上でフォロワーさんたちからアイデアを募って決めたんですよ。20名以上の方が考えてくれた名前の案、一部を紹介させていただきます。「上部デザ子（ウェブデザイン）」「ゆい（UI）」「のこちゃん」「ゆるみちゃん」などなど。どれも個性的で面白いものばかりでした。そんな中でも、さらっと読みやすく、かつ初心者マークを連想しやすい「わかばちゃん」に決定しました。かれこれ5年前の出来事です。当時はこんなに続くことになるとは思っていませんでした。日々支えてくださる読者の皆様に感謝です。わかばちゃんは今後どんなふうに成長していくのか？私自身もとても楽しみです。

著者紹介

youchan



猫が好きなRubyエンジニア。2014年にJavaからRubyに転向し、現在は、RubyKaigiやロサンゼルスで行われたRubyConf 2018に登壇するなどRubyistとして活躍している。Twitterアカウントは@youchan。



Pragmatic Opal 発売中

Amazon
でも
買えるのじゃ!



著：大崎 瑞(youchan) / 出版社：インプレスR&D

Opalは、RubyをJavaScriptに変換する
トランスペイラーで、フロントエンドもRubyで
開発することができます。

Opalのサポートツールを開発してきたyouchanが
OpalをつかったWebアプリケーション開発を
余すところなく解説しています。

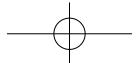
BOOTHでも 販売中じゃ!!

youchanのBOOTH



<https://youchan.booth.pm/>





湊川あい



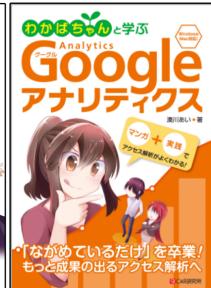
フリーランスの漫画家 / Webデザイナー / 技術書執筆/Live2Dモデラ
ー。『わかばちゃんと学ぶ』シリーズ著者。好きな食べ物は茎わかめ。
Twitterアカウントは@lluminat0ll。



運用☆ちゃん 書籍化決定

4/13発売!!

技術書典でもご購入いただけます



著書

わかばちゃんと学ぶ Webサイト制作の基本

わかばちゃんと学ぶ Git使い方入門

わかばちゃんと学ぶ Googleアナリティクス

Web連載

マンガでわかるGit (リクナビNEXTジャーナル)

マンガでわかるGoogleアナリティクス (KOBITブログ)

マンガでわかるScrapbox (Scrapbox)

マンガでわかるLINE Clova開発 (TECH PLAY Magazine)

マンガでわかる衛星データ活用 (宇宙ビジネス情報サイト 宙畠-sorabatake-)

個人制作誌

マンガでわかるDocker ①～③

告白に学ぶHTTPステータスコード～エラー編～

マンガでわかるWebデザイン 設定資料集

その他 漢川あいの、わかば家。 (pixiv BOOTH) でダウンロード販売中

免責事項

本書の内容は、情報提供のみを目的としております。正確性には留意しておりますが、正確性を保証するものではありません。この本の記載内容に基づく結果について、著者は一切の責任を負いません。

会社名、商品名については、一般に各社の登録商標です。TM 表記等については記載しておりません。また、特定の会社、製品、案件について、不当に貶める意図はありません。

本書の一部あるいは全部について、無断での複写・複製・アップロードはお断りします。

マンガでわかるRuby 制作秘話

2019年7月27日 初版発行

著者： youchan・湊川あい

イラスト・マンガ： 湊川あい

(C)2019 youchan / Ai Minatogawa
