Npm install -g gatsby-cli

Gatsby new projeAdi => Default template

* dosyaAdi.js ile pages klasoru altinda olsuturdugumuz her dosya aslinda url olarakda ulasilabilir duruma geliyor. Example: localhost:8000/dosyaAdi.js

**Link kullanimi**

import { Link } from "gatsby"

Yukaridaki gibi import ettikten sorna asagidaki gibi kullanabiliriz. Bu bize sayfalar arasi refesh yapmadan gecisi saglayacak.

<Link to="/page-2/">Go to page 2</Link>

**Layout kullanimi**

/src/components larda layout.js adinda genelde dosya olsuturulur ve icine asagidaki yapida bir sistem kurulur ve Header ve Footer gibi sabit olacak seyler icinde tutulur degisecek cseyler children yapisiyla dynamic olarak cekilir.

    <>

      <Header />

        <main>{children}</main>

      <Footer />

    </>

**InlineCSS**

<div

        style={{

          margin: `0 auto`,

          maxWidth: 960,

          padding: `0 1.0875rem 1.45rem`,

        }}

      >

**GlobalCSS**

Kucuk projelerde layout.css acilir ve icinde tum komponentlarda gecen cssleri duzenleyebiliriz. Layout.css ‘i sadece layout.js icine import etsek yeterli olur diger componentlara page lere import etmemize gerek yoktur.

**CSSModules**

Olusturmak istedigimiz page veya component icin dosyaAdi.module.css/scss diye isimlendirerek css dosyamizi olusturur ve bunu birdegisken adi tanimlayarak (styles gibi) import ederiz. Ve classname leri styles.name gibi kullanariz dosyamizin icinde.

İndex.module.scss

.name { font-size: 25px; }

İndex.js

İmport styles from ‘./index.module.scss’

<div className={styles.name}> … </div>

**Styled-Components**

Ilk once Gastby Plugins kismina gibip gatsby-plugin-styled-components isimli plugini veya sadece styled components yazarak da aratabilirsiniuz ve daha sorna oradaki yukleme adimlarini takip ediyorsunuz.

buttons.js => Icinde degisik button componentlarimizi barindirabilir ve bunlari daha sornadan diger klasorlerde import edip kullanabiliriz asagidaki gibi.

import styled from 'styled-components';

export const SuccessButton = styled.button`

  background: green;

`

export const DangerButton = styled.button`

  background: red;

`

Body.js

import { DangerButton } from '../components/Buttons.js';

<DangerButton>Uyari Butonu</DangerButton

## Configuration options

gatsby-config.js icinde sadece asagida ki property ‘ler degerler olabilir ayarlanabilir.

1. [siteMetadata](https://www.gatsbyjs.org/docs/gatsby-config/#sitemetadata) (object)
2. [plugins](https://www.gatsbyjs.org/docs/gatsby-config/#plugins) (array)
3. [pathPrefix](https://www.gatsbyjs.org/docs/gatsby-config/#pathprefix) (string)
4. [polyfill](https://www.gatsbyjs.org/docs/gatsby-config/#polyfill) (boolean)
5. [mapping](https://www.gatsbyjs.org/docs/gatsby-config/#mapping-node-types) (object)
6. [proxy](https://www.gatsbyjs.org/docs/gatsby-config/#proxy) (object)
7. [developMiddleware](https://www.gatsbyjs.org/docs/gatsby-config/#advanced-proxying-with-developmiddleware) (function)

**EXAMPLE siteMetadata**

Gatsby.config icinde siteMetadata yi asagida ki sekilde ayarlayalim. Ve daha sorna useStaticQuery leri asagidaki gibi kullanabiliriz

  siteMetadata: {

    title: `Gatsby Default Starter`,

    description: `Kick off your next, great Gatsby project with this default starter. This barebones starter ships with the main Gatsby configuration files you might need.`,

    author: `@mucahidyazar`,

    data: ["item 1", "item 2"],

    person: {

      name: "peter",

      age: 28

    }

  },

**useStaticQuery**

Once useStaticQuery ve graphql metodlarini gatsby kutuphanesinden import ediyoruz sonra asagidaki sekilde cekiyoruz verilerimizi.

import React from "react"

import { useStaticQuery, graphql } from "gatsby"

const Header = () => {

  const data = useStaticQuery(graphql`

    {

      site {

        siteMetadata {

          title

          description

          author

          data

          person {

            name

            age

          }

        }

      }

    }

  `)

  return <pre> {JSON.stringify(data, null, 4)} </pre>

}

export default Header

**alias**

Asagidaki info gibi alias lar query tanimlarinda belirlersek siteMetadata gibi uzun isimleri datayi kullanirken kullanmak zorunda kalmayiz. Yani siteMetadata yerine info yazmamiz yeterli olur return degerindeki gibi.

import React from "react"

import { useStaticQuery, graphql } from "gatsby"

const Header = () => {

  const data = useStaticQuery(graphql`

    {

      site {

**info:** siteMetadata {

          title

          description

          author

          data

          person {

            name

            age

          }

        }

      }

    }

  `)

  return <pre> {data.site.info.title} </pre>

}

export default Header

**PageQuery**

PageQueryler asagidaki gibi tanimlanir ve herbir pageQuery ‘e tanimlandiktan sonra props icinden erisebiliriz. Yani pageQueryler kisaca propsun icine yerlesirler.

import React from "react"

import HeaderJson from "../examples/Header"

import { graphql } from "gatsby"

const examples = (props) => {

  const { data } = props

  return (

    <div>

      <HeaderJson />

    </div>

  )

}

export const data = graphql`

  {

    site {

      info: siteMetadata {

        title

        description

        author

        data

        person {

          name

          age

        }

      }

    }

  }

`

export default examples

**Resim Yukleme Kullanma**

Gatsby Pluginslerden gatsby-source-filesystem pluginini yukluyoruz oradaki talimatlara gore.

Daha sorna gatsby.config.js dosyasindaki pluginslere asagidaki kismi ekliyoruz. Zaten pluginiun ysafasindada bu yukleme talimatini goreceksiniz.

    {

      resolve: `gatsby-source-filesystem`,

      options: {

        name: `images`,

        path: `${\_\_dirname}/src/images`,

      },

    },

Artik resimler icin olusturdugumuz dosyaya asagidaki sekilde erisebiliriz.

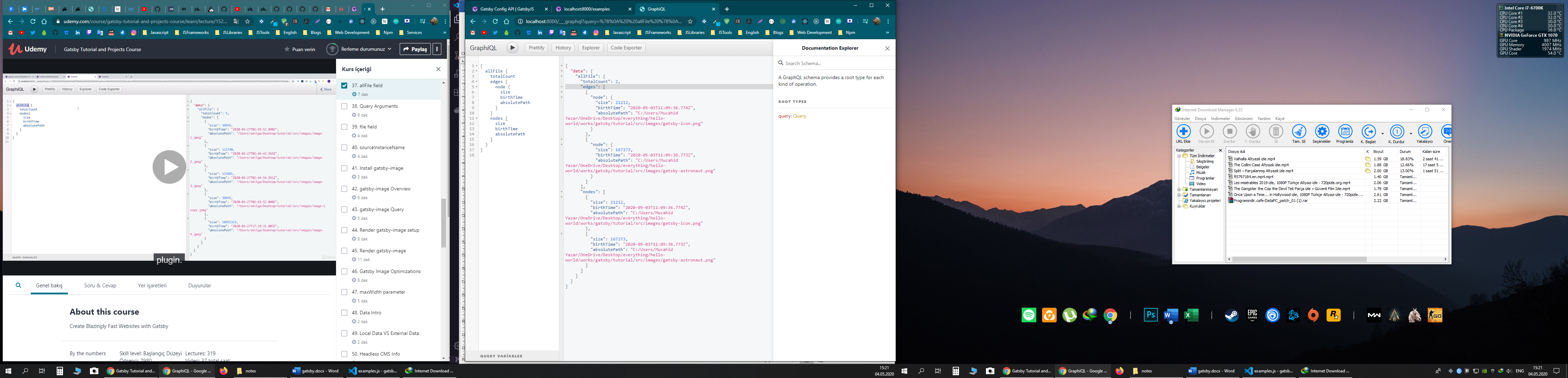
Not: Illa resimler olmak zorunda degil bu sekilde istediginiz turde istediginiz sekilde dosyalar kullanabilir erisebilirsiniz.

totalCount => Toplam dosya sayisi

edges = eski method

nodes = yeni metod

Ulasmak icin ya edges yada nodes kullanabiliriz. Her iki yontyemde kullanilabilir istege bagli ama genelde nodes kullanilir.



**TERIMLER**

relativeDirectory => klasor konumu yoludur.

###Eger images gatsby.config.js de dosya yolu olarak atanmissa

/images/cat-images => relativeDirectory: { eq: “cat-images” }

relativePath => dosya konumu yoludur.

###Eger images gatsby.config.js de dosya yolu olarak atanmissa

/images/cat-images => relativePath: { eq: “cat-images/image-001.jpg” }

Eq => Euqal demek yani esittir anlaminda

relativeDirectory: { eq: “cat-images” }

relativePath: { eq: “cat-images/image-001.jpg” }

**QUERIES**

### 5 tane dosya goster demek

LIMITING

allFile(limit: 5)

### 1. Dosyayi atla gosterme demek

SKIPPING

allFile(skip: 1)

### size ‘a gore sirala order ‘da fazladan aza dogru olsun demek

SORTING

allFile(sort: { fields: size, order: ASC })

###Elbette cok fazla secenek var fakat asagida ki yontemle diyelim images icinde copy adinda bir klasorumuz var onun icindeki resimleri sadece alabiliriz.

FILTER

allFile(filter: { relativeDirectory:{ eq:"copy"} })

Istersek coklu queries de yapabiliriz asagidaki gibi

allFile(limit: 5, skip: 1)

{

file(relativePath: { eq: "copy/image-4.jpg" }) {

size

relativePath

}

}

sourceInstaceName => gatsby.config.js dosyasinda ki gatsby-cource-filesystem ile tanimaldigimiz yollarin isimlerini filtreler. Yani sadece postlari imageleri gibi olusturdugumuz pathleri filtreleyebiliriz.

allFile(filter:{sourceInstanceName: { eq: "posts" }})

**GATSBY-IMAGE**

Plugins ler sayfasindaki gatsby-image eklentisine gidiyoruz ve yukleme talimatlarini gerceklestiriyoruz.

npm install --save gatsby-image

npm install --save gatsby-transformer-sharp gatsby-plugin-sharp

plugins: [

{

resolve: `gatsby-source-filesystem`,

options: {

name: `images`,

path: path.join(\_\_dirname, `src`, `images`),

},

},

`gatsby-plugin-sharp`,

`gatsby-transformer-sharp`,

],