Fast sites with Gatsby.js

18 giugno, 19:00

MUG

ospitato da Yalp

Via Tommaso da Modena, 8
TREVISO - tel 0422.541865

Matteo Granzotto
Frontend Dev
@Blundert

🍍 Matteo Granzotto 🌴

Frontend Developer and Founder at **Wavelop**.
Basketball, music, comics and retrogaming lovers.

- **WEB APP: front end, back and, devops**
- **MOBILE APP: native, hybrid**

Gatsby

Gatsby is a free and open source **framework** based on **React** that helps developers build blazing fast websites and apps

React.js, Webpack, modern JavaScript and CSS

GraphQL

Gatsby.js builds your site as “static” files

JAMSTACK

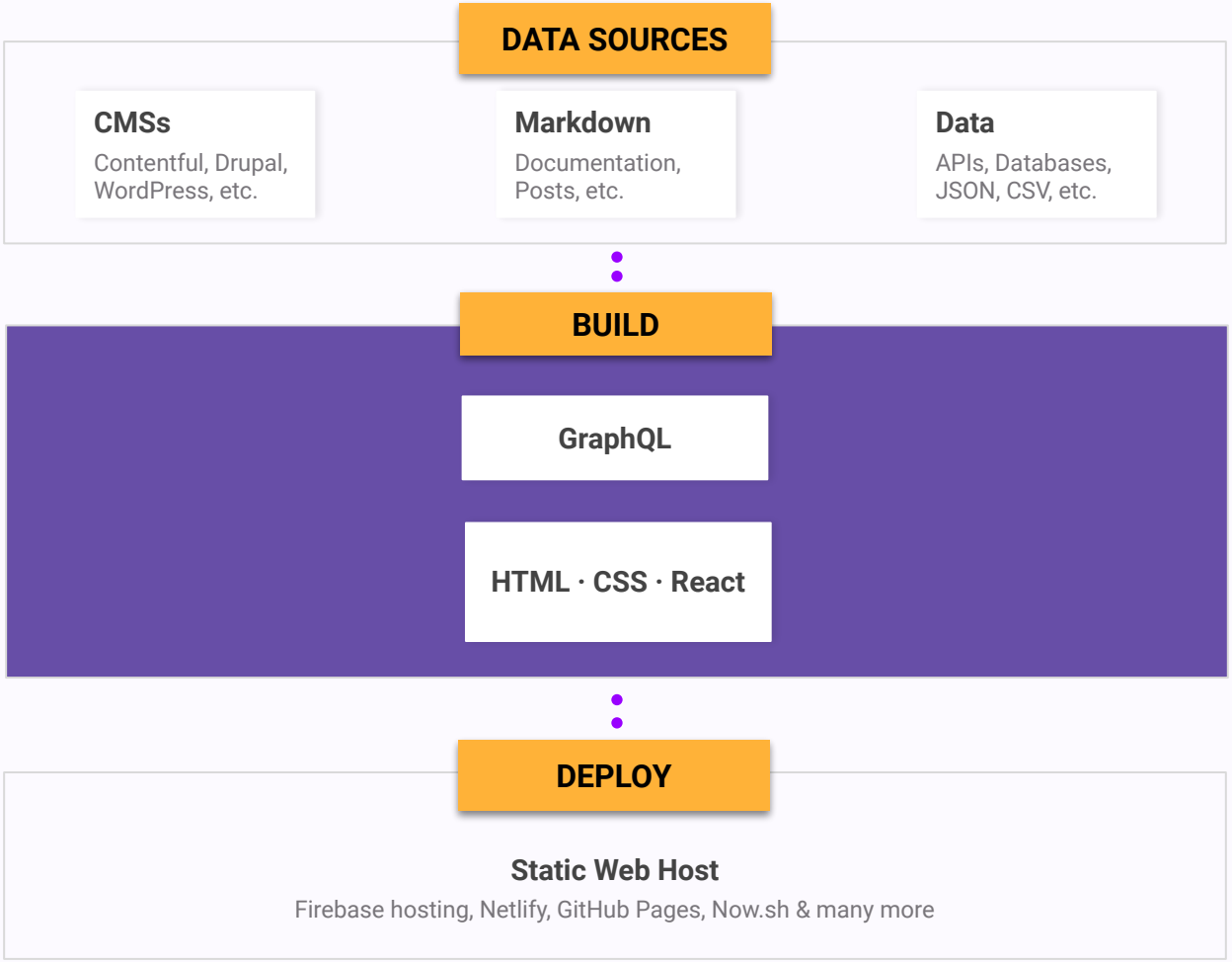
Every website is a web app and every web app is a website - JAMstack

PWA

Gatsby.js is a static PWA (Progressive Web App) generator

```
<link rel="prefetch" href="..">
```

Gatsby prefetches resources for other pages so clicking around the site feels incredibly fast

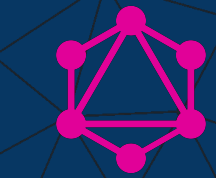

DATA SOURCES
CMSs
Contentful, Drupal, WordPress, etc.
Markdown
Documentation, Posts, etc.
Data
APIs, Databases, JSON, CSV, etc.
BUILD
GraphQL
HTML · CSS · React
DEPLOY
Static Web Host
Firebase hosting, Netlify, GitHub Pages, Now.sh & many more

JAMSTACK

JavaScript
Any dynamic programming during the request/response cycle is handled by JavaScript, running entirely on the client.

APIs
All server-side processes or database actions are abstracted into reusable APIs, accessed over HTTPS with JavaScript.

Markup
Templated markup should be prebuilt at deploy time, usually using a site generator for content sites, or a build tool for web apps.



GraphQL

Describe your data

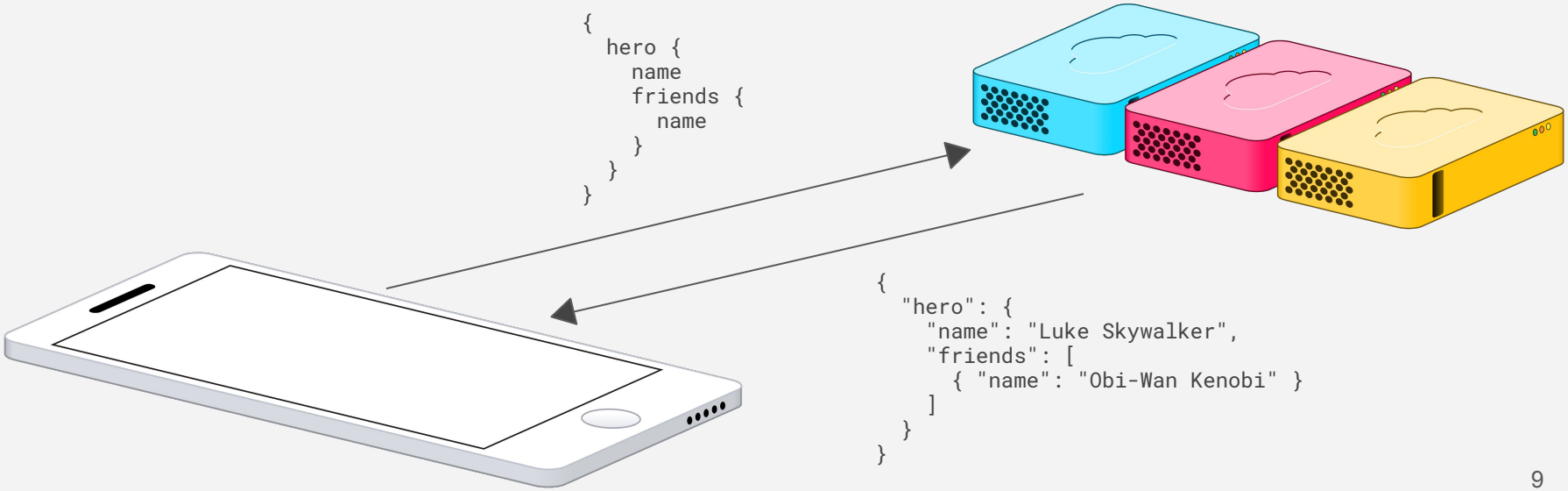
```
type Project {
  name: String
  tagline: String
  contributors: [User]
}
```

Ask for what you want

```
{
  project(name: "GraphQL") {
    tagline
  }
}
```

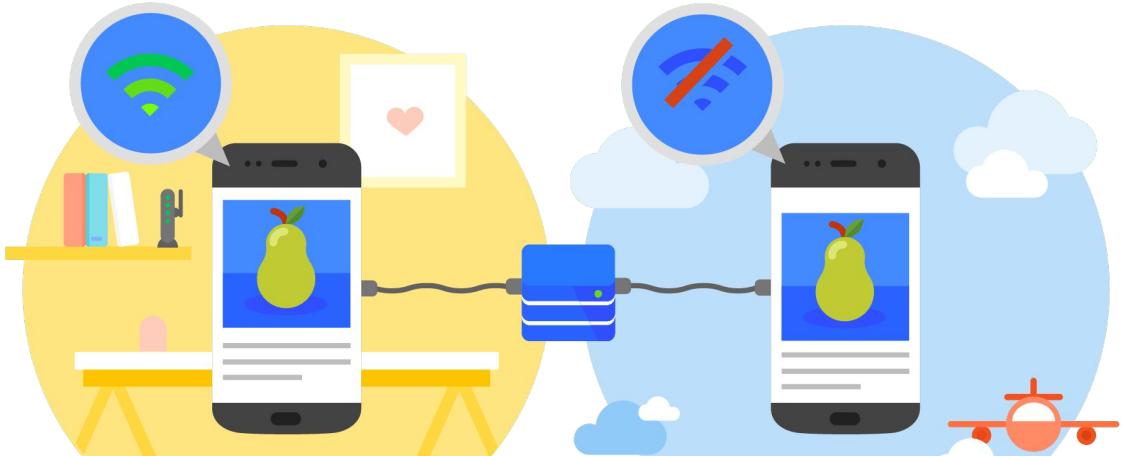
Get predictable results

```
{
  "project": {
    "tagline": "A query
               language for APIs"
  }
}
```

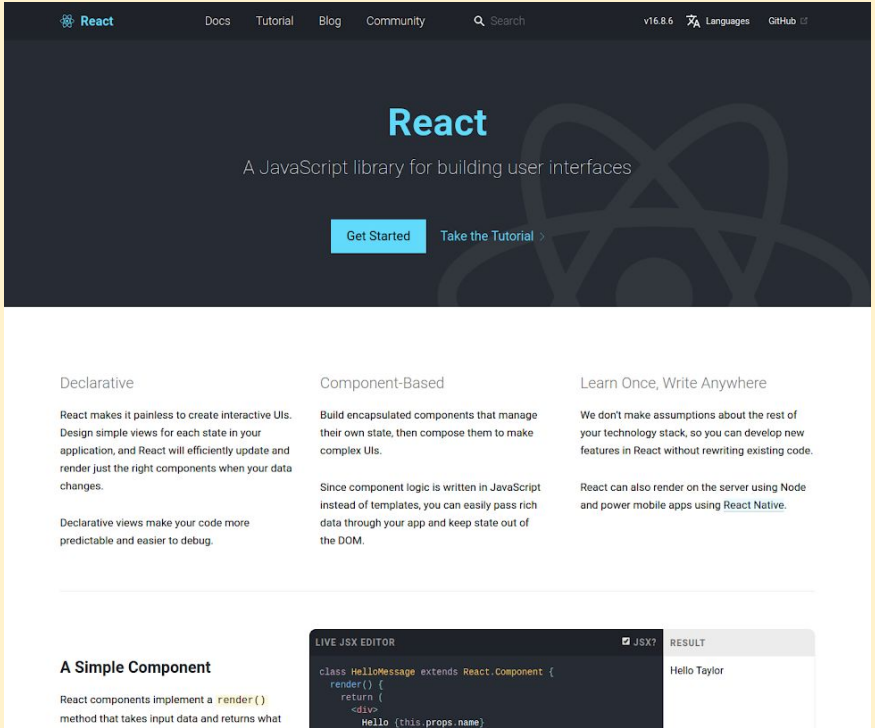


Progressive web app

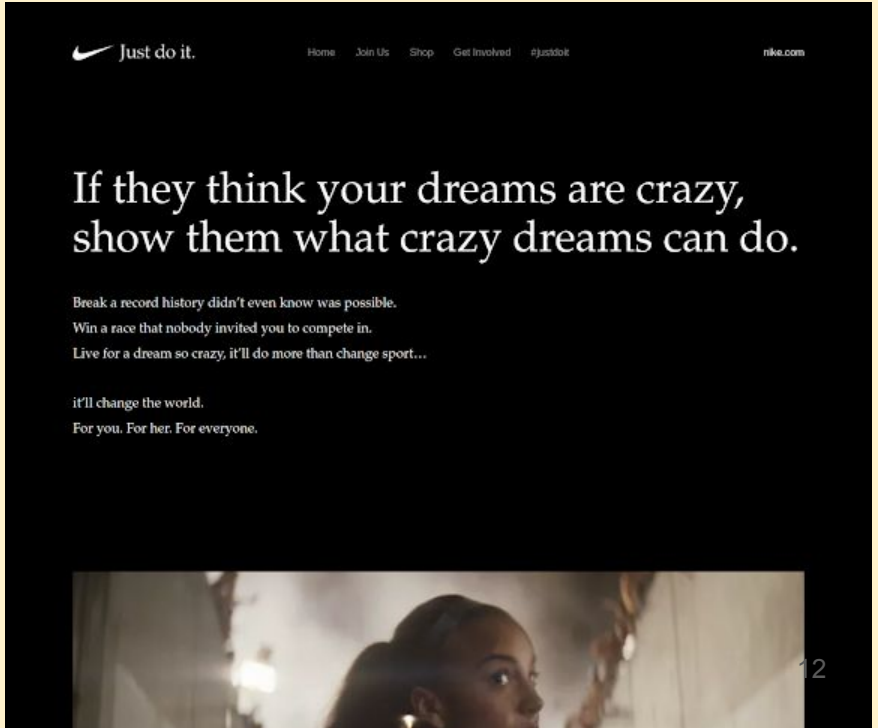
- **Progressive** - progressive improvement
- **Responsive**
- **Connectivity independence**
- **App-like**
- **Immediate Update Deployment**
- **Safety**
- **Discoverability**
- **Easy installation**
- **Linkable**

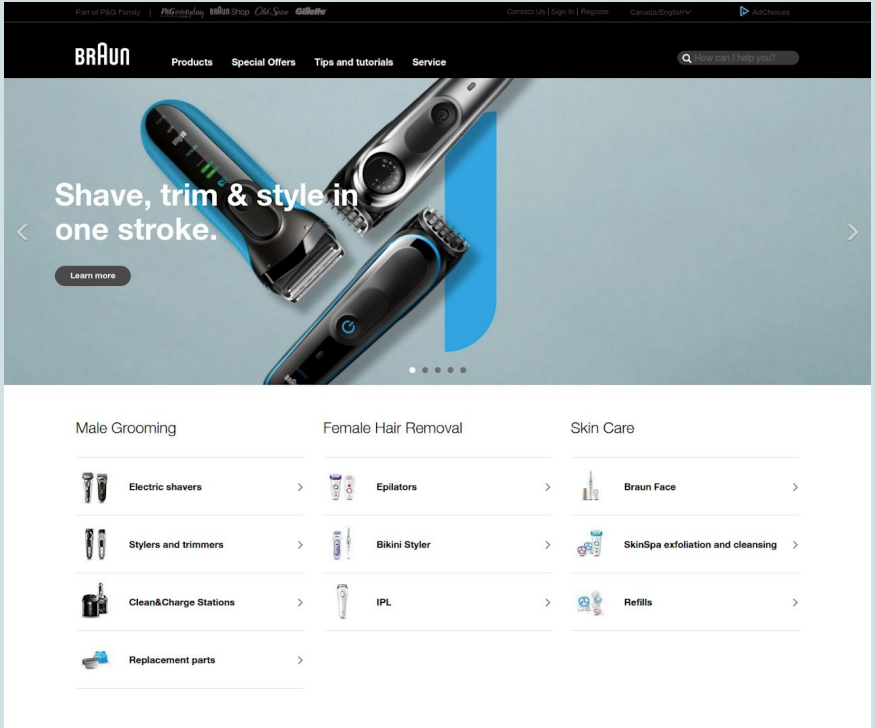


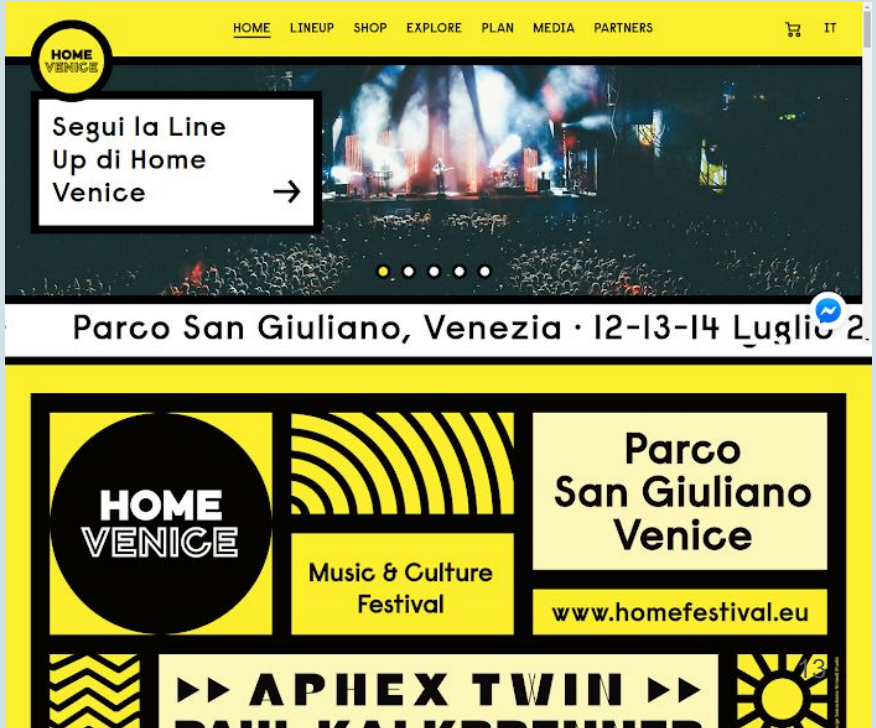
Who uses Gatsby? 


React
Docs
Tutorial
Blog
Community
Search
v16.8.6
Languages
GitHub
React
A JavaScript library for building user interfaces
Get Started
Take the Tutorial
Declarative
React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.
Declarative views make your code more predictable and easier to debug.
Component-Based
Build encapsulated components that manage their own state, then compose them to make complex UIs.
Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM.
Learn Once, Write Anywhere
We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.
React can also render on the server using Node and power mobile apps using React Native.
A Simple Component
React components implement a render() method that takes input data and returns what
LIVE JSX EDITOR
JSX?
class HelloMessage extends React.Component {
render() {
return (

Hello {this.props.name}
RESULT
Hello Taylor


Just do it.
Home
Join Us
Shop
Get Involved
#justdoit
nike.com
If they think your dreams are crazy, show them what crazy dreams can do.
Break a record history didn't even know was possible.
Win a race that nobody invited you to compete in.
Live for a dream so crazy, it'll do more than change sport...
it'll change the world.
For you. For her. For everyone.


BRAUN
Products
Special Offers
Tips and tutorials
Service
Shave, trim & style in one stroke.
Learn more
Male Grooming
Electric shavers
Stylers and trimmers
Clean&Charge Stations
Replacement parts
Female Hair Removal
Epilators
Bikini Styler
IPL
Skin Care
Braun Face
SkinSpa exfoliation and cleansing
Refills


HOME
LINEUP
SHOP
EXPLORE
PLAN
MEDIA
PARTNERS
IT
HOME VENICE
Segui la Line Up di Home Venice
Parco San Giuliano, Venezia · 12-13-14 Luglio 2
HOME VENICE
Music & Culture Festival
Parco San Giuliano Venice
www.homefestival.eu
APHEX TWIN

Use case 

Simple blog from Markdown data source

Data: markdown structure

Frontmatter

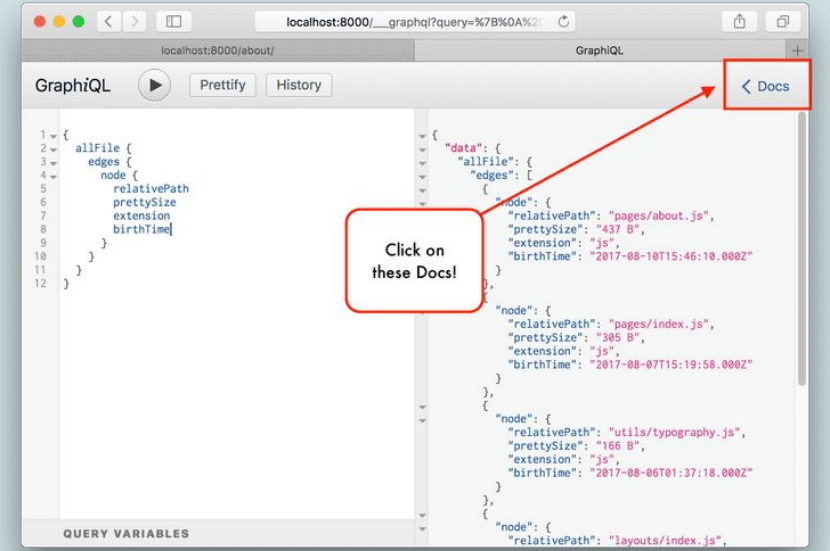
```yaml
---
path: '/en/mysql-backup-script/'
view: 'story'
language: 'en'
translations:
 - path: '/it/mysql-backup-script/'
   language: 'it'
title: 'Mysql Backup Script | Wavelop'
description: 'A simple shell script for backup a MySQL database and
store it in a server with an FTP connection.'
keywords: 'mysql, bash script, backup, tutorial, crontab'
socials:
 socialTitle: 'Mysql Backup Script | Wavelop'
 socialType: 'website'
---
```

Markdown

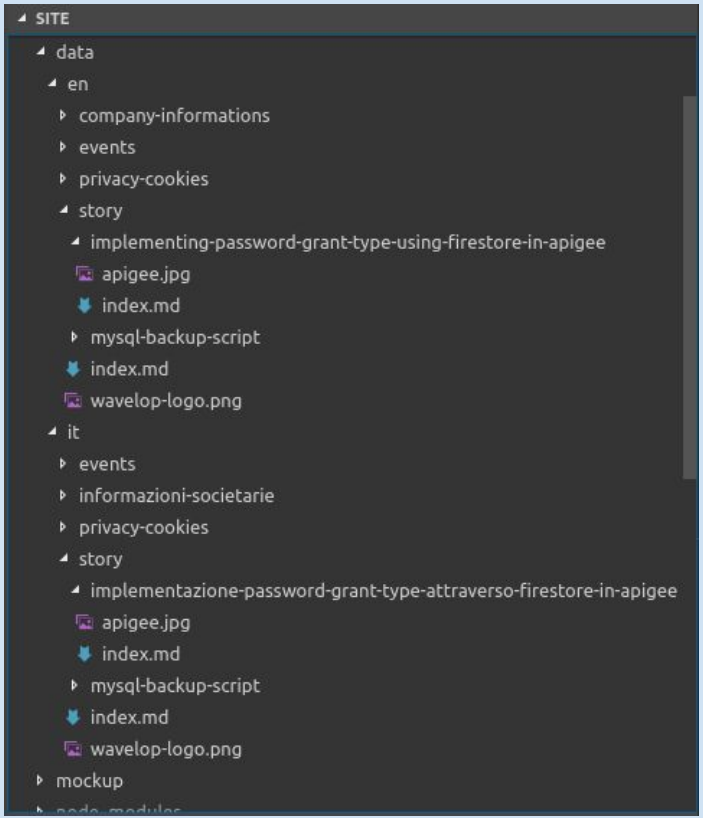
```markdown
### Index

- [Mysql dump command](#dump)
- [Upload file to server](#upload)
```

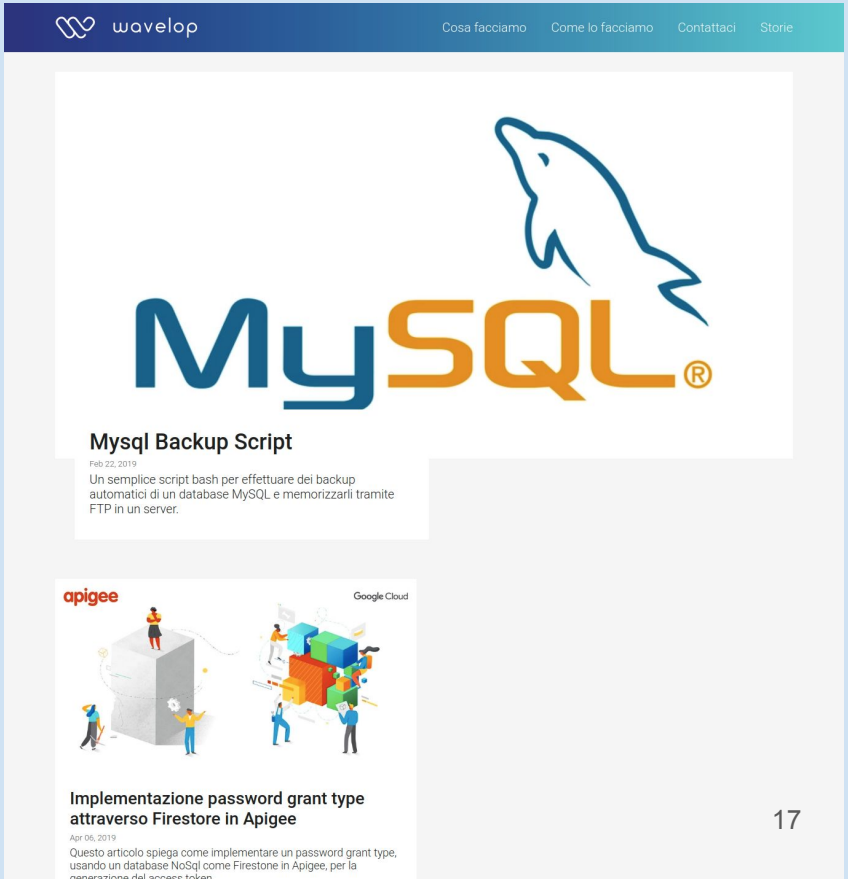
Data: GraphQL

- Gatsby forces you to use GraphQL in order to only get the data that you need
- So, any API that you try and fetch data from, you have to convert it into GraphQL before you can use it in Gatsby



Data: directories structure





Data: create nodes

Plugin

gatsby-config.js

```
{
    resolve: `gatsby-source-filesystem`,
    options: {
       path: `${__dirname}/data/`,
       name: 'data'
    }
}
```

Create Node

gatsby-node.js

```
const { createFilePath } = require(`gatsby-source-filesystem`)
exports.onCreateNode = ({ node, getNode }) => {
  if (node.internal.type === `MarkdownRemark`) {
    const value = createFilePath({ node, getNode });
    createNodeField({
      name: `slug`,
      node,
      value
    });
  }
}
```

Data: create pages

gatsby-node.js

```
exports.createPages = ({ graphql, actions }) => {
  const { createPage } = actions

  return new Promise((resolve, reject) => {
    const storyTemplate =
path.resolve('./src/views/Story/index.tsx')
    resolve(
      graphql(
        `
          {
            allMarkdownRemark(limit: 1000) {
              edges {
                node {
                  fields {
                    slug
                  }
                  frontmatter {
                    title
                    description
                    path
                  }
                }
              }
            }
          }
        `
      )
```

```
      .then(result => {
        if (result.errors) {
          reject(result.errors)
        }
        result.data.allMarkdownRemark.edges.forEach(({ node }) => {
          const path = node.fields.slug
          createPage({
            path,
            component: storyTemplate,
            context: {
              path,
            },
          })
        })
      })
    )
  })
}
```

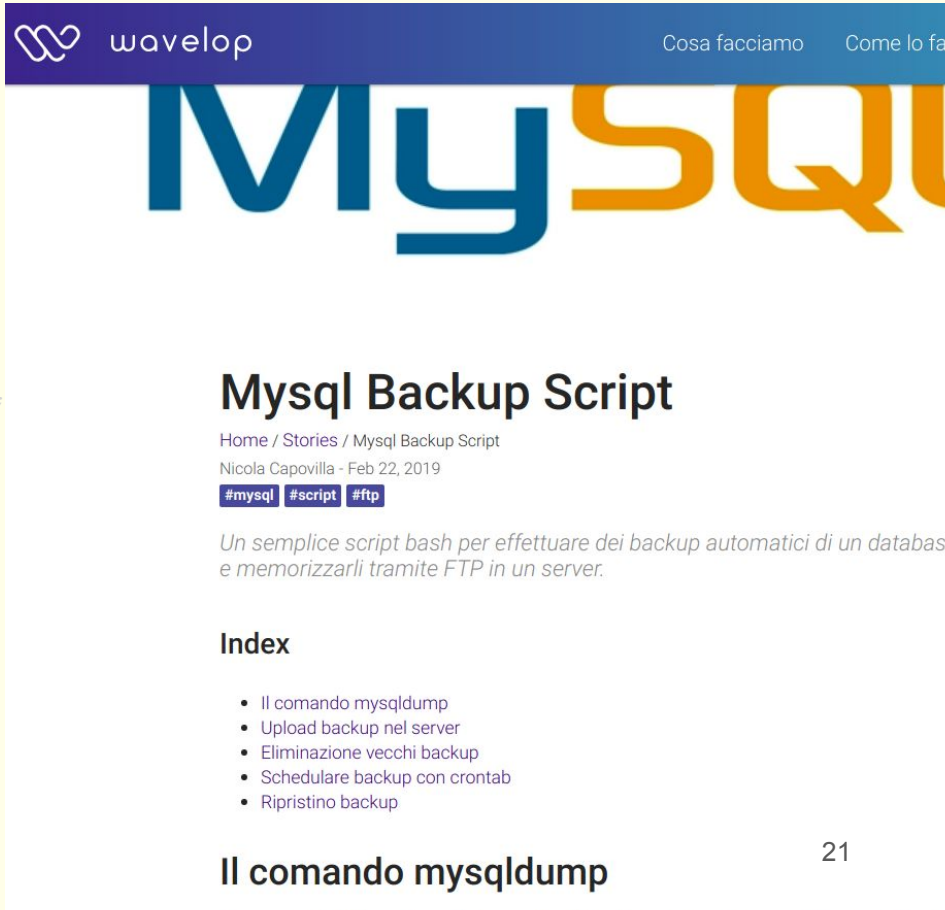
Rendering

```
---
path: '/it/story/mysql-backup-script/'
storyMetaData:
 date: '2019-02-22'
 author: Nicola Capovilla
 coverImage: './mysql.png'
 title: 'Mysql Backup Script'
 summary: 'Un semplice script bash per effettuare dei backup
automatici di un database MySQL e memorizzarli tramite FTP in un
server.'
 tags:
   - tag: mysql
   - tag: script
   - tag: ftp
---
### Index
- [Il comando mysqldump ](#dump)
- [Upload backup nel server ](#upload)

<h2 name="dump">Il comando mysqldump </h2>
Il comando da terminale per generare un backup di un database MySQL
è **mysqldump**.
```

```
export const pageQuery = graphql`
 query{
   markdownRemark{
     htmlAst
     tileInfo: frontmatter {
         path
         meta: storyMetaData {
           title
           author
           date(formatString: "YYYYMMDD")
           summary
           coverImage {
              …
           }
           tags {
             tag
           }
         }
       }
     }
   }
`;
```

Rendering

```
const renderAst = new RehypeReact({
 createElement: React.createElement,
 components: {
   h1: Headline1,
   h2: Headline2,
   p: Paragraph,
   blockquote: Blockquote,
   ul: List,
 }
}).Compiler;

render() {
   const { markdownRemark, related } = this.props.data;
   const storyMetaData = this.props.data.markdownRemark.tileInfo.meta;
   return (
     <Layout {...this.props}>
         <BlogStrip>
             <Title>{storyMetaData.title}</Title>
             <Breadcrumbs {...this.props} />
             <MetaInfo>{storyMetaData.author}</MetaInfo>
             <div>
               {storyMetaData.tags.map(node => (
                   <Tag key={node.tag}>{`#${node.tag}`}</Tag>
                ))}
             </div>
             <Summary>{storyMetaData.summary}</Summary>

             {renderAst(markdownRemark.htmlAst)}
         </BlogStrip>
     </Layout>
   );
 }
```



Google Analytics

`gatsby-plugin-google-analytics`

Ricerca

`gatsby-plugin-lunr`

Style

`gatsby-plugin-styled-components`
`gatsby-remark-prismjs`

Immagini

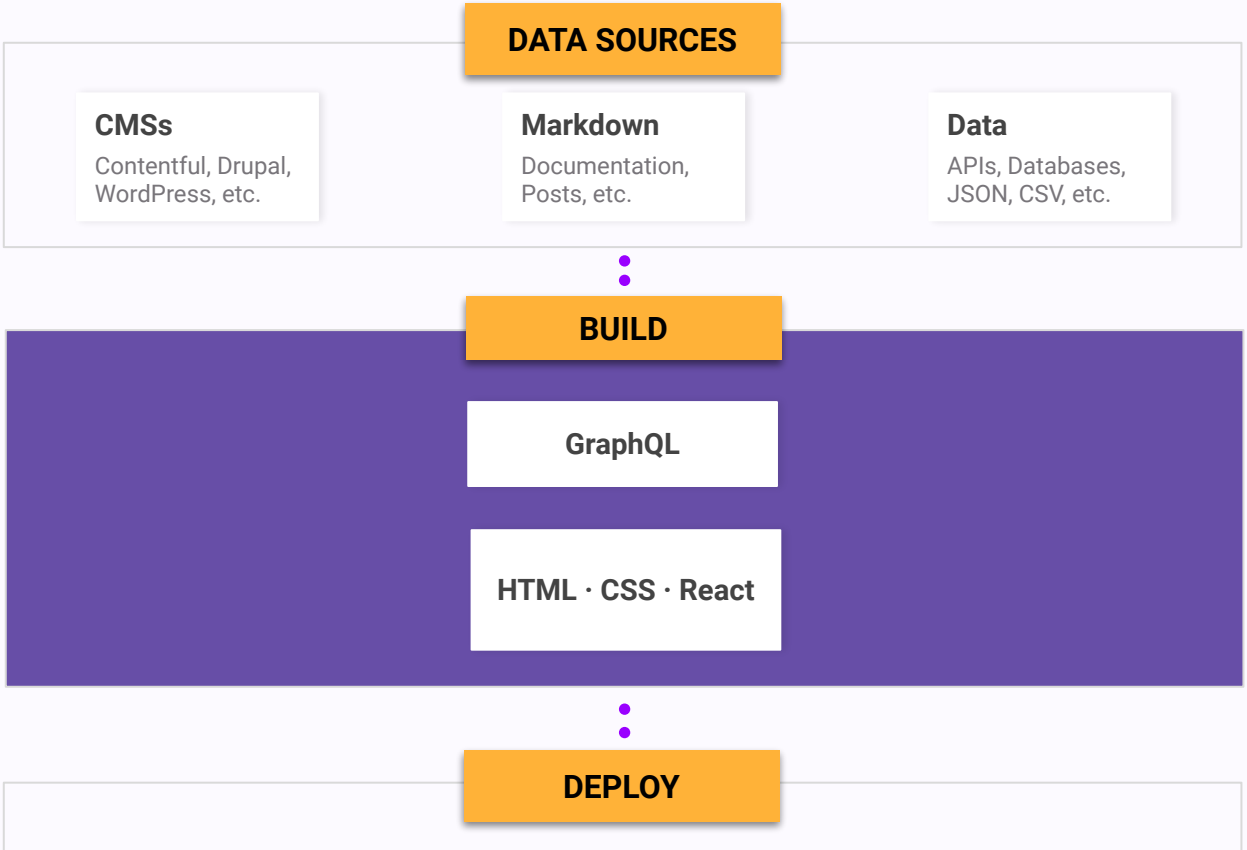
`gatsby-plugin-sharp`
`gatsby-transformer-sharp`
`gatsby-remark-images`

SEO

`gatsby-plugin-robots-txt`
`gatsby-plugin-manifest`
`gatsby-plugin-react-helmet`

Utilities

`gatsby-plugin-typescript`
`gatsby-plugin-offline`

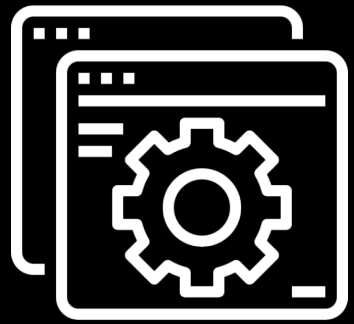


We used Markdown, but with **plugins** and **GraphQL** you can use **any data sources**

Switch to other data sources is easy

Building

```
blundert@wavelop ~/wavelop.com$ gatsby build
success open and validate gatsby-configs — 0.007 s
success load plugins — 0.266 s
success onPreInit — 0.428 s
success delete html and css files from previous builds — 0.042 s
success initialize cache — 0.005 s
success copy gatsby files — 0.019 s
success onPreBootstrap — 0.009 s
success source and transform nodes — 0.084 s
success building schema — 0.279 s
success createPages — 0.083 s
success createPagesStatefully — 0.033 s
success onPreExtractQueries — 0.003 s
success update schema — 0.054 s
success extract queries from components — 0.144 s
success run graphql queries — 0.016 s — 7/7 469.13 queries/second
success write out page data — 0.004 s
success write out redirect data — 0.001 s
success onPostBootstrap — 0.258 s

info bootstrap finished - 3.197 s

success Building production JavaScript and CSS bundles — 4.414 s
success Building static HTML for pages — 2.644 s — 18/18 25.10 pages/second
Generated public/sw.js, which will precache 12 files, totaling 617222 bytes.
info Done building in 10.354 sec
```



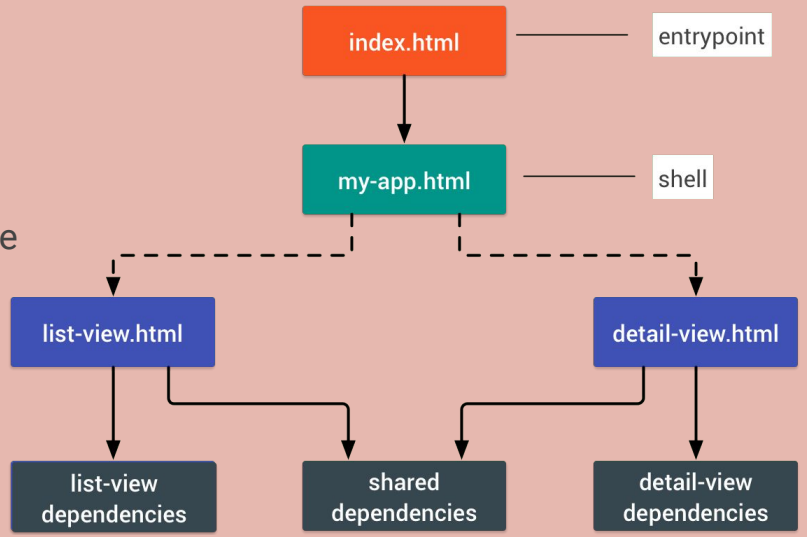
PRPL Pattern

Web site architecture and it stands for:

- Push critical resources for the initial URL route using <link preload> and http/2
- Render initial route
- Pre-cache remaining routes
- Lazy-load and create remaining routes on demand

Gatsby follows the PRPL architectural pattern:

1. Gatsby sites **render** a static HTML version of the initial route and then load the code bundle for the page
2. Then immediately starts **pre-caching** resources for pages linked to from the initial route
3. When a user clicks on a link, Gatsby creates the new page **on demand** on the client



CI / CD

GitLab

\+

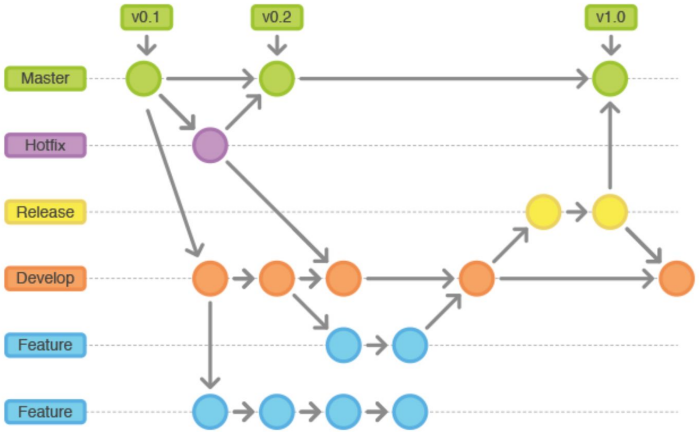
repository

Hosting

Firebase

Build + Deploy

commit in master: PROD
commit in release: STAGING



.gitlab-ci.yml

```
image: node:latest

stages:
  - build
  - deploy

app-build_staging:
  stage: build
  before_script:
    - yarn global add firebase-tools
    - yarn global add gatsby-cli
    - yarn install
  artifacts:
    paths:
      - public/
      - .firebaserc
      - firebase.json
  script:
    - ACTIVE_ENV=test npm run build
  environment:
    name: staging
  only:
    - /^release\/.*$

app-build_production:
  stage: build
  before_script:
    - yarn global add firebase-tools
    - yarn global add gatsby-cli
    - yarn install
  artifacts:
    paths:
      - public/
      - .firebaserc
      - firebase.json
  script:
    - ACTIVE_ENV=production npm run
build
  environment:
    name: production
  only:
    - master

...
```

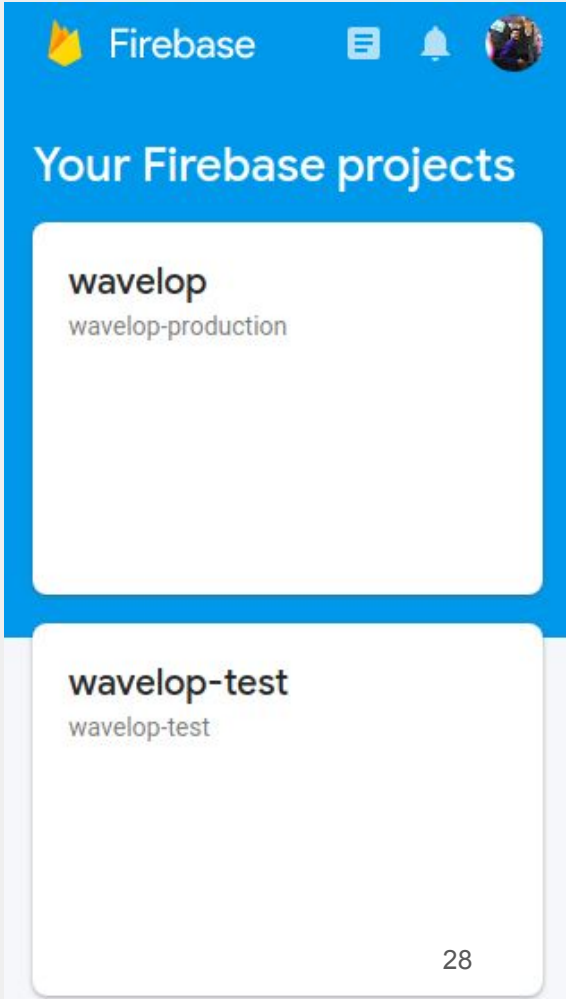
GitLab

.gitlab-ci.yml

```yaml
image: node:latest

stages:
  - build
  - deploy

...

deploy_staging:
  stage: deploy
  before_script:
    - yarn global add firebase-tools
  dependencies:
    - app-build_staging
  script:
    - firebase use test --token
$FIREBASE_TOKEN
    - firebase deploy --only hosting
-m "Pipe $CI_PIPELINE_ID Build
$CI_BUILD_ID @ Hash
${CI_COMMIT_SHA:0:7}" --token
$FIREBASE_TOKEN
  environment:
    name: staging
  only:
    - /^release\/.*$/
```

```yaml
deploy_production:
  stage: deploy
  before_script:
    - yarn global add firebase-tools
  dependencies:
    - app-build_production
  script:
    - firebase use production
--token $FIREBASE_TOKEN
    - firebase deploy --only hosting
-m "Pipe $CI_PIPELINE_ID Build
$CI_BUILD_ID @ Hash
${CI_COMMIT_SHA:0:7}" --token
$FIREBASE_TOKEN
  environment:
    name: production
  only:
    - master
```

+ .firebaserc + firebase.json



https://wavelop.com

SWITCH URL | RUN AUDIT

Last audit: Jun 8, 4:29 PM View Report | Download Report

Performance	Accessibility	Best Practices	SEO
98	100	100	100

Score scale: 0-49 50-89 90-100

First Contentful Paint	1.0 s	First Meaningful Paint	1.0 s
Speed Index	1.6 s	First CPU Idle	2.8 s
Time to Interactive	2.9 s	Estimated Input Latency	470 ms

Indexing 📋

What exactly happens when you type a URL into the browser

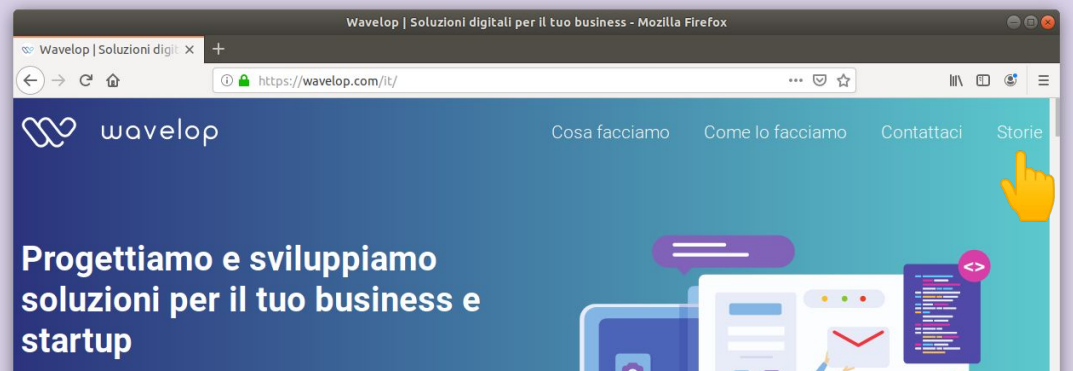
- You'll get an HTML file pre-rendered with the app linked
- You'll get a JavaScript file, which is the React app

Server is serving a static file that's generated

What happens when you click a link on a loaded web page

It's a React app - it will go and run it on the browser side

- Google always requests Initial fetch - so static file
- A user uses React app



Thanks 😀

🎮 matteogranzotto.com
@blundert

🌐 wavelop.com
@wavelop

wavelop