

GraphQL

01 GraphQL?

02 GraphQL VS REST API

03 활용 사례

04 마무리

01 GraphQL?

안녕하세요.
성장과
개발자 **정진혁**입니다.



[GitHub](#) [LinkedIn](#) [Email](#)
[email](#)

ALL FEATURED 블로그 회고 웹공부 PS 프로젝트 스타트업 알고리즘

움직이는 미모티콘 GIF 생성하기

미모티콘 생성하기 미모티콘을 생성하시려면 먼저 아이폰 메시지에 들어가시면 메시지앱에 다음과 같은 옵션이 있습니다. 여기서 앱스토어 옆에 있는 버튼을 클릭하면 아래와 같은 미모티콘들이 나오는 걸 볼 수 있습니다. 미모티콘 리스트 맨 왼쪽에 위치한 플러스 버튼을 클릭하시면 아래와 같이 나만의 미모티콘을 커스터마이징 할 수 있는 페이지가 나오게 됩니다. 이 과정을 쭉 진행하고 나시면...

September 01, 2021 [블로그](#)

신입 개발자의 고민: 나는 잘하고 있는 것일까

💡 나는 잘하고 있는 것일까 입사를 한지 어느덧 반년 가까이 흘러가고 있다. 회사에서 멋진 프로젝트에 참여하며 훌륭한 사람들과 함께 일을 하고 있게 되었고, 남아있던 마지막 학기도 저녁과 주말 시간을 활용해서 마무리하게 되었다. 곁보기에는 순탄하게 잘 진행되고 있는 것 같지만 본격적으로 시작된 직장인 생활은 녹록치 않았다. 특히 조급함과 불안감이 참 나를 힘들게 했다. 이 글을 쓰는 지금 나...

August 10, 2021 [회고 featured](#)

안녕하세요.
개발이
개발자 **김지훈**입니다.



[GitHub](#) [Email](#)

ALL DEV EXPERIENCE

CS 발표회 | cwv
April 28, 2024

CS 발표회 | three.js
April 19, 2024

프로젝트 | Judgement 초기 기획안
April 04, 2024

깃허브 블로그를 오른쪽처럼 커스터마이징 하고 싶었다.

01 GraphQL?

```
function PostCard({ post }) {
  const { id, slug, title, excerpt, date, categories, emoji } = post;
  return (
    <div className="post-card-wrapper">
      <Link className="post-card" key={id} to={slug}>
        <div className="emoji">{emoji}</div>
        <div className="content">
          <div className="title">{title}</div>
          <div className="date">{date}</div>
        </div>
      </Link>
    </div>
  );
}
```

그러기 위해선 PostCard 컴포넌트의 Props에
emoji를 추가해줘야했다.

01 GraphQL?

```
const createPostsPages = ({ createPage, results }) => [
  const categoryTemplate = require.resolve(`./src/templates/category-template.js`);
  const categorySet = new Set(['All']);
  const { edges } = results.data.allMarkdownRemark;

  edges.forEach(({ node }) => {
    const postCategories = node.frontmatter.categories.split(' ');
    postCategories.forEach(category) => categorySet.add(category);
  });

  const categories = [...categorySet];

  createPage({
    path: `/posts`,
    component: categoryTemplate,
    context: {
      currentCategory: 'All',
      edges,
      categories,
    },
  });
]
```

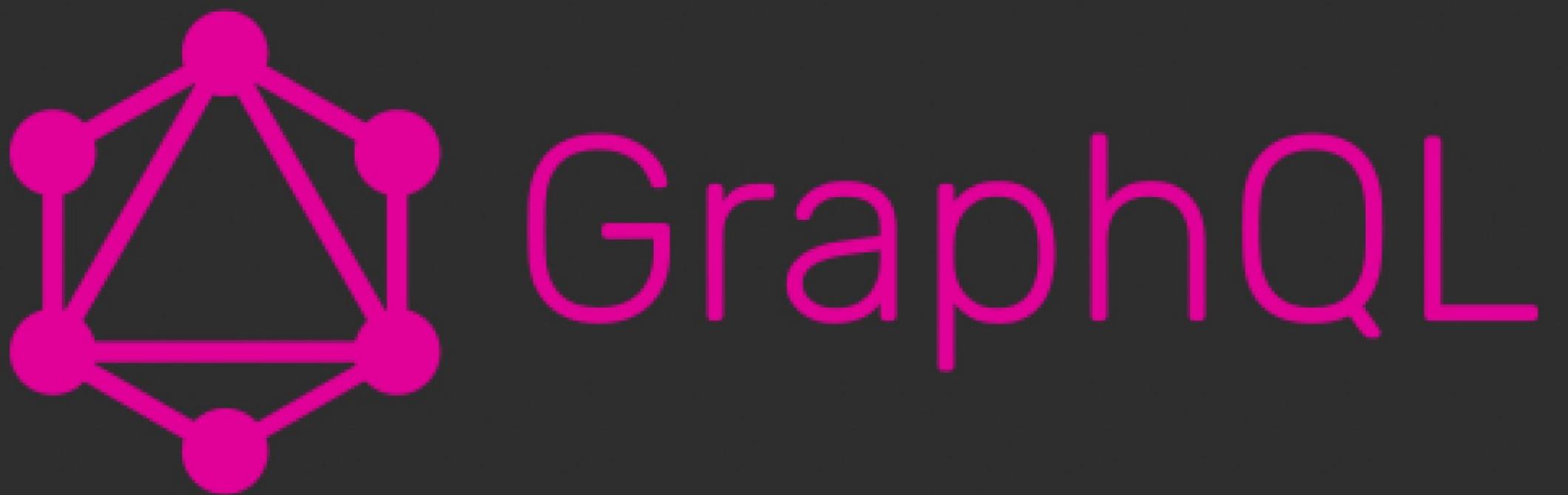
```
const results = await graphql(`

{
  allMarkdownRemark(sort: { order: DESC, fields: [frontmatter__date] }, limit: 1000) {
    edges {
      node {
        id
        excerpt(pruneLength: 500, truncate: true)
        fields {
          slug
        }
        frontmatter {
          categories
          title
          date(formatString: "MMMM DD, YYYY")
          emoji
        }
      }
    }
  }
}

`)
```

그렇게 Props를 타고 가다보니 발견한 코드,
잉 근데 이게 뭐죠?

01 GraphQL?



페이스북에서 만든 쿼리 언어 Like SQL

01 GraphQL?

```
{  
  actors {  
    name  
    id  
  }  
}
```

SELECT ACTOR_ID, ACTOR_NAME FROM ACTORS;

But, SQL이 데이터베이스에 저장된 데이터를 불러온다면,
GraphQL은 클라이언트가 데이터를 서버로부터 가져오는 것을 목적으로 함

02 GraphQL VS REST API



GraphQL과 RESTful API는 모두 웹 서비스에서 데이터를 교환하기 위한 프로토콜
하지만, **엔드포인트**에서 큰 차이가 존재

02 GraphQL VS REST API

가령, 우리가 중국집 메뉴에 대한 데이터를 얻으려고 한다고 하면,
REST API는 아래와 같은 요청이 필요할 것

jung-gugjib.com/menu/

jung-gugjib.com/menu/{세부메뉴 인덱스}

jung-gugjib.com/menu/식사부/{식사메뉴 인덱스}

jung-gugjib.com/menu/요리부/{요리메뉴 인덱스}

...

under-fetching 과
over-fetching 가능성 존재

02 GraphQL VS REST API

하지만, GraphQL은 하나의 엔드포인트로 요청해도 충분하다.

jung-gugjib.com/graphql

02 GraphQL VS REST API

```
query {  
    식사부 (식사부_idx : 3) {  
        name,  
        price,  
    }  
}
```

즉, 원하는 응답값만 쿼리로 작성하여 요청하면,
그것에 맞는 응답을 해준다는 것!

정해진 응답값만 받아올 수 있는 REST API에 비해
직관적이고 효율적일 수 있음

Who's using GraphQL?

Facebook's mobile apps have been powered by GraphQL since 2012. A GraphQL spec was open sourced in 2015 and is now available in many environments and used by teams of all sizes.



많은 기업에서 활용중

03 활용 사례

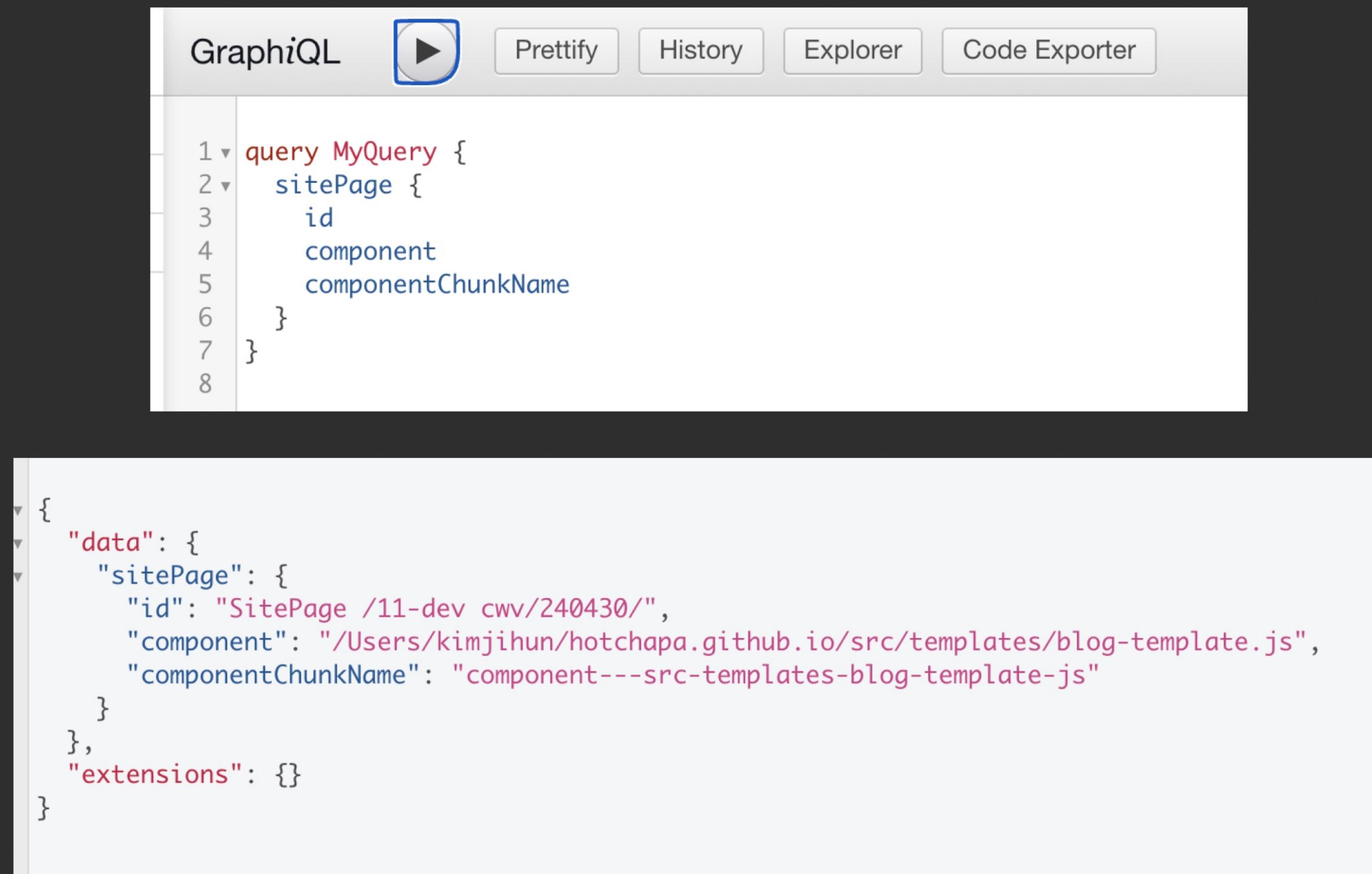


Gatsby

```
You can now view hotchapa.github.io in the browser.  
:::  
:: http://localhost:8000/  
:::  
View GraphQL, an in-browser IDE, to explore your site's data and schema  
:::  
:: http://localhost:8000/__graphql  
:::
```

Gatsby 블로그는 GraphQL을 활용하여
데이터 레이어를 구축, UI 구성에 효율성을 높여줌

03 활용 사례



The image shows the GraphiQL interface, which is a web-based tool for interacting with GraphQL servers. It consists of two main sections: a query editor at the top and a results viewer at the bottom.

Query Editor (Top):

```
1 query MyQuery {  
2   sitePage {  
3     id  
4     component  
5     componentChunkName  
6   }  
7 }  
8
```

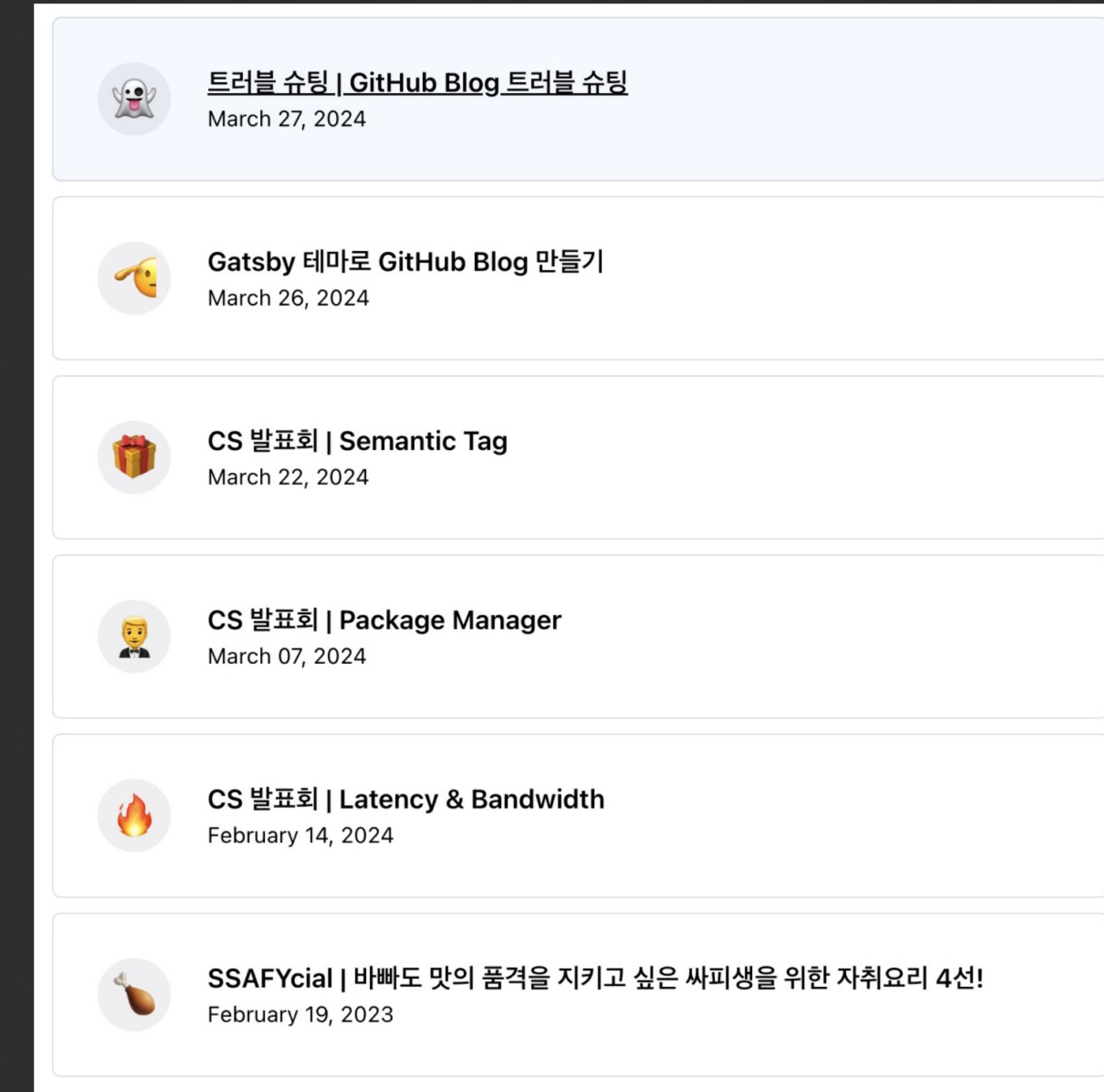
Results Viewer (Bottom):

```
{  
  "data": {  
    "sitePage": {  
      "id": "SitePage /11-dev cwv/240430/",  
      "component": "/Users/kimjihun/hotchapa.github.io/src/templates/blog-template.js",  
      "componentChunkName": "component---src-templates-blog-template-js"  
    }  
  },  
  "extensions": []  
}
```

이렇게 GraphQL을 통해 블로그 데이터 조회도 가능하다

04 마무리

```
const results = await graphql(`\n  {\n    allMarkdownRemark(sort: { order: DESC, fields: [frontmatter__date] }, limit: 1000) {\n      edges {\n        node {\n          id\n          excerpt(pruneLength: 500, truncate: true)\n          fields {\n            slug\n          }\n          frontmatter {\n            categories\n            title\n            date(formatString: "MMMM DD, YYYY")\n            emoji\n          }\n        }\n      }\n    }\n    next {\n      fields {\n        slug\n      }\n    }\n  }\n`)\n\nzoomkoding, 3년 전 • feat: let the blog-template have next and prev ...
```



요청 쿼리에 emoji 필드를 추가해서,
기술블로그에 원하는 UI를 구성할 수 있었음

04 마무리

블로그의 경우, 커스터마이징을 하는 상황이 자주 생김.
이런 경우, 백엔드 의존도가 높은 REST API는
수정 효율성이 떨어질 수 있음 ex) 앤드포인트 커뮤니케이션

반면에 **GraphQL**을 활용하면,
클라이언트 단에서 쿼리를 통해 필요한 데이터 구조를 자유롭게 설계할 수 있기 때문에
개발자가 프론트엔드에서 보여줄 데이터를 더 **유연하게** 처리할 수 있게 도와줌

반대로 고정된 요청과 응답이 필요할 땐,
오히려 REST API 보다 요청 크기가 커질 수 있음.
캐싱 과정 역시 복잡

⇒ GraphQL은 무적이 아님

서비스에 맞는 방식을 고르자.