Gatsby

Super Fast

Gatsby

Super Fast

今日伝えたいこと

- Gatsbyを使うと
  爆速サイト＆ブログが簡単にできます！

- 今日は詳細な仕組みを説明するけど
  30分あれば簡単にできちゃいます！！

- Gatsbyは旬の技術テンコ盛りなので
  フロントエンド勉強の題材としても最高

自己紹介

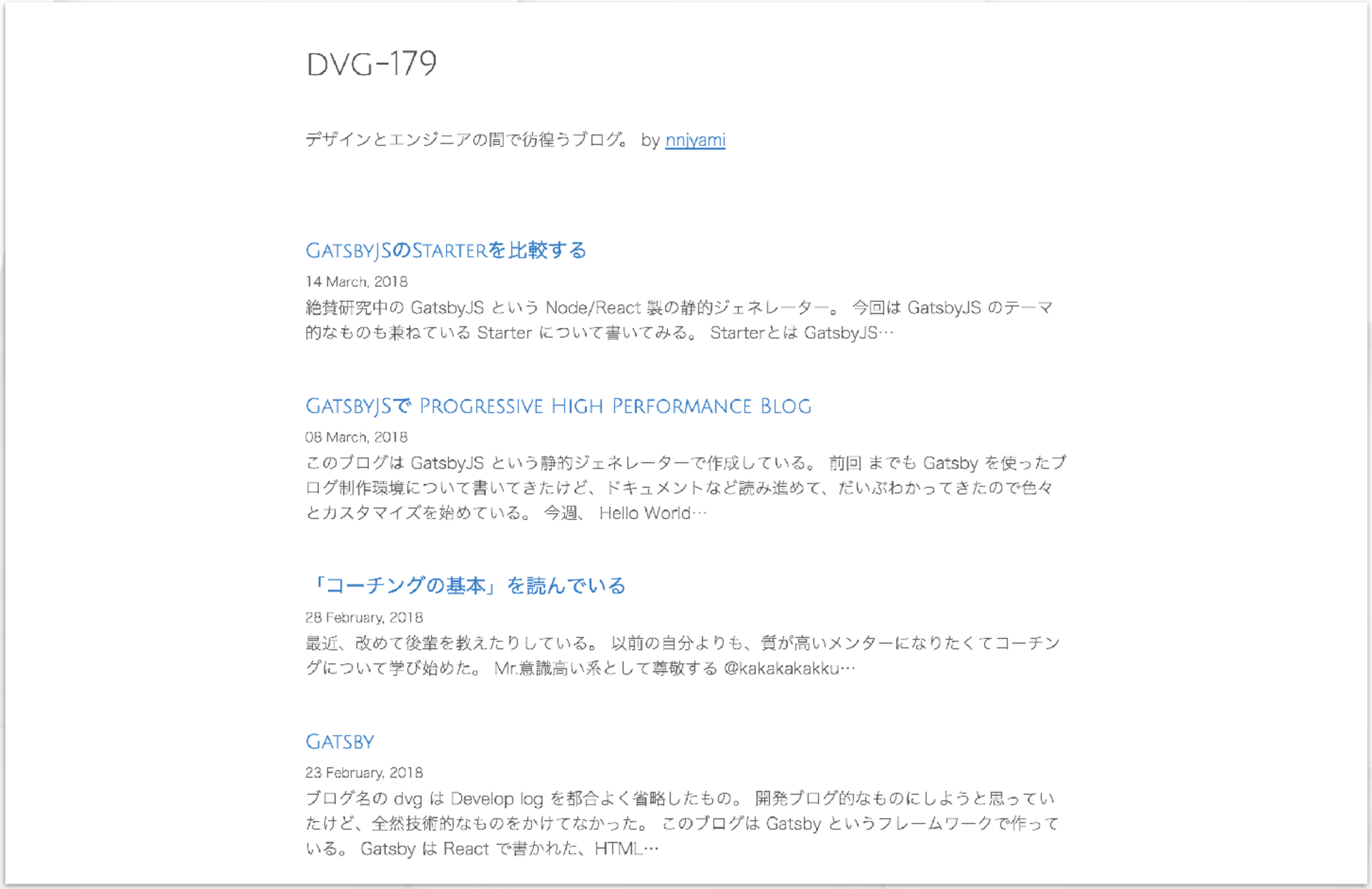
山下 浩一郎

@nnjyami（んじゃみ）

身長 179 cm　→　179.jp

京都で働くレガシーなフロントエンドエンジニア
（JS, PHP, Sass, HTML, …）

今年のテーマは「 **Output** 」



new 開発ブログ – **dvg.179.jp（Gatsby）**

https://dvg.179.jp

Gatsby による表示速度 & 画面遷移の速さを体感してください！！！

dvg.179.jp (2018.2〜)

Sakura VPS

DNS: Cloudflare

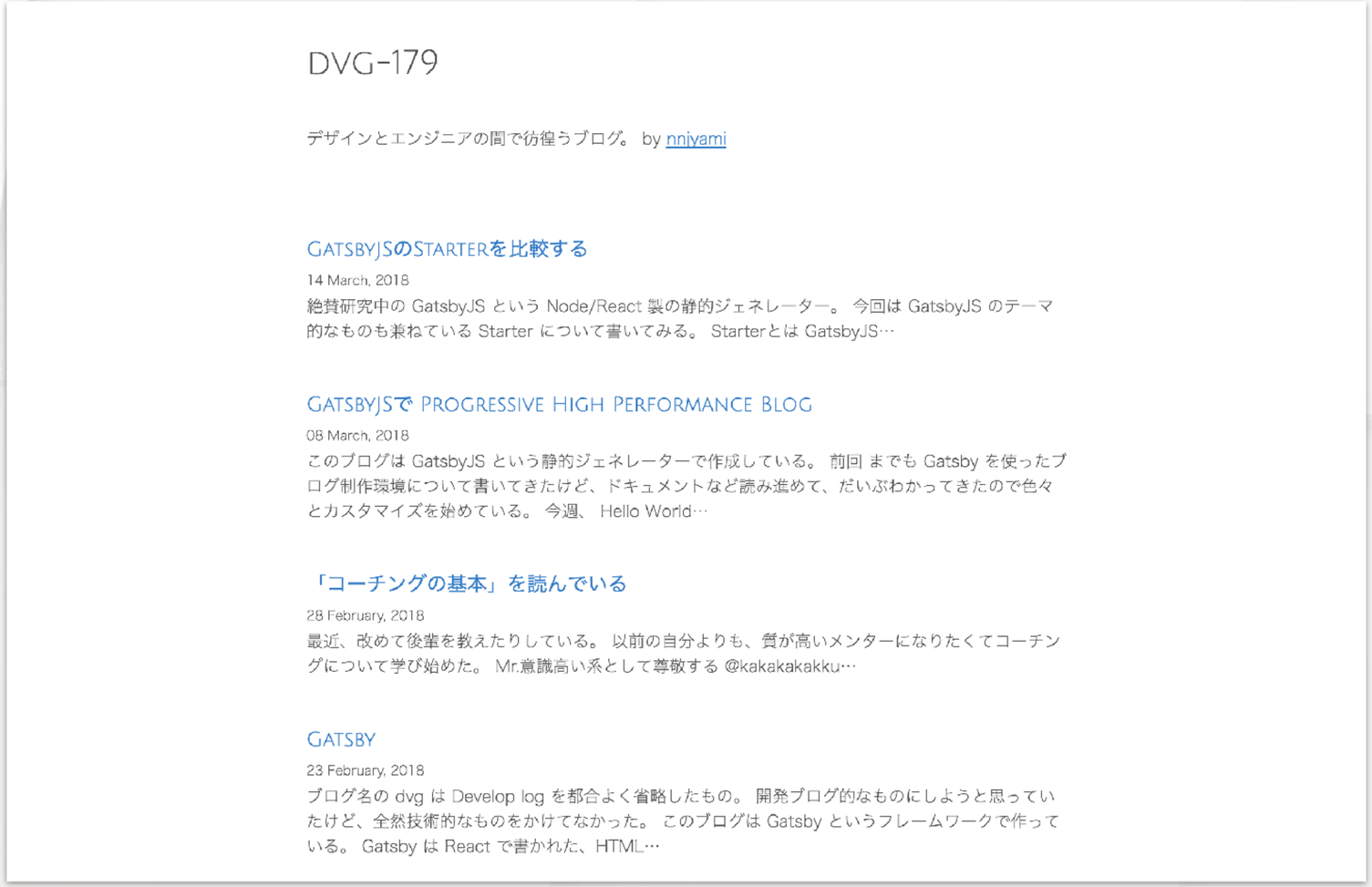
CentOS 7

Nginx

Node

GatsbyJS

– 勉強・実験用
  – セルフホスティング（VPS）
  – Nginx
– あえてサーバー側に Node



dvg.179.jp (2018.2〜)

Sakura VPS

– 勉強・実験用
  – セルフホスティング（VPS）
  – Nginx
– あえてサーバー側に Node

カカカカック (吉田慶章) @kakakakakku・2月24日

Gatsby でブログを構築すると dev.to みたいに爆速になる

（フロントエンド技術すごい） / “Gatsby | dvg - 179”

ブログメンターしていただいてます！

尊敬する Mr. 意識高いエンジニア

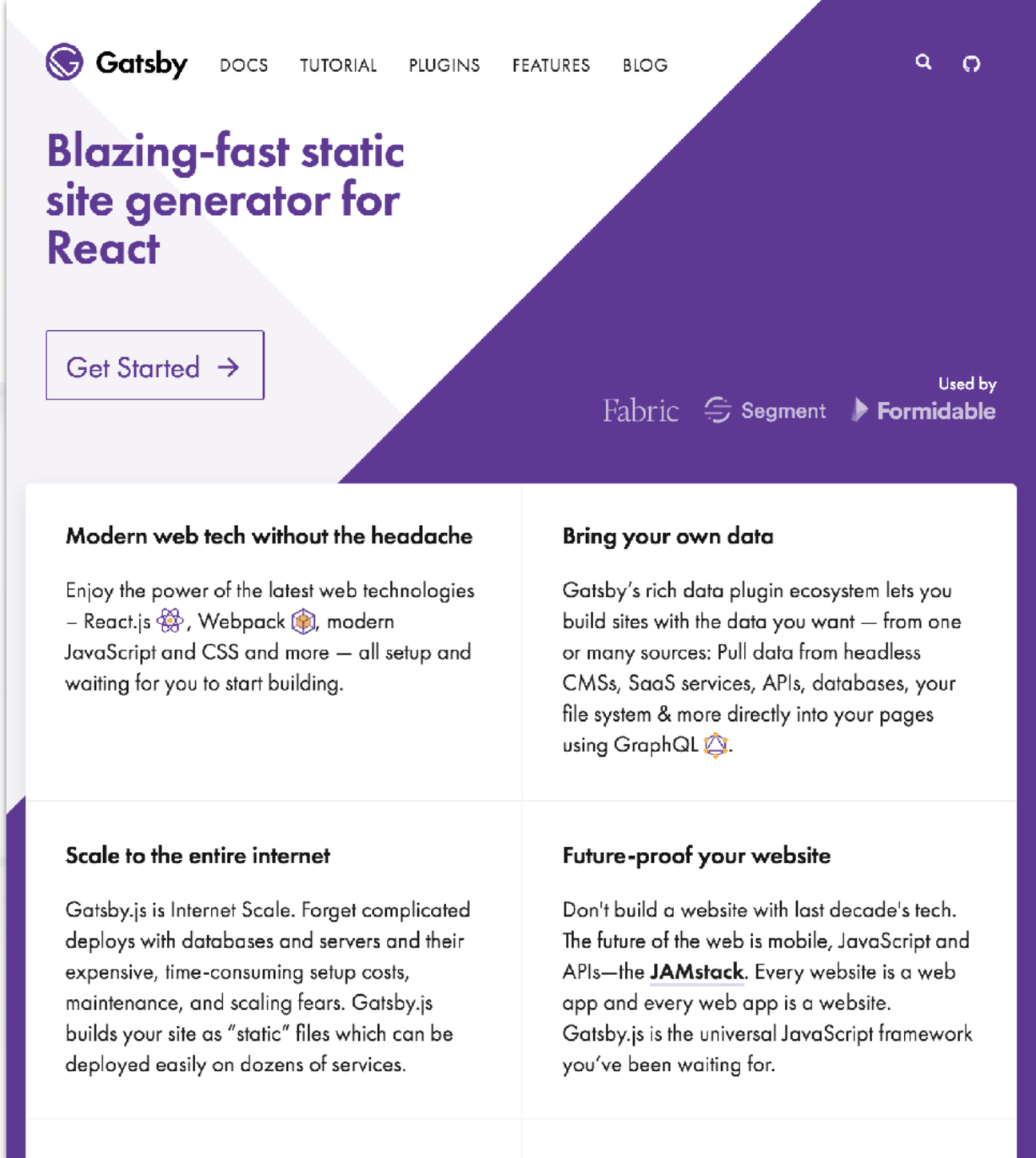
Podcast

omoiyari.fm

#22
Trello があるので眠れない

Gatsbyとは

- **Node** 環境で動く静的ジェネレーター
- テンプレートは **React** による実装
- 表示ページは **SPA** として
  超高速な表示 & ページ遷移
- ほぼ標準で **PWA** としての書き出しに対応
- データソースは **Markdown** をはじめ、
  Wordpress や **JSON** などにも対応



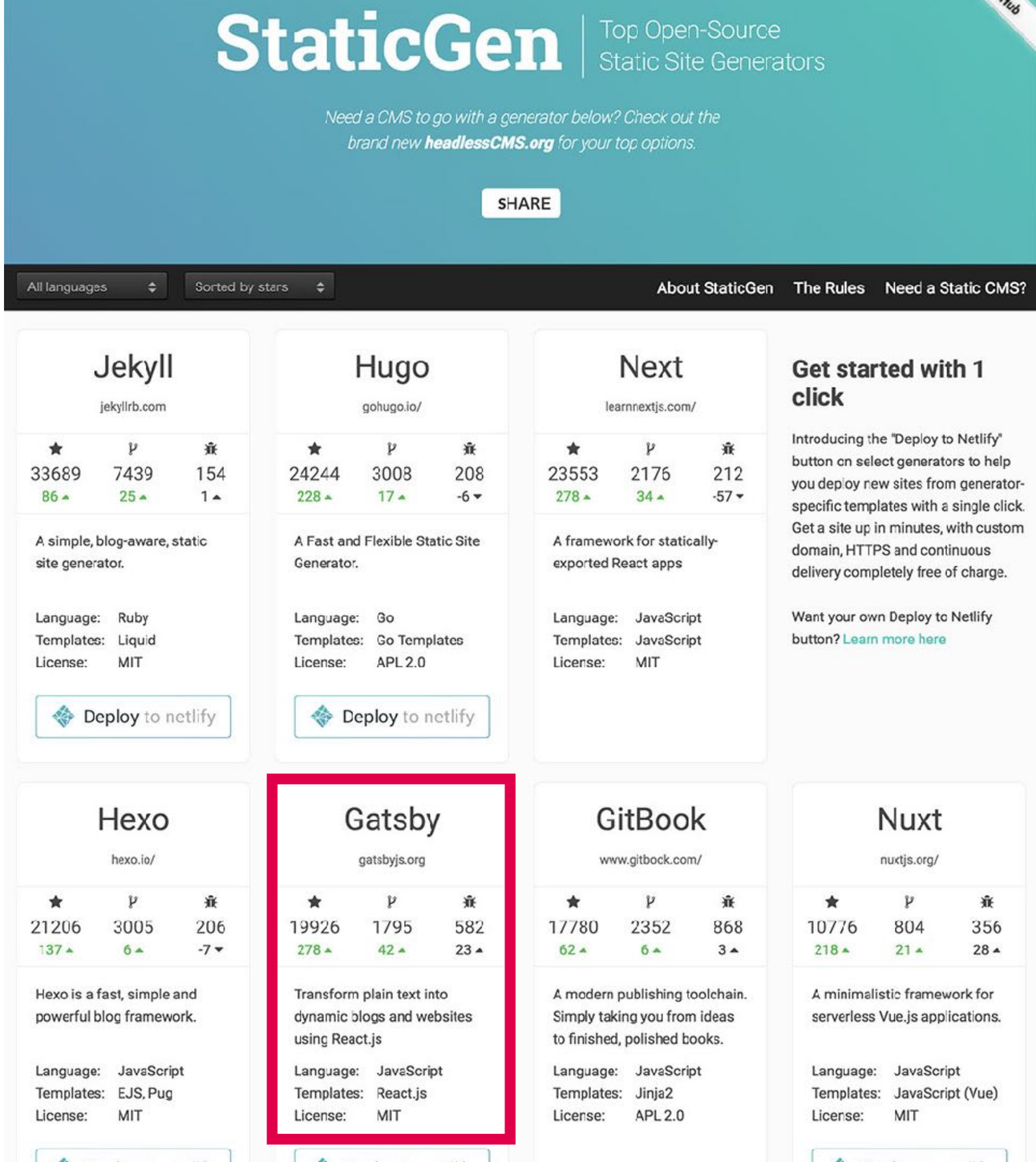
Gatsbyとは

- Author: **Kyle Mathews** @kylemathews

- Github Star: **19,828**
- npm Downloads: **470K/month**
- **v1.1.46** (2018.3)
  - 2015.5 – First Commit
  - 2016.8 – @kylemathews working full-time now on @gatsbyjs!
  - 2017.7 – v1.0.0 リリース

静的ジェネレーターの比較

- フロントエンドの技術で完結するものは少ない
  Jekyll : Ruby
  Hugo : Go
- テンプレートのカスタマイズ性
- 記事は手軽に Markdown で書きたい
- 新しい技術に触れたい



Front-End Developer Handbook 2018

Cody Lindley氏

- 「開眼! JavaScript ―言語仕様から学ぶJavaScriptの本質」（O'Reilly）
- 「jQuery Cookbook」（O'Reilly）
- 「JavaScript Enlightenment」（O'Reilly）

Front-End Developer Handbook 2018

In 2018 expects

1. Nothing will change or slow the usage or popularity of **React** for many years to come.

2. **GraphQL** will replace a lot of REST API's this year.

3. The web will continue to become more native-like with offline capabilities and seamless mobile experiences.

4. HTML 5.3 is coming.

5. Keep an eye on turbo, a blazing fast NPM client.

6. Expect to learn and use CSS transforms 3d, CSS transitions, CSS flexbox, CSS filters, CSS grid

7. JavaScript usage will continue to grow with no slowdown in sight.

8. Still waiting on Web Assembly to peak. This will likely require tooling.

9. Universal/isomorphic JavaScript solutions continue to evolve e.g. next.js and Sapper.

10. Web components still lurk and wait for significant traction from developers.

11. I believe the end is in sight for CSS pre-processors as PostCSS, CSSnext, and CSS in JS take over.

12. Older server centric application patterns show up again but with a new spin. The pendulum could start to swinging away from strick SPA applications. People will begin to pull back on the complexity of single page applications and return to things like pjax (A mix of SPA and Server-side Rendering. See https://stimulusjs.org).

13. Progressive Web Applications hopefully will catch fire. If they don't, I fear they never will.At least not in there current form.

14. "Chatbots created on the basis of artificial intelligence and neural networks will continue to evolve helping to increase communication online. I wonder what it will lead to, but this is unconditional web development trends 2018". Nods.

15. Vue.js usage will likely overtake all Angular usage.

16. AR/AV, AI, and chat bots will continue to evolve and find there sweet spot.

17. JavaScript Symbol and Generators will likely go unnoticed by most front-end developers.

18. More developers will divorce themselves from plain JavaScript and try to marry another.But, just like in marital divorce one always takes most of the same problems with them to the greener grass and little actually changes.Preferences and values just get re-prioritized and history will repeat itself.

19. Webpack 4 will happen, and be better, due to competition!

20. Continued exploration for the ideal CSS solution for a tree of UI components will not cease.

21. State management gets a reset and people start to simplify. Hopefully, this will be the year for solutions like mobx to shine.

Front-End Developer Handbook 2018

In 2018 expects … 要約すると

1. React
2. GraphQL
3. = PWA
4. HTML5.3
5. new NPM Client
6. CSS
7. JS
8. Web Assembly
9. Universal/isomorphic JavaScript
10. Web components
11. CSS
12. SPA + SSR
13. PWA
14. Chatbots / VoiceUI
15. Vue.js
16. AR/AV, AI, and chat bots
17. JavaScript Symbol and Generators
18. JS Poetics
19. Webpack4
20. CSS
21. State management

Front-End Developer Handbook 2018

In 2018 expects

… 要約すると

1. React

2. GraphQL

3. = PWA

4. HTML5.3

5. new NPM Client

6. CSS

7. JS

8. Web Assembly

9. Universal/isomorphic JavaScript

10. Web components

11. CSS

12. SPA + SSR

13. PWA

14. Chatbots / VoiceUI

15. Vue.js

16. AR/AV, AI, and chat bots

17. JavaScript Symbol and Generators

18. JS Poetics

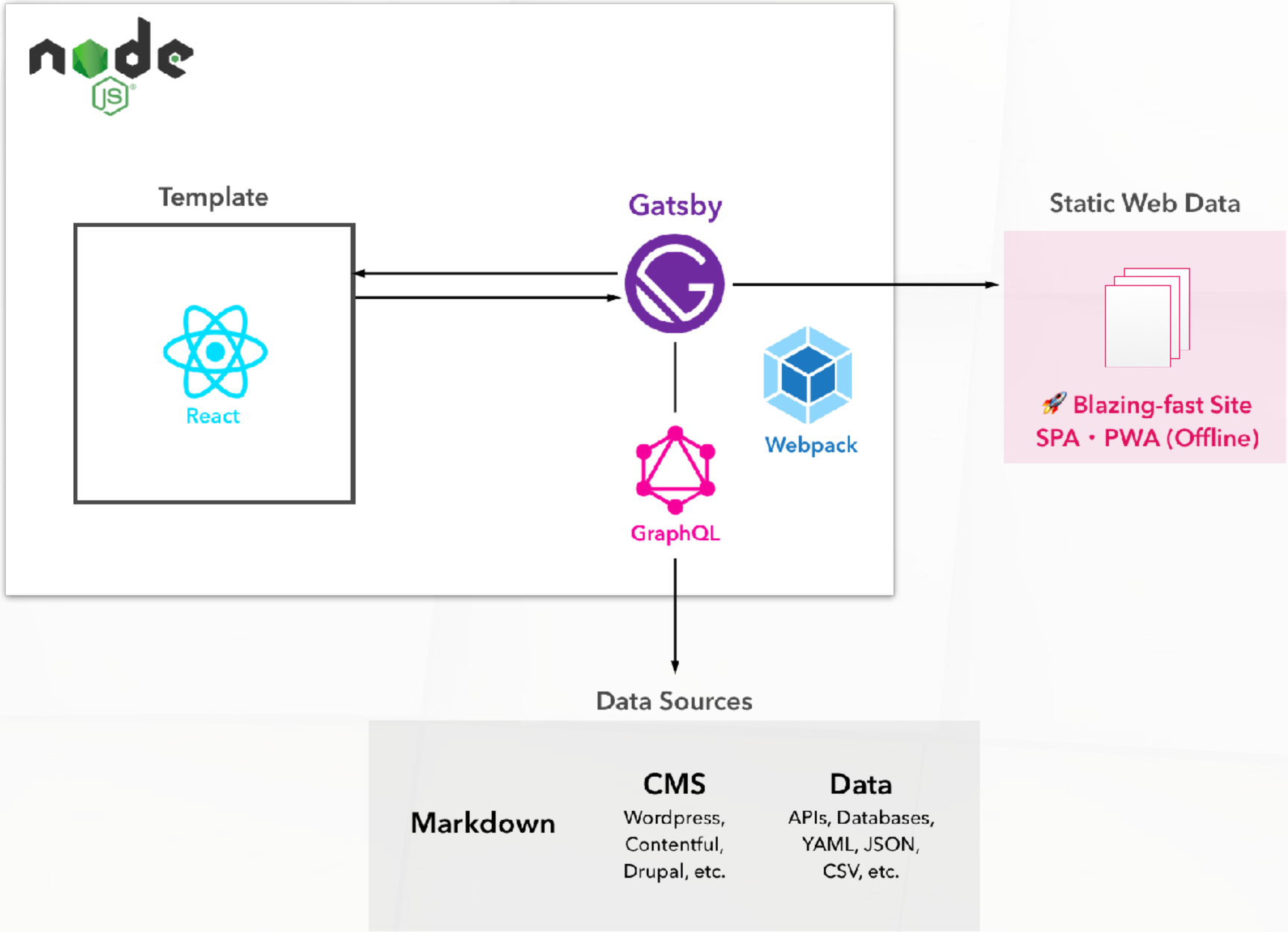
19. Webpack4 ※ v.2

20. CSS

21. State management

Gatsbyを使うメリット

- 旬な技術にまとめて&手軽に触れられる
  - PWA/SPA, GraphQL, HTTP2, Service Worker, etc.
- カスタマイズしやすいテンプレート構造（React）
- 表示ページが爆速な PWA/SPA として出力できる
- データーソースは Markdown や Wordpress, JSON など色々な方法が選択可能


node
JS
Template
React
Gatsby
Webpack
GraphQL
Static Web Data
🚀 Blazing-fast Site
SPA • PWA (Offline)
Data Sources
Markdown
CMS
Wordpress,
Contentful,
Drupal, etc.
Data
APIs, Databases,
YAML, JSON,
CSV, etc.

Gatsbyが向く用途

- Blog
- ドキュメント ← PWA にしてオフライン対応できれば 👩
- ポートフォリオ ← 爆速なポートフォリオならイメージアップ？？
- 静的ページだけのサイト（キャンペーンサイト / LP など）

※ 静的ジェネレーターなので、大規模なサイトや動的なページには向かない

Gatsby周辺

- **Starter ＝ テンプレート + α**
  - Official: 3種類　Community: 30種類以上
- **豊富なプラグイン + React 資産**
- **わかりやすいドキュメント（英語） + チュートリアル（7章）**
- **日本語の情報は少ない**
  - Qiita: 11本　Gatsby製ブログ: 数本

Get Started

Gatsby is a blazing-fast static site generator for React.

For more detailed step-by-step instructions on getting started, see the **tutorial**.

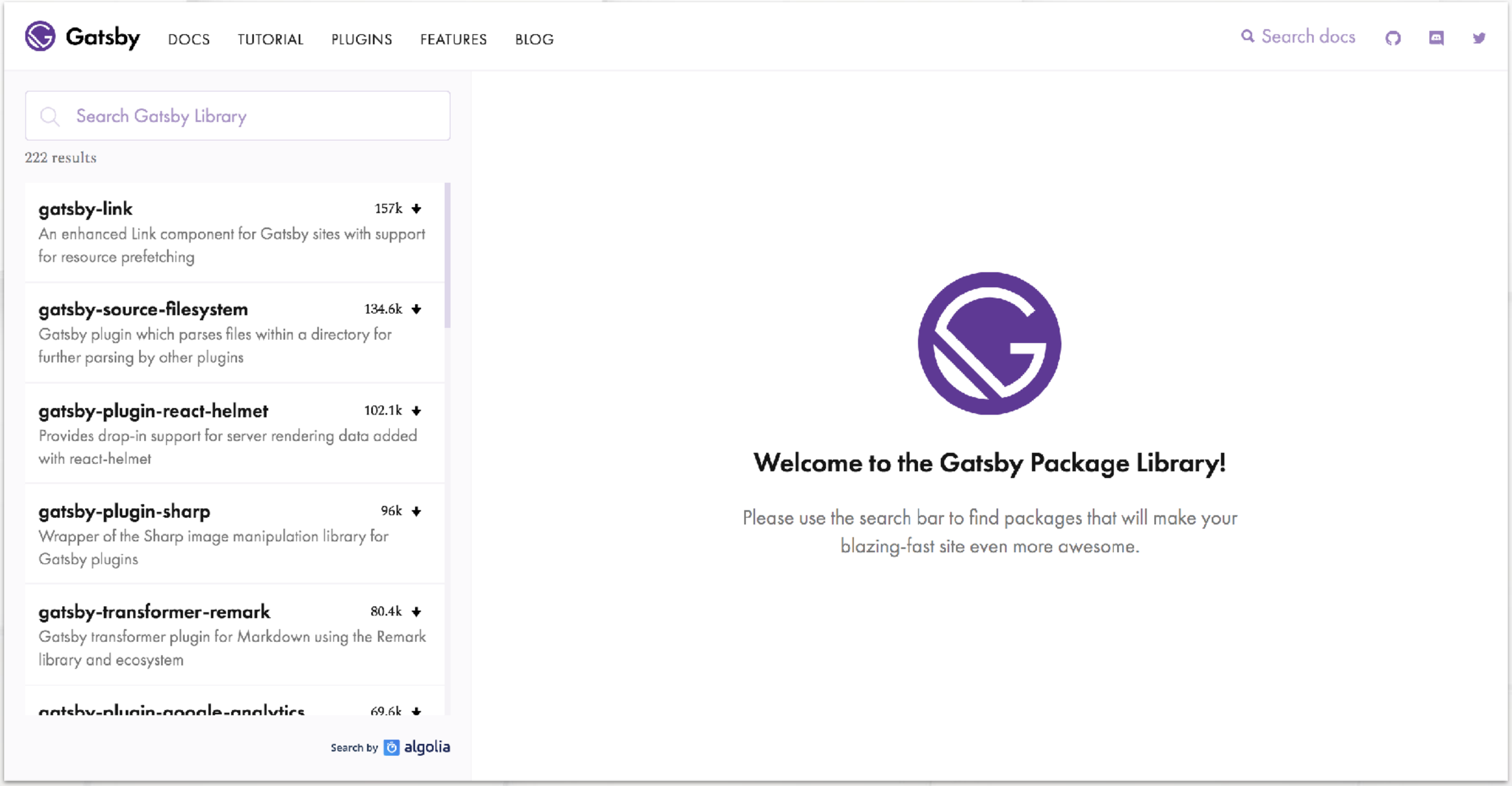
Install Gatsby's command line tool

```
npm install --global gatsby-cli
```

Using the Gatsby CLI

1. Create a new site. `gatsby new gatsby-site`
2. `cd gatsby-site`
3. `gatsby develop` — Gatsby will start a hot-reloading development environment accessible at `localhost:8000`
4. Try editing the javascript pages in `src/pages`. Saved changes will live reload in the browser.
5. `gatsby build` — Gatsby will perform an optimized production build for your site generating static HTML and per-route JavaScript code bundles.
6. `gatsby serve` — Gatsby starts a local HTML server for testing your built

Gatsby Document



Gatsby Plugins

Gatsbyクイックスタート

1. npm で Gatsby をインストール `npm install --global gatsby-cli`

2. Gatsby のプロジェクトを新規作成する `gatsby new [DIR] [URL_OF_STARTER]`

   ※ Starterのテンプレート部分は後から変更は可能だが、簡単には変えられない

3. 必要な設定を行う（gatsby-config.json などでサイト名の変更など）

4. テスト `gatsby develop`

5. ビルド `gatsby build`

6. 書き出された public/ ディレクトリ以下を deploy ！

“ Gatsby is an early example of a web site compiler ”

Gatsby は web サイトコンパイラーの初期モデルです

“ I designed Gatsby with the goal that when using it,
it'd really really hard to be a slow site ”

Gatsby を使用すると、遅いサイトを作ることがすんげー難しくなるように Gatsby を設計しました。

– Kyle Mathews / Gatsby Blog

Gatsbyの詳細

Gatsbyの主要なAPI

```
gatsby new [SITE_DIRECTORY] [URL_OF_STARTER_GITHUB_REPO]
```

– Gatsbyの新規サイトを指定のStarterで作成する

```
gatsby build
```

– 現在のサイトで静的ファイルを生成する

```
gatsby develop
```

– 現在のサイトで 開発サーバーを立ち上げる

  – Hot Reload（JSファイルだけでなく Markdown ファイルも）

  – GraphiQL（GraphQL IDE）

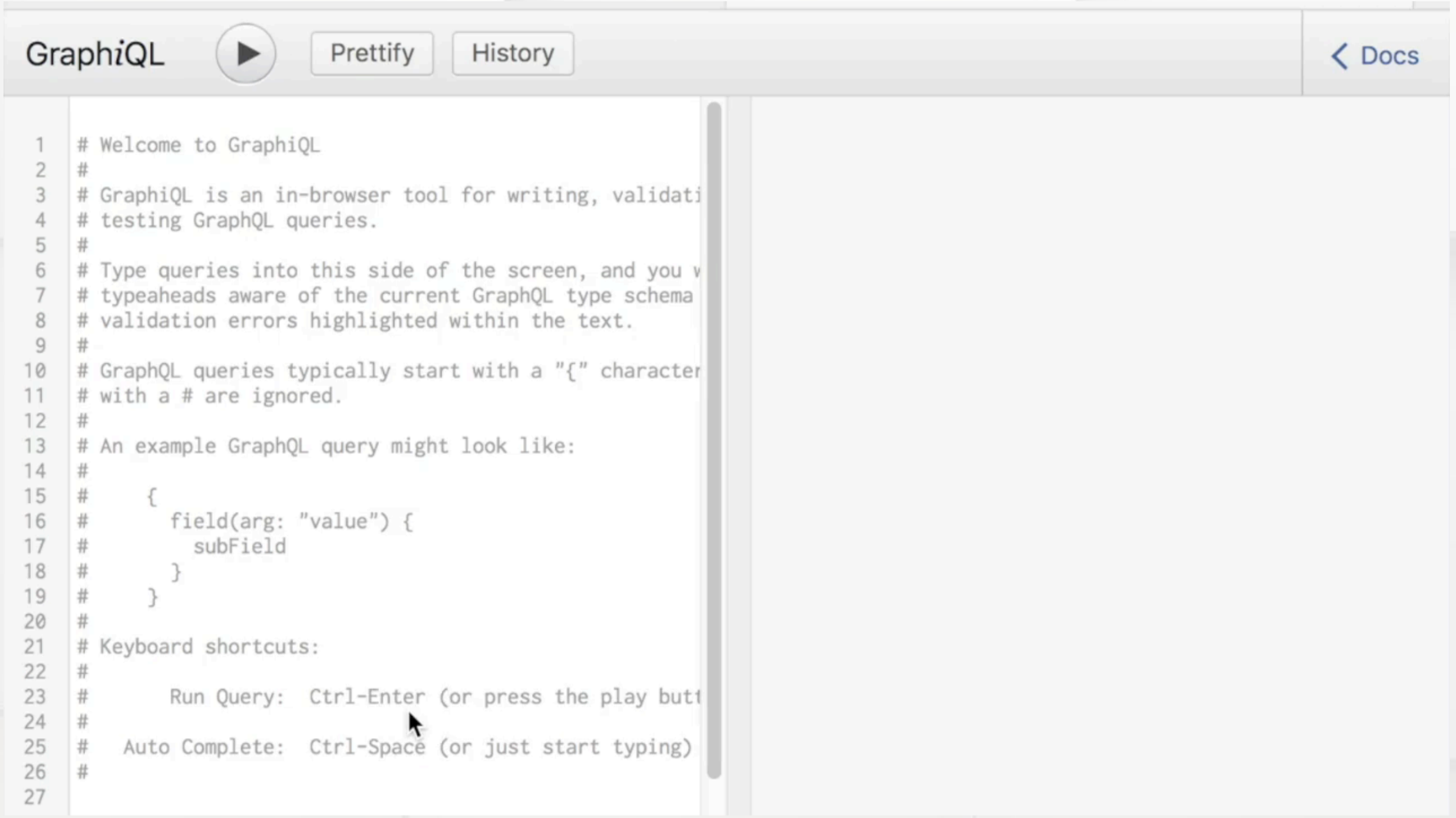
gatsby build - 28sec

```
MacBookPro:179-dvg nnjyami$ gatsby build
success delete html and css files from previous builds — 0.096 s
success open and validate gatsby-config.js — 0.012 s
success copy gatsby files — 0.017 s
success onPreBootstrap — 1.157 s
success source and transform nodes — 0.300 s
success building schema — 0.518 s
success createLayouts — 0.010 s
success createPages — 0.078 s
success createPagesStatefully — 0.023 s
success onPreExtractQueries — 0.001 s
success update schema — 0.274 s
success extract queries from components — 0.099 s
success run graphql queries — 0.175 s
success write out page data — 0.006 s
success write out redirect data — 0.001 s
success onPostBootstrap — 0.001 s

info bootstrap finished - 6.225 s

success Building CSS — 5.553 s
success Building production JavaScript bundles — 11.979 s
success Building static HTML for pages — 4.492 s
Total precache size is about 288 kB for 5 resources.
info Done building in 28.316 sec
```

gatsby develop - 8.5sec

```
cBookPro:179-dvg nnjyami$ gatsby develop
delete html and css files from previous builds — 0.088 s
success open and validate gatsby-config.js — 0.010 s
info One or more of your plugins have changed since the last time you ran Gatsby. As
a precaution, we're deleting your site's cache to ensure there's not any stale
data
success copy gatsby files — 0.019 s
success onPreBootstrap — 1.144 s
success source and transform nodes — 0.567 s
success building schema — 0.535 s
success createLayouts — 0.010 s
success createPages — 0.075 s
success createPagesStatefully — 0.030 s
success onPreExtractQueries — 0.001 s
success update schema — 0.250 s
success extract queries from components — 0.089 s
success run graphql queries — 1.787 s
success write out page data — 0.004 s
success write out redirect data — 0.001 s
success onPostBootstrap — 0.001 s

info bootstrap finished - 8.555 s

 DONE  Compiled successfully in 4434ms


You can now view gatsby-starter-blog-no-styles in the browser.

  http://localhost:8000/

View GraphiQL, an in-browser IDE, to explore your site's data and schema

  http://localhost:8000/___graphql

Note that the development build is not optimized.
To create a production build, use gatsby build

info changed file at /Users/nnjyami/Data/dvg/179-dvg/src/pages/201803-skills-of-manager/index.md
 WAIT  Compiling...

 DONE  Compiled successfully in 729ms

 WAIT  Compiling...
```

Gatsby Build & Develop

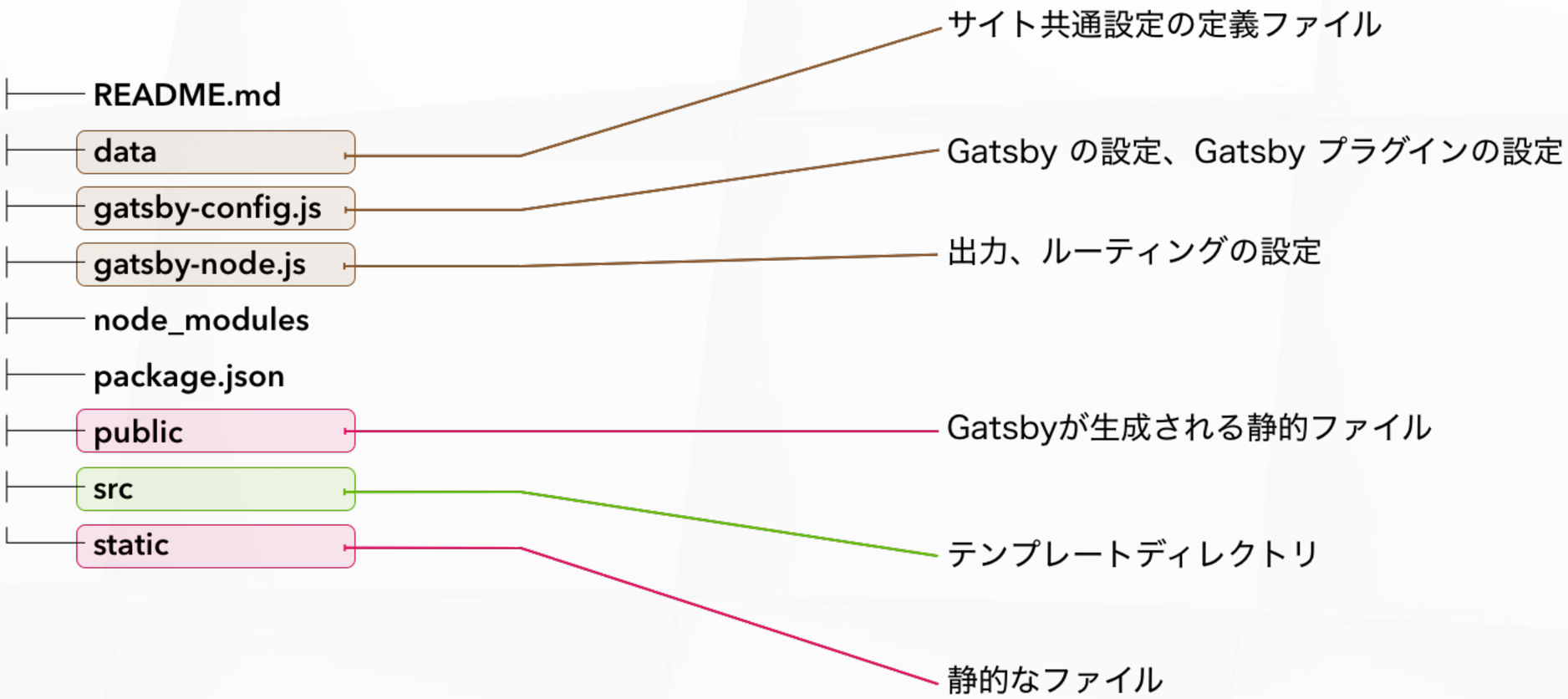


GraphiQL – gatsby develop 時に起動する（http://localhost:8000/___graphql）

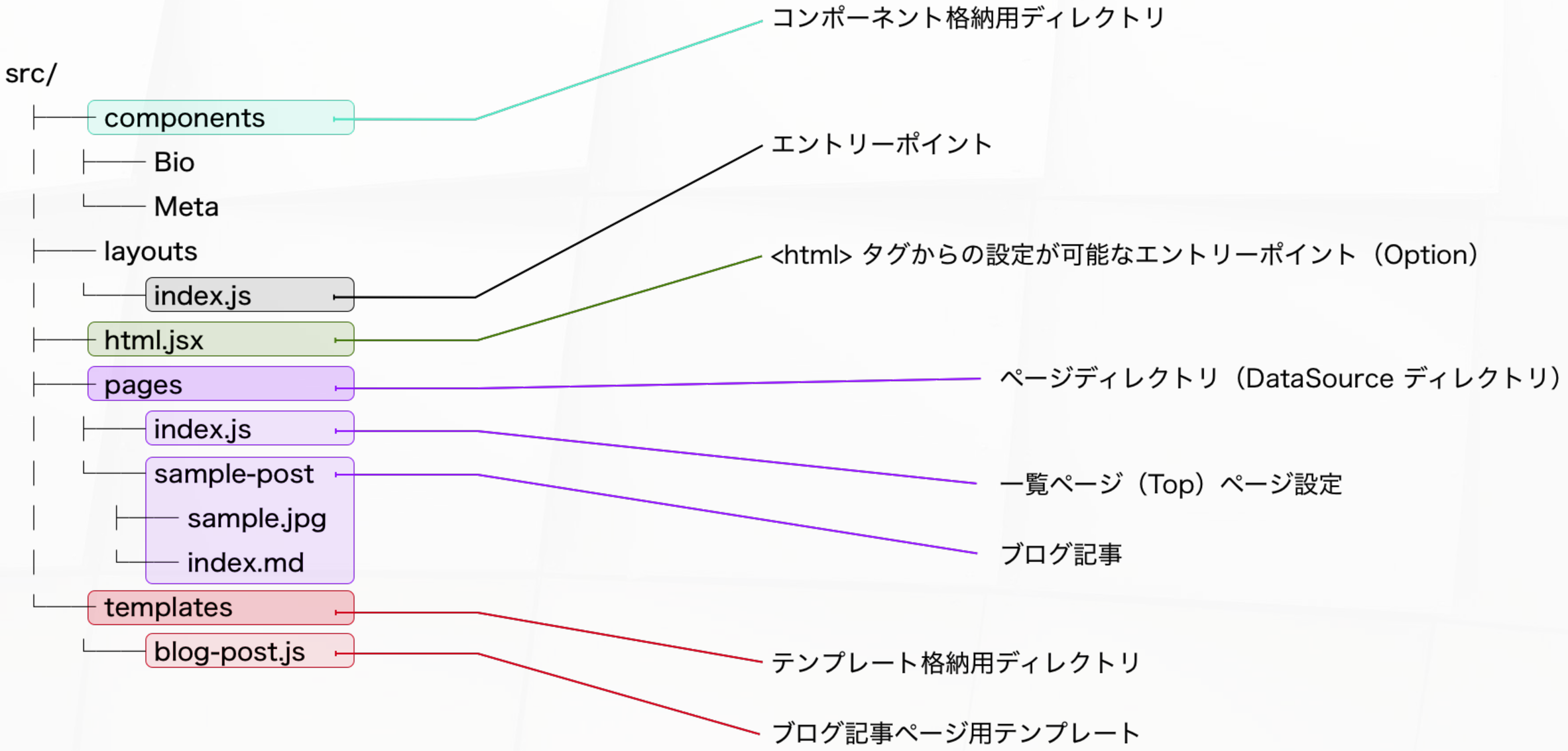
Gatsbyの構成要素

1. Starter & テンプレート
2. ジェネレーター
   - GraphQL をインターフェースにデータを取得
   - React のテンプレートから静的ファイルを書き出す
   - パフォーマンスに最適化した書き出し
3. 表示

1. Starter & テンプレート



1. Starter & テンプレート



src/templates/blog-post.js

```
import React from 'react'
import Helmet from 'react-helmet'
import Link from 'gatsby-link'
import get from 'lodash/get'

import Bio from '../components/Bio'

class BlogPostTemplate extends React.Component {
  render() {
    const post = this.props.data.markdownRemark
    const siteTitle = get(this.props, 'data.site.siteMetadata.title')

    return (
      <div>
        <Helmet title={`${post.frontmatter.title} | ${siteTitle}`} />
        <h1>{post.frontmatter.title}</h1>
        <p>
          {post.frontmatter.date}
        </p>
        <div dangerouslySetInnerHTML={{ __html: post.html }} />
        <hr />
        <Bio />
      </div>
    )
  }
}

export default BlogPostTemplate

export const pageQuery = graphql`
  query BlogPostByPath($path: String!) {
    site {
      siteMetadata {
        title
        author
      }
    }
    markdownRemark(frontmatter: { path: { eq: $path } }) {
      id
      html
      frontmatter {
        title
        date(formatString: "MMMM DD, YYYY")
      }
    }
  }
```

GraphQLで指定したデータが
Props として渡される

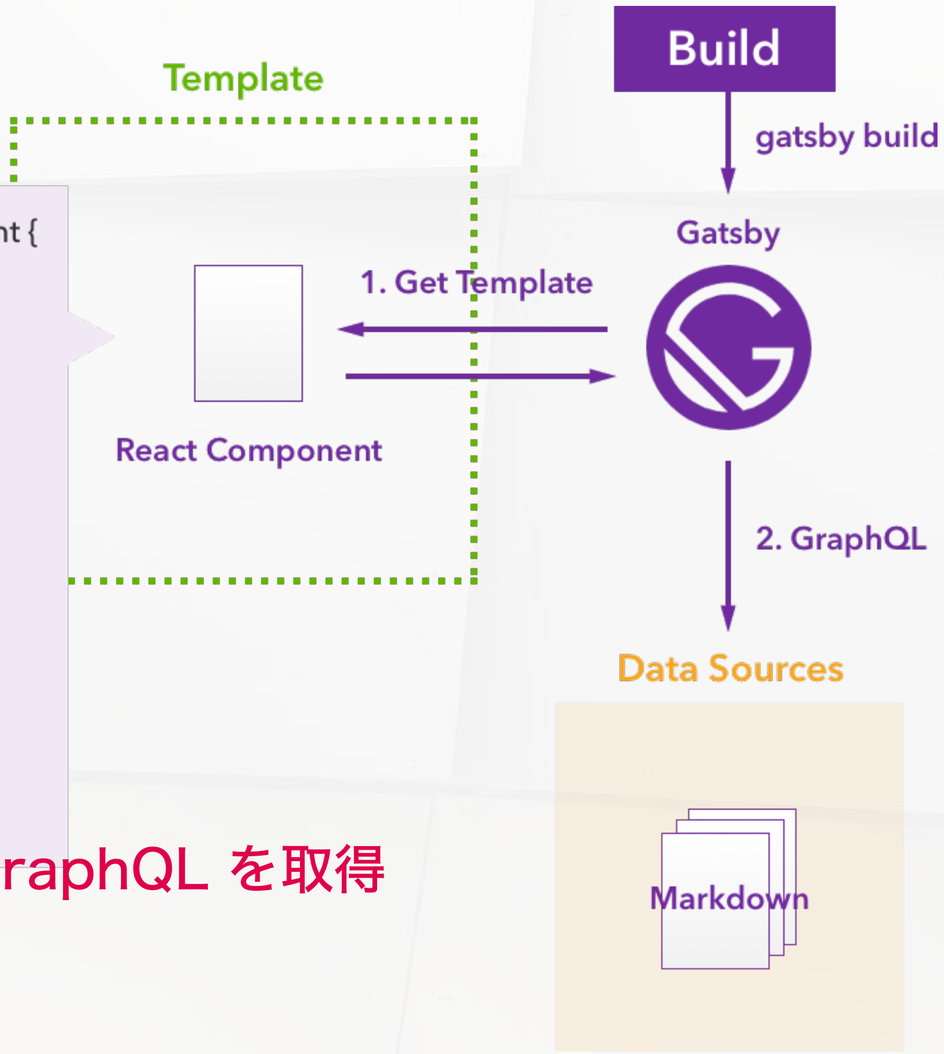
GraphQL による
取得データの指定

2. ジェネレーター

テンプレートに記述した GraphQL を
取得して Data Source からデータを取得

```
export default class BlogPostTemplate extends React.Component {
 render() {
  return (
   <article>
    ...
   </article>
  )
 }
}

export const pageQuery = graphql`
  ...
`
```

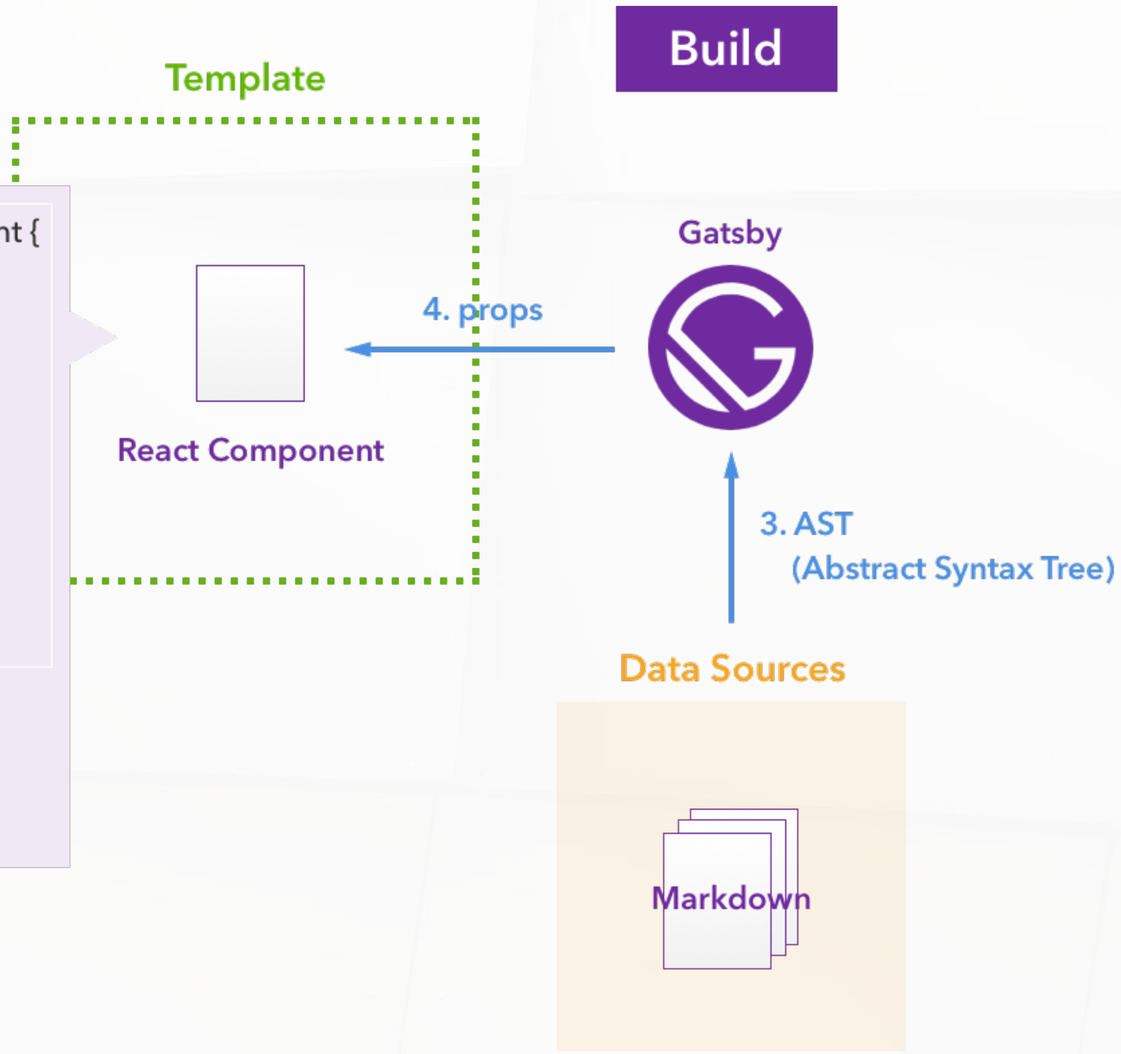
テンプレート内の GraphQL を取得



2. ジェネレーター

Data Source から取得したデータを AST（JSON）に変換し
React のコンポーネントに props として渡す

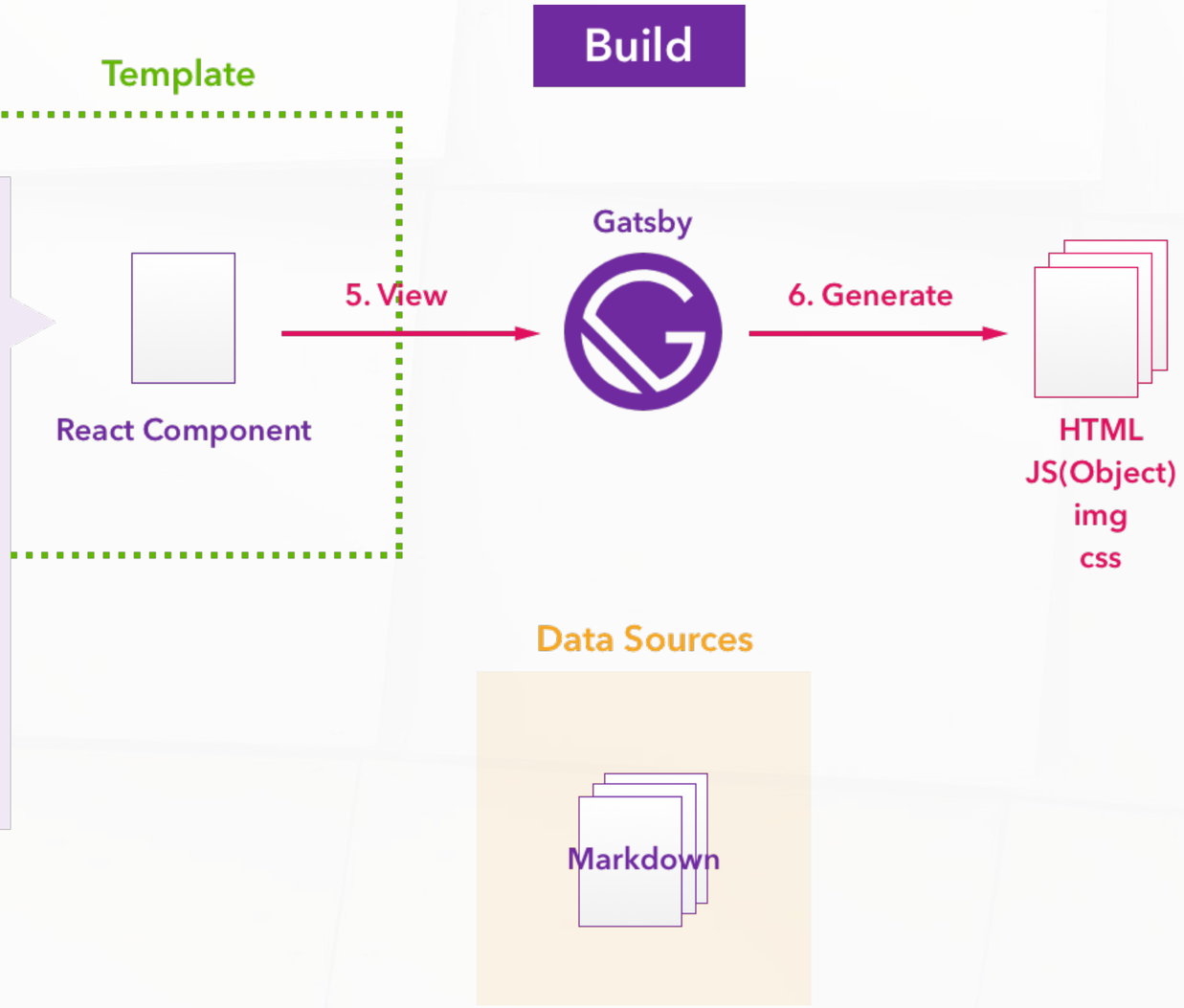
```
export default class BlogPostTemplate extends React.Component {
 render() {
  return (
   <article>
     ...
   </article>
  )
 }
}

export const pageQuery = graphql`
  ...
`
```



2. ジェネレーター

HTML ファイルや JS として
パフォーマンスに考慮した形で書き出す
また必要な画像やCSSも合わせて書き出し

```
export default class BlogPostTemplate extends React.Component {
 render() {
  return (
   <article>
     ...
   </article>
  )
 }
}

export const pageQuery = graphql`
  ...
`
```



3. 表示

- **Blazing-fast / 猛烈な速さ & パフォーマンスの追求**
  - CSS インライン展開
  - 投機的な先読み　<link rel=“preload” />
  - Service Worker (キャッシュコントロール・オフライン対応)
  - JS のファイル分割（Webpack）
  - PRPL Pattern
  - HTTP2 / push
- **SPA によるスムーズなページ遷移**
- **ほぼ標準で PWA 化が可能**

3. 表示

– GatsbyのHTMLを見る

投機的な先読み

下層コンテンツなど

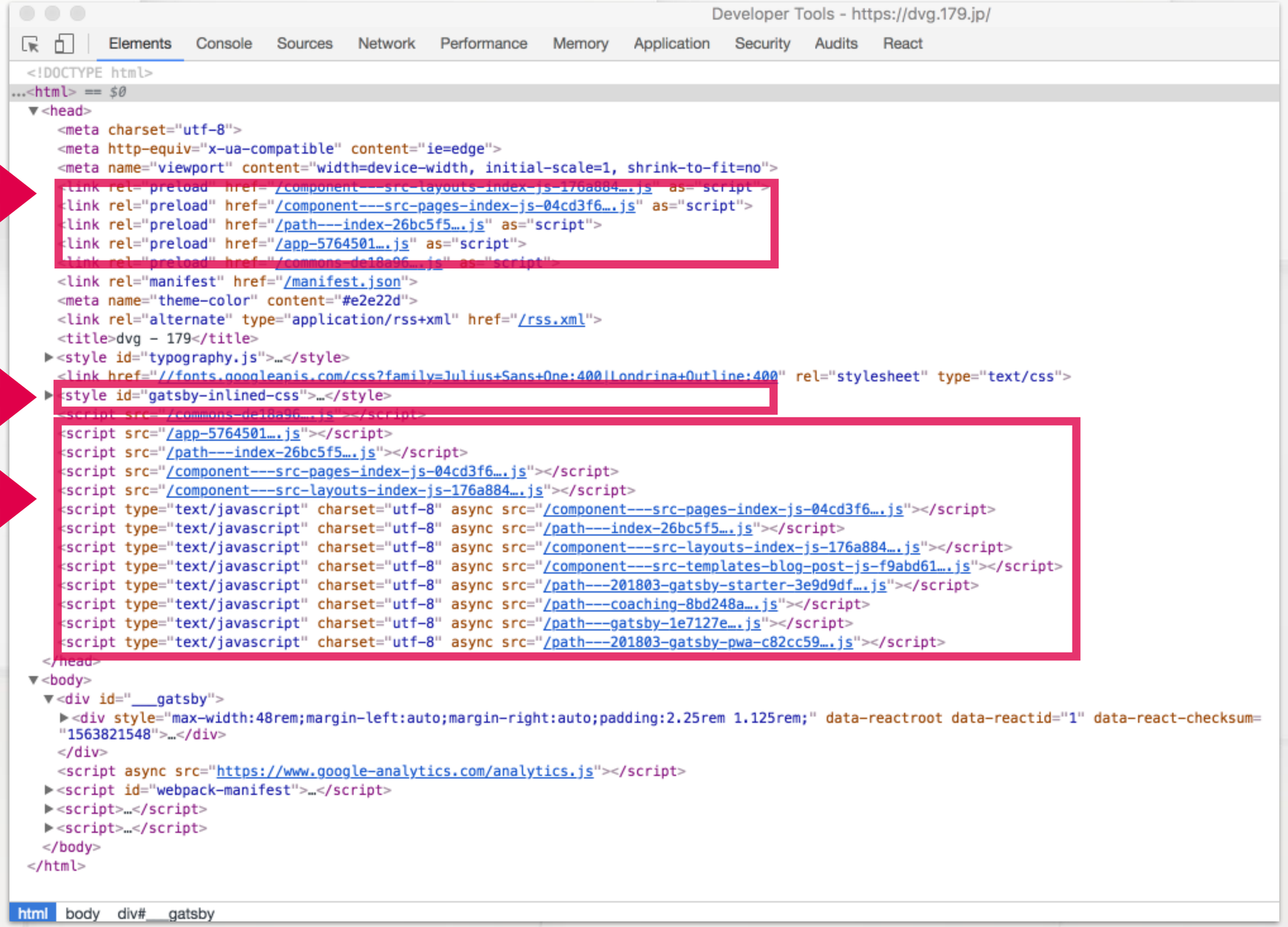
Webpack JSONP

インラインに CSS 展開

JSのファイル分割

最初に必要なJSと

後からでいいJSを分類



PRPL Pattern

Push

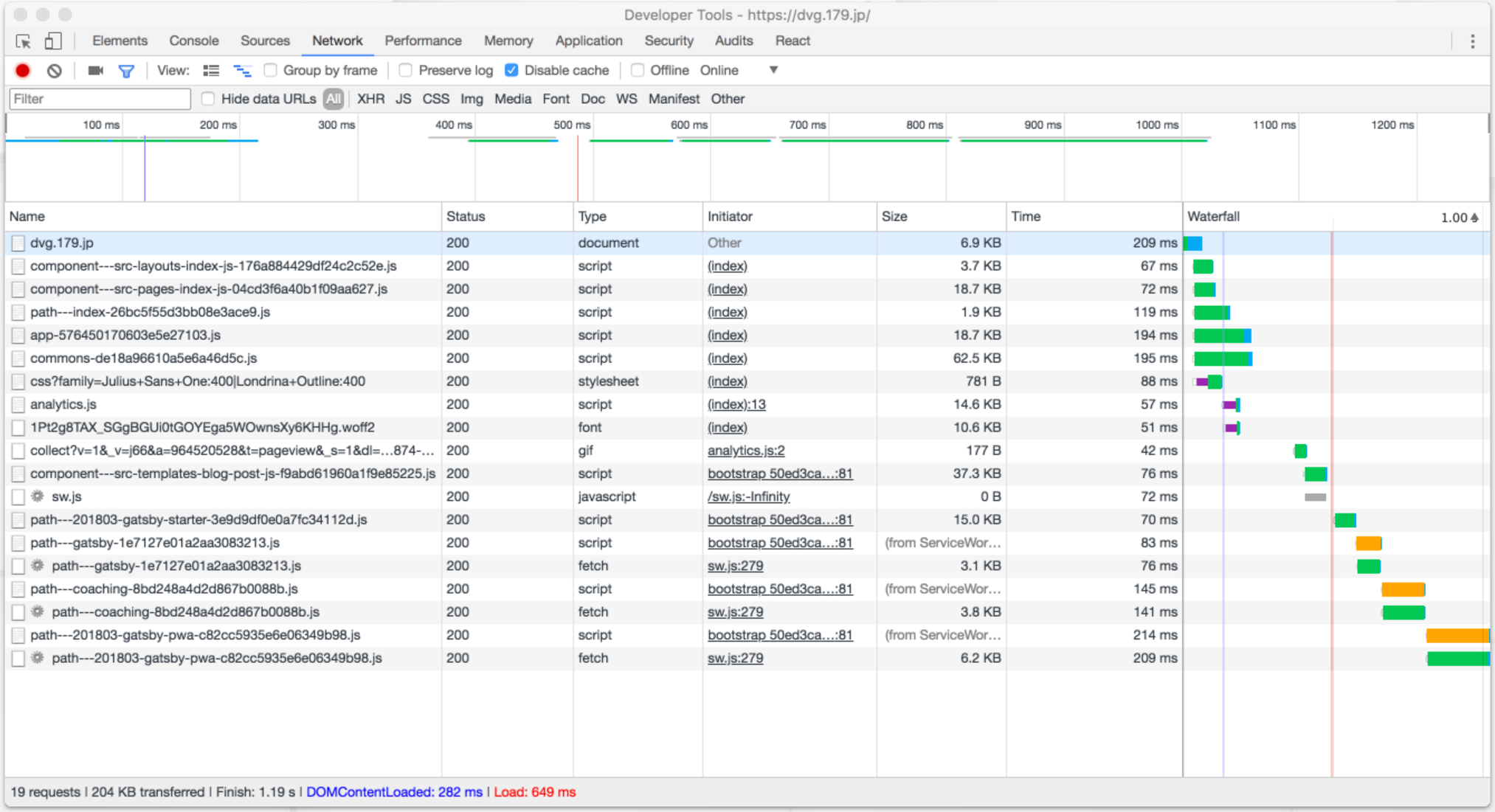
Render

Pre-cache

Lazy-load

3. 表示

- Network（ServiceWorkerインストール前）

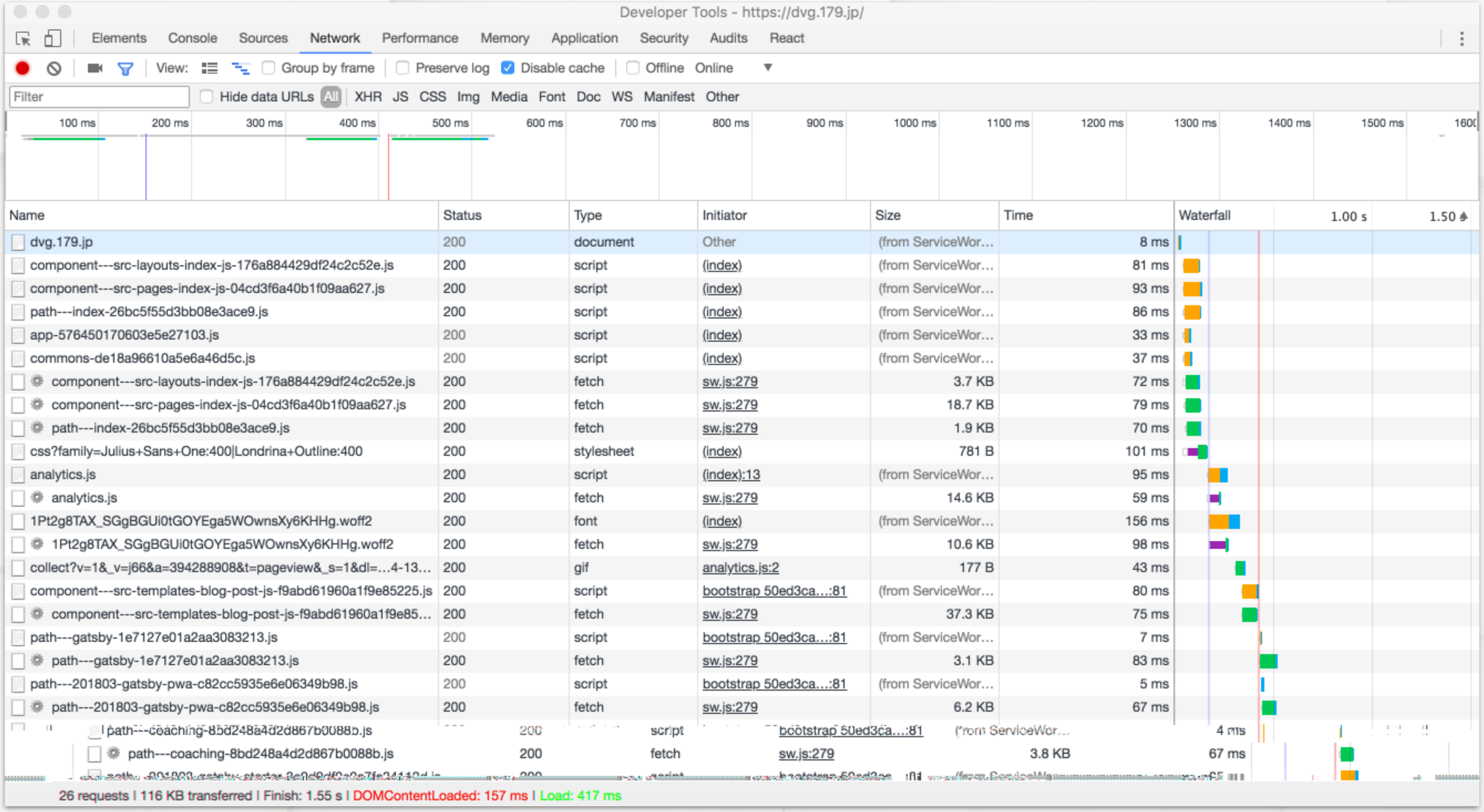


DOMContentLoaded
282ms

Load
649ms

3. 表示

- Network（ServiceWorkerインストール後）



DOMContentLoaded
157ms

Load
417ms

3. 表示

- Network vs http://abehiroshi.la.coocan.jp/

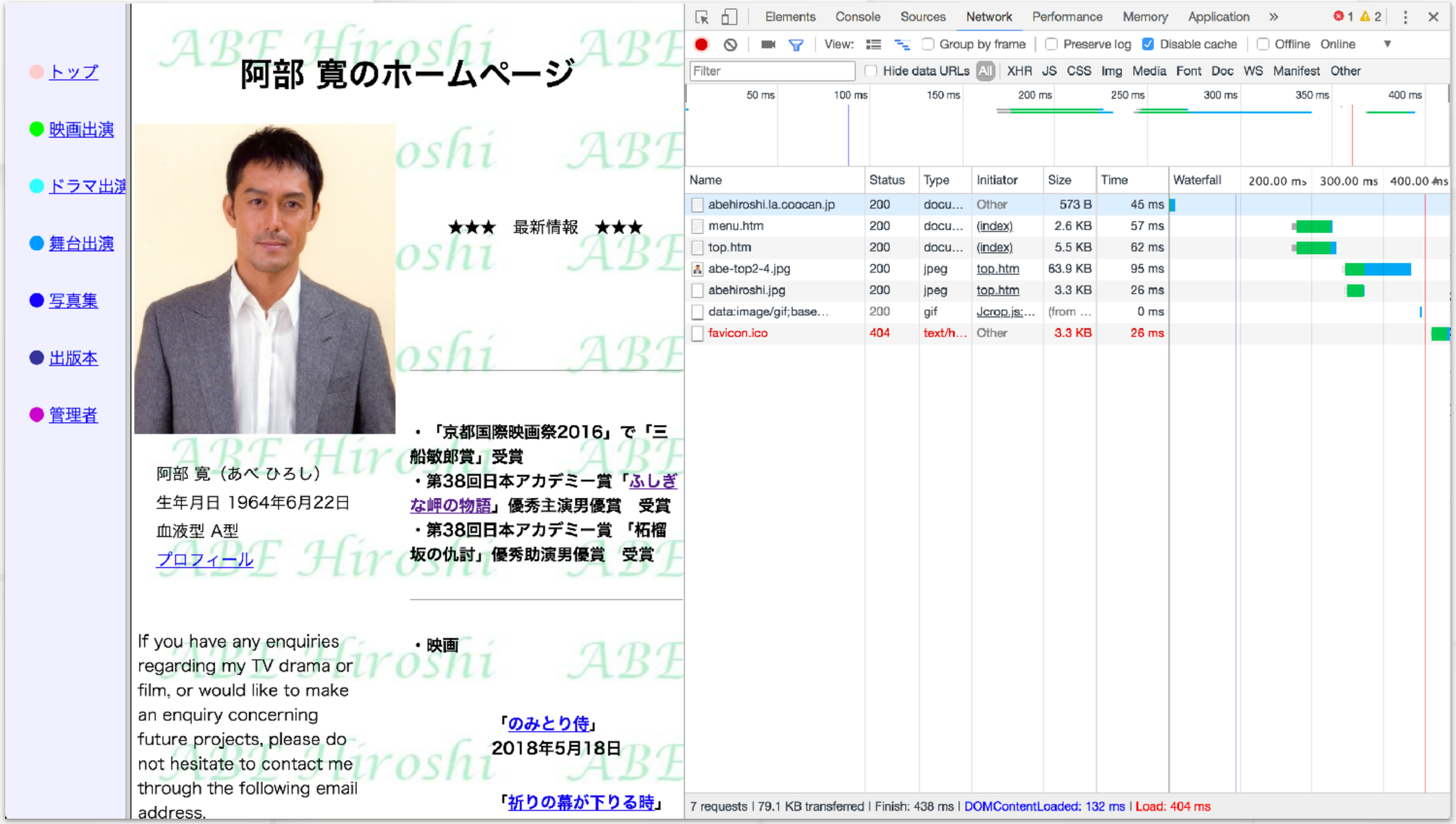
DOMContentLoaded
157ms

Load
417ms

VS

DOMContentLoaded
132ms

Load
404ms



3. 表示

– Auditsによる評価

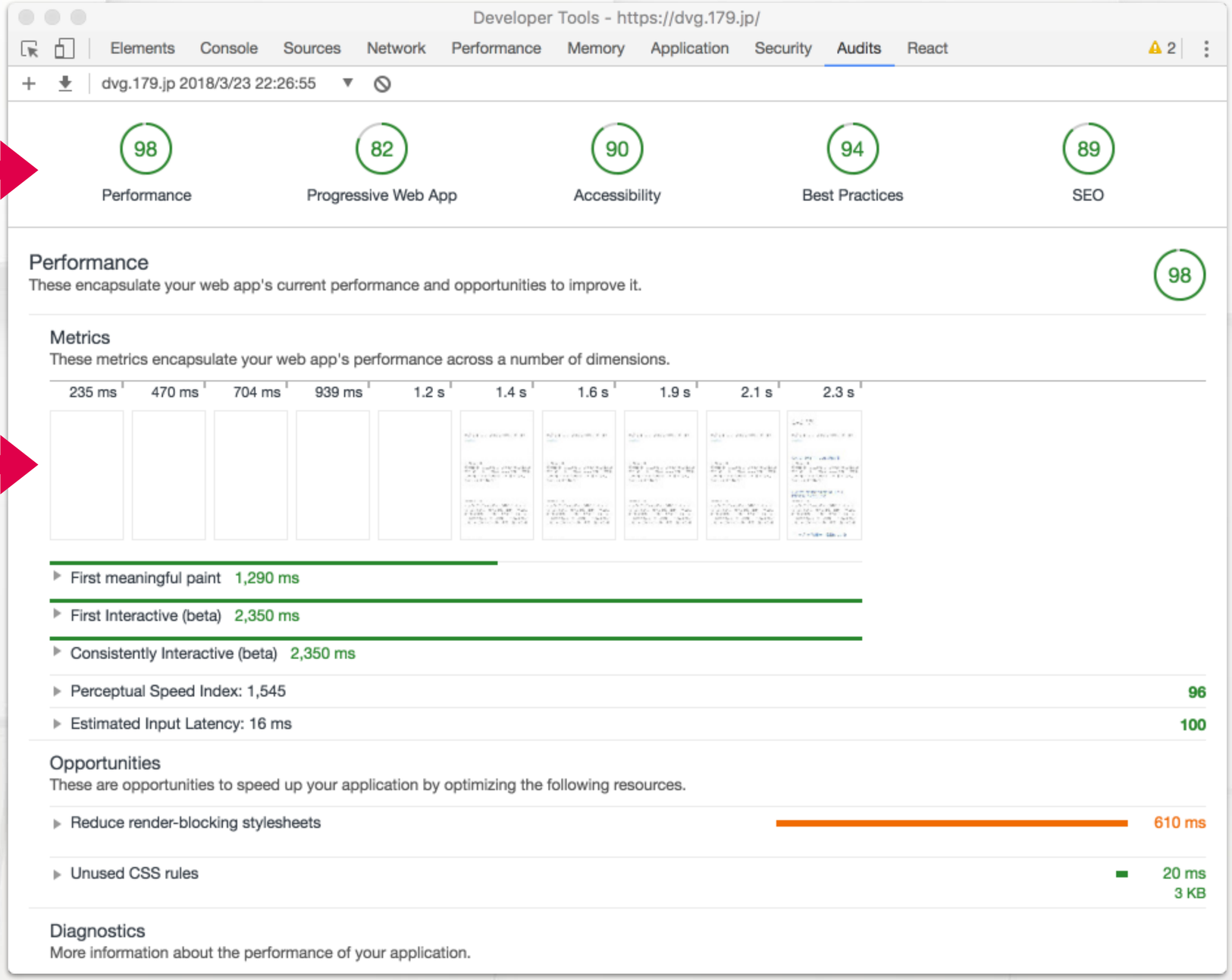
標準状態で高スコア

※ PWAのみプラグイン追加

少しの改善で満点が目指せそう

– Google Web Fonts の設定

– First meaningful paint



Gatsby
Super Fast

まとめ

- Gatsbyを使うと
  爆速サイト＆ブログが簡単にできます！

- 今日は詳細な仕組みを説明したけど
  30分あれば簡単にできちゃいます！！

- Gatsbyは旬の技術テンコ盛りなので
  フロントエンド勉強の題材としても最高

Gatsby Super Fast!!!