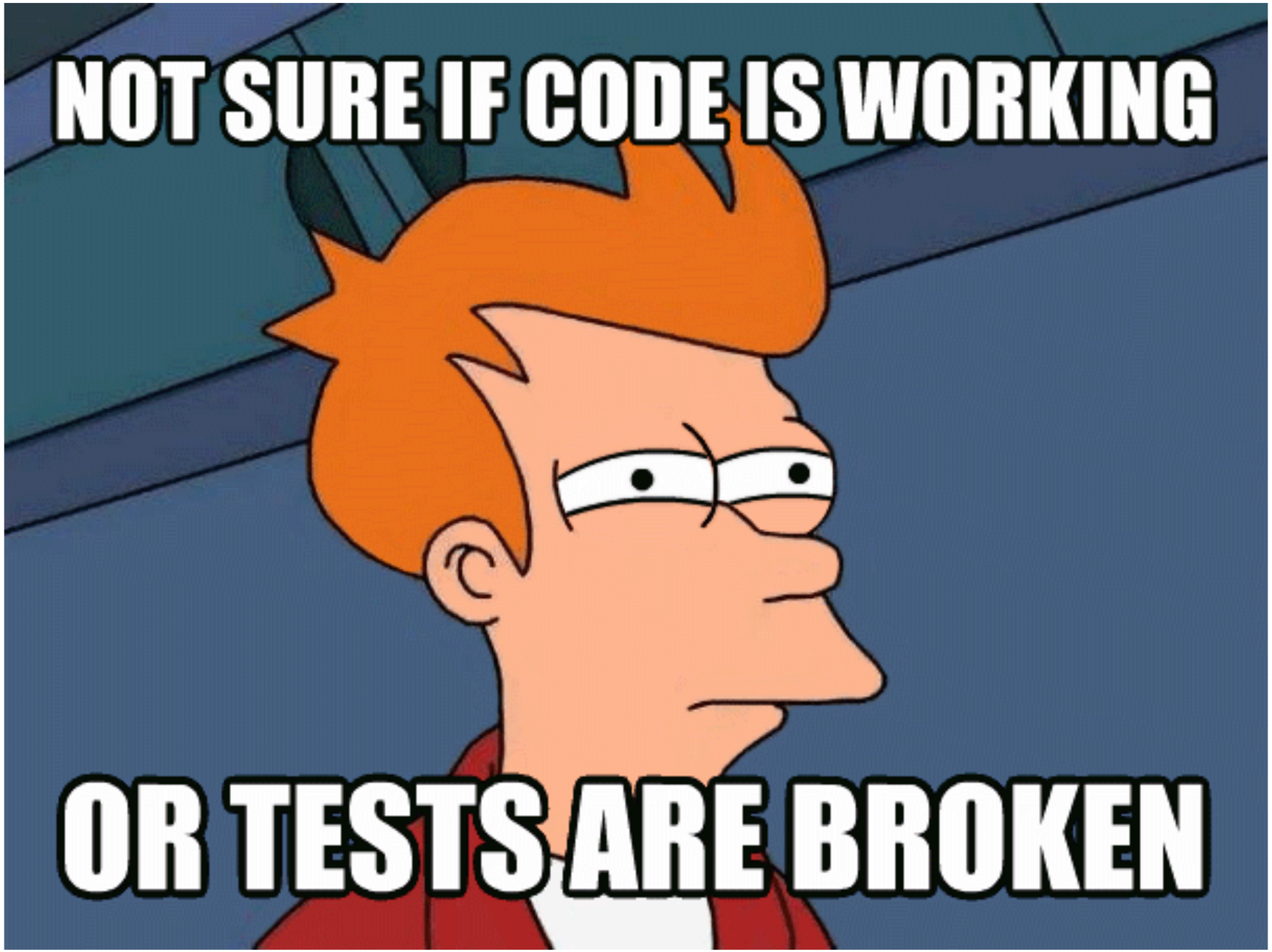
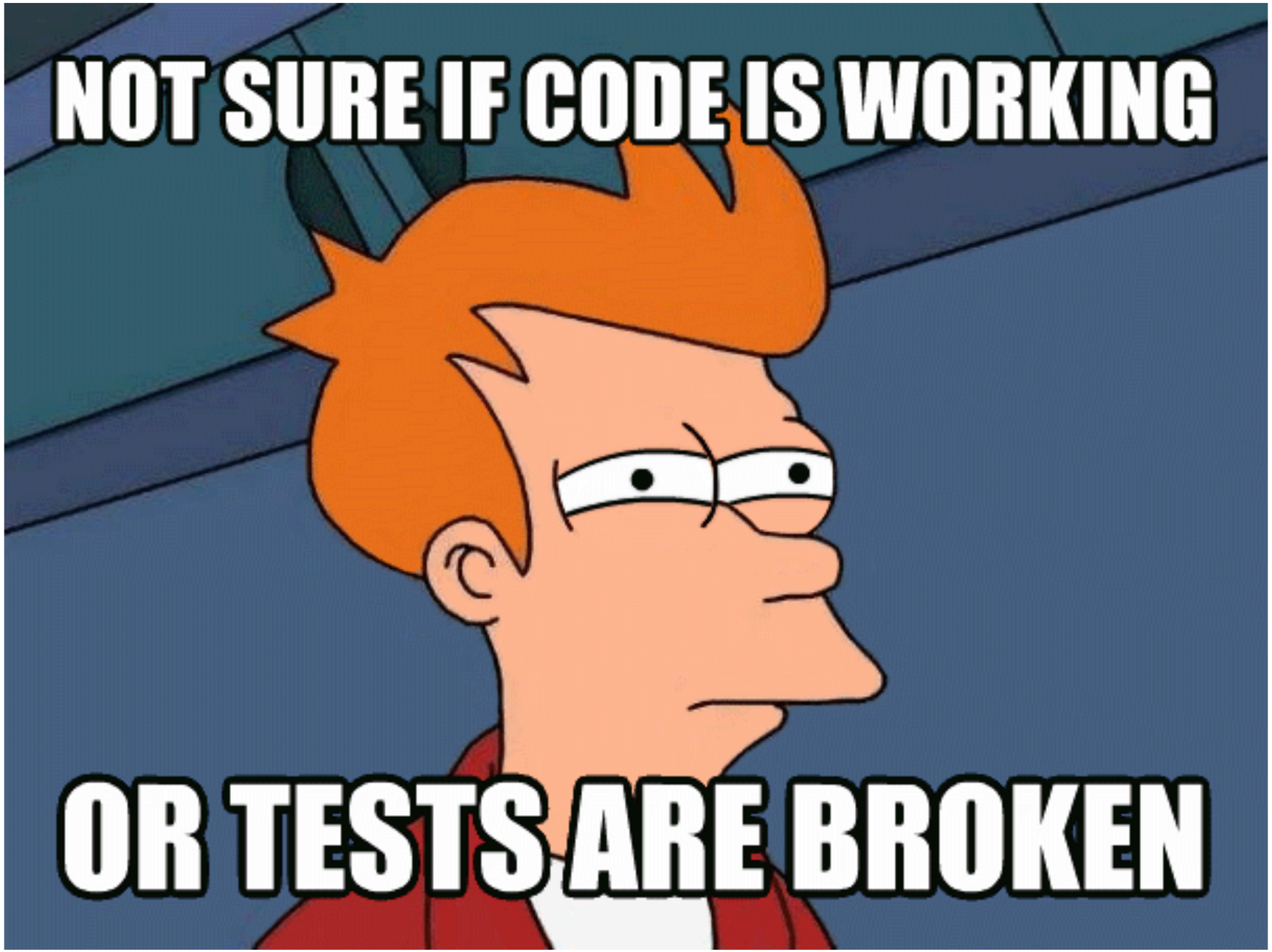
Testing GraphQL in JavaScript Applications

FOSDEM 2019
03/02/2019

What is this about?

@gethackteam


NOT SURE IF CODE IS WORKING
OR TESTS ARE BROKEN


NOT SURE IF CODE IS WORKING
OR TESTS ARE BROKEN

Who Am I?

@gethackteam

Roy Derks

@gethackteam

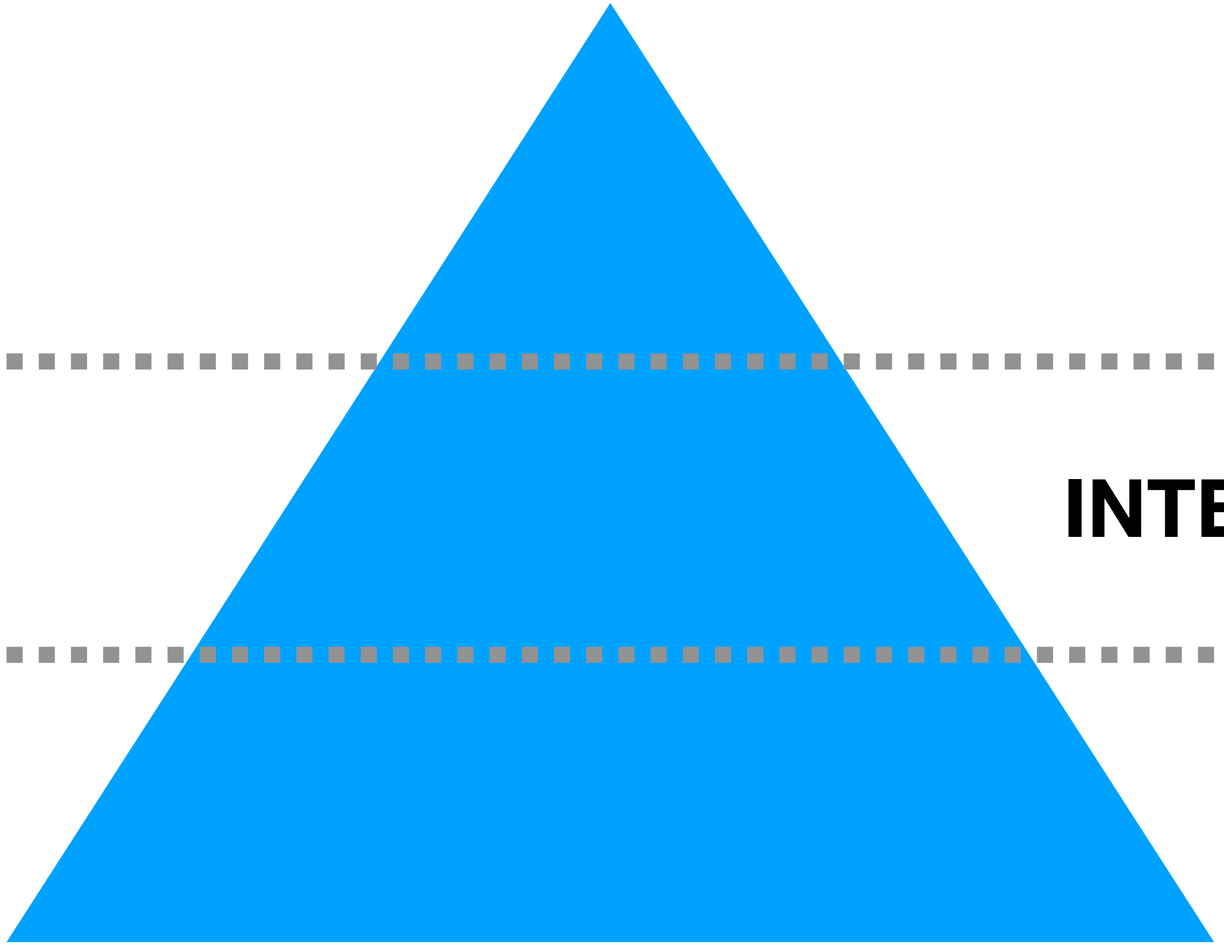
#javascriptEverywhere

#reactjs #ReactNative #GraphQL

Who is this for?

@gethackteam

Who is this for?



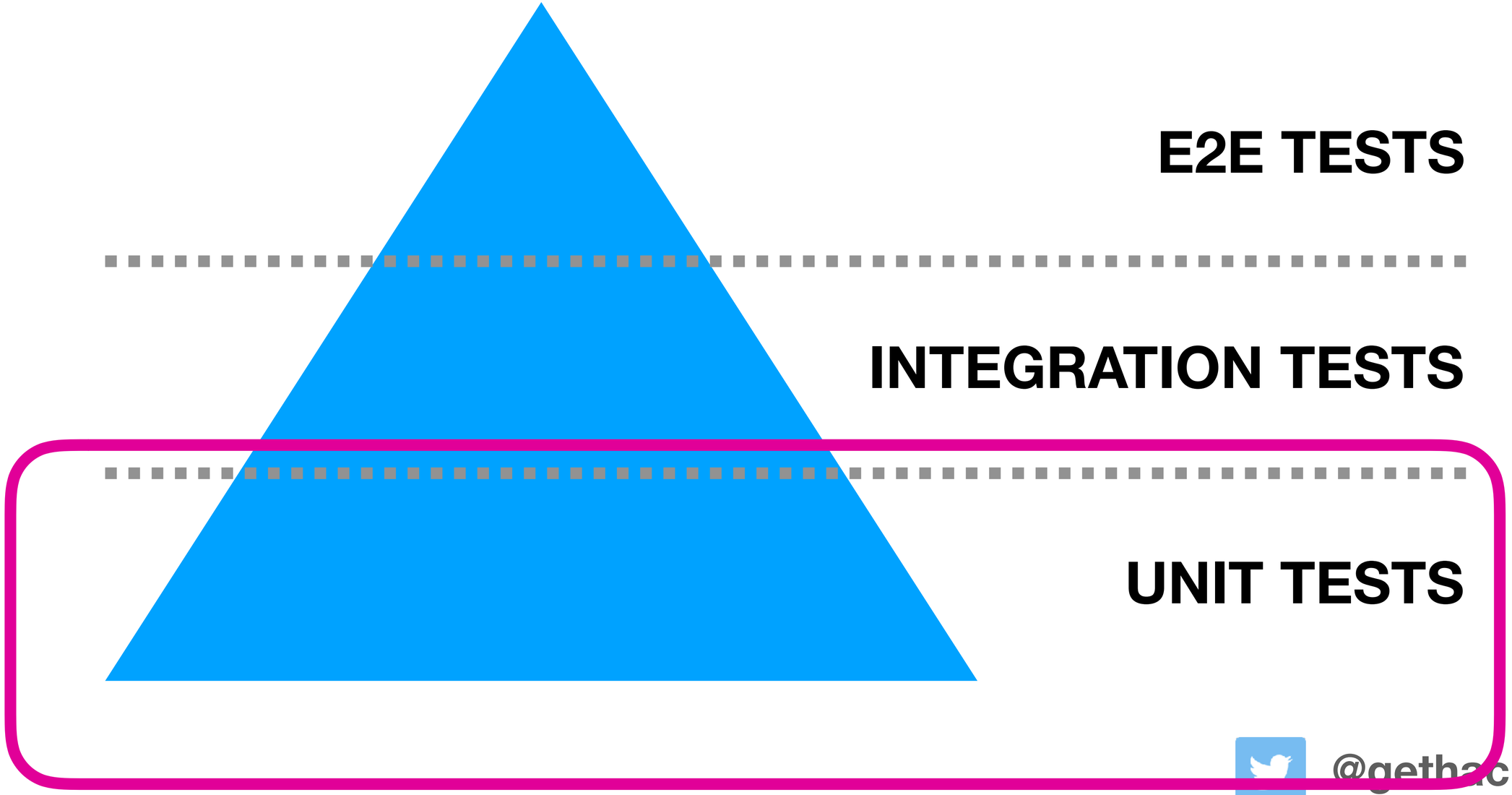
E2E TESTS

INTEGRATION TESTS

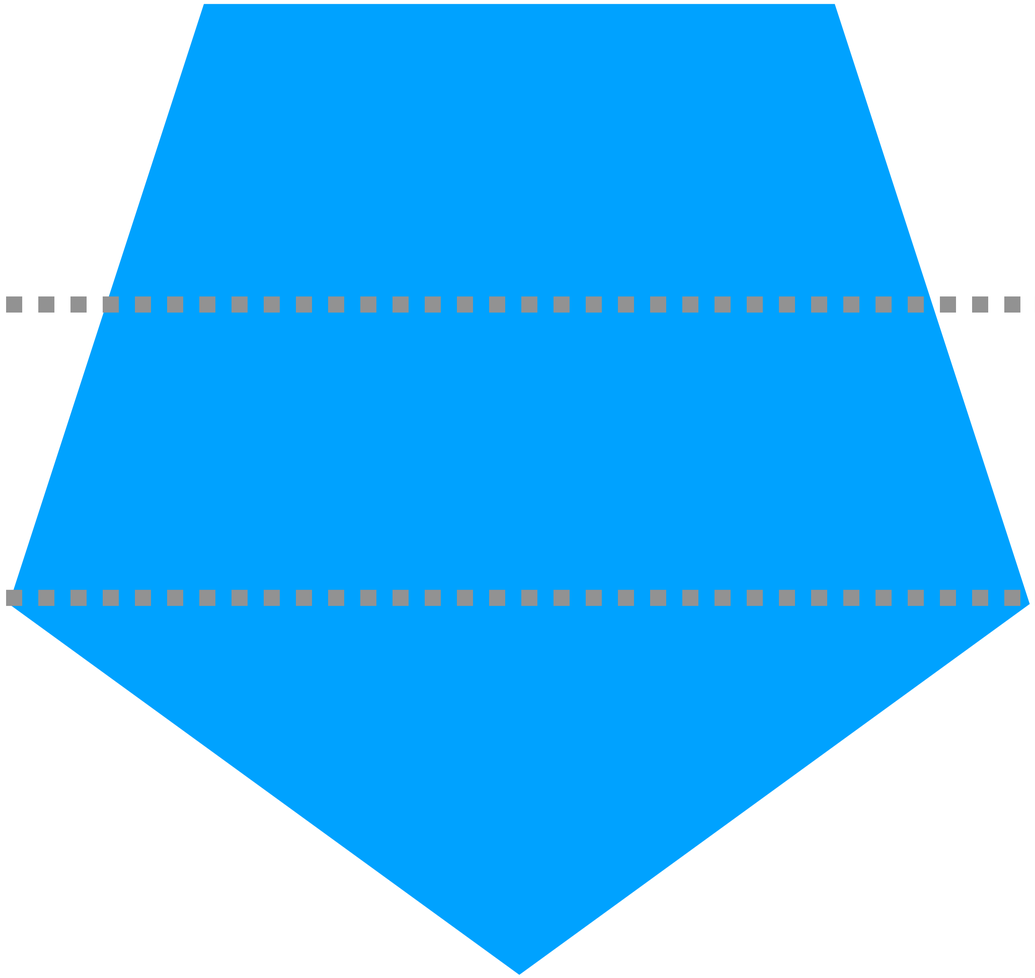
UNIT TESTS

@gethackteam

Who is this for?



Who is this for?



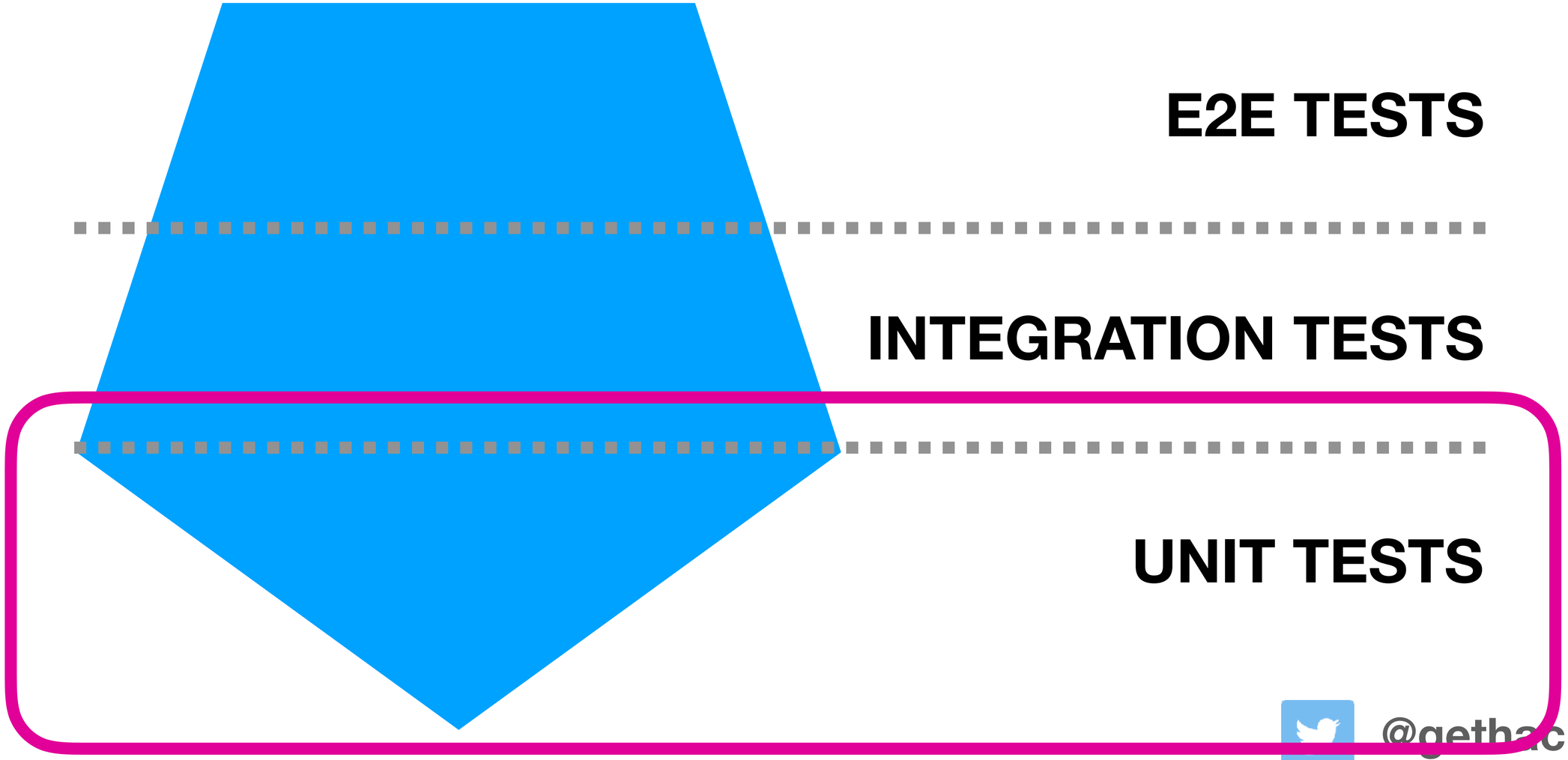
E2E TESTS

INTEGRATION TESTS

UNIT TESTS

@gethackteam

Who is this for?



Why write tests?

@gethackteam

Why write tests?

Check if our code behaves as expected

Why write tests?

Check if our code behaves as expected

Check if our code handles errors and edge cases correctly

Why write tests?

Check if our code behaves as expected

Check if our code handles errors and edge cases correctly

Check if our syntax is handled correctly

Testing JavaScript

@gethackteam

Testing JavaScript

@gethackteam

Enzyme

Testing JavaScript

@gethackteam

Enzyme

Testing JavaScript

@gethackteam

Enzyme

Testing JavaScript

@gethackteam

Let’s look at an example

@gethackteam

Let’s look at an example

```
type Product {
  id: Int!
  title: String!
  thumbnail: String!
  reviews: Review
  offers: [Offer]
}
type Review {
  productId: Int!
  count: Int
  average: Float
}
type Offer {
  productId: Int!
  reseller: String
  price: Float
}
```

```
type Query {
  products: [Product]
  product(id: Int!): Product
}
type Mutation {
  addOffer(productId: Int!, reseller: String, price: Float): Offer
}
```

@gethackteam

Let's look at an example

```
type Product {
  id: Int!
  title: String!
  thumbnail: String!
  reviews: Review
  offers: [Offer]
}
type Review {
  productId: Int!
  count: Int
  average: Float
}
type Offer {
  productId: Int!
  reseller: String
  price: Float
}
```

Query

```
type Query {
  products: [Product]
  product(id: Int!): Product
}
type Mutation {
  addOffer(productId: Int!, reseller: String, price: Float): Offer
}
```

@gethackteam

Let's look at an example

```
type Product {
  id: Int!
  title: String!
  thumbnail: String!
  reviews: Review
  offers: [Offer]
}
type Review {
  productId: Int!
  count: Int
  average: Float
}
type Offer {
  productId: Int!
  reseller: String
  price: Float
}
```

Query

```
type Query {
  products: [Product]
  product(id: Int!): Product
}
type Mutation {
  addOffer(productId: Int!, reseller: String, price: Float): Offer
}
```

Mutation

@gethackteam

Test Components

@gethackteam

Testing Component

```
import TestRenderer from "react-test-renderer";

it("should render without error", () => {
  TestRenderer.create(
    <Products />
  );
});
```

@gethackteam

Testing Component

```
import TestRenderer from "react-test-renderer";

it("should render without error", () => {
  TestRenderer.create(
    <Products />
  );
});
```

Invariant Violation: Could not find "client" in the context of Query or as passed props. Wrap the root component in an <ApolloProvider>

@gethackteam

Testing Component

```
import { ApolloProvider } from "react-apollo";
import ApolloClient from "apollo-boost";
import TestRenderer from "react-test-renderer";

const client = new ApolloClient({
  uri: "<CLIENT_URL>"
});

it("should render without error", () => {
  TestRenderer.create(
    <ApolloProvider client={client}>
      <Products />
    </ApolloProvider>
  );
});
```

@gethackteam

Testing Component

```
import { ApolloProvider } from "react-apollo";
import ApolloClient from "apollo-boost";
import TestRenderer from "react-test-renderer";

const client = new ApolloClient({
  uri: "<CLIENT_URL>"
});

it("should render without error", () => {
  TestRenderer.create(
    <ApolloProvider client={client}>
      <Products />
    </ApolloProvider>
  );
});
```

Tests against “live” data

@gethackteam

MockedProvider

```
import TestRenderer from "react-test-renderer";
import { MockedProvider } from "react-apollo/test-utils";

it("should render without error", () => {
  TestRenderer.create(
    <MockedProvider mocks={[]}>
      <Products />
    </MockedProvider>
  );
});
```

@gethackteam

MockedProvider

```
import TestRenderer from "react-test-renderer";
import { MockedProvider } from "react-apollo/test-utils";

it("should render without error", () => {
  TestRenderer.create(
    <MockedProvider mocks={[]}>
      <Products />
    </MockedProvider>
  );
});
```

Tests against mock data

@gethackteam

MockedProvider

```
import TestRenderer from "react-test-renderer";
import { MockedProvider } from "react-apollo/test-utils";

it("should render without error", () => {
  TestRenderer.create(
    <MockedProvider mocks={[]}>
      <Products />
    </MockedProvider>
  );
});
```

```
{
  "request": {
    "query": "<QUERY>",
    "variables": { /* variables */ }
  },
  "result": {
    "data": {
      /* result */
    }
  }
}
```

Tests against mock data

@gethackteam

Test Schemas

@gethackteam

Test Schemas

easygraphql-tester

```
import EasygraphqlTester from "easygraphql-tester";

it("Should be a valid query", () => {
  schema = '<GRAPHQL_SCHEMA>';
  tester = new EasygraphqlTester(schema);

  tester.test(true, '<QUERY>');
});

it("Should be a valid mutation", () => {
  schema = '<GRAPHQL_SCHEMA>';
  tester = new EasygraphqlTester(schema);

  tester.test(true, '<MUTATION>', { /* variables */ });
});
```

@gethackteam

Test Schemas

easygraphql-tester

```
import EasygraphqlTester from "easygraphql-tester";

it("Should be a valid query", () => {
  schema = '<GRAPHQL_SCHEMA>';
  tester = new EasygraphqlTester(schema);

  tester.test(true, '<QUERY>');
});

it("Should be a valid mutation", () => {
  schema = '<GRAPHQL_SCHEMA>';
  tester = new EasygraphqlTester(schema);

  tester.test(true, '<MUTATION>', { /* variables */ });
});
```

Tests schema directly

@gethackteam

Test Schemas

easygraphql-tester

```
import EasygraphqlTester from "easygraphql-tester";

it("Should be a valid query", () => {
  schema = '<GRAPHQL_SCHEMA>';
  tester = new EasygraphqlTester(schema);

  tester.test(true, '<QUERY>');
});

it("Should be a valid mutation", () => {
  schema = '<GRAPHQL_SCHEMA>';
  tester = new EasygraphqlTester(schema);

  tester.test(true, '<MUTATION>', { /* variables */ });
});
```

Tests schema directly

Test with (in)valid fields

@gethackteam

Test Schemas

```
import EasygraphqlTester from "easygraphql-tester";

it("Should return valid data types", () => {
  const { products } = tester.mock(GET_PRODUCTS_QUERY);

  expect(Array.isArray(products)).toBe(true);
});
```

@gethackteam

Test Schemas

Tests schema mocks

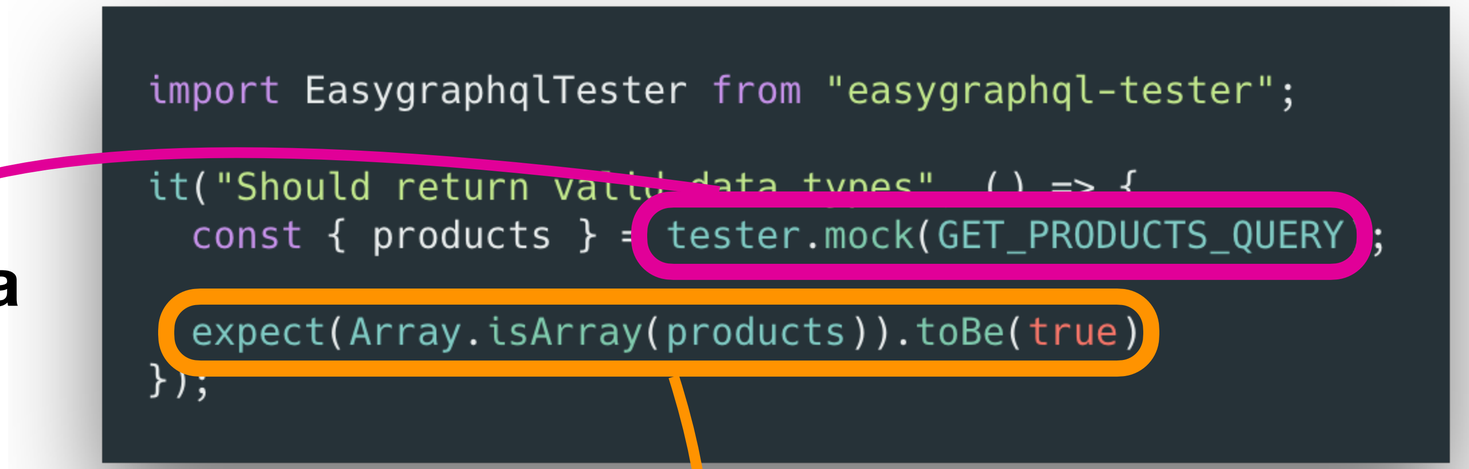
```
import EasygraphqlTester from "easygraphql-tester";

it("Should return valid data types", () => {
  const { products } = tester.mock(GET_PRODUCTS_QUERY);

  expect(Array.isArray(products)).toBe(true);
});
```

@gethackteam

Test Schemas



Tests schema mocks

Test format of result

@gethackteam

https://github.com/royderks/
react-apollo-testing

@gethackteam

To summarise

@gethackteam

Want to learn more?

 @gethackteam **#javascriptEverywhere**

https://www.apollographql.com/

https://easygraphql.com/