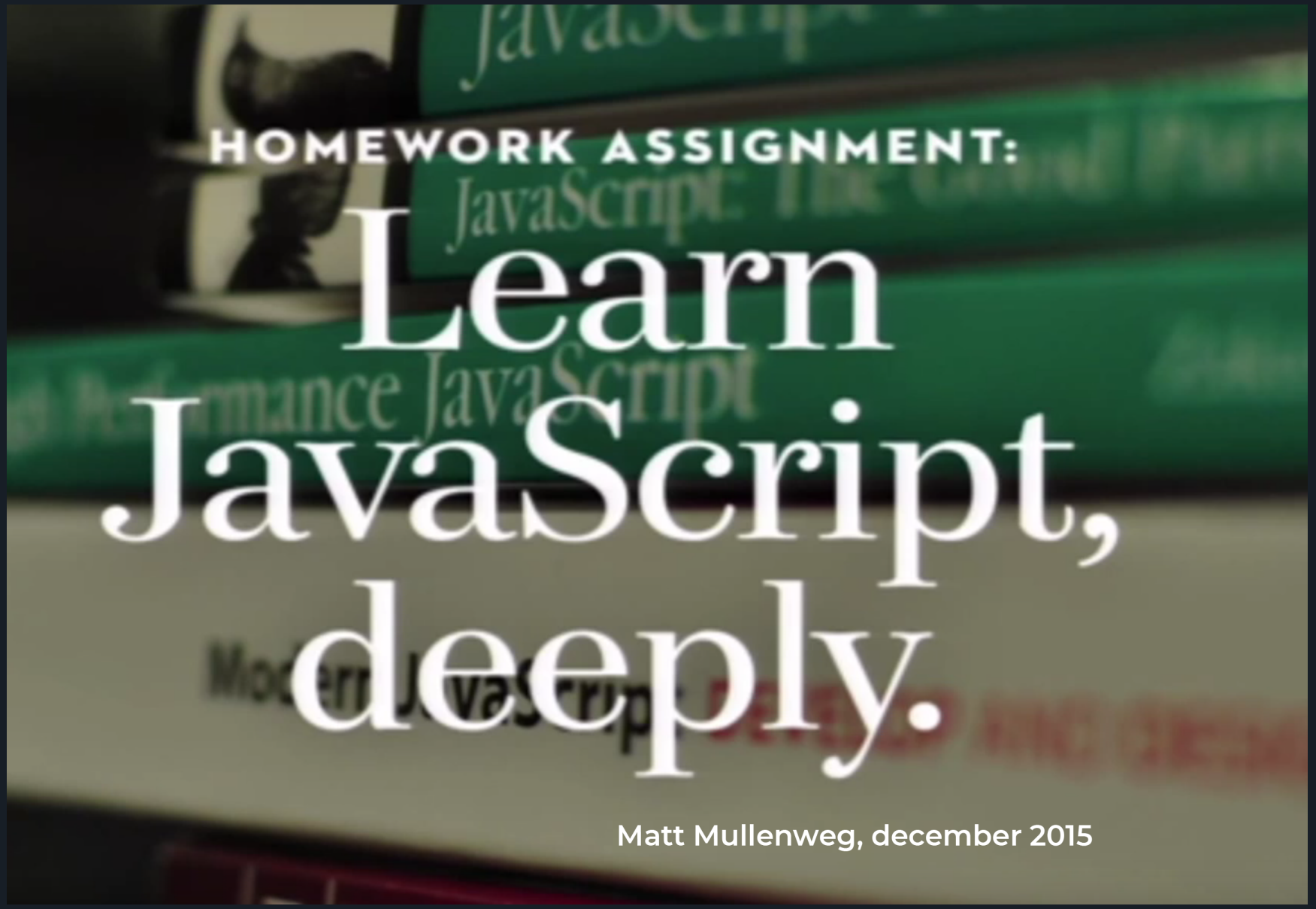
HEADLESS WORDPRESS

with GraphQL & ReactJS

TO FOLLOW ALONG

https://ebeldev.github.io/headlesswordpress-slides/


HOMEWORK ASSIGNMENT:
Learn JavaScript, deeply.
Matt Mullenweg, december 2015

THE GOAL FOR TODAY

- Use Wordpress as a backend service api

& use new stack to deliver the front-end

Front-end stack for today = ReactJS and GraphQL

ÉTIENNE BÉLANGER

Self-taugh web developper

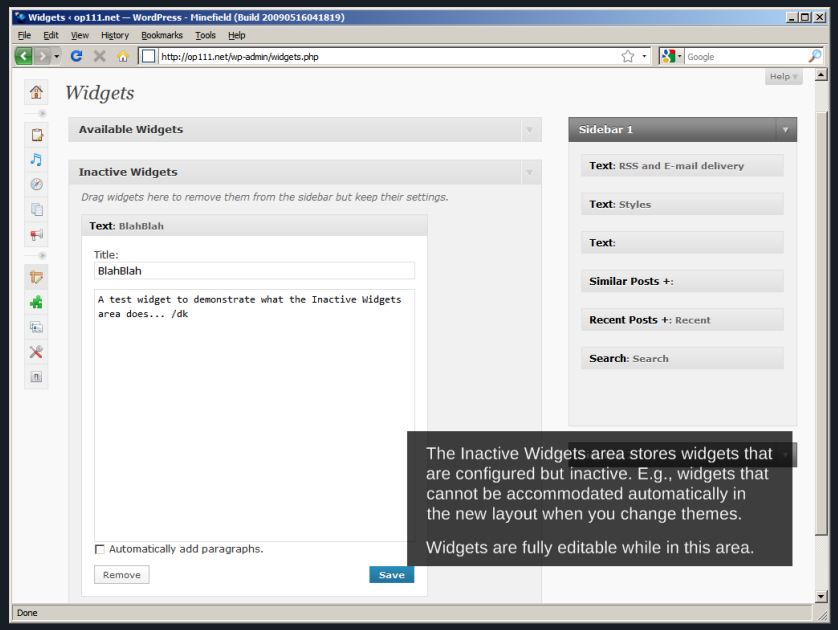
WordPress lover for more then 10 years

Work @ DistrictWeb as full-stack developer

Contact

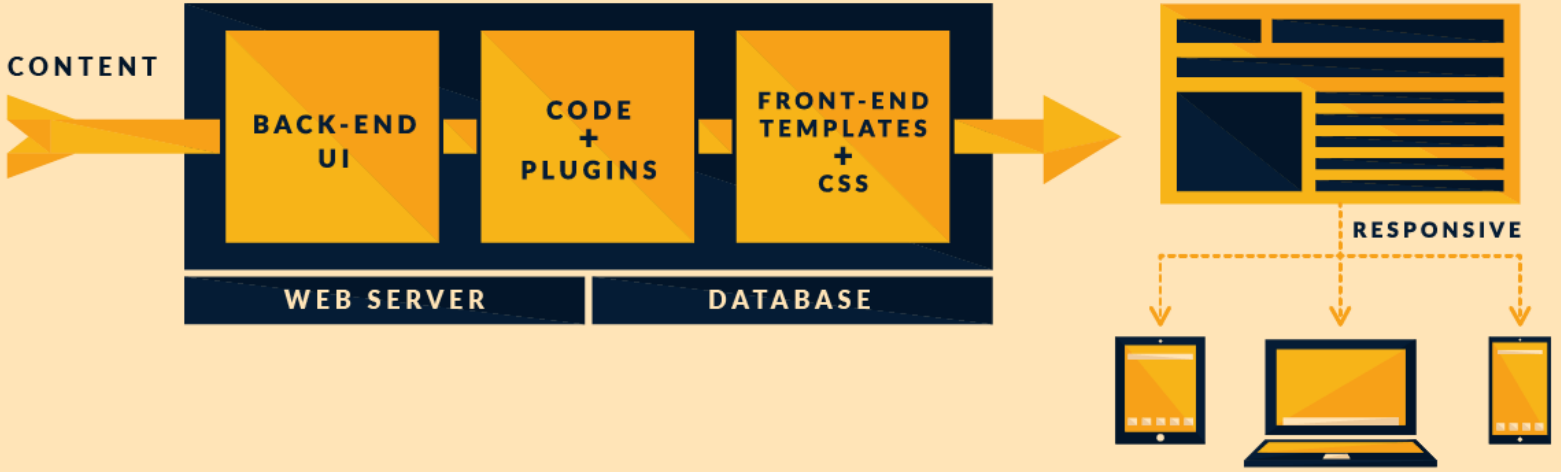
- etienne@etiennebelanger.com
- belangeretienne
- @ebeldev
- @ebeldev

THE WEB HAS CHANGED A LOT

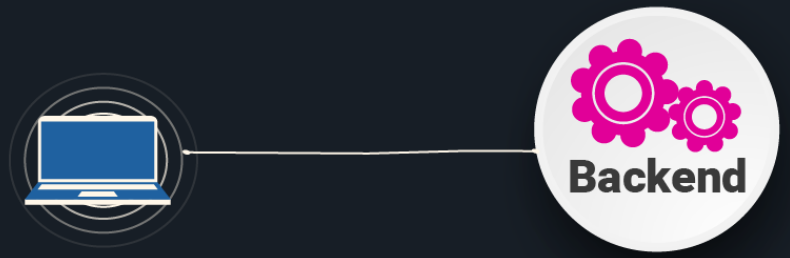


AND SO DID WORDPRESS

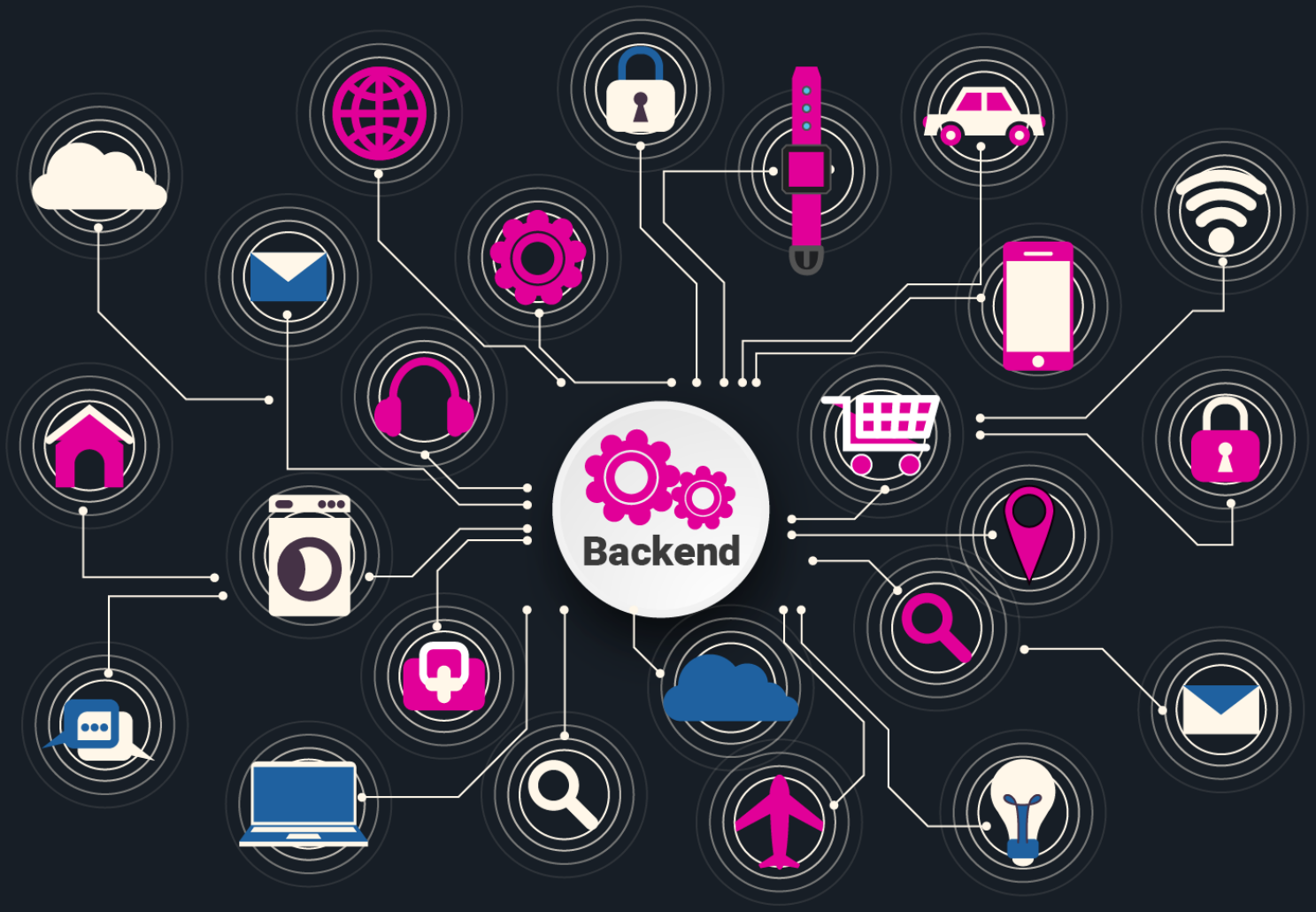
HOW "TRADITIONAL" CMS WORKS

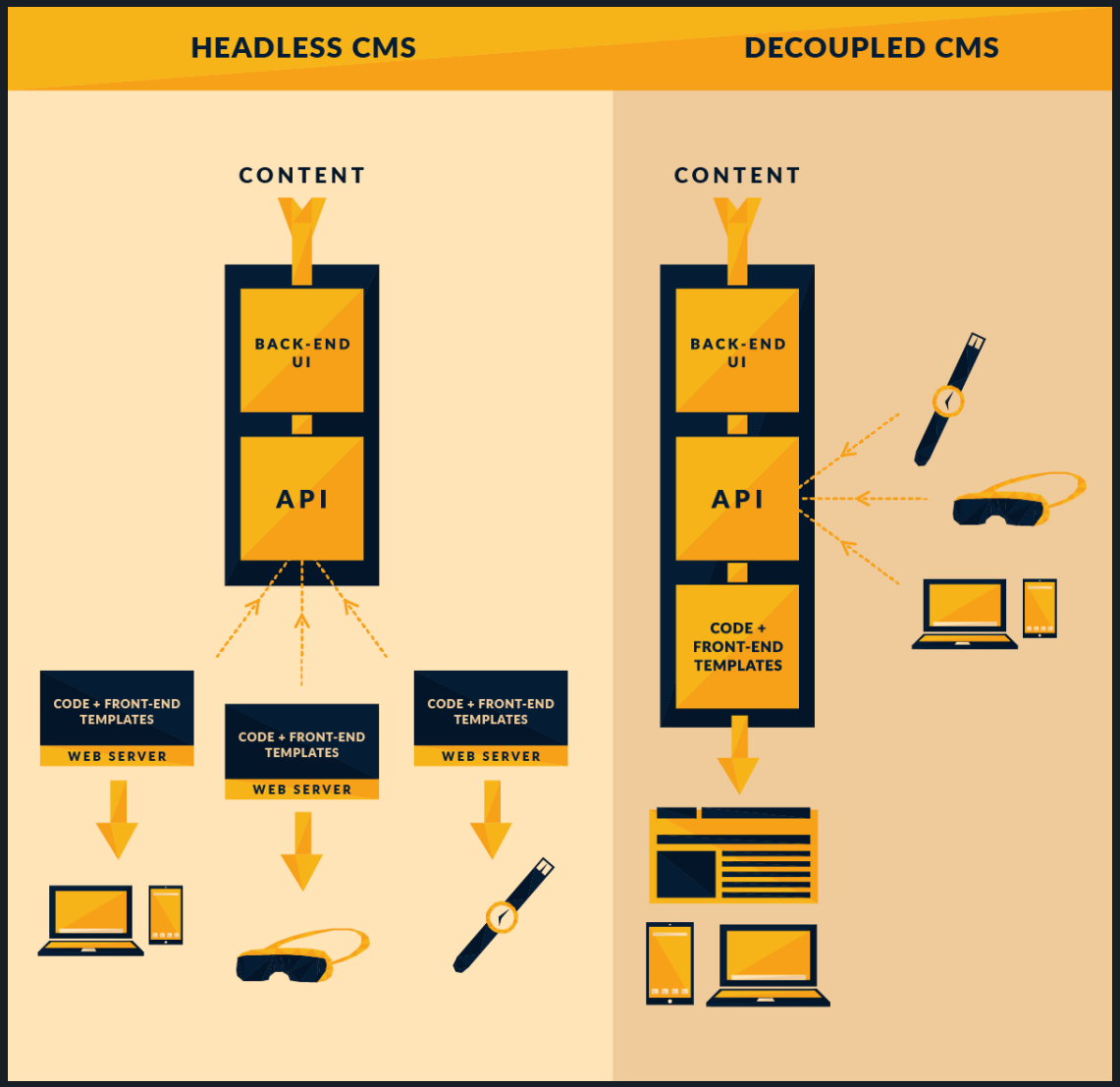


coredna


Backend


Backend


Backend


HEADLESS CMS
DECOUPLED CMS
CONTENT
BACK-END UI
API
CODE + FRONT-END TEMPLATES
WEB SERVER
CODE + FRONT-END TEMPLATES
WEB SERVER
CODE + FRONT-END TEMPLATES
WEB SERVER
CONTENT
BACK-END UI
API
CODE + FRONT-END TEMPLATES

ADVANTAGES OF HEADLESS CMS

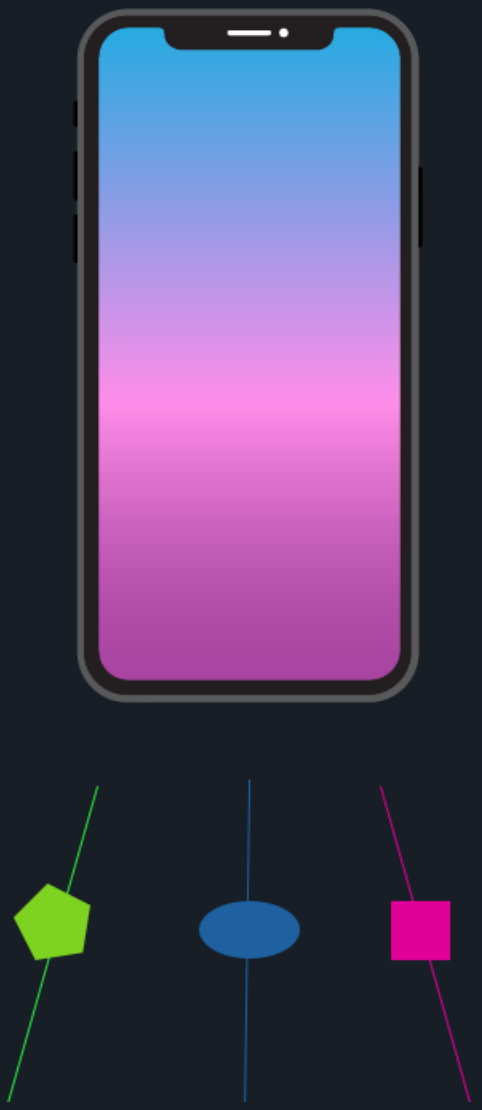
separate content and presentation

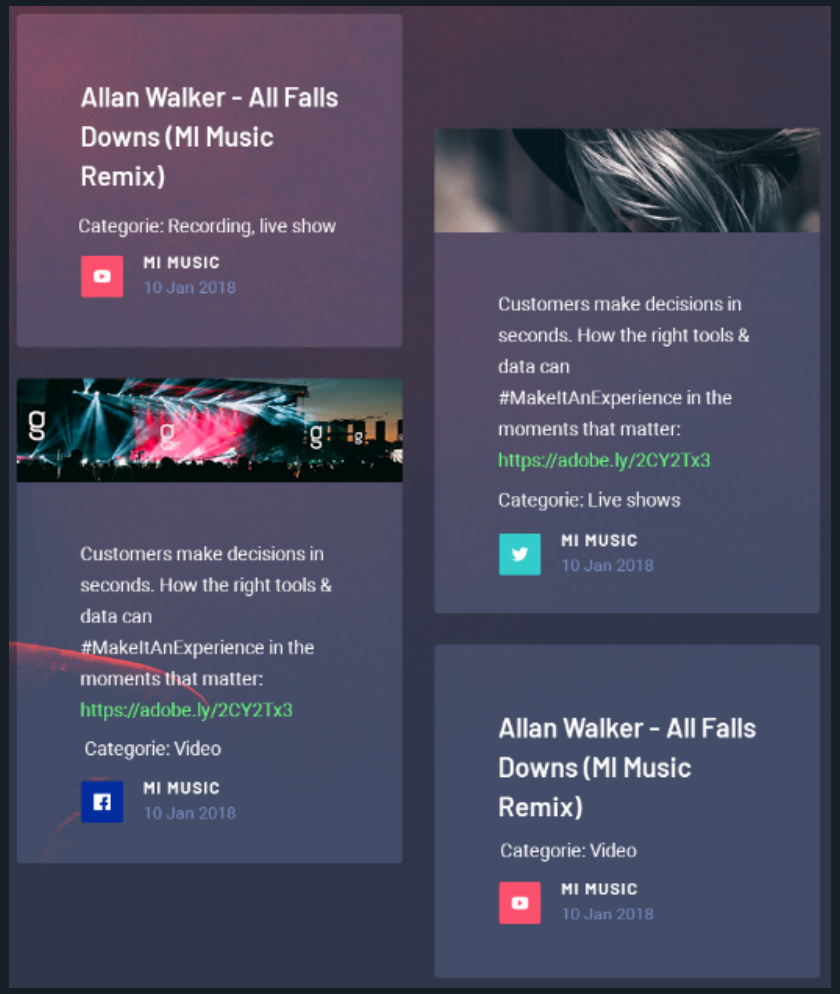
generate unique user experiences

make your content available to mobile apps, kiosk, VR experiences, and other mediums

REST API

/wp-json/wp/v2/





REST API

/wp-json/wp/v2/

/posts

/media/"id"

/users/"id"

/categories/"id"

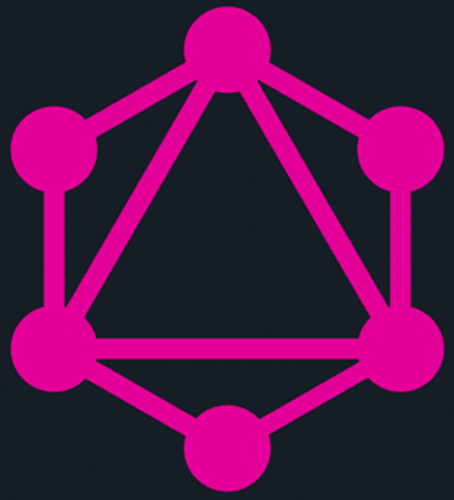
REST API

desavantages

Over-fetching = *a client downloads more information than is actually needed*

Under-fetching = *a specific endpoint doesn't provide enough of the required information*

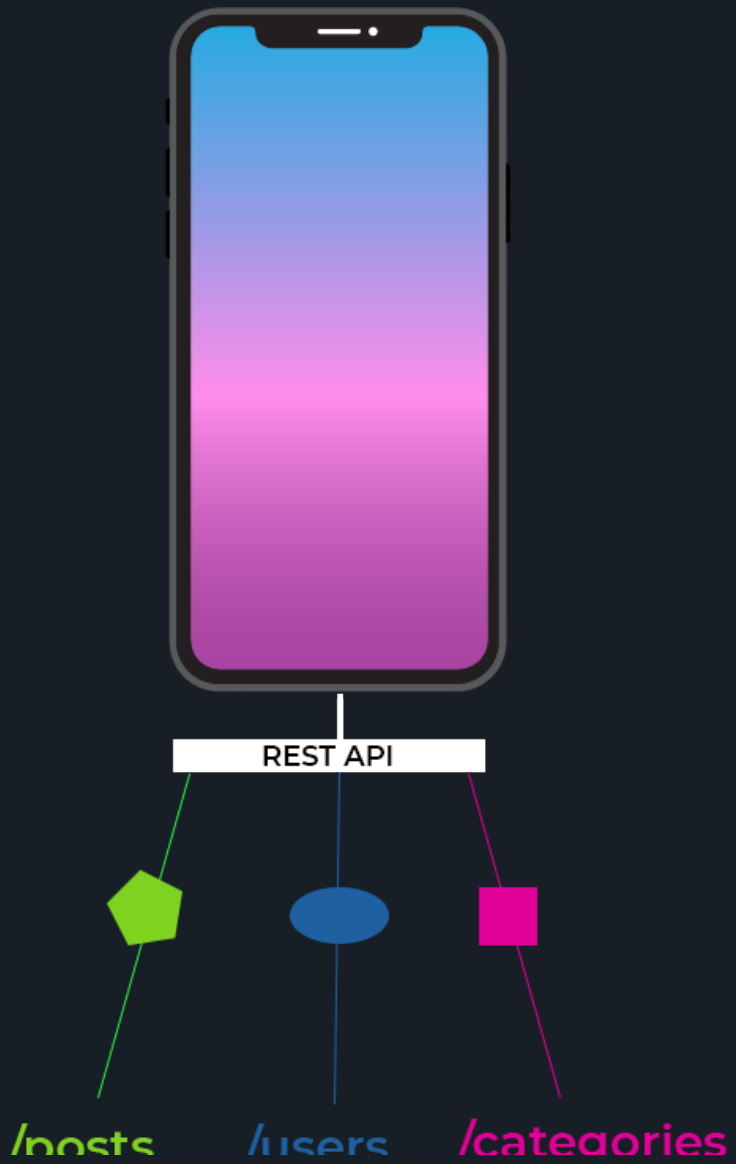
Maintainability and Scalability *having too many endpoints*



GraphQL

Open-sourced by Facebook in 2015

A query language for your API




```
{
  posts {
    users {
      name
      avatar
    }
    categories {
      name
    }
}
```

GRAPHQL

Type of operations you can do on your api

Query *get your data*

Mutation *modify your data*

Subscription *subscribe to action*

GRAPHQL *HAS TWO PARTS*

SERVER

Responsible for the SCHEMA of your API and resolvers of the data

```
type Query {
 posts: [Post]
}

type Post {
 title: String
 content: String
 author: Author
}

type Author {
 name: String
 avatar: String
 posts: [Post]
}
```

CLIENT

Responsible for making the query to the s

```
query {
 posts {
  title
  author {
   name
   avatar
  }
 }
}
```

```
{"data": {
  "posts": {
   "edges": [{
     "node": {
      "title": "Hello World",
      "author": {
           name: "My super name",
       avatar: "http://www.imagesource.com/au
          }
     }
   }]
```

```
    }
}}
```

GRAPHQL *OPTIONS*

SERVER

Custom

Prisma

Framework for self-hosted GraphQL server

WordExpress

Framework to help develop Wordpress applications (server + client)

wp-graphql

WordPress plugin
https://github.com/wp-graphql/wp-graphql

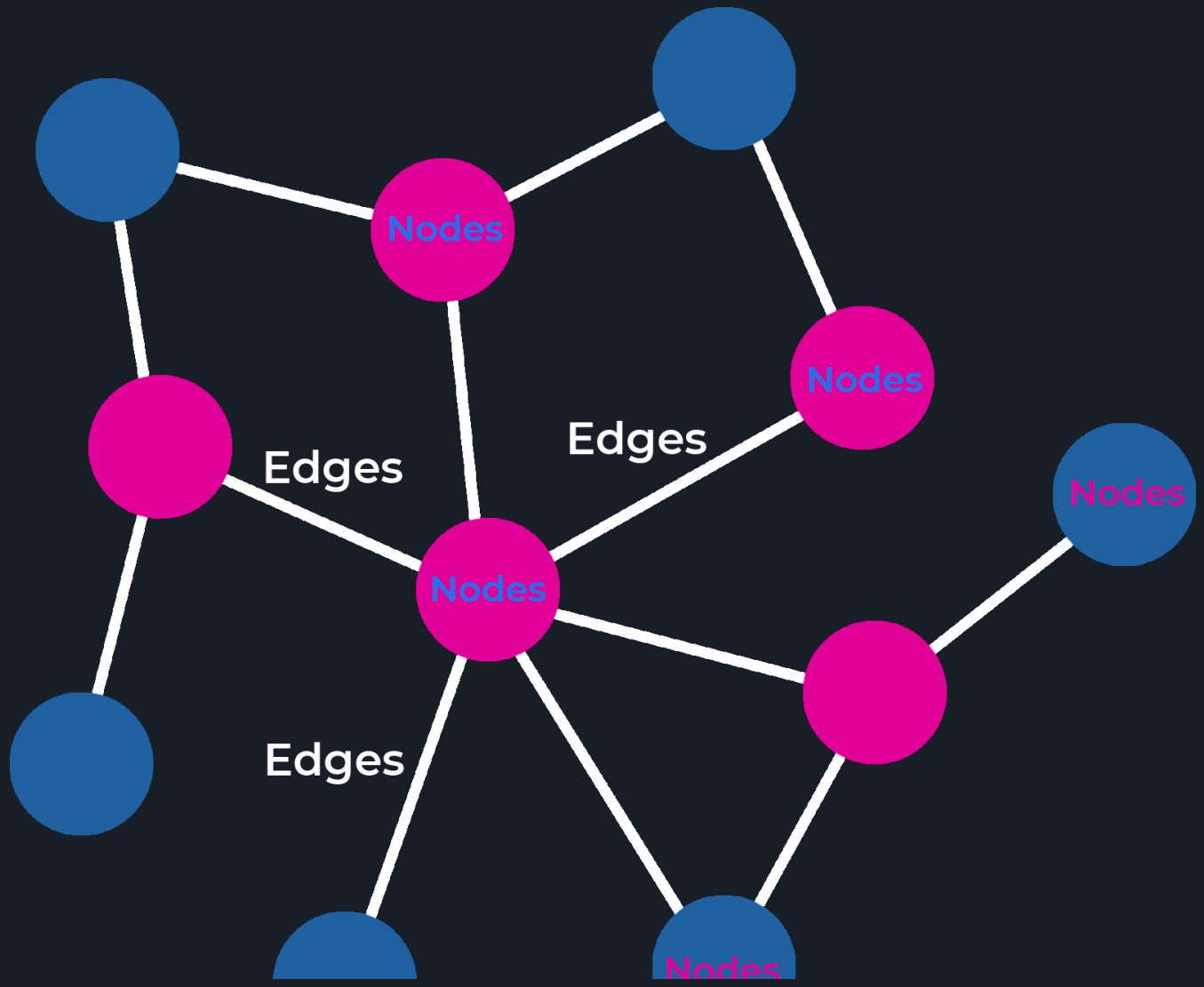
CLIENT

Relay

From Facebook, for more advance developpers

Apollo-client

Apollo-boost = no setup, cache, manage loca and server data


Nodes
Nodes
Edges
Edges
Nodes
Nodes
Edges
Nodes

WORDPRESS SETUP

WordPress will serve as the backend system on which GraphQL will be hocked on

Install wp-graphql plugin

EXTENDING WP-GRAPHQL PLUGIN *IN YOUR THEME FUNCTIONS.PHP*

```php
use WPGraphQL\Types;

function wcgql_add_custom_fields_to_graphql ($fields) {
  $fields['custom'] = [
    'type'                      => Types::String(),
        'description'    => __( 'The date the film was released', 'wc
        'resolve'                => function ( \WP_Post $post ) {
          // Get your data
          $customField = get_post_meta( $post->ID, 'custom_field_nam
          // make any modification on your data

          // Return your data
          return $customField;
        },
  ];

  return $fields;
}

add_filter( 'graphql_post_fields', 'wcgql_add_custom_fields_to_graph
```

REACTJS SETUP

NodeJS

In your command line

npx create-react-app nameofyourproject

cd nameofyourproject

npm install apollo-boost react-apollo graphql —-save

- apollo-boost: Apollo-Client
- react-apollo: View layer integration in react
- graphql: Parses GraphQL queries

npm install react-router-dom —-save

npm start

localhost:3000

APOLLO-BOOST CLIENT SETUP *CREATE A PROVIDER*

```
import React, { Component } from 'react';
import ApolloClient from "apollo-boost";
import { ApolloProvider } from "react-apollo";

const client = new ApolloClient({
uri: "http://localhost/wordcamp2018/graphql"
});

class App extends Component {
render() {
return (
<apolloprovider client="{client}">
        <div>
        <h1>Welcome to Headless WordPress</h1>
        </div>
</apolloprovider>
);
}
}

export default App;
```

APOLLO-BOOST CLIENT SETUP *CREATE A CLIENT*

```
import React, { Component } from 'react';
import ApolloClient from "apollo-boost";
import { ApolloProvider } from "react-apollo";

const client = new ApolloClient({
uri: "http://localhost/wordcamp2018/graphql"
});

class App extends Component {
render() {
return (
<apolloprovider client="{client}">
        <div>
        <h1>Welcome to Headless WordPress</h1>
        </div>
</apolloprovider>
);
}
}

export default App;
```

APOLLO-BOOST CLIENT SETUP *WRAP YOUR PROVIDER*

```
import React, { Component } from 'react';
import ApolloClient from "apollo-boost";
import { ApolloProvider } from "react-apollo";

const client = new ApolloClient({
uri: "http://localhost/wordcamp2018/graphql"
});

class App extends Component {
render() {
return (
<apolloprovider client="{client}">
        <div>
        <h1>Welcome to Headless WordPress</h1>
        </div>
</apolloprovider>
);
}
}

export default App;
```

APOLLO-BOOST CLIENT IN COMPONENT *IMPORT LIBRARIES*

```
1 import React, { Fragment } from 'react';
2 import { Query } from 'react-apollo';
3 import gql from 'graphql-tag';
```

APOLLO-BOOST CLIENT IN COMPONENT *CREATE YOUR QUERY*

```
1  import React, { Fragment } from 'react';
2  import { Query } from 'react-apollo';
3  import gql from 'graphql-tag';
4
5  const GET_POSTS = gql`
6   query GET_POSTS {
7    posts {
8     edges {
9      node {
10      id
11      title
12      date
13     }
14    }
15   }
16  }
17 `;
```

APOLLO-BOOST CLIENT IN COMPONENT *OUTPUT THE DATA*

```
export default function Posts() {
 return (
  <query query="{GET_POSTS}">
   {({ data, loading, error }) => {
    if (loading) return <p>A spinner</p>;
    if (error) return <p>ERROR</p>;

        return (
         <fragment>
          {data.posts &&
           data.posts.posts &&
           data.posts.posts.map(post => (
            console.log(post)
          ))}
         </fragment>
    );
   }}
  </query>
 );
};
```

See the slides

https://ebeldev.github.io/headlesswordpress-slides/

EXTRA

GraphQL + Wordpress with Jason Bahl
https://www.youtube.com/watch?v=tYXlHb2eyQw

GraphQL + Wordpress Learning site
https://edwincromley.gitbooks.io/wp-graphql/content/

Learn GraphQL howtographql.com

Apollo GraphQL apollographql.com