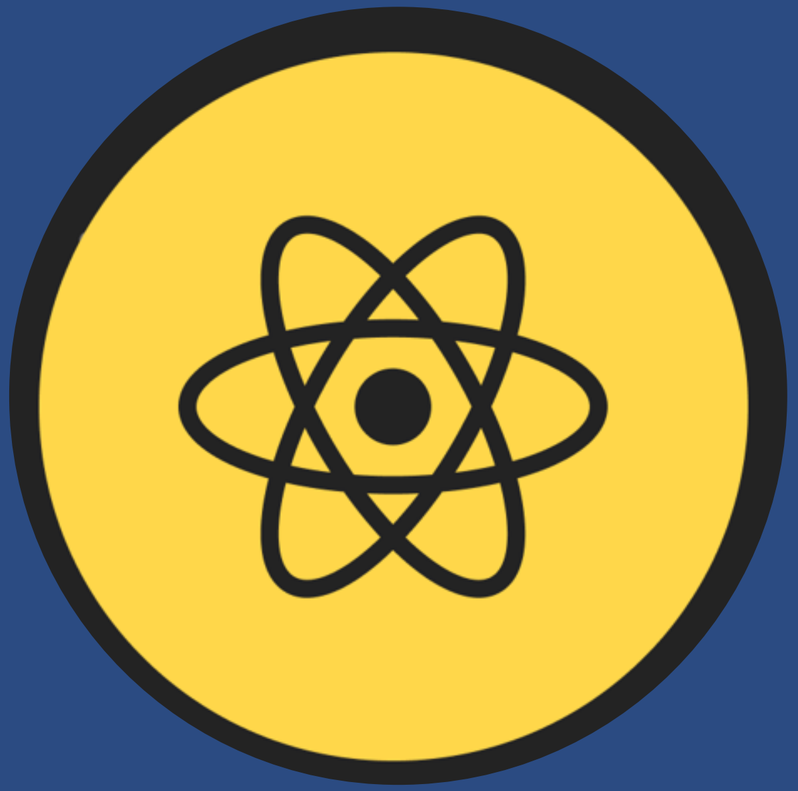
IoT LAB KIIT

Web : CodeHackday

Seminar on NEXT.JS

DL-07 CAMPUS 15 KIIT UNIVERSITY

VS







Code
using HTML CSS

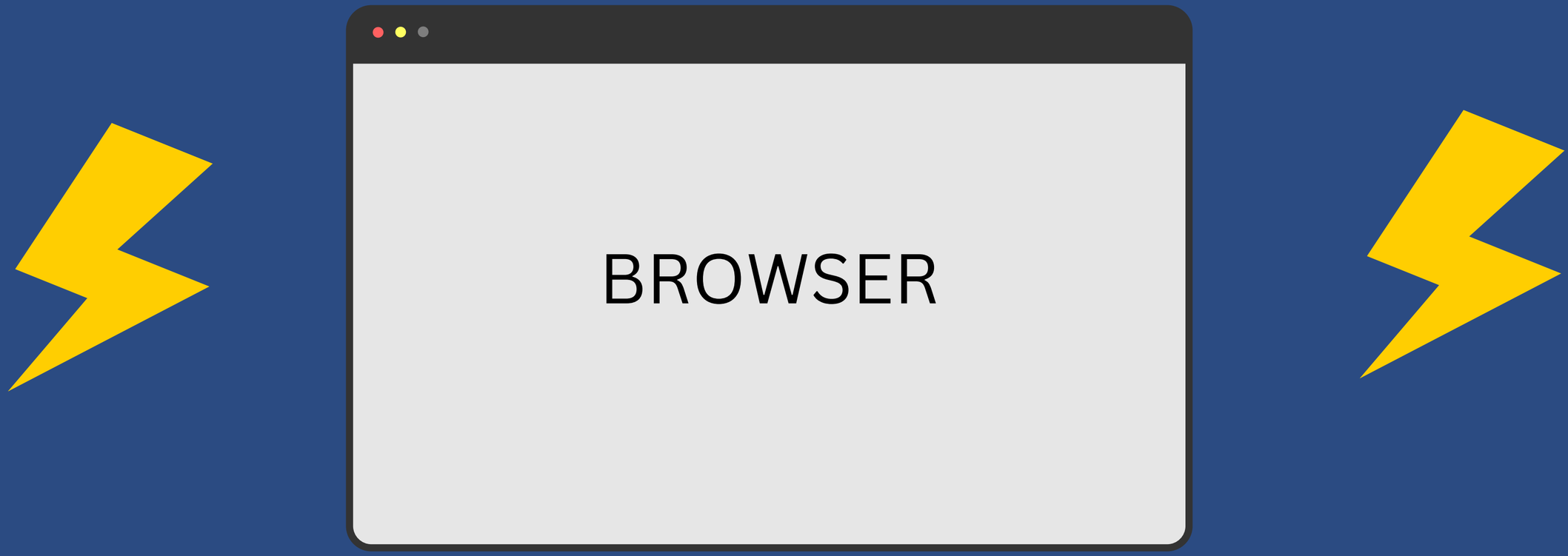
```
<div class="container">
  <div class="card">
    <img src="https://images.unsplash.com/photo-1577805947697-89e18249d767?
crop=entropy&cs=tinysrgb&fm=jpg&ixid=MnwzMjM4NDZ8MHwxfHJhbmRvbXx8fHx8fHx8fDE2NjY1MjQ1NTk&ixlib=rb-
4.0.3&q=80" alt="">
    <div class="info-card">
      <h1>Deliche</h1>
      <p>Coffee, smothies & Sundaes</p>
    </div>
  </div>
  <div class="card">
    <img src="https://images.unsplash.com/photo-1484300681262-5cca666b0954?
crop=entropy&cs=tinysrgb&fm=jpg&ixid=MnwzMjM4NDZ8MHwxfHJhbmRvbXx8fHx8fHx8fDE2NjY1MjQ1NTk&ixlib=rb-
4.0.3&q=80" alt="">
    <div class="info-card">
      <h1>Grillz</h1>
      <p>Steakhouse & Barbecue</p>
    </div>
  </div>
  <div class="card">
    <img src="https://images.unsplash.com/photo-1613564834361-9436948817d1?
crop=entropy&cs=tinysrgb&fm=jpg&ixid=MnwzMjM4NDZ8MHwxfHJhbmRvbXx8fHx8fHx8fDE2NjY1MjU1ODM&ixlib=rb-
4.0.3&q=80" alt="">
    <div class="info-card">
      <h1>Pizzarazi</h1>
      <p>Napoli Style Pizza</p>
    </div>
  </div>
  <div class="card">
    <img src="https://images.unsplash.com/photo-1615361200141-f45040f367be?
crop=entropy&cs=tinysrgb&fm=jpg&ixid=MnwzMjM4NDZ8MHwxfHJhbmRvbXx8fHx8fHx8fDE2NjY1MjU1NjE&ixlib=rb-
4.0.3&q=80" alt="">
    <div class="info-card">
      <h1>Shioji</h1>
      <p>Sushi & Handrolls</p>
```

Code
using REACT.JS

```
<body>
    <div id="root"></div>
    <script type="text/babel">
      ReactDOM.render(
        <section>
          <Card />
          <Card />
          <Card />
        </section>,
        document.getElementById("root")
      );
    </script>
 </body>
```

```
function Card() {
  return (
    <div class="card">
    <img src="https://images.unsplash.com/photo-
1577805947697-89e18249d767?
crop=entropy&cs=tinysrgb&fm=jpg&ixid=MnwzMjM4NDZ8MHwxfHJhb
mRvbXx8fHx8fHx8fDE2NjY1MjQ1NTk&ixlib=rb-4.0.3&q=80"
alt="">
    <div class="info-card">
      <h1>Deliche</h1>
      <p>Coffee, smothies & Sundaes</p>
    </div>
  </div>
  );
}
```

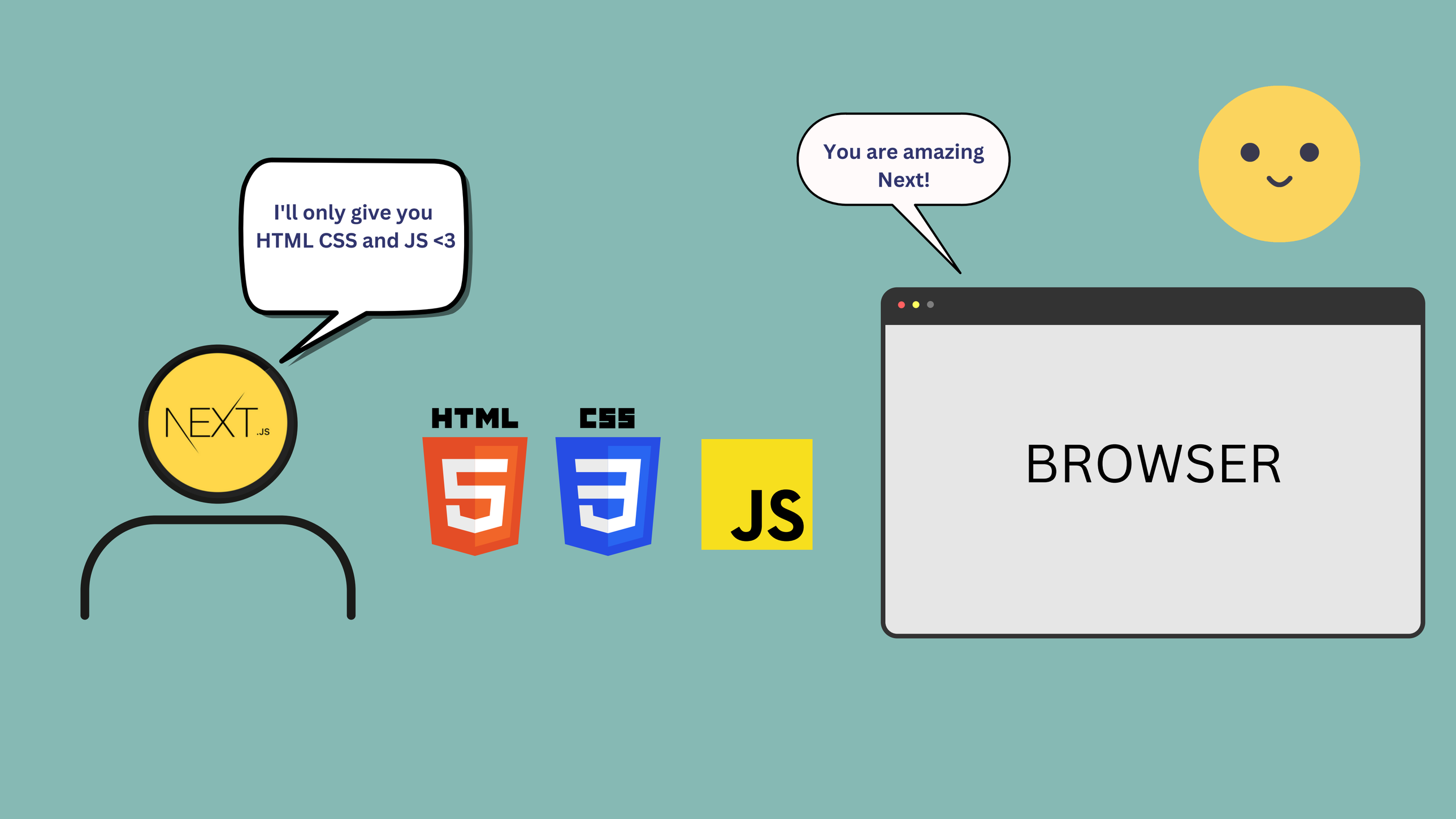
"I am powerful in rendering HTML CSS and build JS"



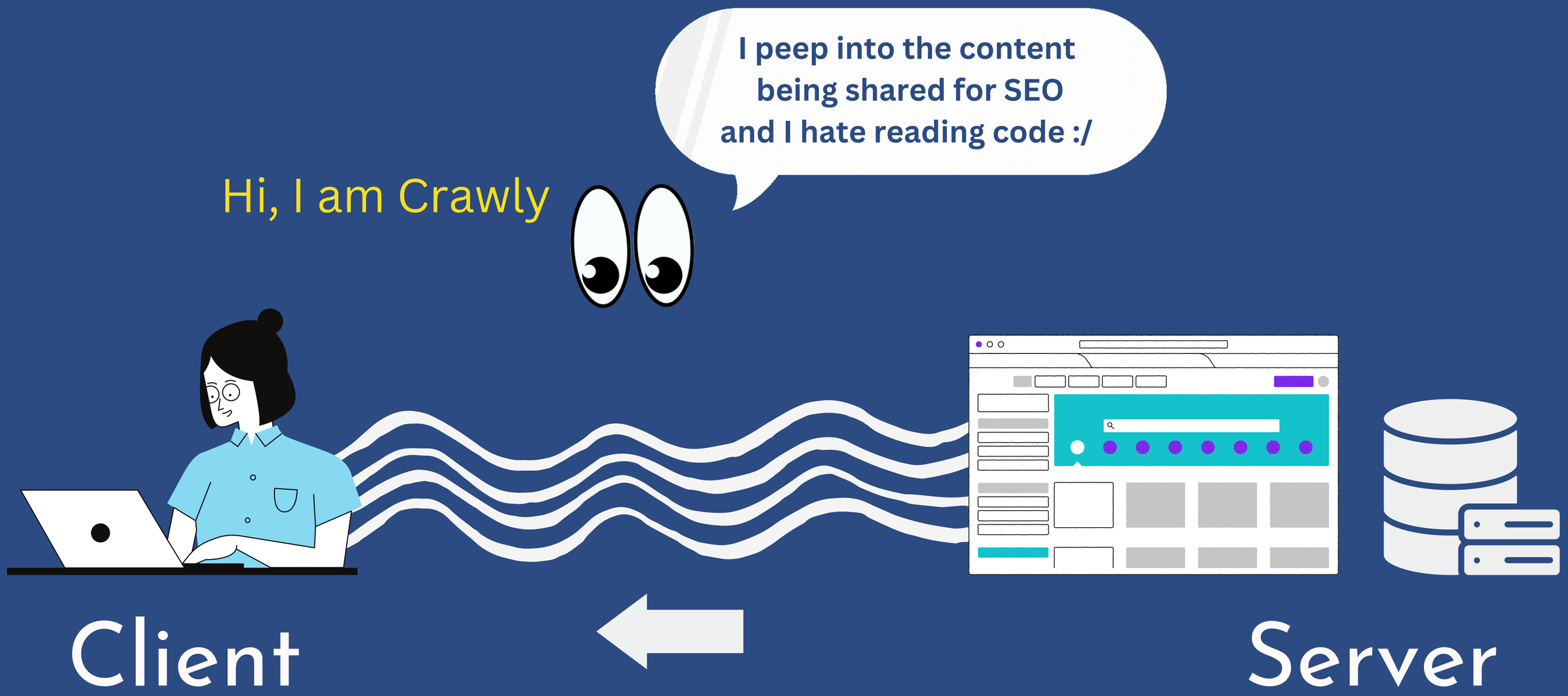
Take the whole code
duh!
I am little lazy

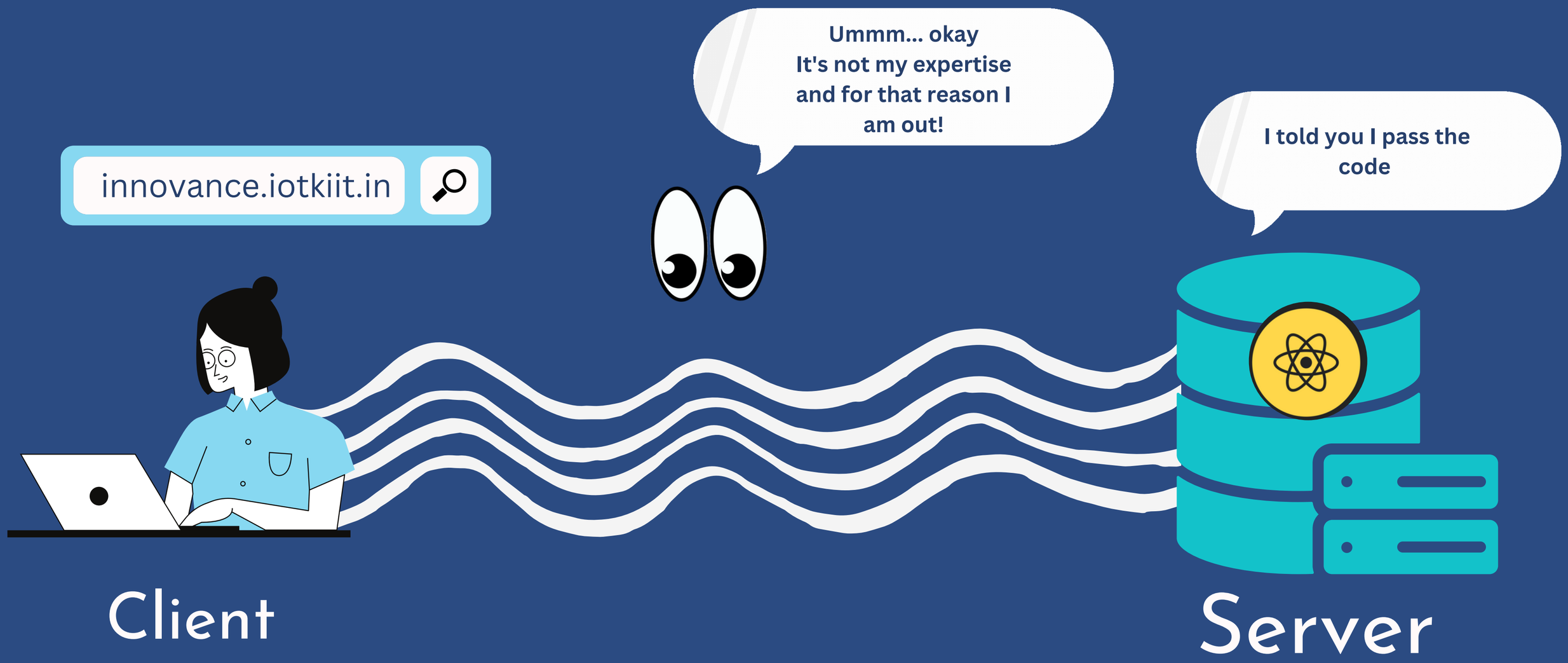
But.. but.. I like html
css and js only.
Nevermind!.... Okay :(

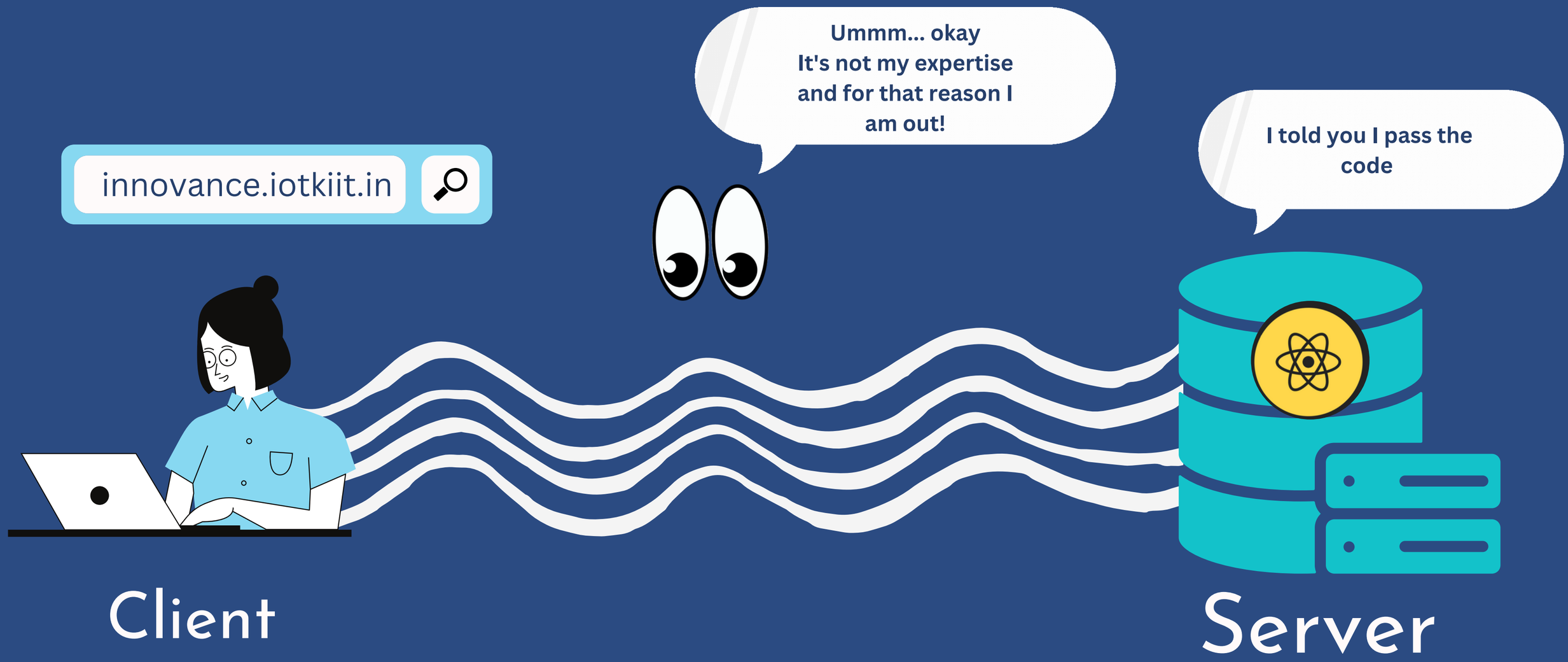
BROWSER


I'll only give you
HTML CSS and JS <3
NEXT.JS
HTML
CSS
JS
You are amazing
Next!
BROWSER

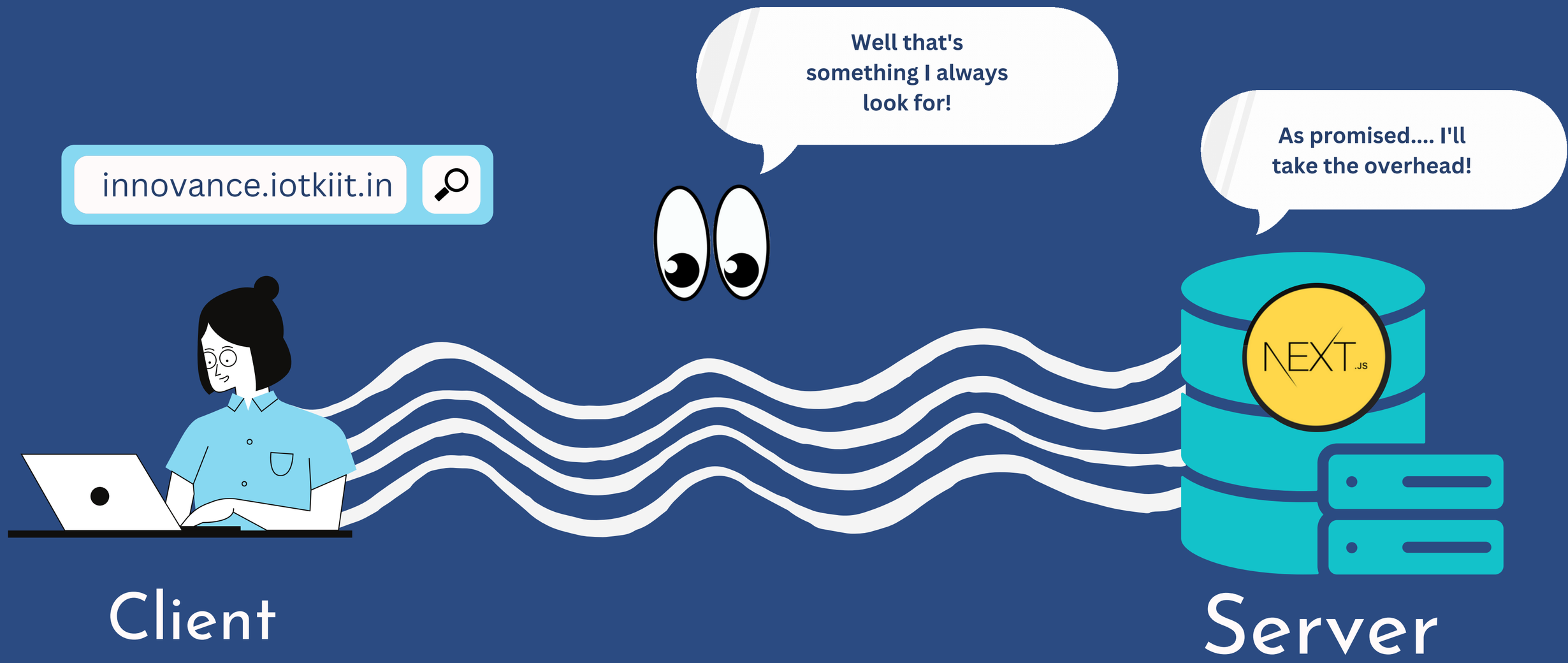
~~Server-side rendering~~


I peep into the content being shared for SEO and I hate reading code :/
Hi, I am Crawly
Client
Server


innovance.iotkiit.in
Ummm... okay
It's not my expertise
and for that reason I
am out!
I told you I pass the
code
Client
Server



COMPARATIVELY BAD SEO


innovance.iotkiit.in
Well that's something I always look for!
As promised.... I'll take the overhead!
NEXT.JS
Client
Server


innovance.iotkiit.in
Well that's something I always look for!
As promised.... I'll take the overhead!
NEXT.JS
Client
Server
Better SEO

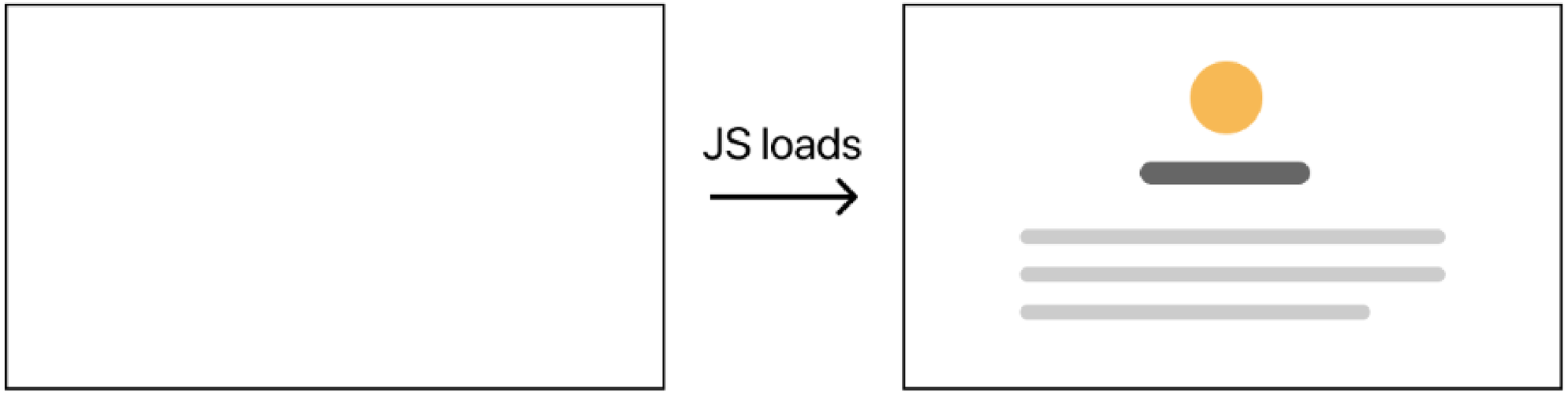
~~pre-rendering~~

No Pre-rendering (Plain React.js app)

Initial Load:

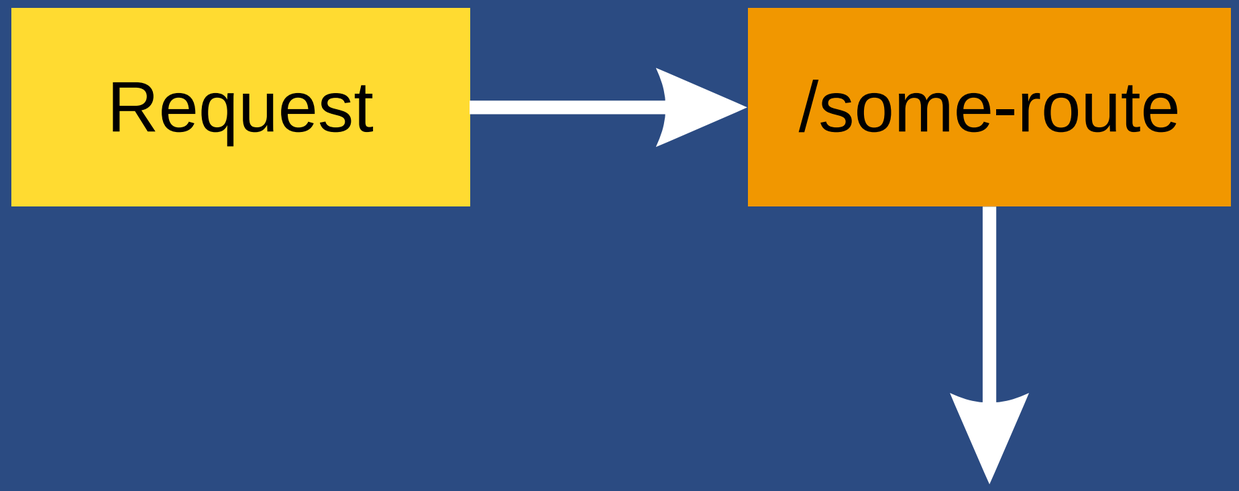
App is not rendered

Hydration: React components are initialized and App becomes interactive

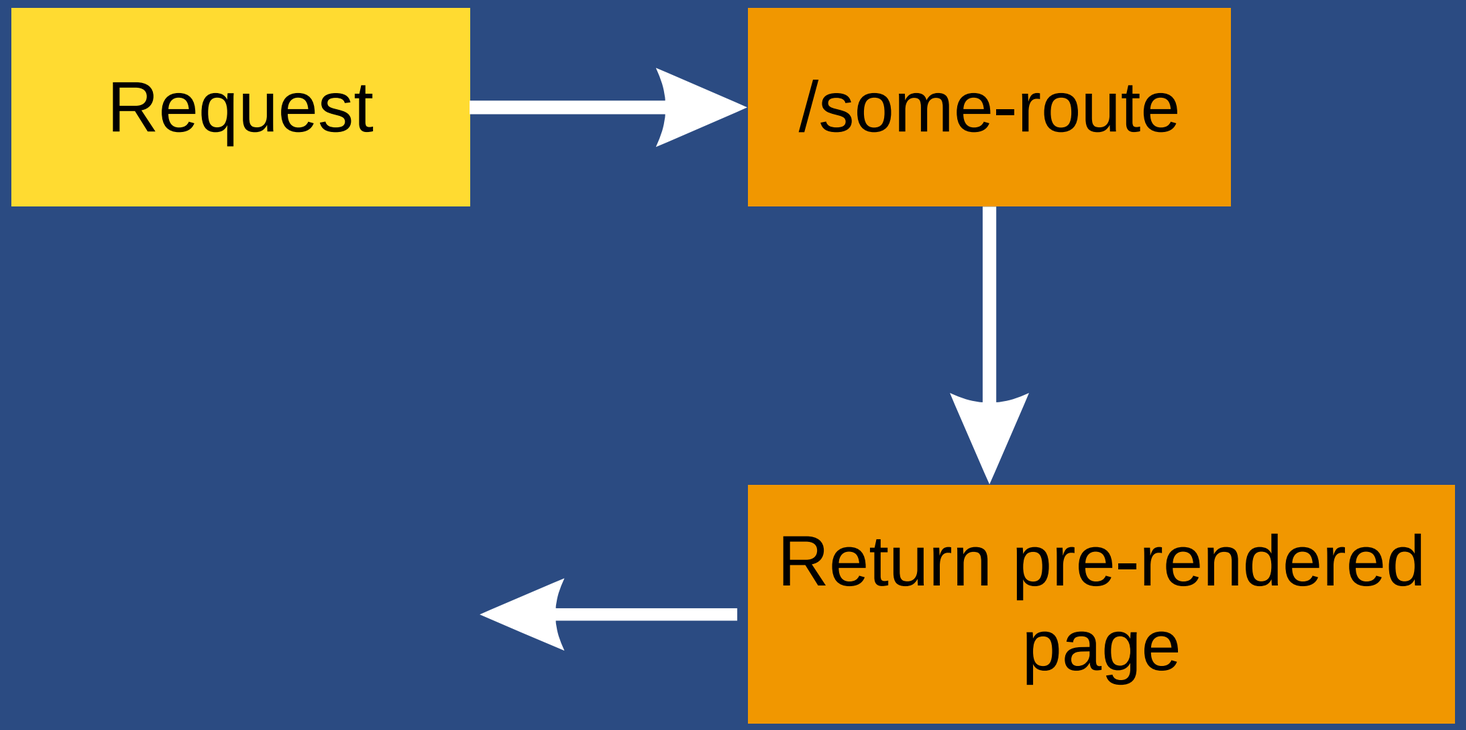


Page Pre-rendering

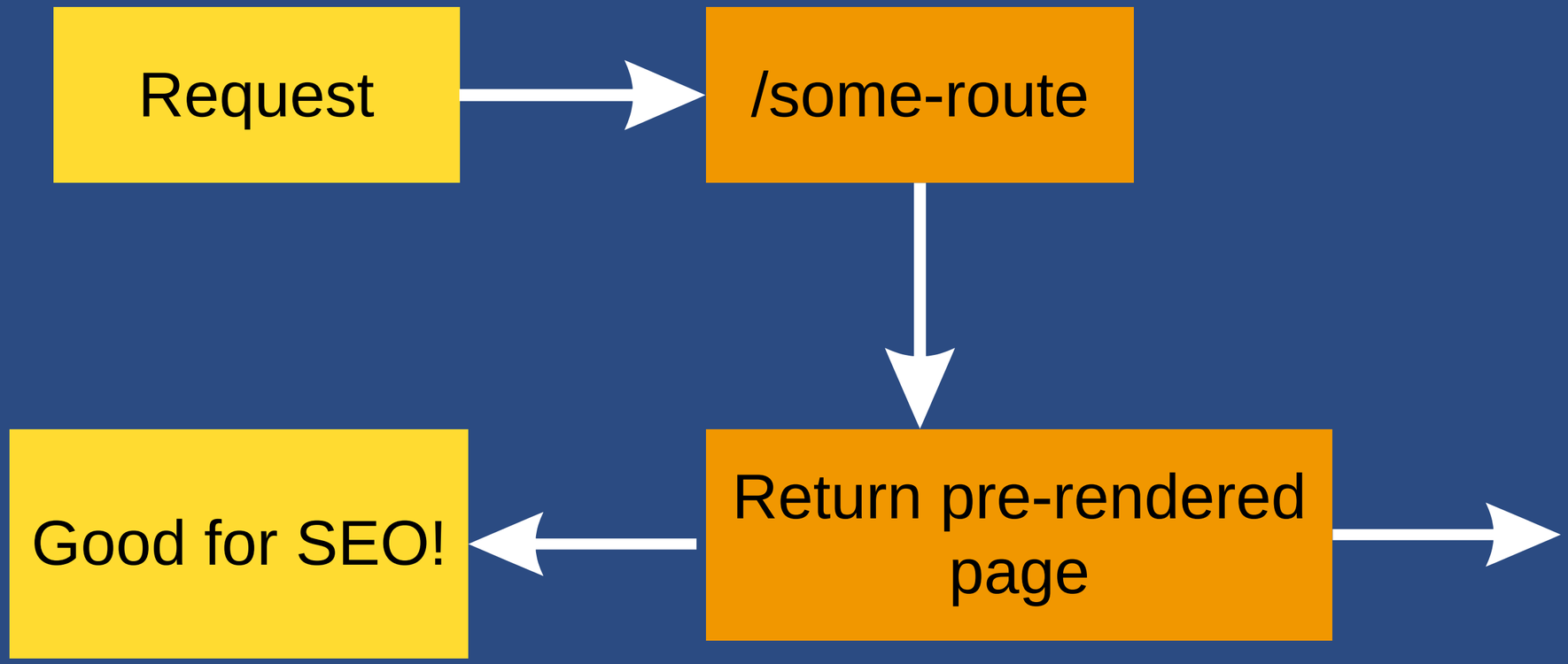
Page Pre-rendering



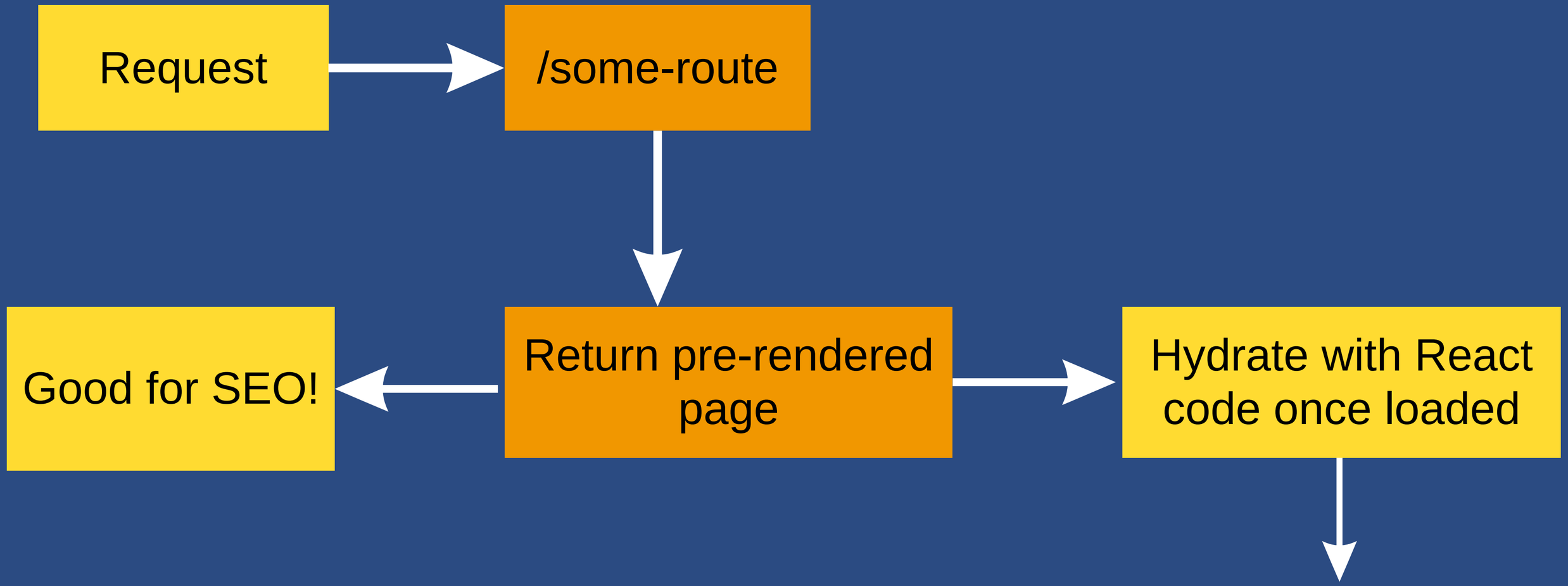
Page Pre-rendering



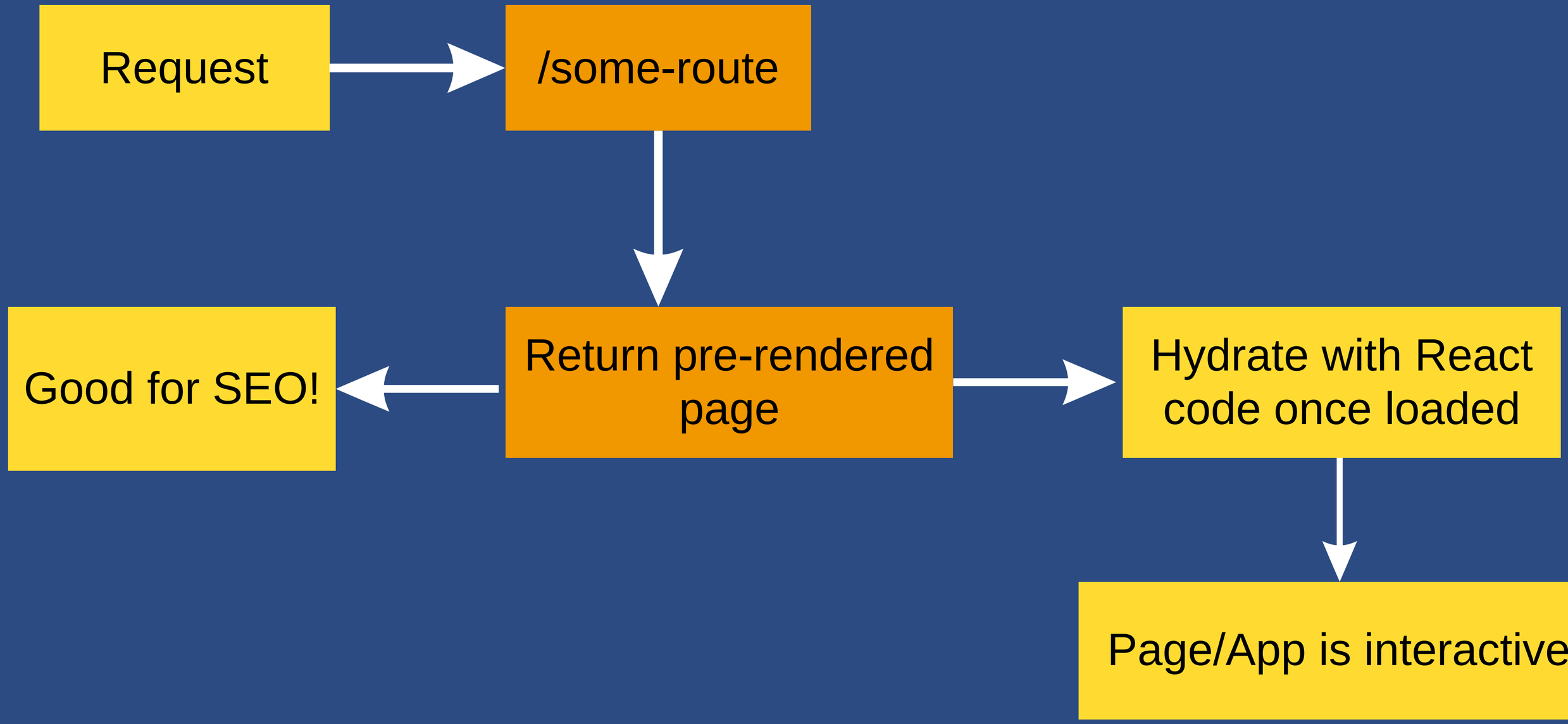
Page Pre-rendering



Page Pre-rendering



Page Pre-rendering

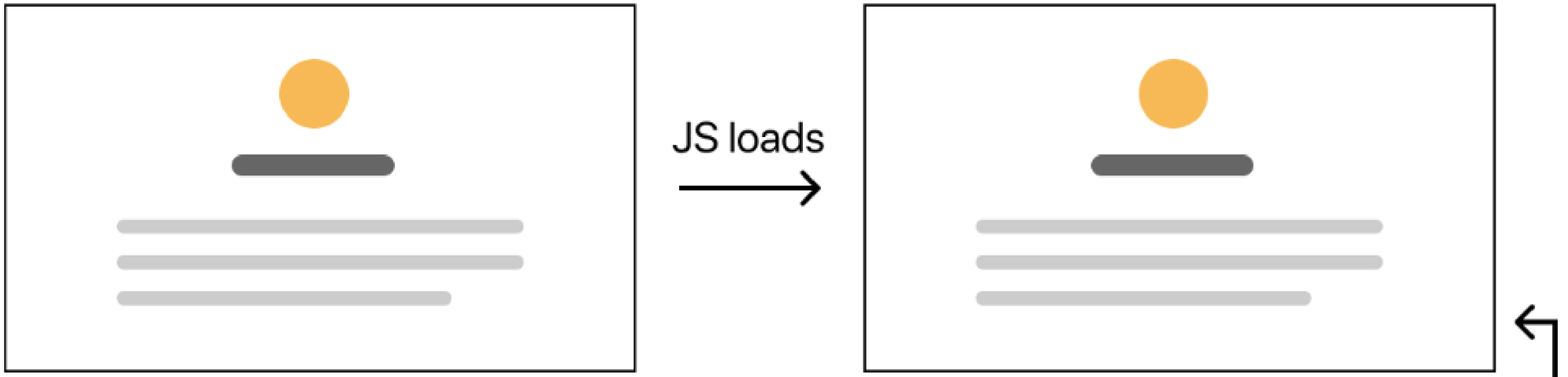


Pre-rendering (Using Next.js)

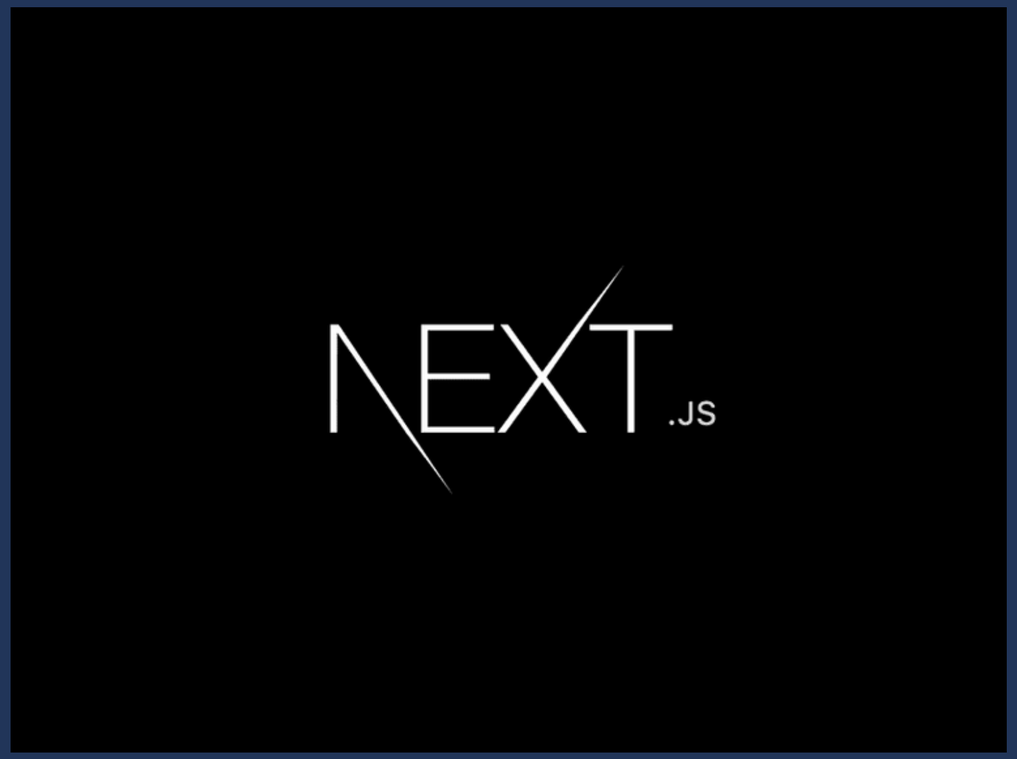
Initial Load:

Pre-rendered HTML is displayed

Hydration: React components are initialized and App becomes interactive



If your app has interactive components like `<Link />`, they'll be active after JS loads



SETUP AND INSTALLATION

SYSTEM REQUIREMENT

- Node.js 14.6.0 or newer
- MacOS, Windows (including WSL), and Linux are supported

SETUP

```
npx create-next-app@latest
# or  yarn create next-app
# or   pnpm create next-app
```

AFTER INSTALLATION

- Run npm run dev or yarn dev or pnpm dev to start the development server on http://localhost:3000

Creating Pages

Adding pages using REACT.JS

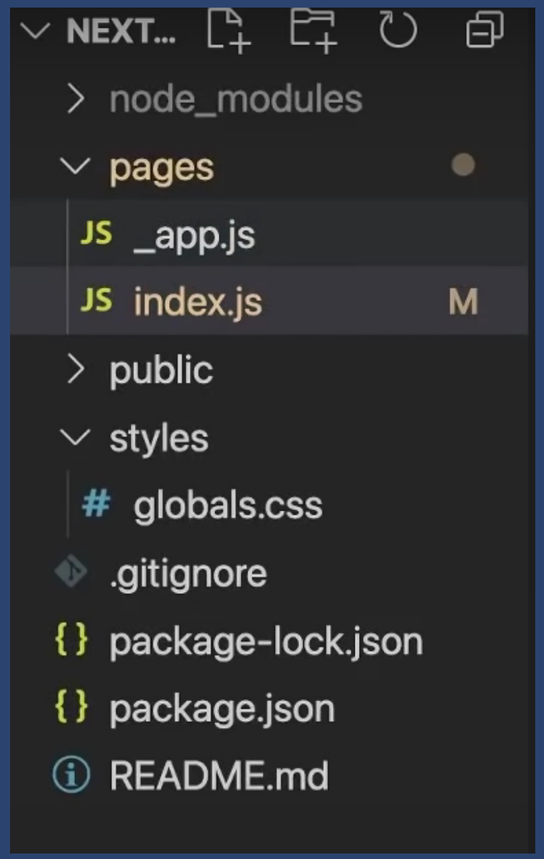
React Router is the standard library for routing with react. Once set up, you'll be able to navigate between multiple components within react, synchronized to changing URL paths.

```
import ReactDOM from "react-dom";
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Layout from "./pages/Layout";
import Home from "./pages/Home";
import Blogs from "./pages/Blogs";
import Contact from "./pages/Contact";
import NoPage from "./pages/NoPage";

export default function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Layout />}>
          <Route index element={<Home />} />
          <Route path="blogs" element={<Blogs />} />
          <Route path="contact" element={<Contact />} />
          <Route path="*" element={<NoPage />} />
        </Route>
      </Routes>
    </BrowserRouter>
  );
}

ReactDOM.render(<App />, document.getElementById("root"));
```

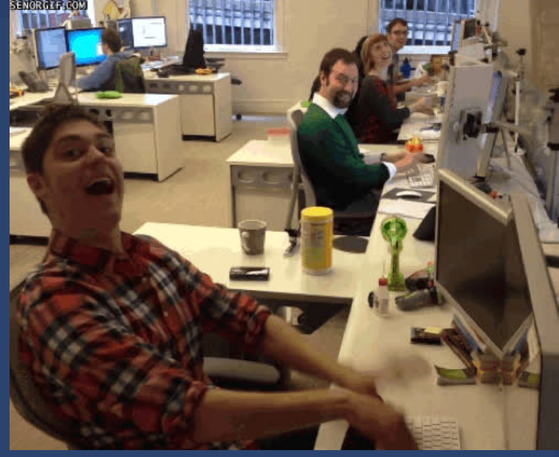
Adding pages using NEXT.JS

In Next.js, a page is a React Component exported from a .js, .jsx, .ts, or .tsx file in the pages directory. Each page is associated with a route based on its file name.

Create a pages folder,
and add an index.js file.



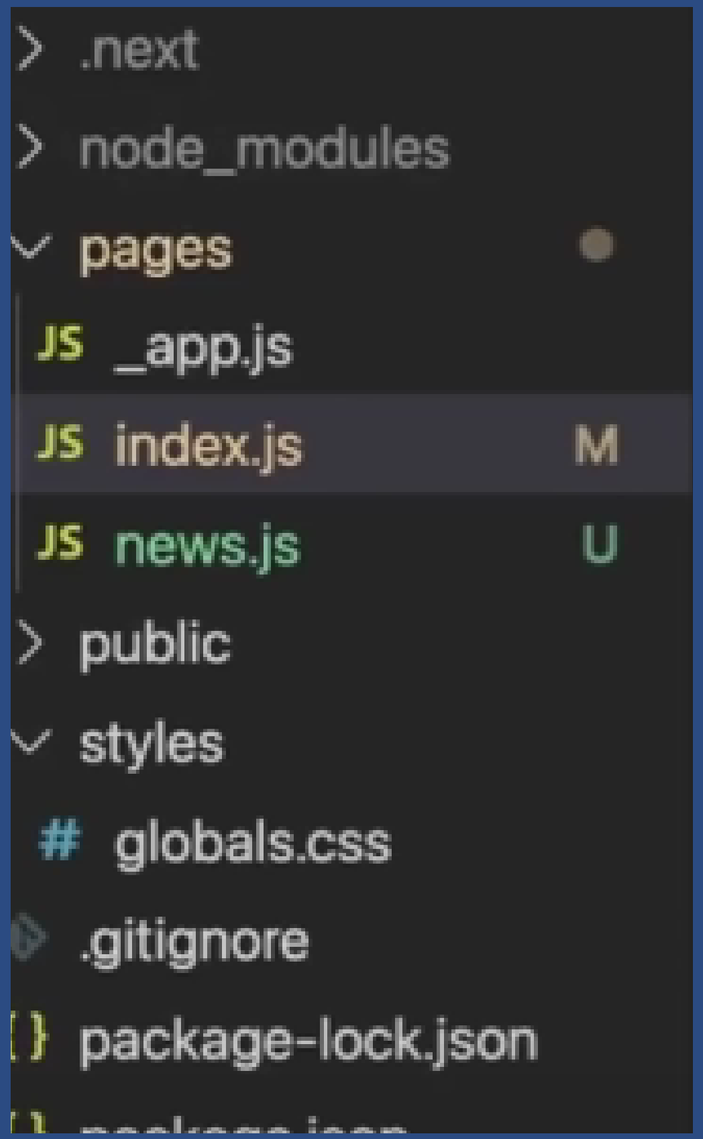

```
//our domain.com/

function Homepage(){
  return <h1> Home page </h1>
}
export default Homeepage;
```



Start the development
server using
npm run dev
in cmd and visit
localhost:3000

Next JS Routing

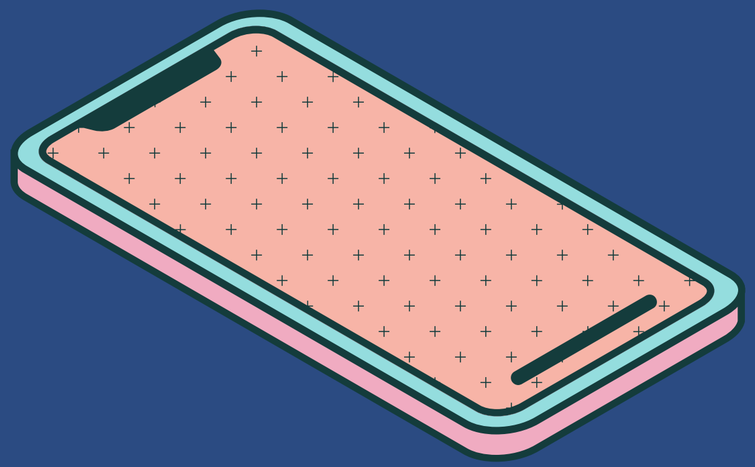
- Next.js has a file-system based router built on the concept of pages.
- When a file is added to the pages directory, it's automatically available as a route.
- The files inside the pages directory can be used to define most common patterns



Nested Routes

The router supports nested files. If we create a nested folder structure, files will automatically be routed in the same way still.

- pages/blog/first-post.js → /blog/first-post
- pages/dashboard/settings/username.js → /dashboard



Dynamic Routing

- Defining routes by using predefined paths is not always enough for complex applications.

- To match a dynamic segment, we can use the bracket syntax. This allows you to match named parameters.

- Any route like will be matched by pages/post/[pid].js. The matched path parameter will be sent as a query parameter to the page, and it will be merged with the other query parameters.

...continued

The route /post/abc will have the following query object:

The route /post/abc?foo=bar will have the following query object:

Extracting Dynamic Route Data

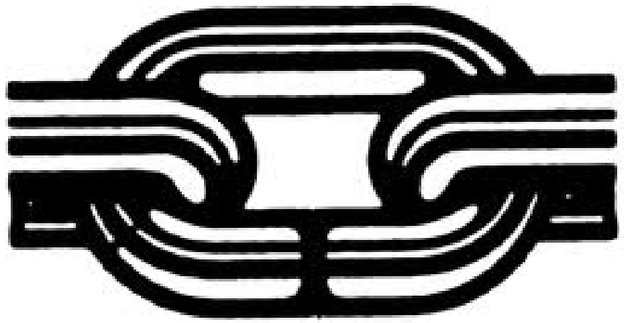
- To extract the value in the url nextjs provides us with a hook useRouter imported from next/router

- Here newsId is the identifier in square brackets of the file name.

```
import { useRouter } from 'next/router';
function DetailPage(){
  const router = useRouter();

  const newsId = router.query.newsId;

}
export default DetailPage;
```

Linking Between Pages

- When linking between pages on websites, we use the <a> HTML tag.
  - In Next.js, you can use the Link Component next/link to link between pages in your application.
  - <Link> allows you to do client-side navigation and accepts props that give you better control over the navigation behavior.



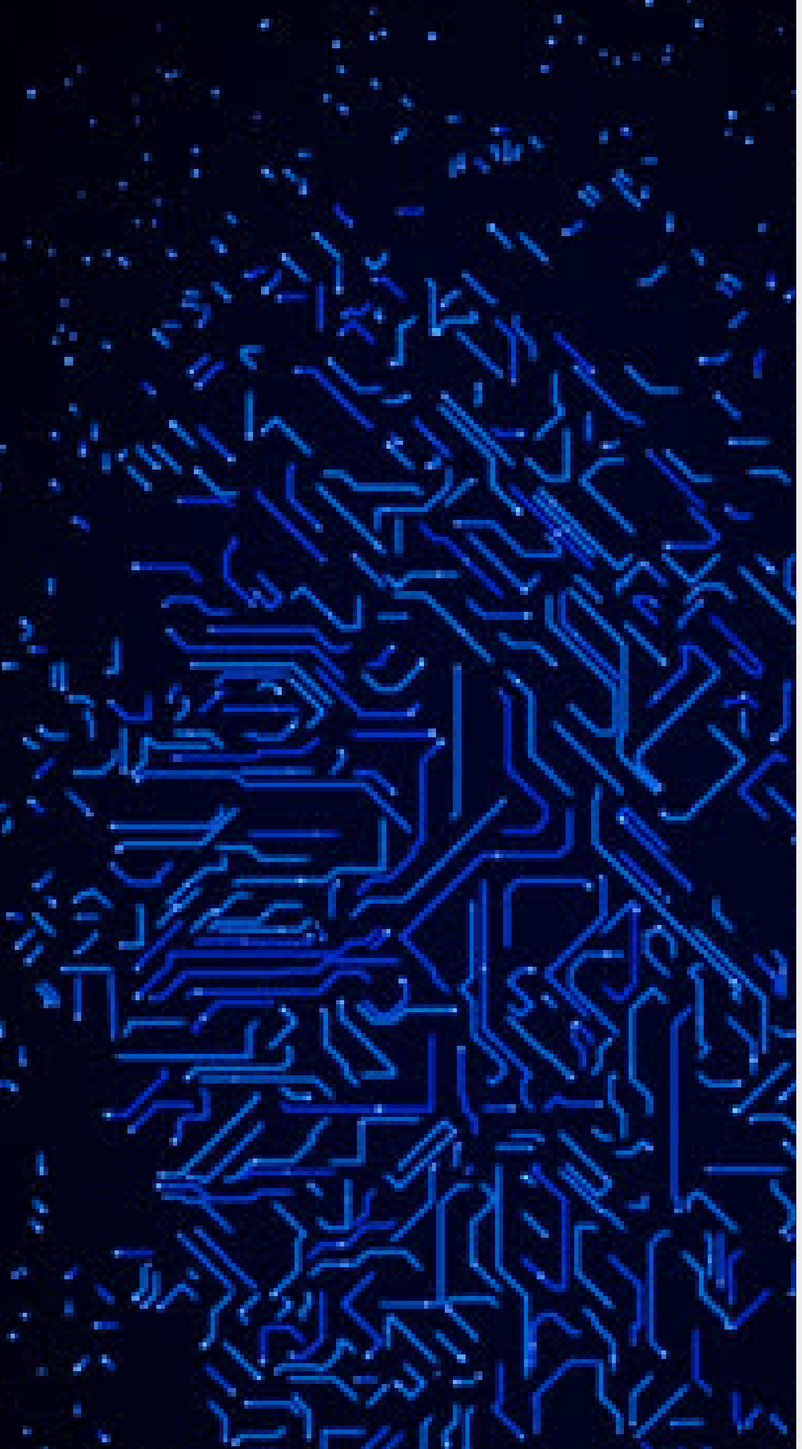
...continued

- Open pages/index.js, and import the Link component from next/link:

```
import Link from
'next/link';
```

- Then link the first-post page

```
<h1 className="title">
  Read <Link
href="/posts/first-
post">this page!</Link>
</h1>
```



Pre-rendering and Data Fetching

- By default, Next.js pre-renders every page. This means that Next.js generates HTML for each page in advance, instead of having it all done by client-side JavaScript. Pre-rendering can result in better performance and SEO.

- In practice, this means that for a fully client-side rendered app, the user will see a blank page while the rendering work is being done

CORE FEATURES

Server-side Rendering

- Automatic page pre-rendering: Great for SEO and initial load
- Blending client side and server-side

CORE FEATURES

Server-side Rendering

- Automatic page pre-rendering: Great for SEO and initial load
- Blending client side and server-side

File-based Routing

- Define pages and routes with files and folder instead of code
- Less code ,less work, Highly Understandable.

CORE FEATURES

Server-side Rendering

- Automatic page pre-rendering: Great for SEO and initial load
- Blending client side and server-side

File-based Routing

- Define pages and routes with files and folder instead of code
- Less code ,less work, Highly Understandable.

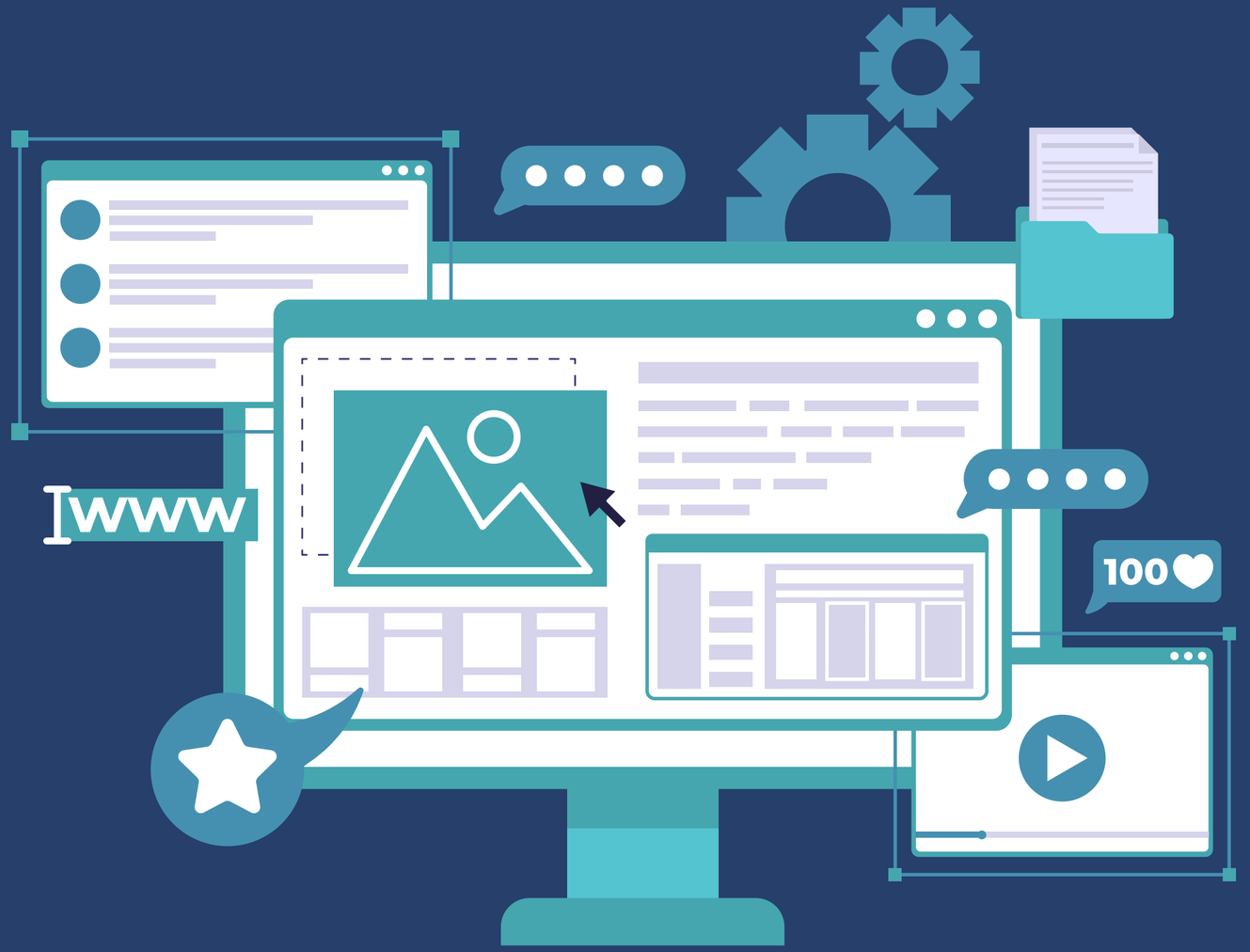
Full-Stack Capabilites

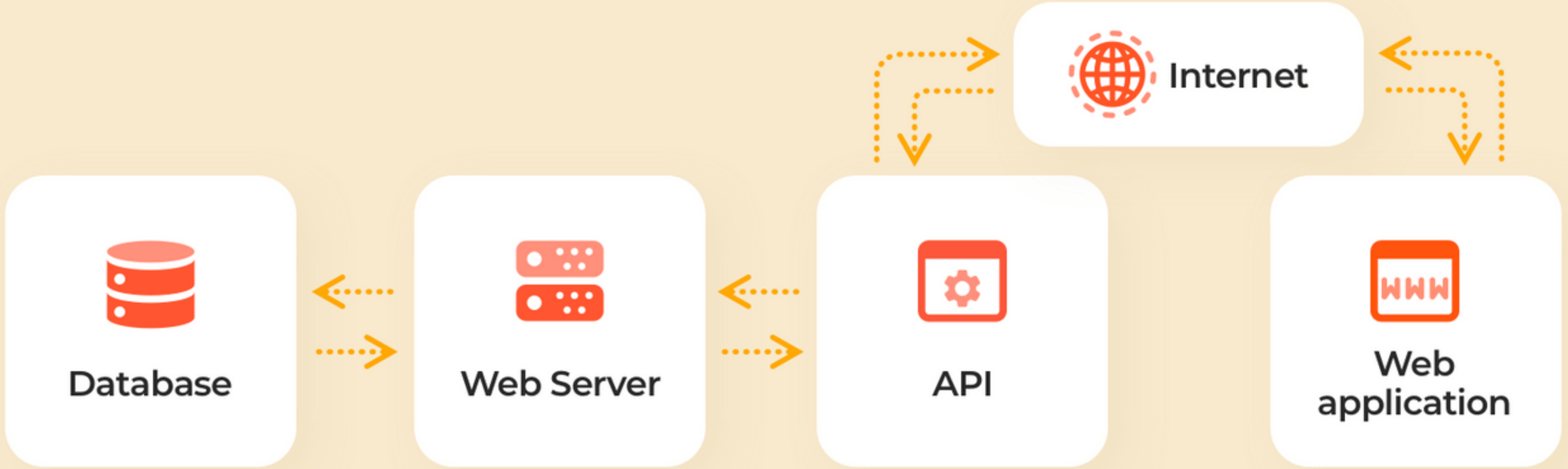
- Easily add backend code to the to Next/React Apps
- Storing data,getting data, authentication etc.

Visit http://kahoot.com

It's time for Quiz

Code-


www
100


Internet
Database
Web Server
API
WWW
Web application

What is API?

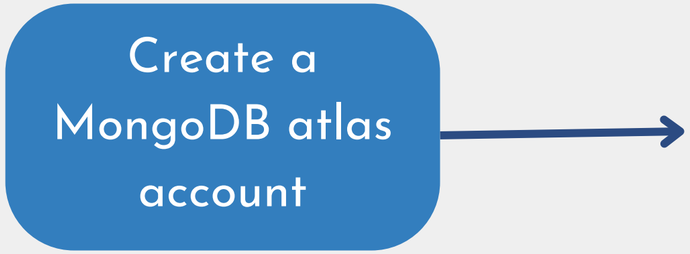
API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API.

Why there need of API?

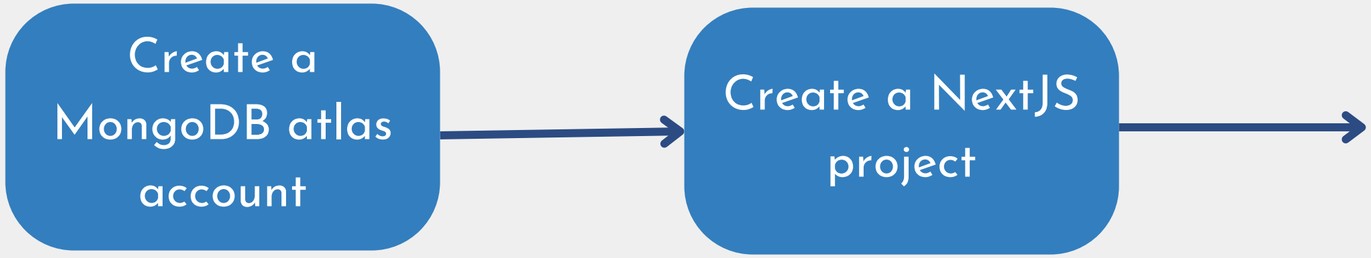
APIs are needed to bring applications together in order to perform a designed function built around sharing data and executing pre-defined processes..

What is mongoDB ?

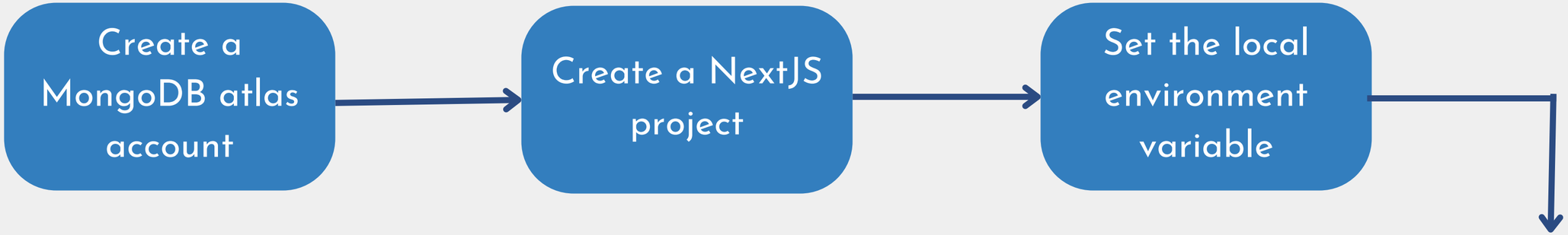
Connecting and Querying to a MongoDB database



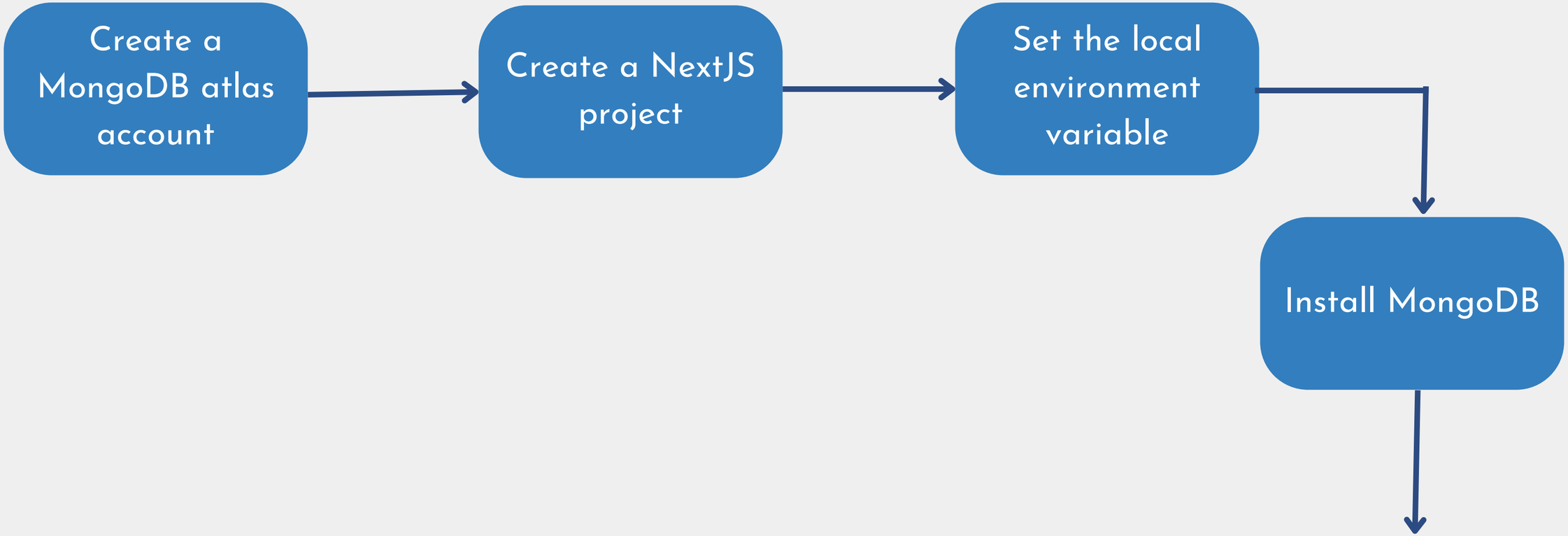
Connecting and Querying to a MongoDB database



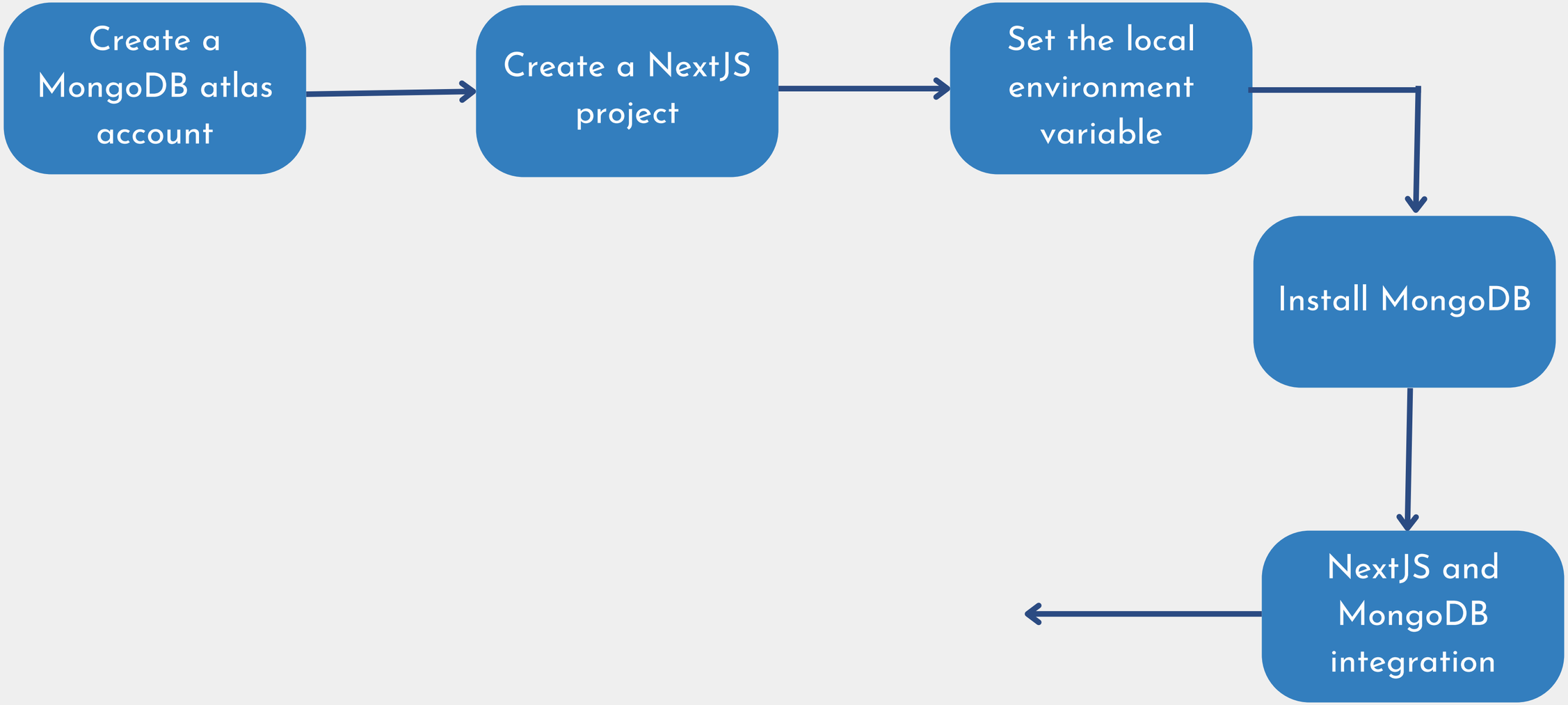
Connecting and Querying to a MongoDB database



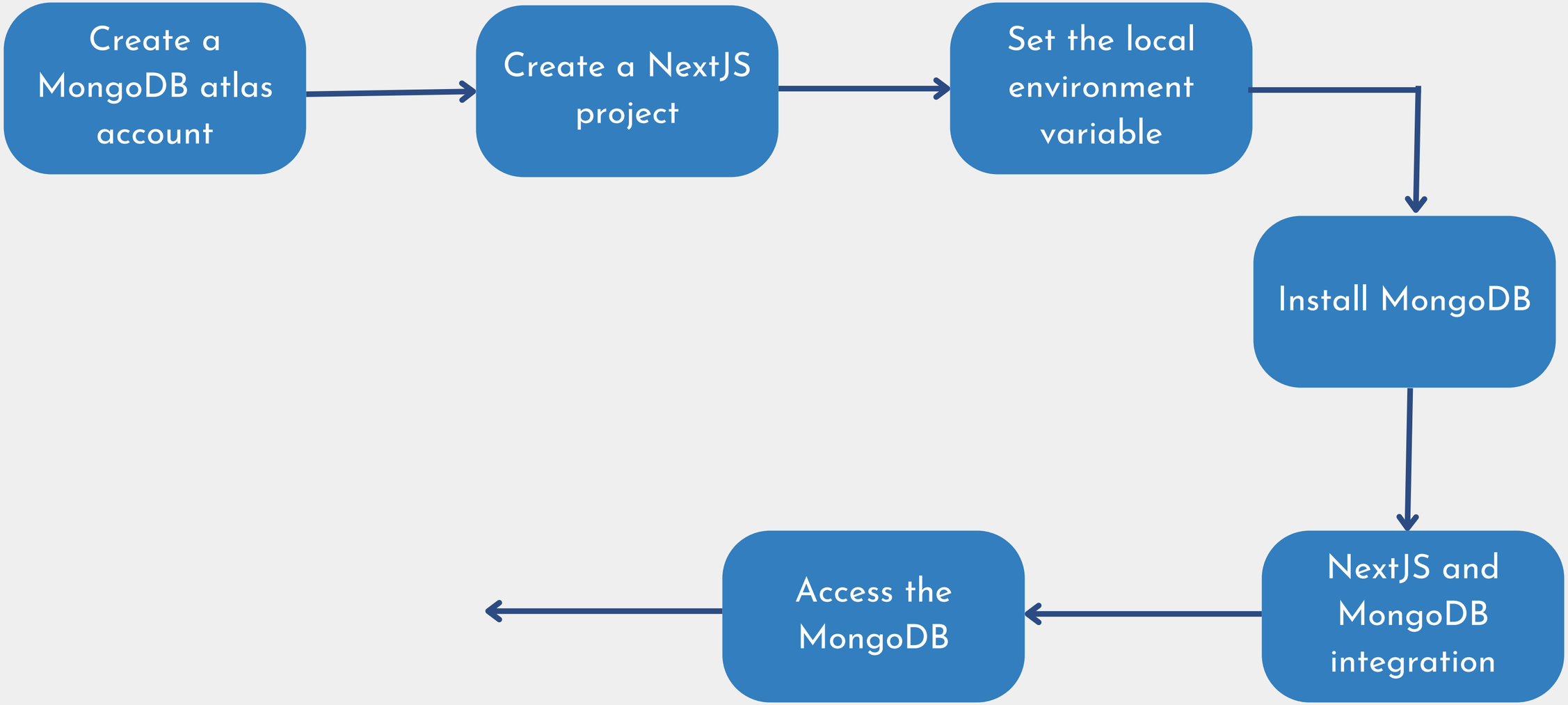
Connecting and Querying to a MongoDB database



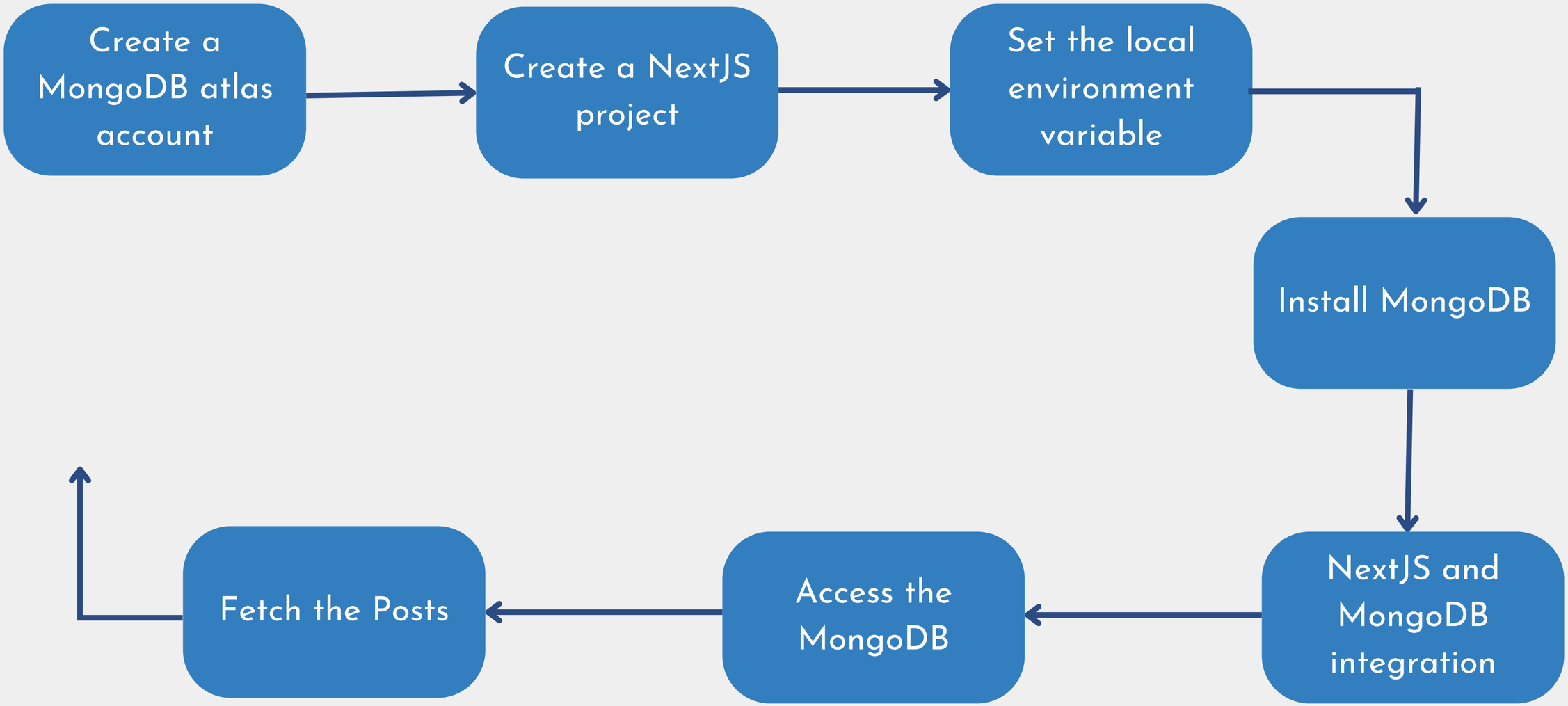
Connecting and Querying to a MongoDB database



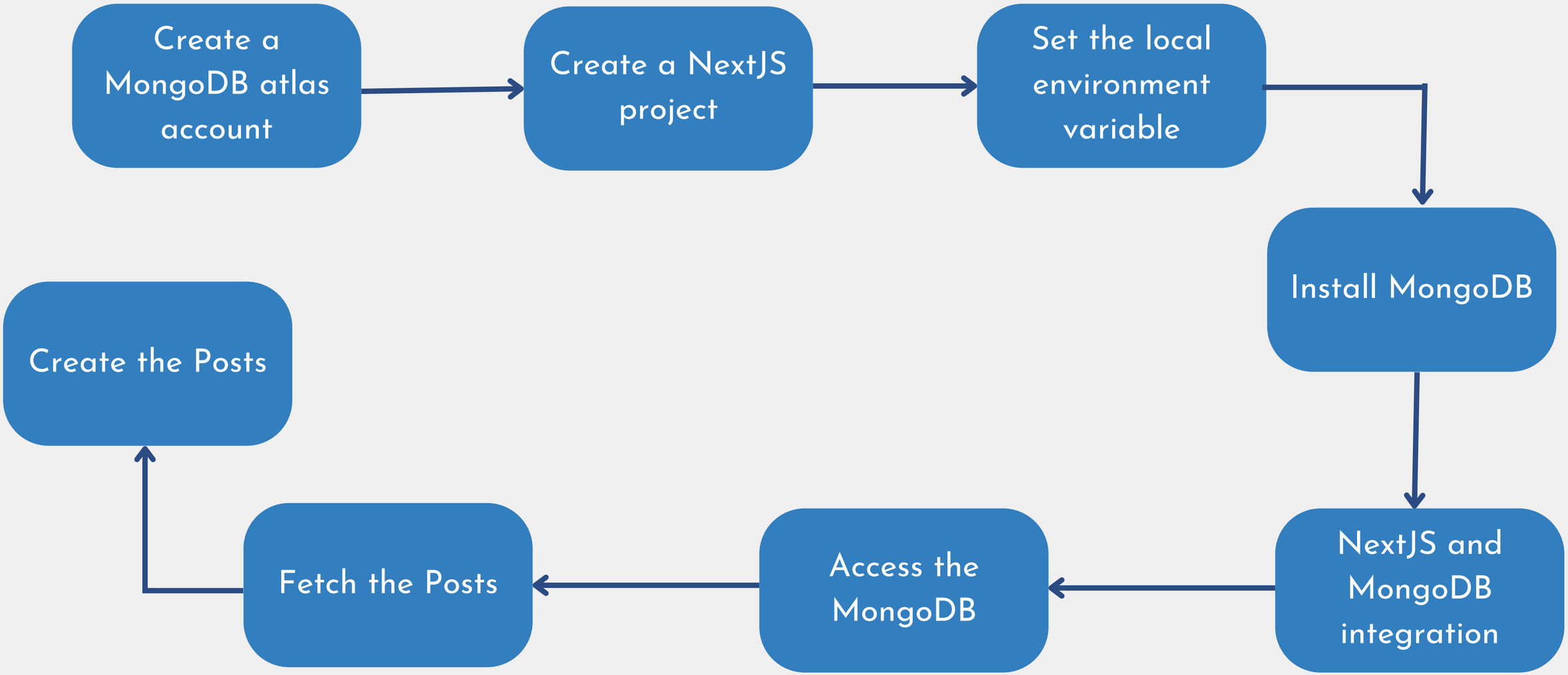
Connecting and Querying to a MongoDB database



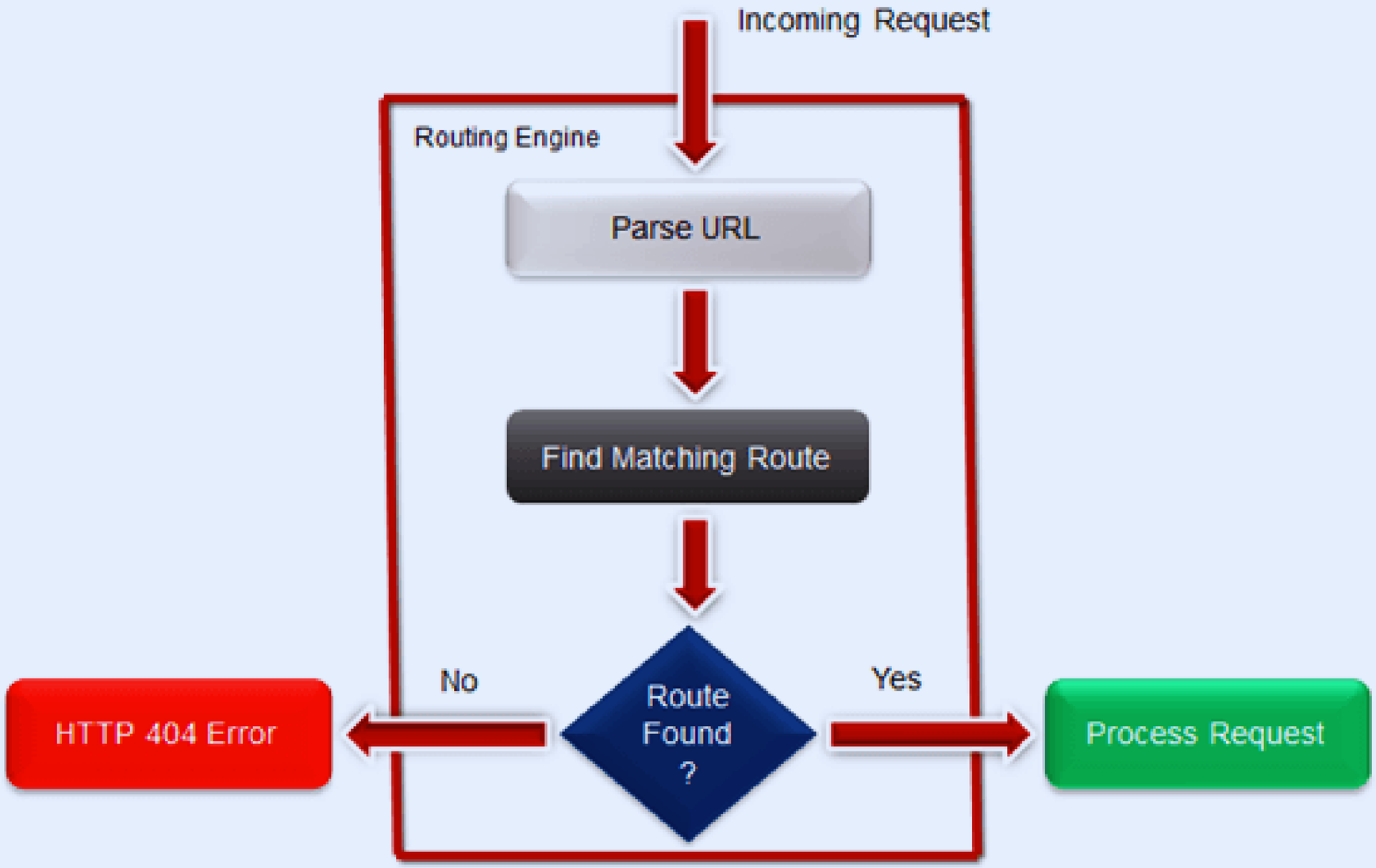
Connecting and Querying to a MongoDB database



Connecting and Querying to a MongoDB database



HOW ROUTING WORKS



API Routes in NextJS

- Any file inside the folder pages/api is mapped to /api/* and will be treated as an API endpoint instead of a page.

- They are server-side only bundles and won't increase your client-side bundle size.

- The following API route pages/api/user.js returns a json response with a status code of 200:.

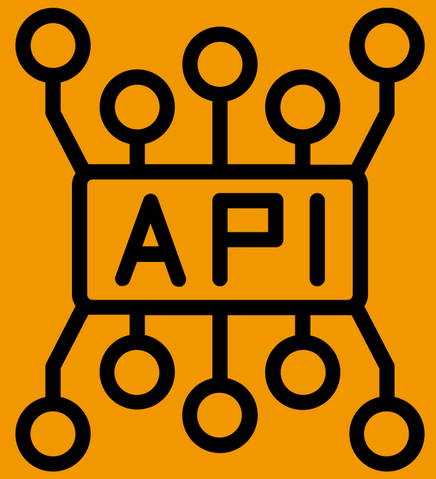
```
export default function
handler(req, res) {
  res.status(200).json({ name:
'John Doe' })
}
```

For an API route to work, you need to export a function as default (a.k.a request handler), which then receives the following parameters:

req: An instance of http.IncomingMessage, plus some pre-built middlewares

res: An instance of http.ServerResponse, plus some helper functions

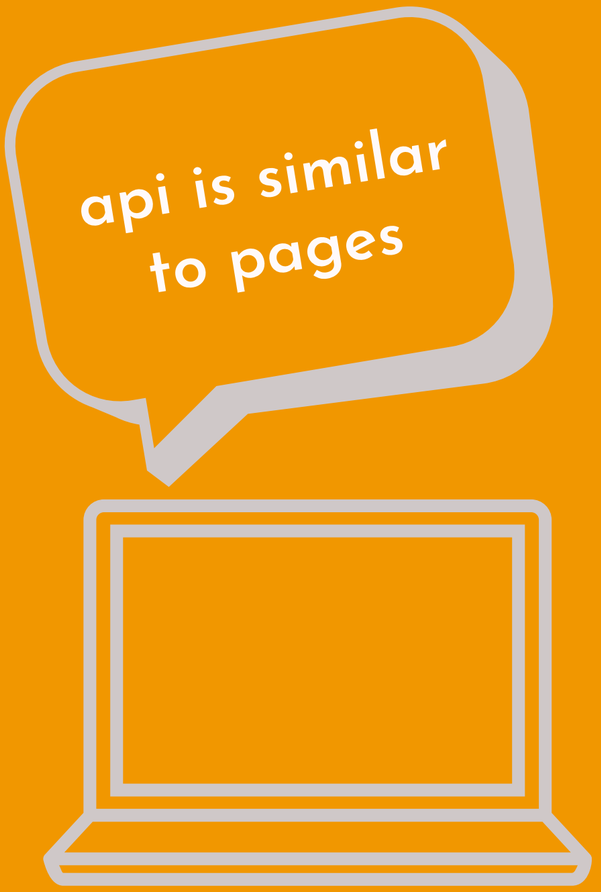
```
export default function
handler(req, res) {
  if (req.method === 'POST') {
    // Process a POST request
  } else {
    // Handle any other HTTP method
  }
}
```



Dynamic API Routes

- API routes support dynamic routes, and follow the same file naming rules used for pages.
- For example, the API route pages/api/post/[pid].js has the following code:

```
export default function
handler(req, res) {
  const { pid } = req.query
  res.end(`Post: ${pid}`)
}
```



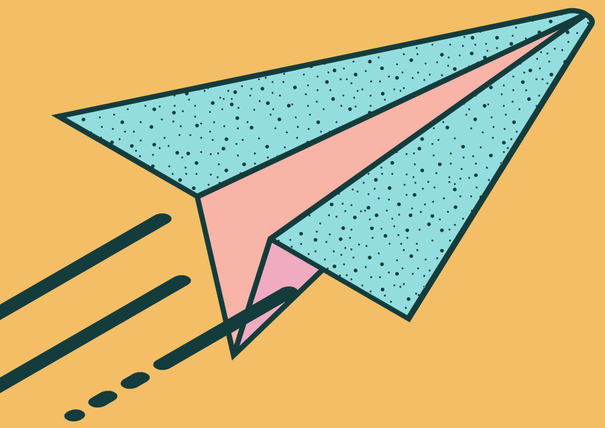
```
export async function getStaticPaths() {
  return {
    paths: [{ params: { id: '1' } }, {
params: { id: '2' } }],
    fallback: false, // can also be true
or 'blocking'
  }
}

// `getStaticPaths` requires using
`getStaticProps`
export async function
getStaticProps(context) {
  return {
    // Passed to the page component as
props
    props: { post: {} },
  }
}
```

getStaticPaths()

- If a page has Dynamic Routes and uses getStaticProps, it needs to define a list of paths to be statically generated.

- When we export a function called getStaticPaths from a page that uses dynamic routes, Next.js will statically pre-render all the paths specified by getStaticPaths.

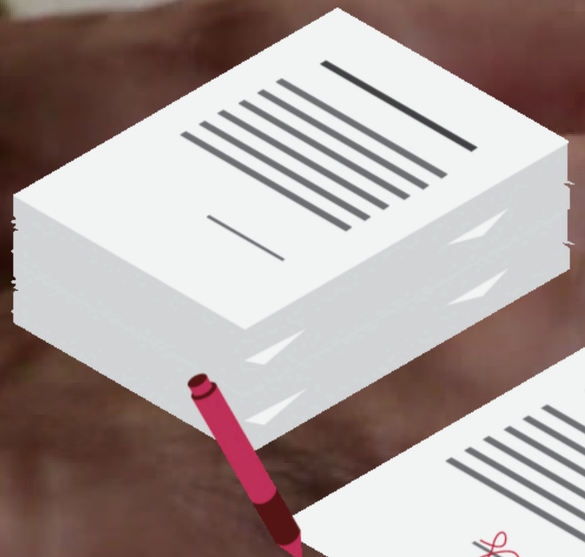
getStaticprops



- If you export a function called getStaticProps (Static Site Generation) from a page, Next.js will pre-render this page at build time using the props returned by getStaticProps.

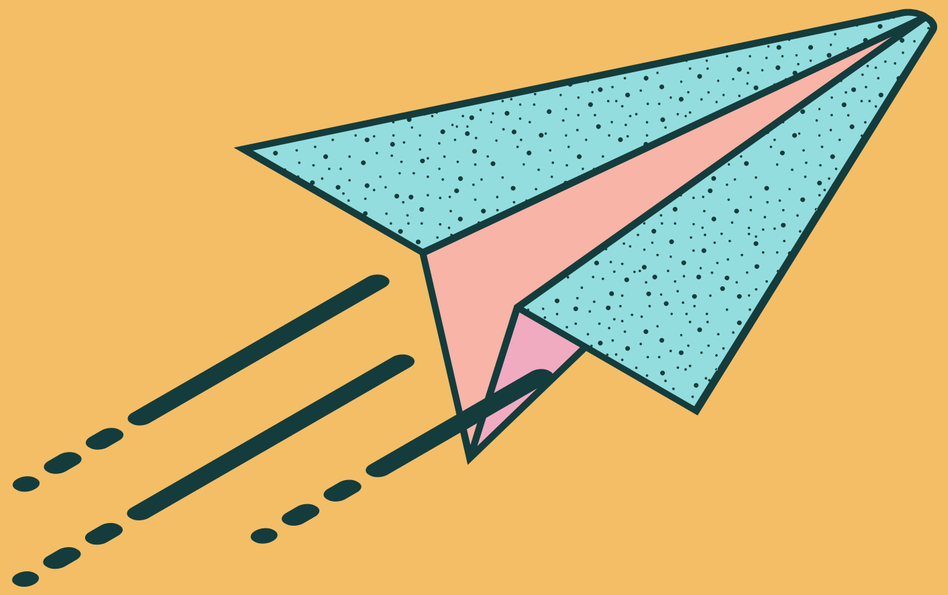
- getStaticProps always runs on the server and never on the client. You can validate code written inside getStaticProps is removed from the client-side bundle with this tool.

Using getStaticProps ()



```
function Homepage(props){
  return <MeetupList meetups=
{props.meetups}/>
}
export async function getStaticProps(){
  //fetch data from an API
  return{
    props:{
      meetups:
      // will be passed to the page
component as props
    },
      revalidate:10;
      //this number is no. of seconds
next.js will wait until it regenerate
the page for an upcoming event

  };
}
export default Homepage;
```

getServerSideProps



- If we export a function called getServerSideProps from a page, Next.js will pre-render this page on each request using the data returned by getServerSideProps.

- This function only runs on server-side and never runs on the browser and is rendered at run time.

Using getServerSideProps to fetch data at request time

```
export async function
getServerSideProps() {
  // Fetch data from external API
  const res = await
fetch(`https://.../data`)
  const data = await res.json()

  // Pass data to the page via props
  return { props: { data } }
}
```

getStaticProps()

- A method that tells the Next component to populate props and render into a static HTML page at *build time*.

getStaticProps()

- A method that tells the Next component to populate props and render into a static HTML page at *build time*.

- Prior to hosting the page, the developer converts the React into raw HTML pages, and only then are the pages hosted and served to clients.

getStaticProps()

- A method that tells the Next component to populate props and render into a static HTML page at *build time*.

- Prior to hosting the page, the developer converts the React into raw HTML pages, and only then are the pages hosted and served to clients.

getServerSideProps()

- A method that tells the Next component to populate the props and render into a static HTML page at *run time*.

getStaticProps()

- A method that tells the Next component to populate props and render into a static HTML page at *build time.*

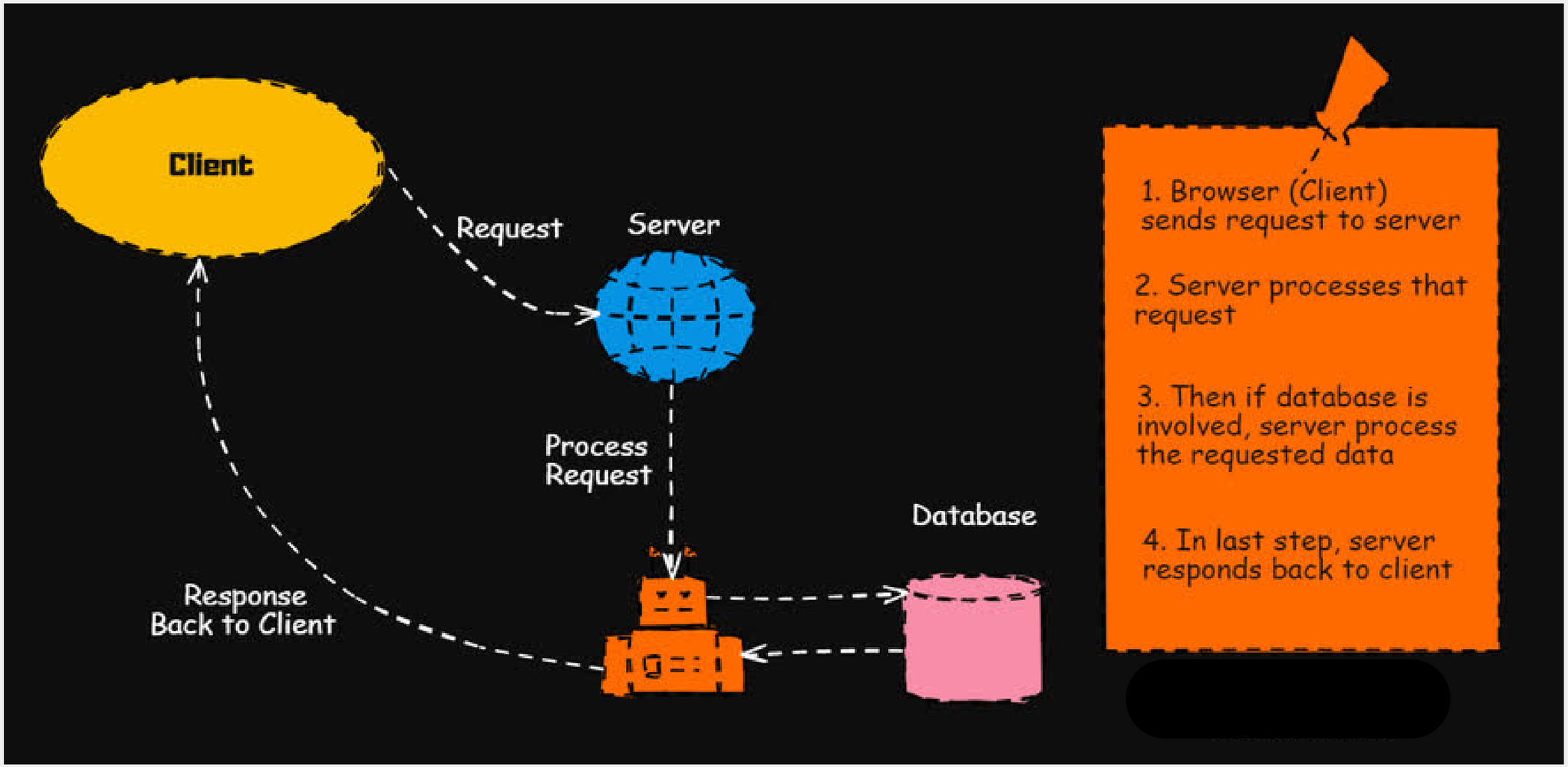
- Prior to hosting the page, the developer converts the React into raw HTML pages, and only then are the pages hosted and served to clients.

getServerSideProps()

- A method that tells the Next component to populate the props and render into a static HTML page at *run time.*

- The site is live and hosted with React components sitting on the server awaiting a request. Next Js tells React to go ahead and render these components into HTML in real time, send the HTML to the client.

HTTP Requests



Deploying Next.js as a Static Site

- To deploy the Next.js app to App Platform as a Static Site, we'll use Next's built-in commands to generate all of your HTML files.
- We first call next build, followed by next export.
- To make it easier, create a single npm run export command that calls both.
- Open package.json in your editor and add the following export script to the file.

```
"scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "export": "npm run build && next
export -o _static"
},
```

Stats on NEXT.JS

Next.js became popular because it made it easier to create full-stack applications with React.js. It has built-in mechanisms for prerendering, data fetching, and intelligent page routing.

Stats on NEXT.JS

Next.js is used by the biggest companies like Netflix, Twitch, Starbucks, Nike, Hulu and more.
It is considered one of the fastest-growing React frameworks.
Google favors websites built with the Next.js framework by ranking them higher
Next.js saw over 6 million downloads since its inception

Deploy to Vercel

Vercel is a serverless platform for static and hybrid applications built to integrate with your headless content, commerce, or database.

Deploy to Vercel

Vercel is a serverless platform for static and hybrid applications built to integrate with your headless content, commerce, or database.

Create a Vercel Account →

Deploy to Vercel

Vercel is a serverless platform for static and hybrid applications built to integrate with your headless content, commerce, or database.

Create a Vercel Account → Import your nextjs repository on Vercel →

Deploy to Vercel

Vercel is a serverless platform for static and hybrid applications built to integrate with your headless content, commerce, or database.

Create a Vercel Account → Import your nextjs repository on Vercel → When you deploy, your Next.js app will start building. It should finish in under a minute.

Deploy to Vercel

Vercel is a serverless platform for static and hybrid applications built to integrate with your headless content, commerce, or database.

Create a Vercel Account → Import your nextjs repository on Vercel → When you deploy, your Next.js app will start building. It should finish in under a minute.

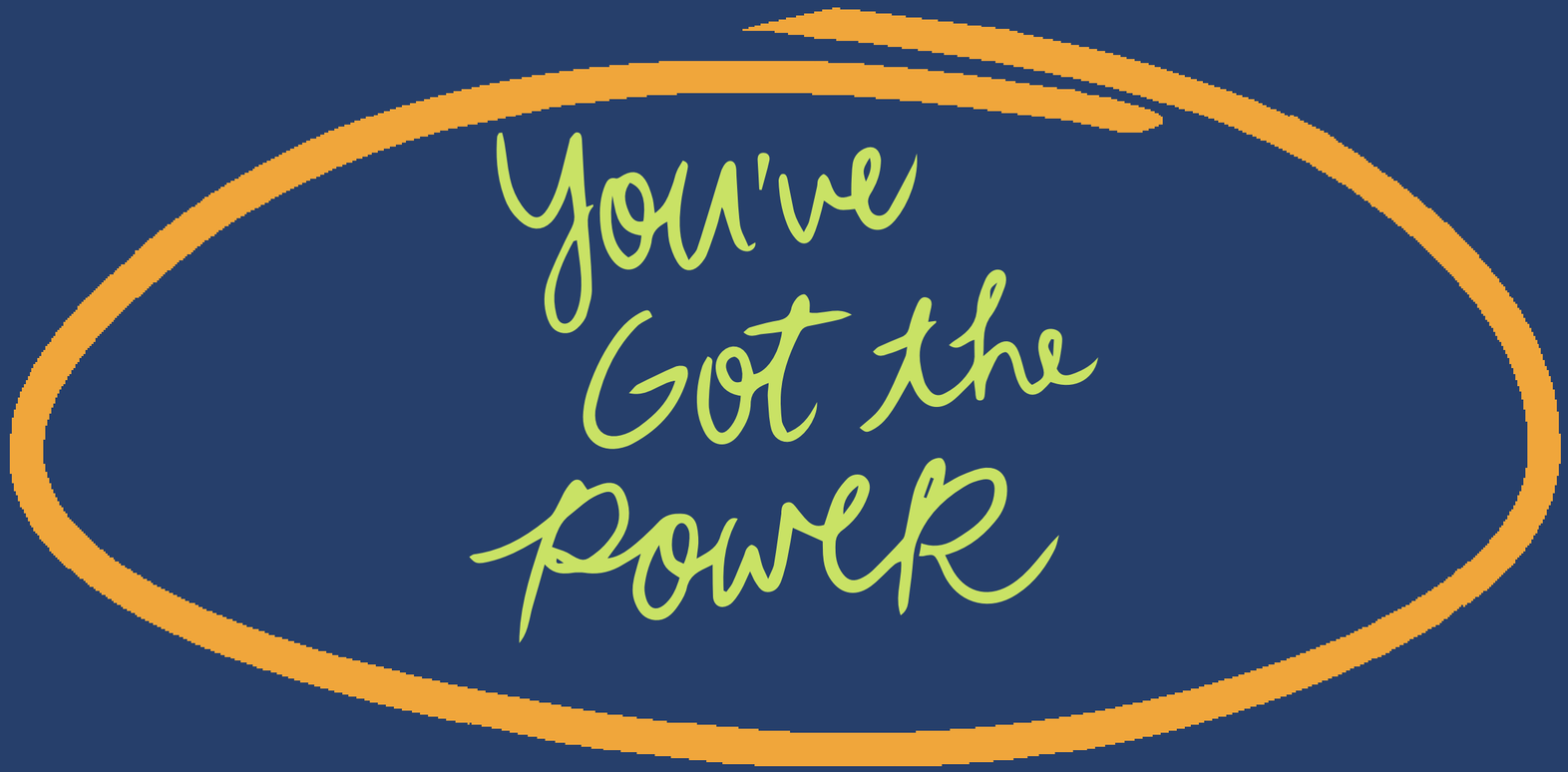
When it's done, you'll get deployment URLs. Click on one of the URLs and you should see the Next.js starter page live.

Visit http://kahoot.com

It's time for Quiz

Firr se Prize Milenge!

Code-


You've
Got the
Power

THANK YOU!