**Gatsby & GraphQL**

**Gatsby**

*Npm install -g Gatsby-cli*

*Gatsby new project\_name*

This will generate a project with files/folders included. Predominantly you will be working out of the src folder.

Src > components

> Images

> Pages

Look in the package.json file to view available scripts. *Npm start* will compile and watch for changes via hot reloading.

There are existing components included within a Gatsby project such as the Link component, which is similar to an anchor tag, however uses the prop *to* as opposed to the attribute *href* to route:

<Link to=”/”>Homepage</Link>

You can view certain meta data in Gatsby-config.js such as site title. These can be accessed in components. The Gatsby-config.js file is also where you can add plugins.

How to create a page

In src > components > pages:

Add a file, e.g. *about.js*

*About.js*

Const AboutPage = () => (

<div>

<h1>Hello World</h1>

</div>

);

Export default AboutPage;

Similar to nextjs (which is react on the server), gatsby does all the routing for you based on the name of the page. You do not need to specify a route for the about page above, it will automatically be accessible via [*http://localhost:8000/about*](http://localhost:8000/about).

Navigation menu

If you want to create a component, such as a navigation bar, across all components, create a menu component in the components directory and import it into layout.js. Notice how the pre-existing pages are wrapped in this Layout component.

CSS

Some of the pre-existing components use inline CSS via react using the style attribute, you can also create external CSS files and import these in: import ‘./style.css’

**GraphQL**

GraphQL is a query language for APIs created by Facebook which gives clients the power to ask for exactly what they need and nothing more.

GraphiQL is the GraphQL IDE you will often use while building Gatsby websites, it can be accessed via [*http://localhost:8000/\_\_\_graphql*](http://localhost:8000/___graphql).

What does it look like

*Local JSON data:*

{

“user”: {

“id”: 100,

“name”: “Ali Issaee”,

“email”: [rezaa91@hotmail.co.uk](mailto:rezaa91@hotmail.co.uk)

}

}

*GraphQL Query*

{

User(id: 100) {

Id,

Email

}

}

Note how the query above is passed an argument to only return the user with the id of 100, and it is only asking for the id and email fields, it is not concerned with the *name* field.

GraphQL types

GraphQL APIs are organised in terms of types and fields:

Type Query {

user: User

}

Type User {

Name: String

Age: Int

Friends: [User]

}

Example

*Npm I express express-graphql graphql*

*Server.js*

const express = require(‘express’);

const expressGraphQL = require(‘express-graphql’);

const {

GraphQLSchema,

GraphQLObjectType,

GraphQLString,

GraphQLInt,

} = require(‘graphql’)

const app = express();

const schema = new GraphQLSchema({

query: new GraphQLObjectType({

name: ‘HelloWorld’

fields: () => ({

message: {

type: GraphQLString,

resolve: () => ‘Hello World’

},

Age: {

type: GraphQLInt,

resolve: () => 28

})

})

})

app.use(‘/graphql’, expressGraphQL({

schema,

graphiql: true // use IDE in browser via /graphql route

}))

App.listen(5000, () => console.log(‘server running…’));

GraphiQL

In the /graphql route, you should now see the GraphiQL IDE. Click *Docs* on the right hand side to see available queries which will currently only be *HelloWorld*. Type in the query:

{

Message

}

To get the output:

{

“data”: {

“message”: “Hello World”

}

}

Example 2

*Server.js*

Const books = [

{name: “Book 1”, author: “GRRM”, id: 1},

…

]

Const authors = [{}, {}, {}];

Const RootQueryType = new GraphQLObjectType({

Name: “Query”,

Description: “Root query”,

Fields: () => ({

Books: {

Type: new GraphQLList(BookType),

Description: “list of books”,

Resolve: () => books

},

Book: {

Type: BookType,

Description: “A single book”,

Args: {

Id: {type: GraphQLInt}

},

Resolve: (parent, args) => books.find(book => book.id === args.id)

}

})

})

Const BookType = new GraphQLObjectType({

Name: “Book”,

Description: “book list”,

Fields: () => ({

Id: {type: GraphQLInt},

Name: {type: GraphQLString},

Author: {type: GraphQLString}

})

})

// note how fields match the properties in the books array at the top of the file

*GraphiQL*

// get all books

{

Books {

Name

Author

}

}

// get book with id of 1

{

Book(id:1) {

name

}

}

**TEST**

1. Create a new Gatsby project
2. Create a new page in the Gatsby project and access it via its route
3. Create navigation component to be used across all pages
4. For the page created, use inline styling and an external CSS file
5. What is GraphQL?
6. Create an express server with local data, create a graphql query object which can access this local data via /graphql route
7. Create your own graph *type* to be used in queries
8. In GraphiQL IDE, select certain aspects of your data
9. Use an argument to access one specific resource from a collection