

The background features a dark blue to black gradient. It is decorated with several network diagrams consisting of purple nodes connected by lines. Additionally, the letters 'TSS' are repeated in a light blue, semi-transparent font, tilted at various angles across the background.

# Web Applications of the Future

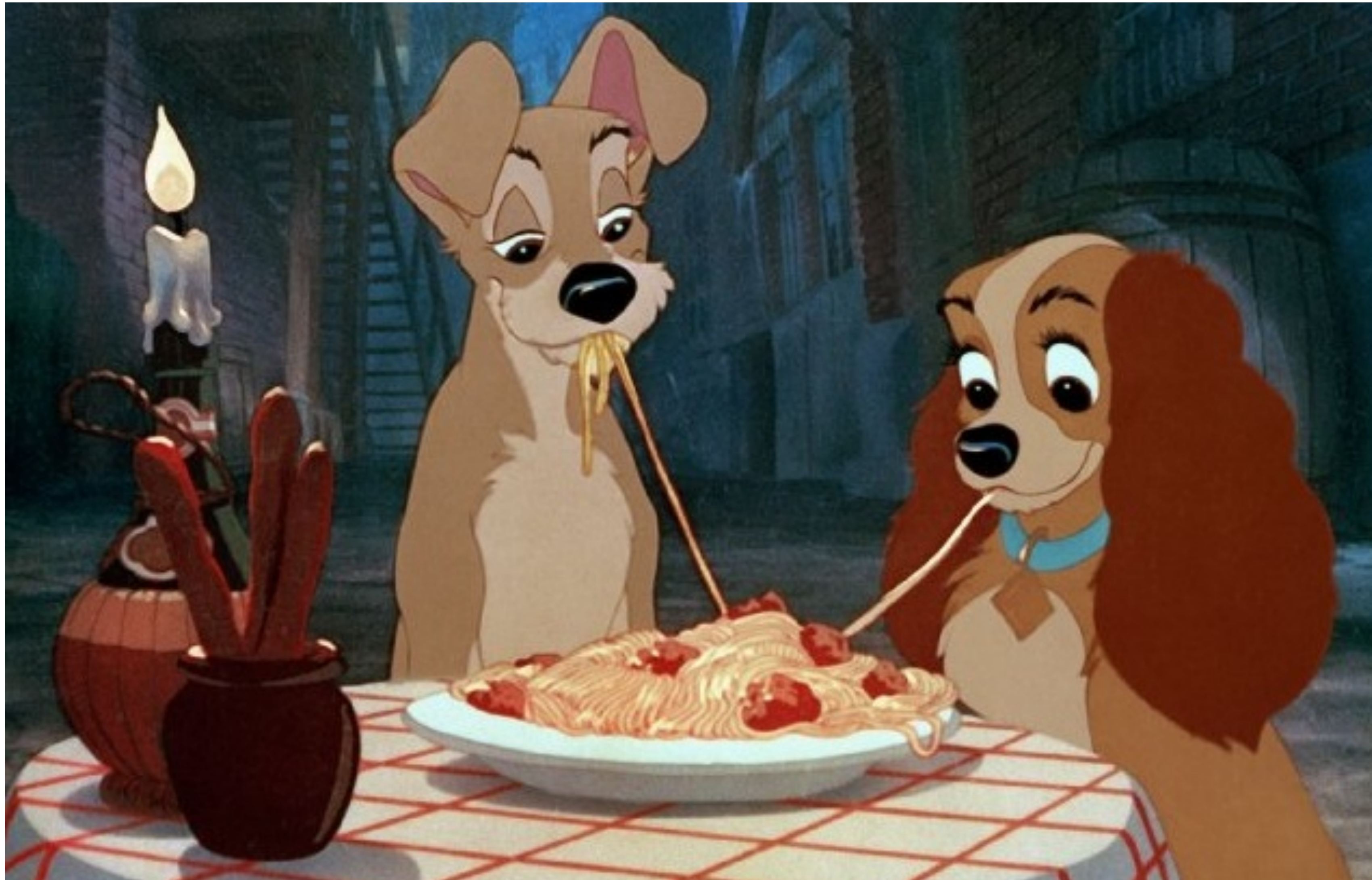
Who is this for?











@gethackteam



```
let date;
```

```
date = new Date(); // Sat, 07 Dec 2022 00:00:00
```



```
date = 'Sat, 07 Dec 2022 00:00:00'
```

```
date = 1575676800; // Sat, 07 Dec 2022 00:00:00
```

```
date = // ...
```





That's why you need  
type systems!  

A little bit about  
myself first...



@gethackteam



# Roy Derks

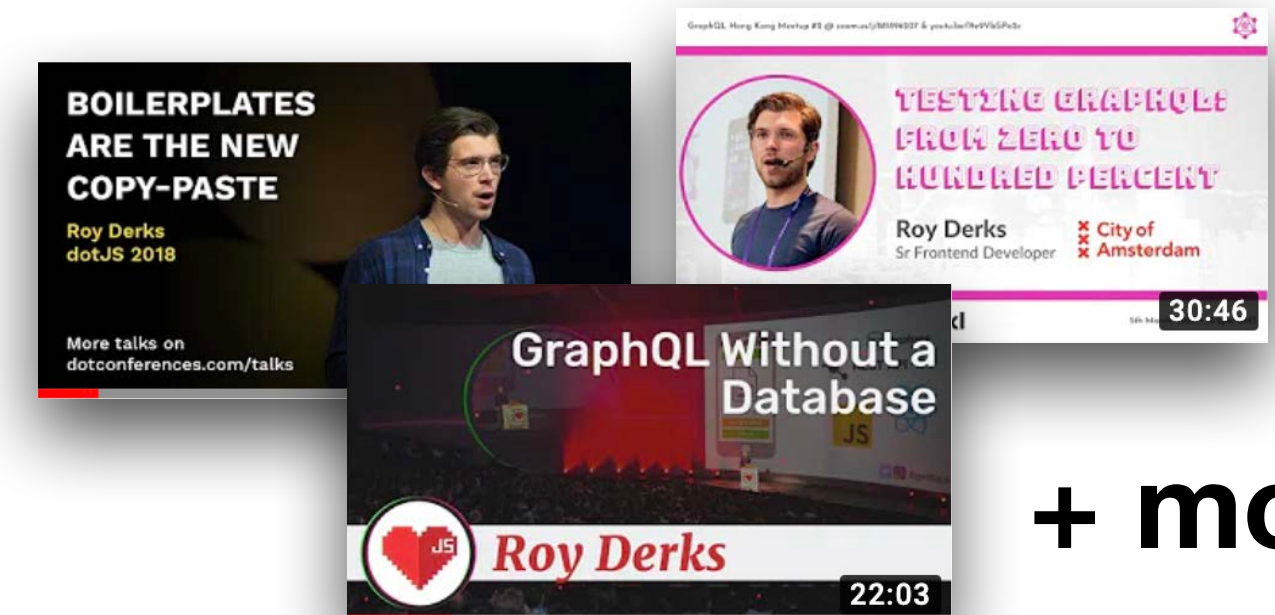


@gethackteam

Hey! I'm Roy  
an entrepreneur and software  
engineer, author and public speaker  
from The Netherlands.

 StepZen


 hackteam



+ more



@gethackteam

What do you already know  
about TypeScript? 



@gethackteam

TypeScript is a **typed superset** of  
JavaScript

That **compiles to plain JavaScript**

And uses the **latest ECMAScript** features



@gethackteam

*superset. [ˈsü·pər, set] (computer science)*  
*A programming language that **contains all the features of a given language** and has been expanded or enhanced to include other features as well.*

*- the internet*



@gethackteam

*superset. [ˈsü·pər, set] (computer science)  
A programming language that contains all  
the features of a given language and has  
been **expanded or enhanced to include  
other features** as well.*

*- the internet*



@gethackteam

TypeScript is a **typed superset** of  
JavaScript

That **compiles to plain JavaScript**

And uses the **latest ECMAScript** features



@gethackteam



```
type Fruit = {
  name: string,
  variety: string
}

// Creates new array with only the name
function createArray(array: Array<Fruit>) {
  return array.map((item) => item.name)
}

console.log(createArray(fruits))
```

```
function createArray(array) {
  return array.map(function (item) {
    return item.name;
  });
}

console.log(createArray(fruits));
```

```
type Fruit = {  
  name: string,  
  variety: string  
}
```

```
// Creates new array with only the name  
function createArray(array: Array<Fruit>) {  
  return array.map((item) => item.name)  
}
```

```
console.log(createArray(fruits))
```

```
function createArray(array) {  
  return array.map(function (item) {  
    return item.name;  
  });  
}  
console.log(createArray(fruits));
```

# Readable for the browser



@gethackteam

TypeScript is a **typed superset** of  
JavaScript

That **compiles to plain JavaScript**

And uses the **latest ECMAScript** features



@gethackteam

Repository

 [github.com/Microsoft/TypeScript](https://github.com/Microsoft/TypeScript)

---

Homepage

 [www.typescriptlang.org/](http://www.typescriptlang.org/)

---

↓ Weekly Downloads

**24,664,744**



Version

**4.5.2**

License

**Apache-2.0**

---

Unpacked Size

**64 MB**

Total Files

**177**

---

Issues

**5093**

Pull Requests

**296**

---

Last publish

**10 hours ago**

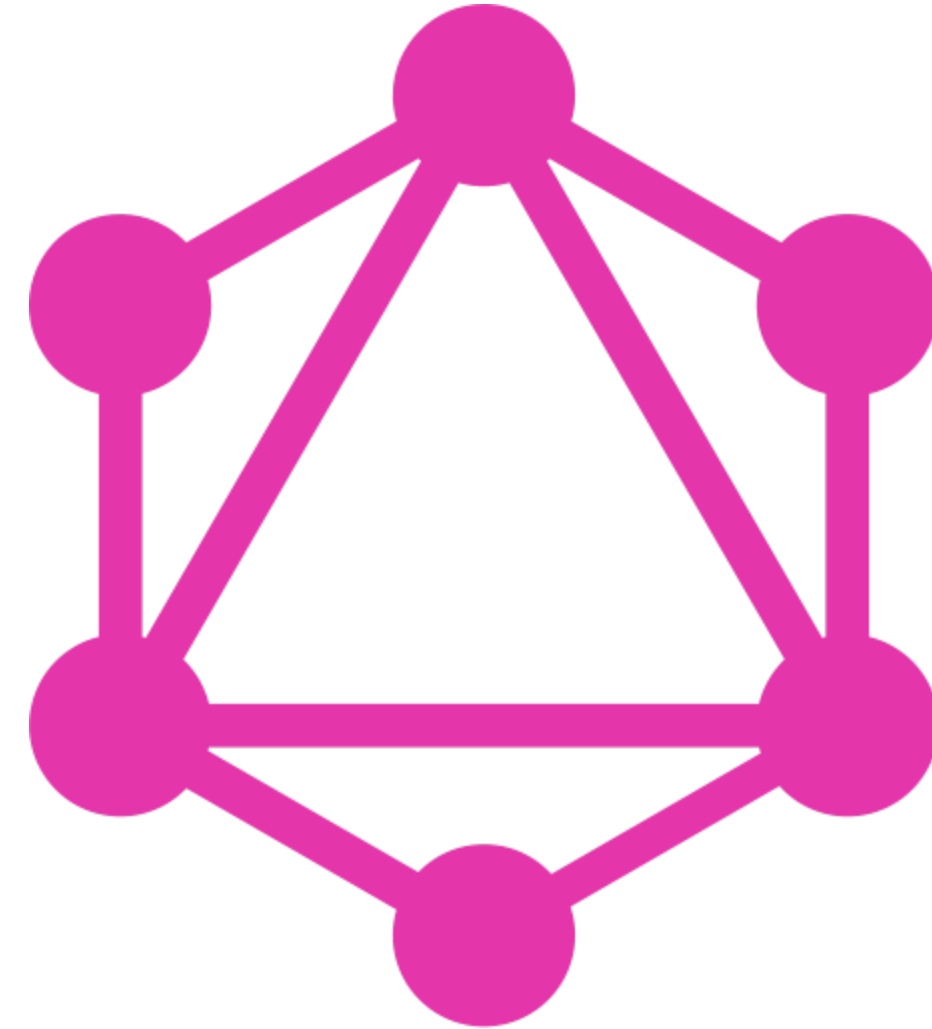
---



**@gethackteam**



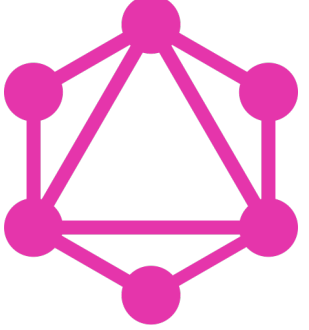
You know what else has types?



You know what else has types?  
**GraphQL!**



@gethackteam

What do you already know  
about GraphQL? 



@gethackteam

GraphQL is a **query language** for APIs

That offers a **single endpoint** for **multiple resources**

And is based on a **type system**



@gethackteam



Query Schema

RUN



```
query GithubQueries {
  github_user(login: "githubteacher") {
    bio
    followers(first: 10) {
      edges {
        node {
          id
          login
        }
      }
    }
  }
}
```

Response Documentation



```
{
  "data": {
    "github_user": {
      "bio": "StepZen revolution",
      "followers": {
        "edges": [
          {
            "node": {
              "id": "MDU6VG9waWNqcXVlcnktdGx1Z2lu",
              "login": "In scelerisque sem at dolor"
            }
          },
          {
            "node": {
              "id": "MDU6VG9waWNqYXZhc2NyaXB0",
              "login": "Duis sagittis ipsum"
            }
          }
        ]
      }
    }
  }
}
```



@gethackteam

Query Schema



```
query GithubQueries {
  github_user(login: "githubteacher") {
    bio
    followers(first: 10) {
      edges {
        node {
          id
          login
        }
      }
    }
  }
}
```

Response Documentation



```
{
  "data": {
    "github_user": {
      "bio": "StepZen revolution",
      "followers": {
        "edges": [
          {
            "node": {
              "id": "MDU6VG9waWNqcXV1cnktdGx1Z22lu",
              "login": "In scelerisque sem at dolor"
            }
          },
          {
            "node": {
              "id": "MDU6VG9waWNqYXZhc2NyaXB0",
              "login": "Duis sagittis ipsum"
            }
          }
        ]
      }
    }
  }
}
```



**Ask what you need, get exactly that**



@gethackteam



Tell me what you want

GraphQL

what you really really want



@gethackteam

GraphQL is a **query language** for APIs

That offers a **single endpoint** for **multiple resources**

And is based on a **type system**



@gethackteam

Query Schema

RUN



```
▼ query TwitterQueries {
▼   twitter_search(query: "einstein") {
▼     edges {
▼       node {
         created_at
         text
         author {
           name
         }
       }
     }
  }
}
```

Response Documentation



```
▼ {
▼   "data": {
▼     "twitter_search": {
▼       "edges": [
▼         {
▼           "node": {
             "author": {
               "name": "stepzen_dev"
             },
             "created_at": "2020-02-04T18:36:02Z",
             "text": "RT @AlbertEinstein: Could
fundamental physical constants not be constant across
space and time? https://t.co/kpqBHTrxvx"
           }
         }
       ]
     }
  }
}
```



@gethackteam

```
Query Schema [RUN] [copy]
```

```
▼ query TwitterQueries {  
  ▼ twitter_search(query: "einstein") {  
    ▼ edges {  
      ▼ node {  
        created_at  
        text  
        author {  
          name  
        }  
      }  
    }  
  }  
}
```

```
Response Documentation [copy]
```

```
▼ {  
  ▼ "data": {  
    ▼ "twitter_search": {  
      ▼ "edges": [  
        ▼ {  
          ▼ "node": {  
            "author": {  
              "name": "stepzen_dev"  
            },  
            "created_at": "2020-02-04T18:36:02Z",  
            "text": "RT @AlbertEinstein: Could  
            fundamental physical constants not be constant across  
            space and time? https://t.co/kpqBHTrxvx"  
          }  
        }  
      ]  
    }  
  }  
}
```

**Define nested relationships**

GraphQL is a **query language** for APIs

That offers a **single endpoint** for **multiple resources**

And is based on a **type system**



@gethackteam

Query Schema

RUN



```
▼ query TwitterQueries {
  ▼ twitter_search(query: 1) {
    ▼ edges {
      ▼ node {
        created_at
        text
        author {
          name
        }
      }
    }
  }
}
```

Response

Documentation

< Twitter\_Tweet

FIELDS

**created\_at:** Date

Creation time of this account.

**description:** String

The text of this user's profile description (also known as bio), if the user provided one.

**id:** ID!

Unique identifier of this user.

**location:** String

The location specified in the user's profile, if the user provided one. As this is a freeform value, it may not indicate a valid location



@gethackteam



Query Schema

```
query TwitterQueries {  
  twitter_search(query: 1) {  
    edges {  
      node {  
        created_at  
        text  
        author {  
          name  
        }  
      }  
    }  
  }  
}
```

Response Documentation

< Twitter\_Tweet

FIELDS

**created\_at: Date**  
Creation time of this account.

**description: String**  
The text of this user's profile description (also known as bio), if the user provided one.

**id: ID!**  
Unique identifier of this user.

**location: String**  
The location specified in the user's profile, if the user provided one. As this is a freeform value, it may not indicate a valid location

**Every field has a type definition**

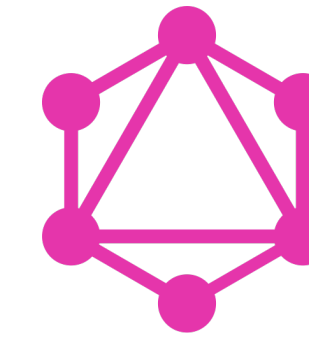
Let's compare the type systems



@gethackteam

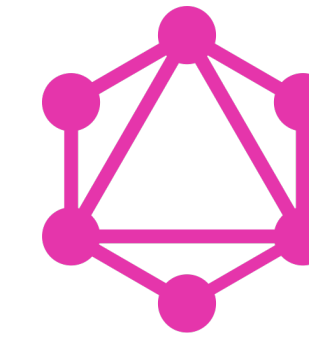


```
type Product = {  
  id: number,  
  title: string,  
  thumbnail: string,  
  price: number,  
  categories: Category[],  
  reviews?: Review  
}
```



```
type Product {  
  id: Int!  
  title: String!  
  thumbnail: String!  
  price: Float!  
  categories: [Category]!  
  reviews: Review  
}
```





```
type Product = {  
  id: number,  
  title: string,  
  thumbnail: string,  
  price: number,  
  categories: Category[],  
  reviews?: Review  
}
```

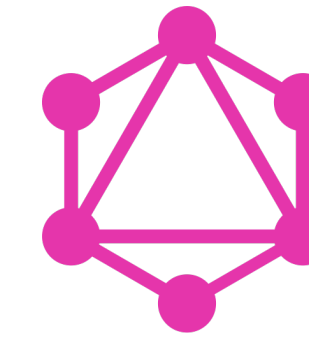
```
type Product {  
  id: Int!  
  title: String!  
  thumbnail: String!  
  price: Float!  
  categories: [Category]!  
  reviews: Review  
}
```

**Both have basic “scalar” types**



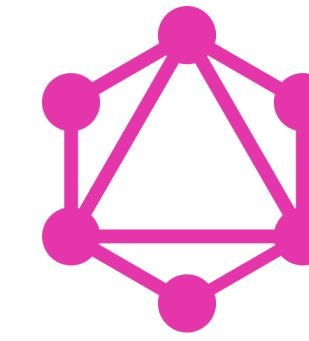


```
type Product = {  
  id: number,  
  title: string,  
  thumbnail: string,  
  price: number,  
  categories: Category[],  
  reviews?: Review  
}
```



```
type Product {  
  id: Int!  
  title: String!  
  thumbnail: String!  
  price: Float!  
  categories: [Category]!  
  reviews: Review  
}
```





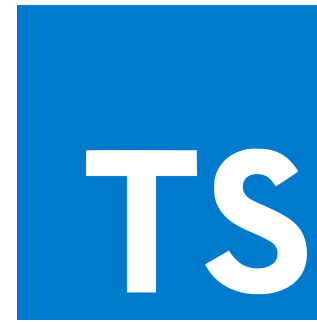
```
type Product = {  
  id: number,  
  title: string,  
  thumbnail: string,  
  price: number,  
  categories: Category[],  
  reviews?: Review  
}
```

```
type Product {  
  id: Int!  
  title: String!  
  thumbnail: String!  
  price: Float!  
  categories: [Category]!  
  reviews: Review  
}
```

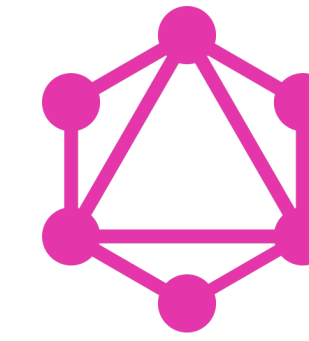
**Can relate to other types**



@gethackteam

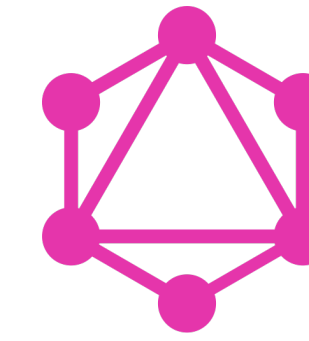


```
type Product = {  
  id: number,  
  title: string,  
  thumbnail: string,  
  price: number,  
  categories: Category[],  
  reviews?: Review  
}
```



```
type Product {  
  id: Int!  
  title: String!  
  thumbnail: String!  
  price: Float!  
  categories: [Category]!  
  reviews: Review  
}
```





```
type Product = {  
  id: number,  
  title: string,  
  thumbnail: string,  
  price: number,  
  categories: Category[],  
  reviews?: Review  
}
```

```
type Product {  
  id: Int!  
  title: String!  
  thumbnail: String!  
  price: Float!  
  categories: [Category]!  
  reviews: Review  
}
```

**And have required/optional flags**



@gethackteam

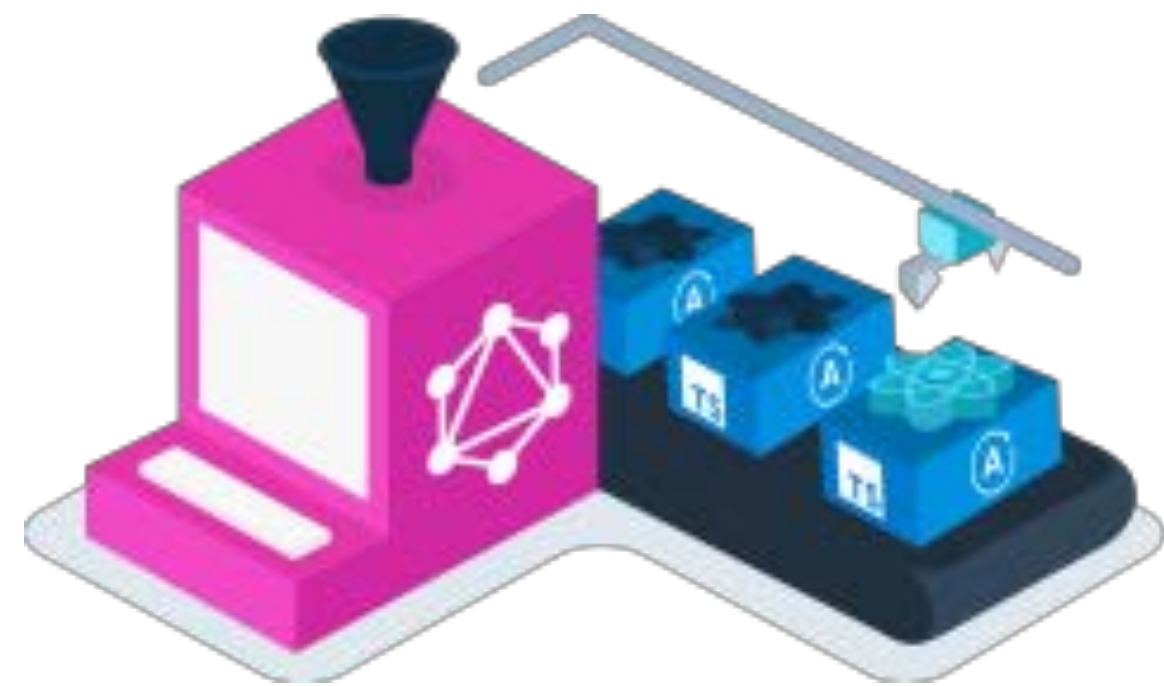


So how can we combine  
TypeScript and GraphQL?



@gethackteam

By using



{ GraphQL }

**code generator**



@gethackteam

A CLI tool that generates TypeScript types from a GraphQL schema.



@gethackteam

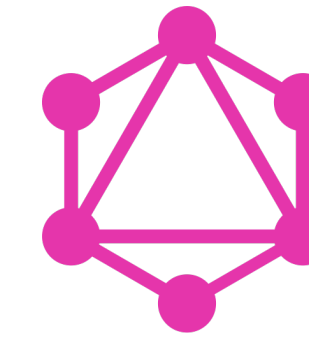
Remember both the type  
definitions?



@gethackteam



```
type Product = {  
  id: number,  
  title: string,  
  thumbnail: string,  
  price: number,  
  categories: Category[],  
  reviews?: Review  
}
```



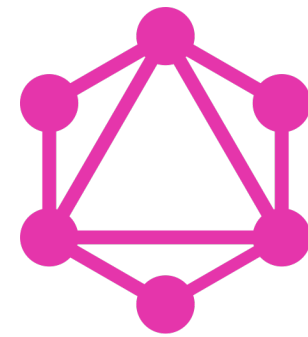
```
type Product {  
  id: Int!  
  title: String!  
  thumbnail: String!  
  price: Float!  
  categories: [Category]!  
  reviews: Review  
}
```



Using the GraphQL schema as  
source of truth



@gethackteam

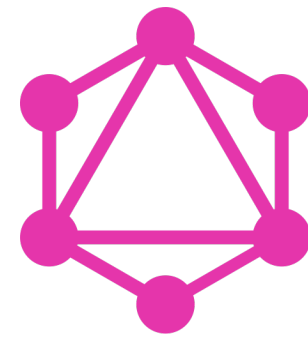


```
type Product {  
  id: Int!  
  title: String!  
  thumbnail: String!  
  price: Float!  
  categories: [Category]!  
  reviews: Review  
}
```



```
type Product = {  
  id: number,  
  title: string,  
  thumbnail: string,  
  price: number,  
  categories: Category[],  
  reviews?: Review  
}
```





```
type Product {  
  id: Int!  
  title: String!  
  thumbnail: String!  
  price: Float!  
  categories: [Category]!  
  reviews: Review  
}
```



```
type Product = {  
  id: number,  
  title: string,  
  thumbnail: string,  
  price: number,  
  categories: Category[],  
  reviews?: Review  
}
```

# Generate TypeScript types



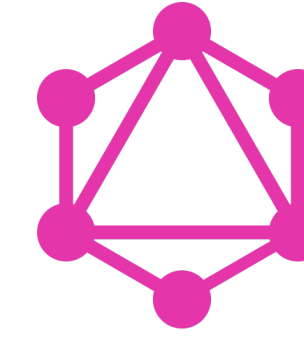
@gethackteam



TS

```
export type Scalars = {
  ID: string,
  String: string,
  Boolean: boolean,
  Int: number,
  Float: number
};

export type Product = {
  __typename?: 'Product',
  id: Scalars['Int'],
  title: Scalars['String'],
  thumbnail: Scalars['String'],
  thumbnailName: Scalars['String'],
  reviews: Review,
  offers: Array<Offer>
};
```



```
type Product {
  id: Int!
  title: String!
  thumbnail: String!
  price: Float!
  categories: [Category]!
  reviews: Review
}
```

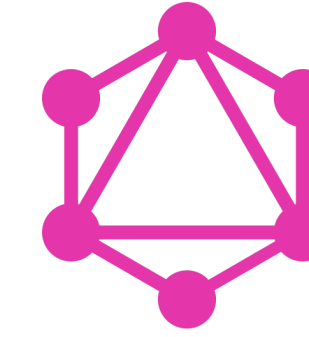


@gethackteam

TS

```
export type Scalars = {
  ID: string,
  String: string,
  Boolean: boolean,
  Int: number,
  Float: number
};

export type Product = {
  __typename?: 'Product',
  id: Scalars['Int'],
  title: Scalars['String'],
  thumbnail: Scalars['String'],
  thumbnailName: Scalars['String'],
  reviews: Review,
  offers: Array<Offer>
};
```



```
type Product {
  id: Int!
  title: String!
  thumbnail: String!
  price: Float!
  categories: [Category]!
  reviews: Review
}
```

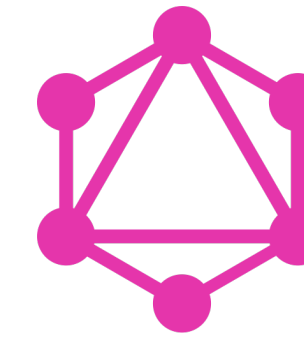


@gethackteam

TS

```
export type Scalars = {  
  ID: string,  
  String: string,  
  Boolean: boolean,  
  Int: number,  
  Float: number  
};
```

```
export type Product = {  
  __typename?: 'Product',  
  id: Scalars['Int'],  
  title: Scalars['String'],  
  thumbnail: Scalars['String'],  
  thumbnailName: Scalars['String'],  
  reviews: Review,  
  offers: Array<Offer>  
};
```



# Map basic TypeScript types to scalar types of GraphQL

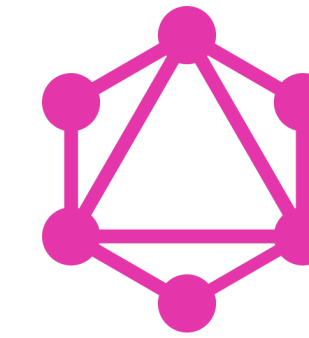
```
type Product {  
  id: Int!  
  title: String!  
  thumbnail: String!  
  price: Float!  
  categories: [Category]!  
  reviews: Review  
}
```



@gethackteam



```
export type Scalars = {  
  ID: string,  
  String: string,  
  Boolean: boolean,  
  Int: number,  
  Float: number  
};  
  
export type Product = {  
  __typename?: 'Product',  
  id: Scalars['Int'],  
  title: Scalars['String'],  
  thumbnail: Scalars['String'],  
  thumbnailName: Scalars['String'],  
  reviews: Review,  
  offers: Array<Offer>  
};
```



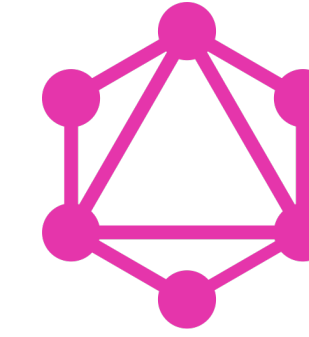
```
type Product {  
  id: Int!  
  title: String!  
  thumbnail: String!  
  price: Float!  
  categories: [Category]!  
  reviews: Review  
}
```



TS

```
export type Scalars = {
  ID: string,
  String: string,
  Boolean: boolean,
  Int: number,
  Float: number
};

export type Product = {
  __typename?: 'Product',
  id: Scalars['Int'],
  title: Scalars['String'],
  thumbnail: Scalars['String'],
  thumbnailName: Scalars['String'],
  reviews: Review,
  offers: Array<Offer>
};
```



```
type Product {
  id: Int!
  title: String!
  thumbnail: String!
  price: Float!
  categories: [Category]!
  reviews: Review
```

**Keep relation to  
non-scalar types**

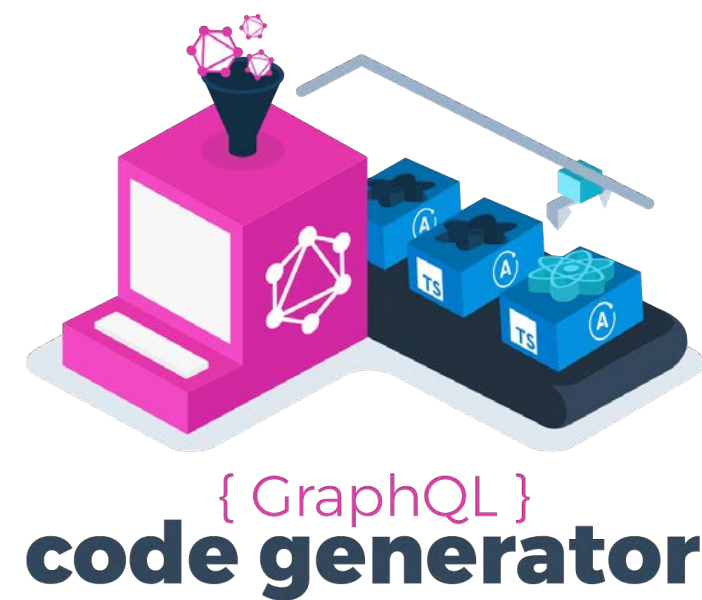


@gethackteam

Let's try it out 😎

~~NEXT~~.JS

+



=



@gethackteam

So should I start converting to  
TypeScript right away? 🤔



@gethackteam



**Roy Derks**

@gethackteam



My JavaScript codebase seconds after adding TypeScript to the project



@gethackteam



But definitely start using a  
**type system!!**



@gethackteam



**Roy Derks**



@gethackteam

**Thank you!**

**Let's stay in touch**

[stepzen.com](https://stepzen.com)

[hackteam.io](https://hackteam.io)