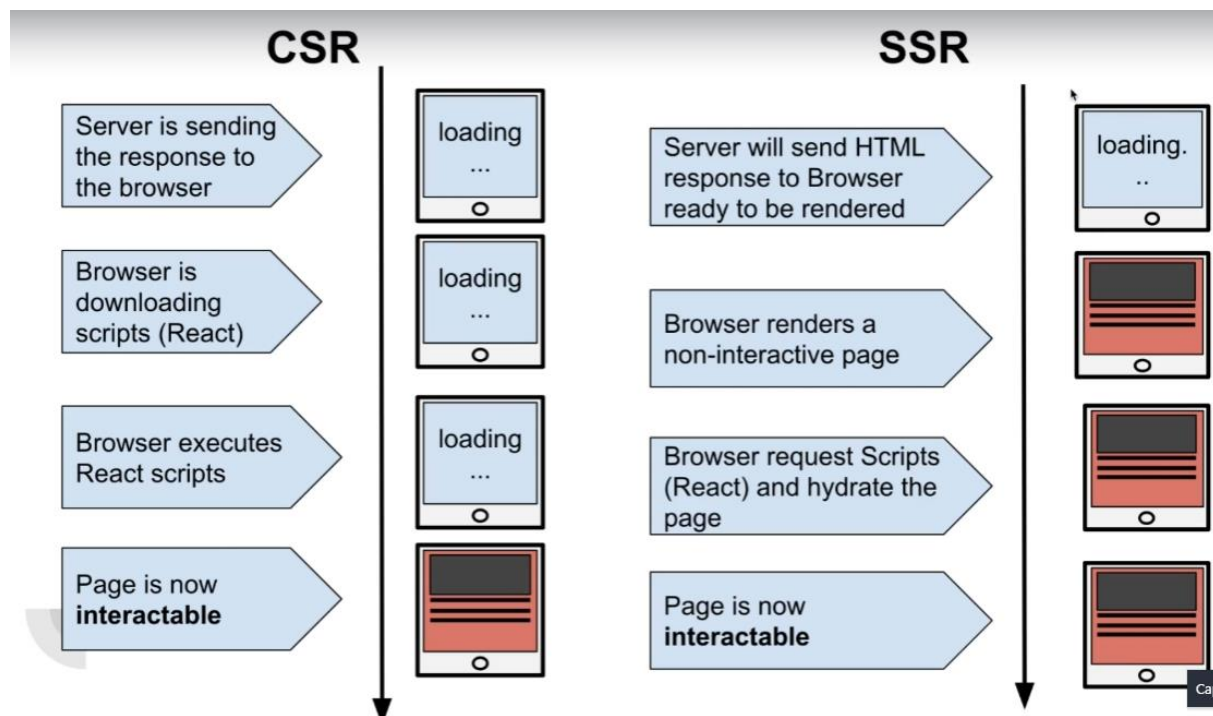


## NEXT JS main benefits:

- SSR itself which is the main purpose of the framework;
- Easy React SSR project scaffolding;
- Low project setup;
- Extensible through Webpack and Babel configuration;
- Easy built-in routing system (convention over configuration)
- Hot Module Reload out of the box;
- Typescript support
- Easy deployment with Zeit's Now hosting;



# Using latest version

Hello all!

My course was launched on Jan 2020, but Next.js team already made an update on their API which might affect you if you install the latest Next.js version.

Since version 9.3, Next.js now comes with a couple new methods to be used when fetching data for a page. So now, instead of only **getInitialProps**, we have:

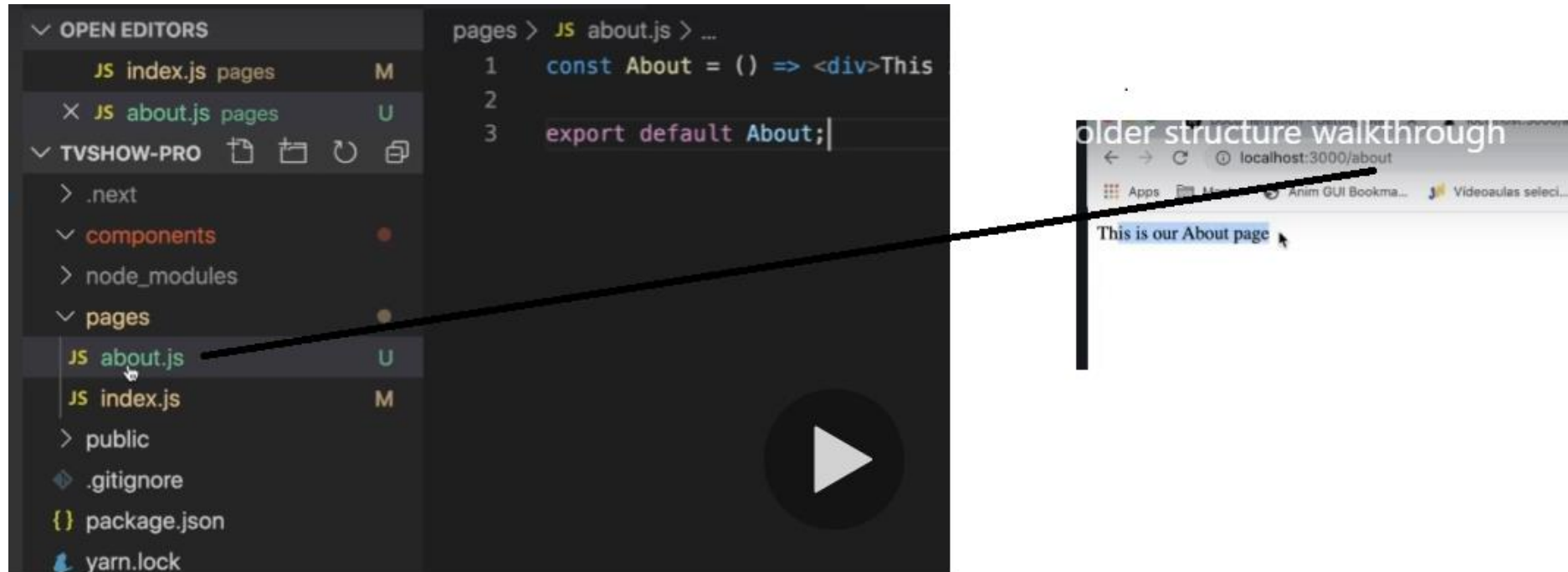
- `getStaticProps`
- `getServerSideProps` (which is equivalent of `getInitialProps` for us)

I'd like to reinforce that **getInitialProps is still compatible with the latest version**. However, if you would like to follow the documentation and start using **getServerSideProps** instead you are more than welcome to do it. But remember: you will need to stick with that till the end. Mixing the 2 methods won't work.

---

## Navigation

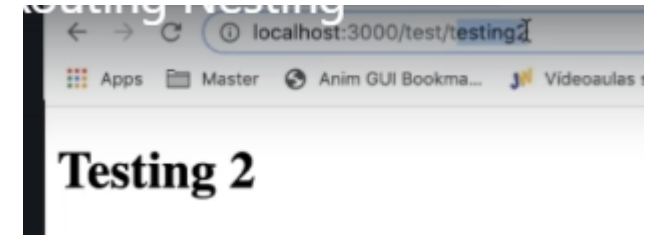
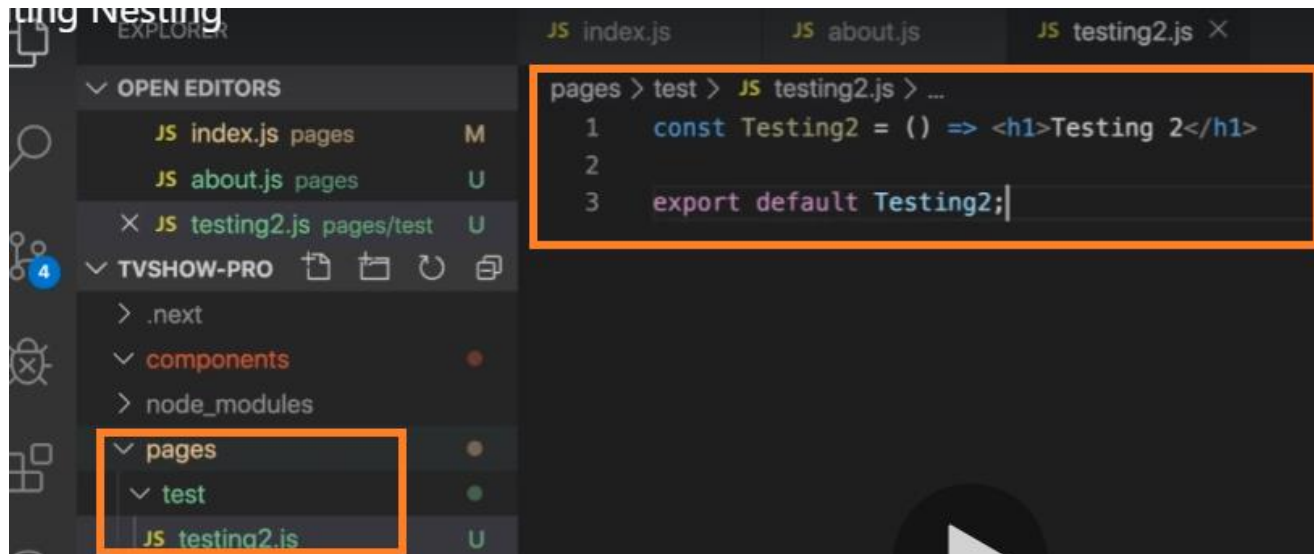
1 file under pages folder = new route where the url is the file name



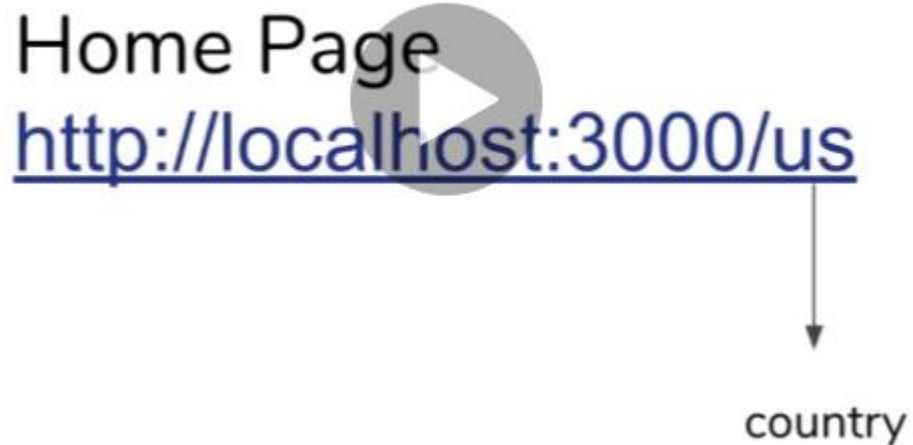
If bad url → Next.js has a 404 by default



## Routing nesting



## Dynamic routing



```
pages > [country] > JS index.js > ...
1  const CountryTest = () => <h1>This is a country Test</h1>
2
3  export default CountryTest;
```



Show Detail Page

<http://localhost:3000/us/5617>

Show id

country

## Dynamic Routing

OPEN EDITORS 1 UNSAVED

JS [showId].js pages/[co... U

TVSHOW-PRO

> .next

> components

> node\_modules

> pages

> [country]

JS [showId].js U

JS index.js U

JS [showId].js

pages > [country] > JS [showId].js > ...

```
1 const ShowDetails = () => <h1>Show Details</h1>
2
3 export default ShowDetails;
```

Fetching data from server side

!!! warning !!! => new version we have `getStaticProps` and `getServerSideProps`

Dynamic routing

Populating `<head>`

Fetching data and component lifecycle

Routing

With `<Link>`

Basic Example

Custom routes (using props from URL)

With URL object

Replace instead of push url

Using a component that supports

```
import fetch from 'isomorphic-unfetch'

function Page({ stars }) {
  return <div>Next stars: {stars}</div>
}

Page.getInitialProps = async ({ req }) => {
  const res = await fetch('https://api.github.com/repos/zeit/next.js')
  const json = await res.json()
  return { stars: json.stargazers_count }
}

export default Page
```

Fetching data from **client**

```
JS [showId].js JS index.js X
pages > [country] > JS index.js > [🔍] Home
1  import { useEffect } from 'react'; 8.4K (gzipped: 3.4K)
2  import axios from 'axios'; 15.2K (gzipped: 5.2K)
3
4  const Home = () => {
5    useEffect(() => {
6      axios.get('http://api.tvmaze.com/schedule?country=US&date=2014-12-01')
7        .then(response => console.log(response.data))
8    }, [])
9
10   return (
11     <h1>This is a country Test</h1>
12   )
13 }
14
15 export default Home;
```

Fetching data from server side

```
showId].js JS index.js
s > [country] > JS index.js > [🔍] Home
import { useEffect } from 'react'; 8.4K (gzipped: 3.4K)
import axios from 'axios'; 15.2K (gzipped: 5.2K)

const Home = (props) => {
  console.log("TCL: Home -> props", props.shows)

  return (
    <h1>This is a country Test</h1>
  )
}

Home.getInitialProps = async () => {
  const response = await axios.get('http://api.tvmaze.com/schedule?country=US&date=2014-12-01')
  return {
    shows: response.data
  }
}

export default Home;
```

It's important to point that first the method `Home.getInitialProps` is executed and **only after the component** is rendered



## Retrieve the country from the url

Home Page

<http://localhost:3000/us>

country

g data from other countries

EXPLORER

OPEN EDITORS

- JS [showId].js pages/[co... U
- JS index.js pages/[country] U

TVSHOW-PRO

- > .next
- > .vscode
- > components
- > node\_modules
- > pages
  - > [country]
    - JS [showId].js U
    - JS index.js U
- > public
- > .gitignore
- { } package.json M
- 🐧 yarn.lock M

pages > [country] > JS index.js > Home.getInitialProps 15.2K (gzipped: 5.2K)

```
1 import axios from 'axios';
2
3 const Home = ({ shows }) => {
4   const renderShows = () => {
5     return shows.map((showItem) => {
6       const { show } = showItem;
7       return <li key={index}>
8         <div>
9           <h3>{show.name}</h3>
10          <div>
11            <div>{show.genres}</div>
12            <div>{show.schedule}</div>
13          </div>
14        </div>
15      </li>
16    });
17  };
18
19  return <ul className="tvshows">{renderShows()}</ul>;
20 };
21
22 Home.getInitialProps = async context => {
23   const country = context.query.country || 'us';
24
25   const response = await axios.get(
26     `http://api.tvmaze.com/schedule?country=${country}&date=2014-12-01`
27   );
28 }
```

[Symbol(kOutHeaders)]: null

pathname: '/[country]',

query: { country: 'us' },

asPath: '/us',

AppTree: [Function: AppTree]

we have acces to a context which contains a lot of information ==> path

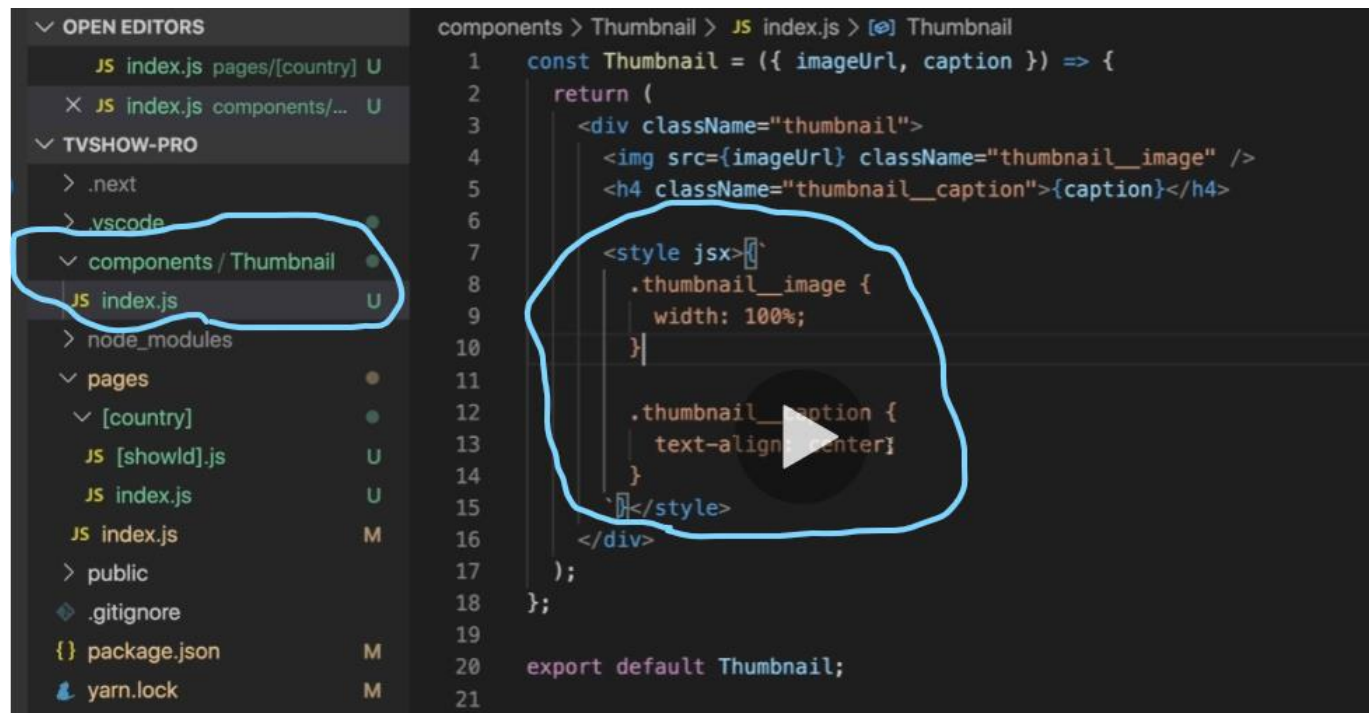
## Styling a component

Using the Built-in css provided by Next.js or we can import a CSS / Sass / Less / Stylus files

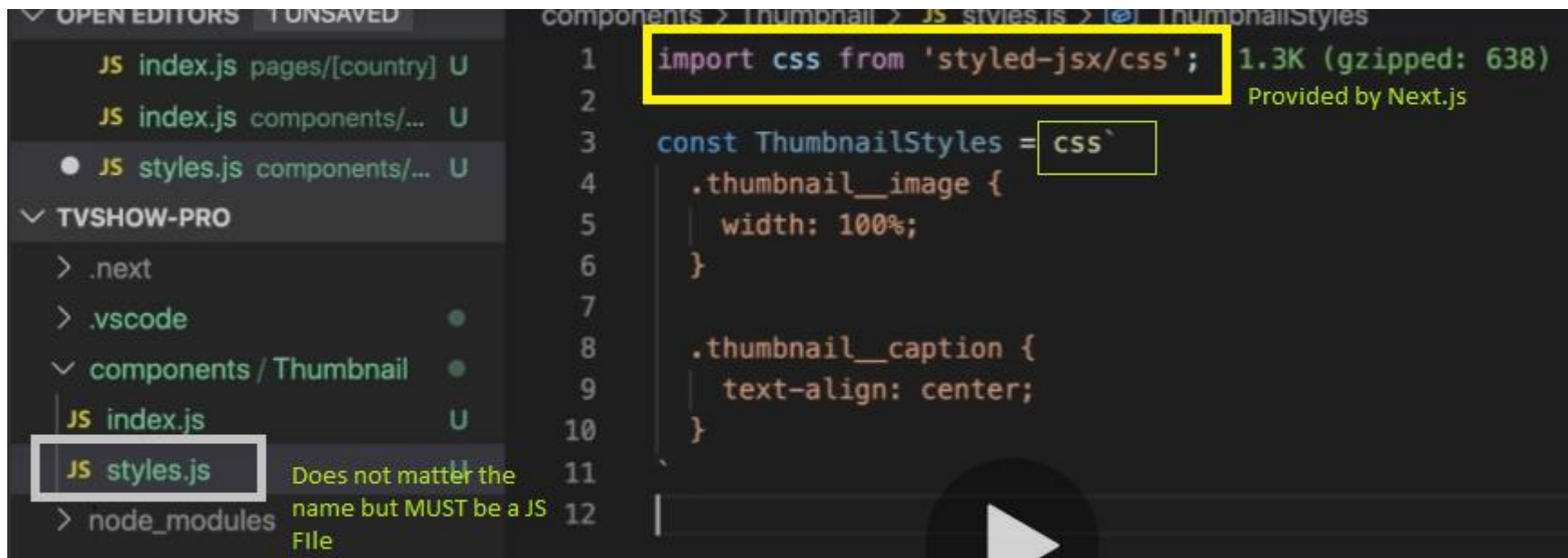
Now add `<style jsx>` to your code and fill it with CSS:

```
export default () => {  
  <div>  
    <p>only this paragraph will get the style :)</p>  
  
    { /* you can include <Component />s here that include  
      other <p>s that don't get unexpected styles! */ }  
  
    <style jsx>{`  
      p {  
        color: red;  
      }  
    `}</style>  
  </div>  
}
```

It's possible to have our css in our component but it's just BERKKKKKKKKKKKKKKKKKK



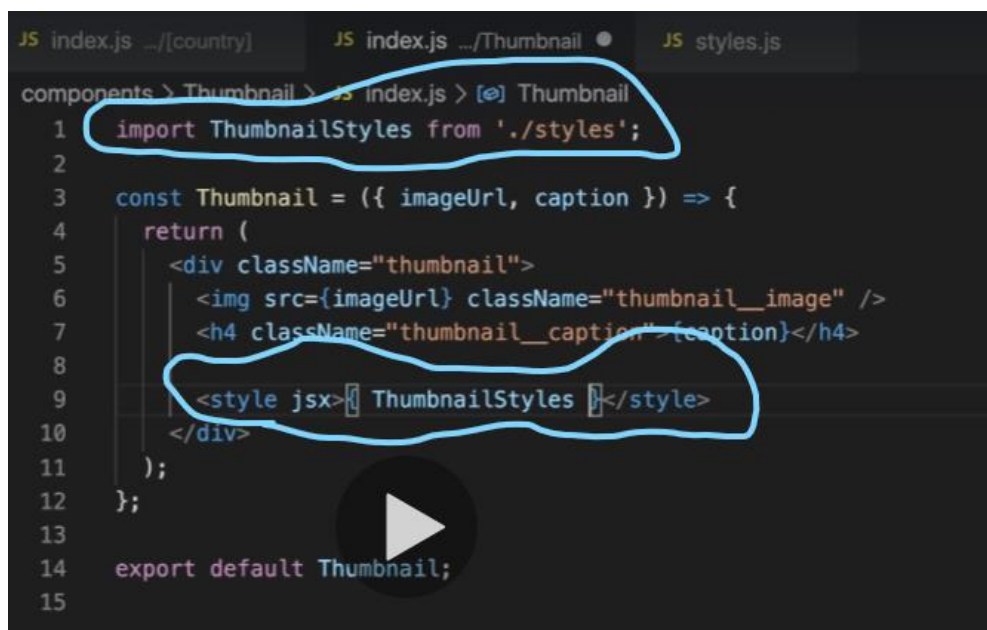
## Extract css from component file



```
1 import css from 'styled-jsx/css';
2
3 const ThumbnailStyles = css`
4   .thumbnail__image {
5     width: 100%;
6   }
7
8   .thumbnail__caption {
9     text-align: center;
10  }
11
12  export default ThumbnailStyles;
```

Does not matter the name but MUST be a JS File

export default ThumbnailStyles;



```
1 import ThumbnailStyles from './styles';
2
3 const Thumbnail = ({ imageUrl, caption }) => {
4   return (
5     <div className="thumbnail">
6       <img src={imageUrl} className="thumbnail__image" />
7       <h4 className="thumbnail__caption">{caption}</h4>
8       <style jsx>{ThumbnailStyles}</style>
9     </div>
10   );
11 };
12
13 export default Thumbnail;
```

### Importing CSS / Sass / Less / Stylus files

To support importing `.css`, `.scss`, `.less` or `.styl` files you can use these modules, which configure sensible defaults for server rendered applications.

- [@zeit/next-css](#)
- [@zeit/next-sass](#)
- [@zeit/next-less](#)
- [@zeit/next-stylus](#)

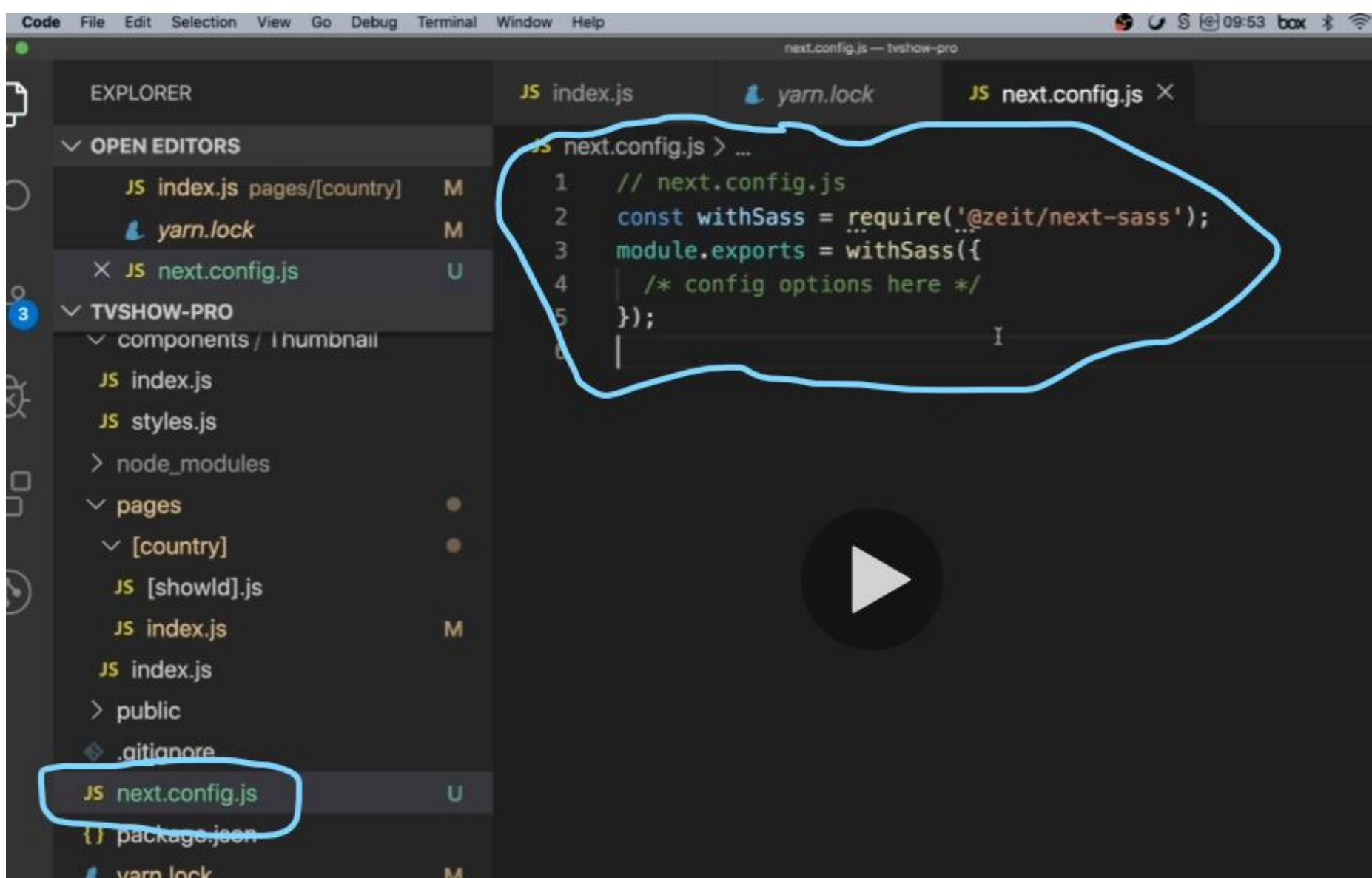
---

### Installation

```
npm install --save @zeit/next-sass node-sass
```

or

```
yarn add @zeit/next-sass node-sass
```





components > ThumbnailWithSass > JS index.js > Thumbnail

```
1  const Thumbnail = ({
2    imageUrl = 'https://via.placeholder.com/210x295?text=?',
3    caption
4  }) => {
5    return (
6      <div className="thumbnail">
7        <img src={imageUrl} className="thumbnail__image" />
8        <div className="thumbnail__caption">{caption}</div>
9      </div>
10    );
11  };
12
13  export default Thumbnail;
14
```

index.js .../ThumbnailWithSass JS index.js .../[country] styles.scss

components > ThumbnailWithSass > styles.scss > .thumbnail > &\_\_caption

```
1  .thumbnail {
2    &__image {
3      width: 100%;
4    }
5
6    &__caption {
7      text-align: center;
8      padding: 10px;
9    }
10 }
```

JS index.js .../ThumbnailWithSass X

JS index.js .../[country]

styles.scss

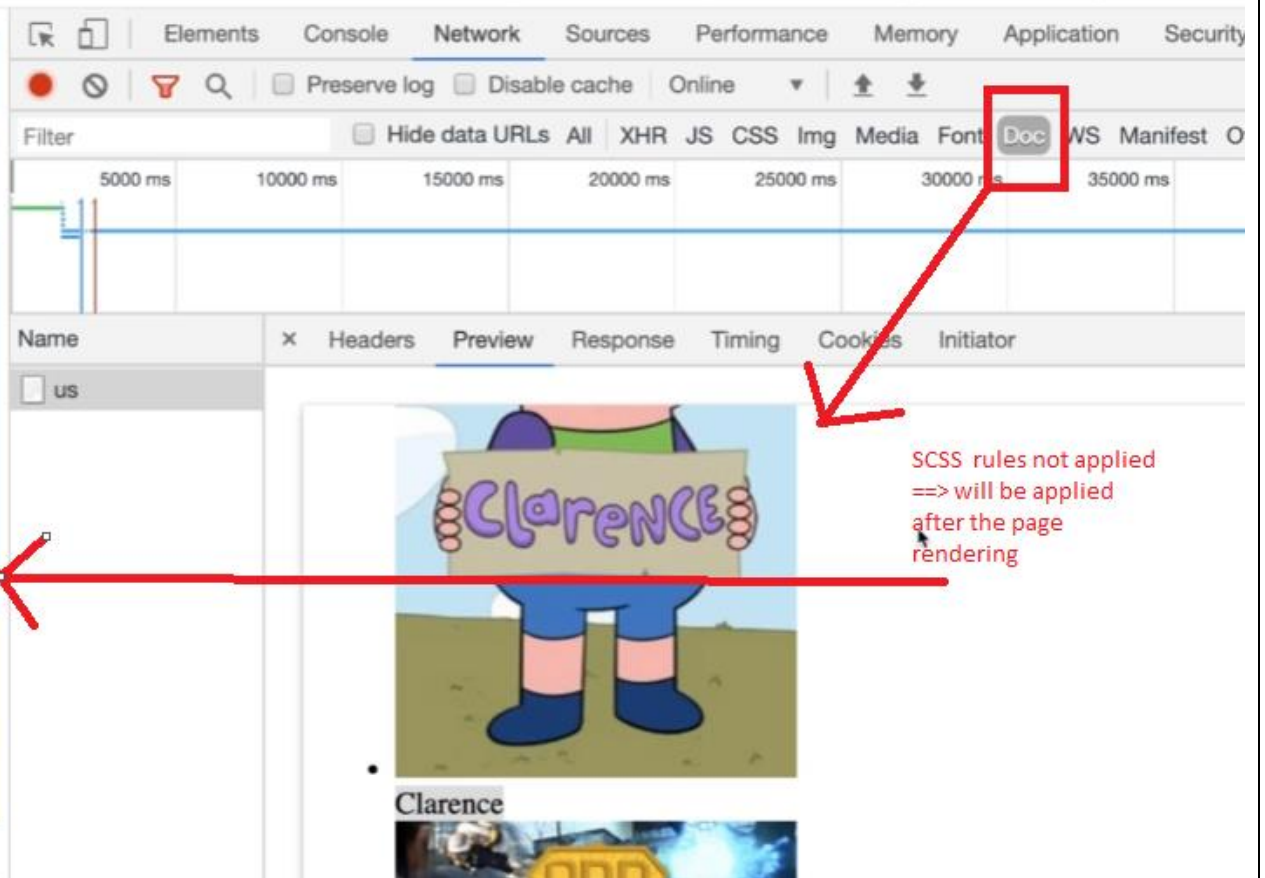
components &gt; ThumbnailWithSass &gt; JS index.js &gt; ...

```
1  import './styles.scss';|
2
3  const Thumbnail = ({
4    imageUrl = 'https://via.placeholder.com/210x295?text=?',
5    caption
6  }) => {
7    return (
8      <div className="thumbnail">
9        <img src={imageUrl} className="thumbnail__image" />
10       <div className="thumbnail__caption">{caption}</div>
11     </div>
12   );
13 };
14
15 export default Thumbnail;
16
```

## Difference between SASS and CSS

With SASS the css rules will be applied after the page was rendered

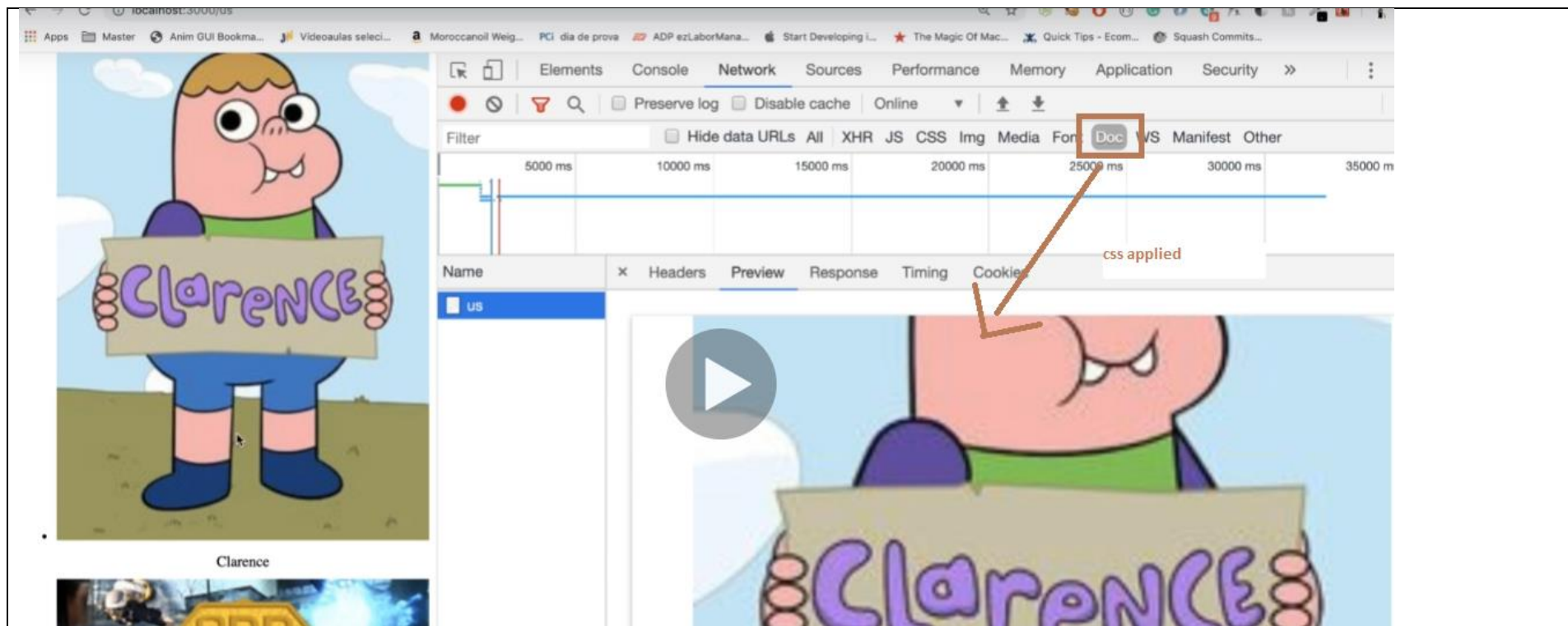
### SASS



SCSS rules not applied  
==> will be applied  
after the page  
rendering

### CSS





### Set a Default thumbnail

Show.image && show.image.medium => will return the image url

```
return (  
  <li key={index}>  
    <Thumbnail  
      imageUrl={show.image && show.image.medium || undefined}  
      caption={show.name}  
    />  
  </li>  
)  
);  
});
```

```

1  import ThumbnailStyles from './styles';
2
3  const Thumbnail = ({
4    imageUrl = 'https://via.placeholder.com/210x295?text=?',
5    caption
6  }) => {
7    return (
8      <div className="thumbnail">
9        <img src={imageUrl} className="thumbnail__image" />
10       <h4 className="thumbnail__caption">{caption}</h4>
11
12       <style jsx>{ThumbnailStyles}</style>
13     </div>
14   );
15 };
16
17 export default Thumbnail;

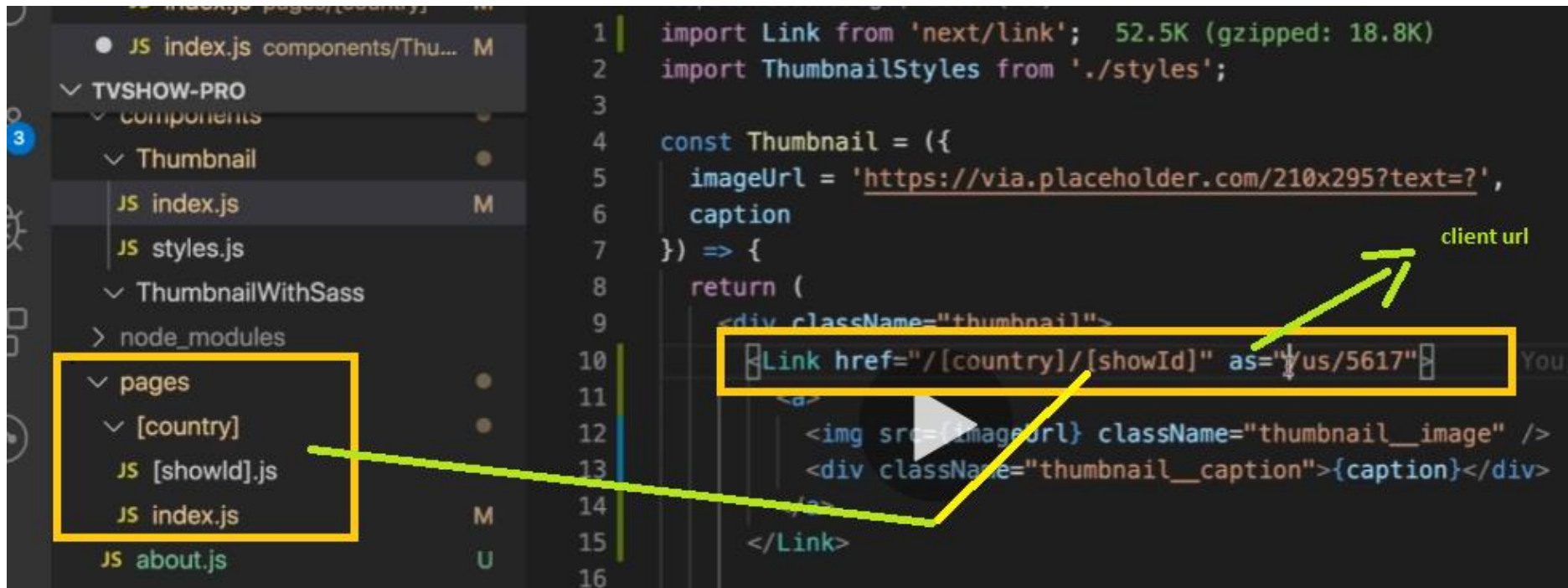
```

## Routing with LINK provided by Next.js

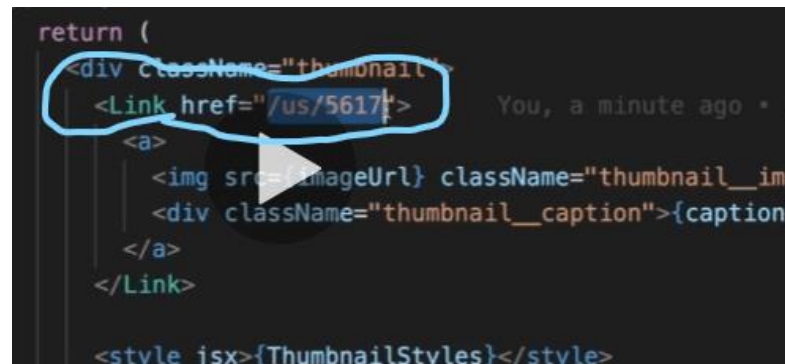
4. For client side routing, use `next/link`:

```
<Link href="/post?slug=something" as="/post/something">
```

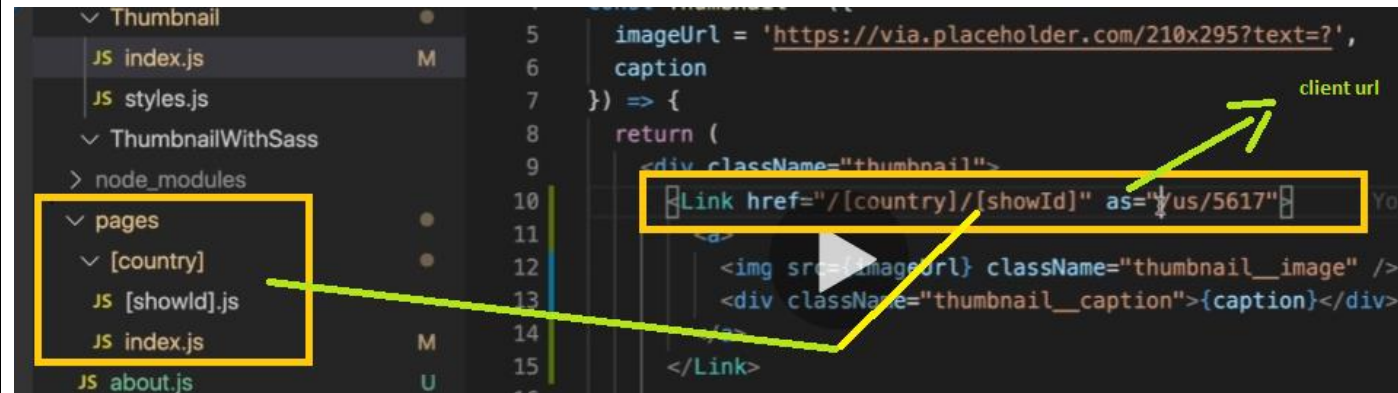
- `href`: the path inside `pages` directory
- `as`: the path used by your server routes



Without the **as** → server side routing → the page will be reloaded



With the **as** → client side routing



## Implement thumbnail with url

[country]/index.js

```
ges > [country] > JS index.js > Home > renderShows > shows.map() callback
Unsaved changes (cannot determine recent change or authors)
1 import axios from 'axios'; 15.2K (gzipped: 5.2K)
2 import Thumbnail from '../components/Thumbnail';
3
4 const Home = ({ shows, country }) => {
5   const renderShows = () => {
6     return shows.map((showItem, index) => {
7       const { show } = showItem;
8
9       return (
10        <li key={index}>
11          <Thumbnail
12            imageUrl={show.image && show.image.medium || undefined}
13            caption={show.name}
14            href="/[country]/[showId]"
15            as={`/ ${country}/${show.id}`}
16          />
17        </li>
18      );
19    });
20  };
21  }
```

Thumbnail/index.js

```
const Thumbnail = ({
  imageUrl = 'https://via.placeholder.com/210x295?text=
  caption,
  href = '',
  as = ''
}) => {
  return (
    <div className="thumbnail">
      <Link href={`${href}`} as={`${as}`} >
```

umbnails with Dynamic urls

The screenshot shows the VS Code interface with the Explorer sidebar on the left and the Source Explorer on the right. The Explorer sidebar shows the project structure with the following files and folders:

- JS index.js pages/[country] M
- JS index.js components/Thu... M
- TVSHOW-PRO
  - Thumbnail
  - JS index.js M
  - JS styles.js
  - ThumbnailWithSass
  - node\_modules
  - pages
    - [country]
      - [showId].js
      - JS index.js M
      - JS about.js U
      - JS index.js
  - public
  - .gitignore
  - package.json
  - varn lock

The Source Explorer on the right shows the code for `Home.getInitialProps` in `index.js`. The code is as follows:

```
34 }
35 }</style>
36 </ul>
37 );
38 };
39
40 Home.getInitialProps = async context => {
41   const country = context.query.country || 'us';
42
43   const response = await axios.get(
44     `http://api.tvmaze.com/schedule?country=${country}&date=2
45   );
46
47   return {
48     shows: response.data,
49     country
50   };
51 };
52
53 export default Home;
54
```



Like I mentioned earlier, we had an update in the latest version of Next.js and now the documentation recommends us to use **getServerSideProps** instead of **getInitialProps**. Like I said, this is optional for us at the moment and we can stick with `getInitialProps` as long as it is used throughout the app. Mixing the 2 methods is not allowed and your code will break. Below is an example of how the `getServerSideProps` would look like:

```
1  export const getServerSideProps = async ({ query }) => {
2    const { showId } = query;
3
4    try {
5      const response = await axios.get(
6        `https://api.tvmaze.com/shows/${showId}?embed=cast`
7      );
8
9      return {
10        props: {
11          show: response.data,
12        },
13      };
14    } catch (error) {
15      return {
16        props: {
17          error: error.error,
18        },
19      };
20    }
21  };
```

Please let me know if you have questions :)

## API Response contains html

```
const ShowDetails = ({ show }) => {
  const { name, image, summary } = show;

  return (
    <div className="show-details">
      <div
        className="show-details__poster"
        style={{ backgroundImage: `url(${image.original})` }}
      ></div>
      <h1>{name}</h1>
      <p>{summary}</p>
    </div>
  );
}
```

## API RESPONSE contains html

```
{
  "image": {
    "medium": "http://static.tvmaze.com/uploads/images/medium_portrait/81/202627.jpg",
    "original": "http://static.tvmaze.com/uploads/images/original_untouched/81/202627.jpg"
  },
  "summary": "<p><b>Under the Dome</b> is the story of a small town that is suddenly and in world by an enormous transparent dome. The town's inhabitants must deal with surviving th for answers about the dome, where it came from and if and when it will go away.</p>",
  "updated": 1573667713,
  "_links": {
    "self": {
      "href": "http://api.tvmaze.com/shows/1"
    }
  }
}
```



## Under the Dome

<p><b>Under the Dome</b> is the story of a small town that is suddenly and inexplicably sealed off from the rest of the world by an enormous transparent dome. The town's inhabitants must deal with surviving the post-apocalyptic conditions while searching for answers about the dome, where it came from and if and when it will go away.</p>

Use

## html-react-parser



npm install html-react-parser  
4 dependencies version 0.10.0  
updated 2 months ago

npm v0.10.0 build passing coverage 100% dependencies up to date

```
es > [country] > JS [showId].js > [⌘] ShowDetails
Unsaved changes (cannot determine recent change or authors)
import axios from 'axios'; 15.2K (gzipped: 5.2K)
import parse from 'html-react-parser'; 16.2K (gzipped: 6.5K)

const ShowDetails = ({ show }) => {
  const { name, image, summary } = show;

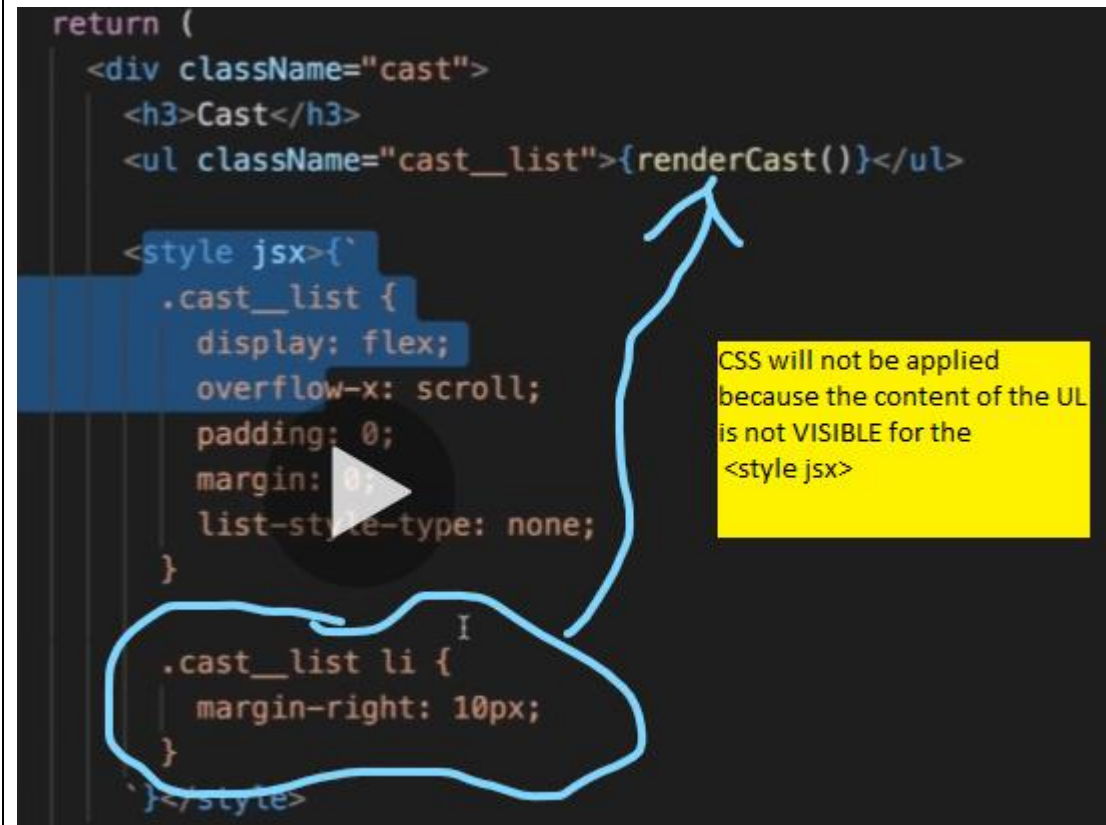
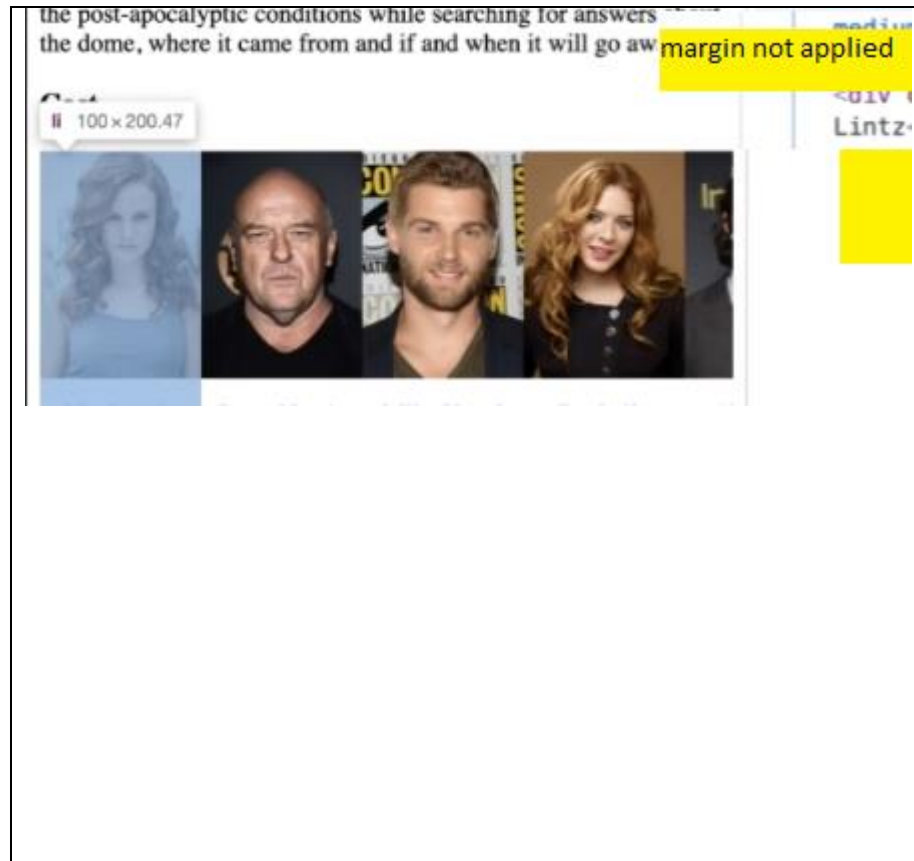
  return (
    <div className="show-details">
      <div
        className="show-details__poster"
        style={{ backgroundImage: `url(${image.original})` }}
      >
        <h1>{name}</h1>
        {parse(summary)}
      </div>
    </div>
  )
}
```



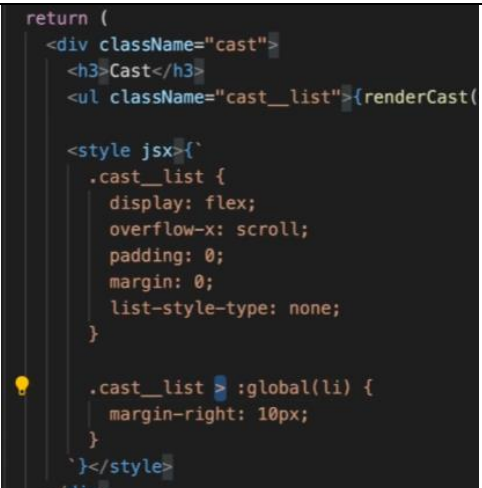
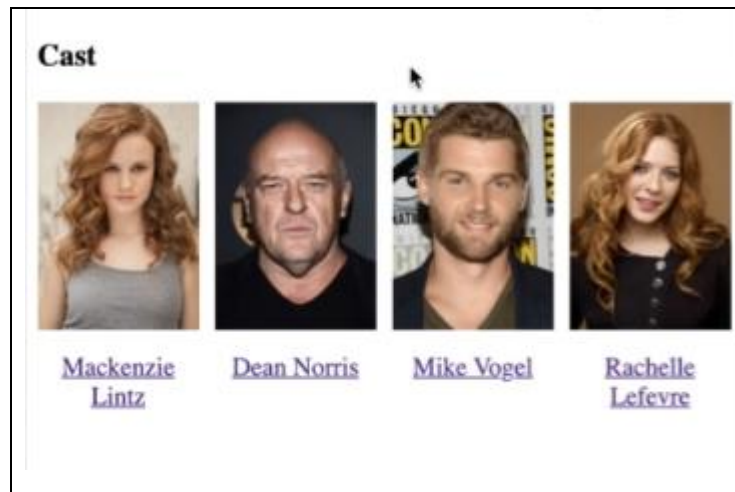
## Under the Dome

**Under the Dome** is the story of a small town that is suddenly and inexplicably sealed off from the rest of the world by an enormous transparent dome. The town's inhabitants must deal with surviving the post-apocalyptic conditions while searching for answers about the dome, where it came from and if and when it will go away.

## CSS ISSUE



Use global to fix the issue





## Tips

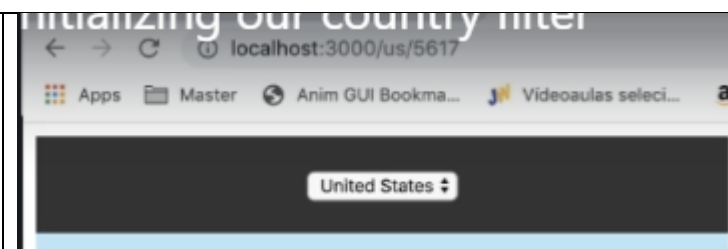
```
{parse(summary)}  
    any  
    {[_embedded.cast.length > 0 && <Cast cast={_embedded.cast} />]}  
  
    <style jsx>{'  
      .show-details poster {
```

Will render the component Cast if length > 0

## useRouter

```
ponents > Header > JS index.js > ...  
import { useRouter } from 'next/router';
```

```
const Header = () => {  
  const router = useRouter();  
  console.log('TCL: Header -> router', router);
```



```
Initializing our country inter  
TCL: Header -> router ServerRouter {  
  route: '/[country]/[showId]',  
  pathname: '/[country]/[showId]',  
  query: { country: 'us', showId: '5617' },  
  asPath: '/us/5617'  
}
```

```
const Header = () => {
  const router = useRouter();
  const [selectedCountry, setSelectedCountry] = useState(router.query.country);

  const handleChange = e => {
    setSelectedCountry(e.target.value);

    // /country
    router.push(`/[country]`, `/${e.target.value}`);
  };
}
```

```
return (
  <div className="header">
    <select value={selectedCountry} onChange={handleChange}>
      {renderCountries()}
    </select>
  </div>
);
```

Right now we're finally able to select a country

#### Client side routing

```
const Header = () => {
  const router = useRouter();
  const [selectedCountry, setSelectedCountry] = useState(router.query.country);

  const handleChange = e => {
    setSelectedCountry(e.target.value);

    // /country
    router.push(`/[country]`, `/${e.target.value}`);
  };
}
```

#### Server side routing

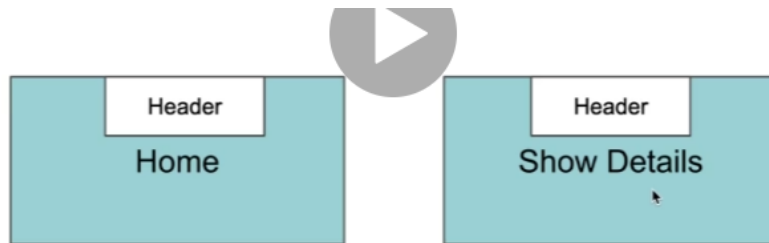
⇒ Will reload the page entirely

```
const handleChange = e => {
  setSelectedCountry(e.target.value);

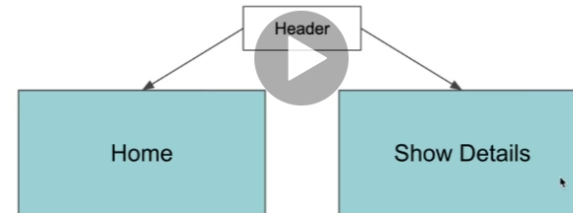
  // /country
  router.push(`/${e.target.value}`);
};
```

## Have only 1 header

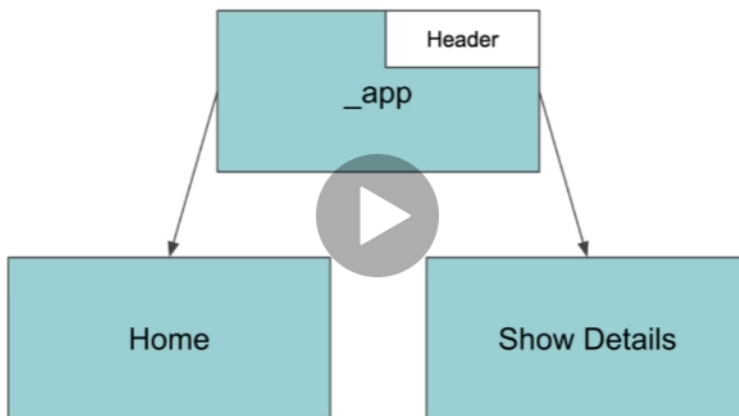
### Current situation



### What we want



⇒ Use `_app`



### Custom `<App>`

#### ► Examples

Next.js uses the `<App>` component to initialize pages. You can override it and control the page initialization. Which allows you to do amazing things like:

- Persisting layout between page changes
- Keeping state when navigating pages
- Custom error handling using `componentDidCatch`
- Inject additional data into pages (for example by processing GraphQL queries)

To override, create the `./pages/_app.js` file and override the App class as shown below:

To override, create the `./pages/_app.js` file and override the App class as shown below:

EXPLORER

1

OPEN EDITORS 1 UNSAVED

- JS index.js components/Hea... U
- JS \_app.js pages U

TVSHOW-PRO

- > .next
- > .vscode
- > components
- > node\_modules
- pages
  - [country]
  - JS [showId].js M
  - JS index.js M
  - JS \_app.js U
  - JS index.js
- > public
- .gitignore
- package.json M
- yarn.lock M

8

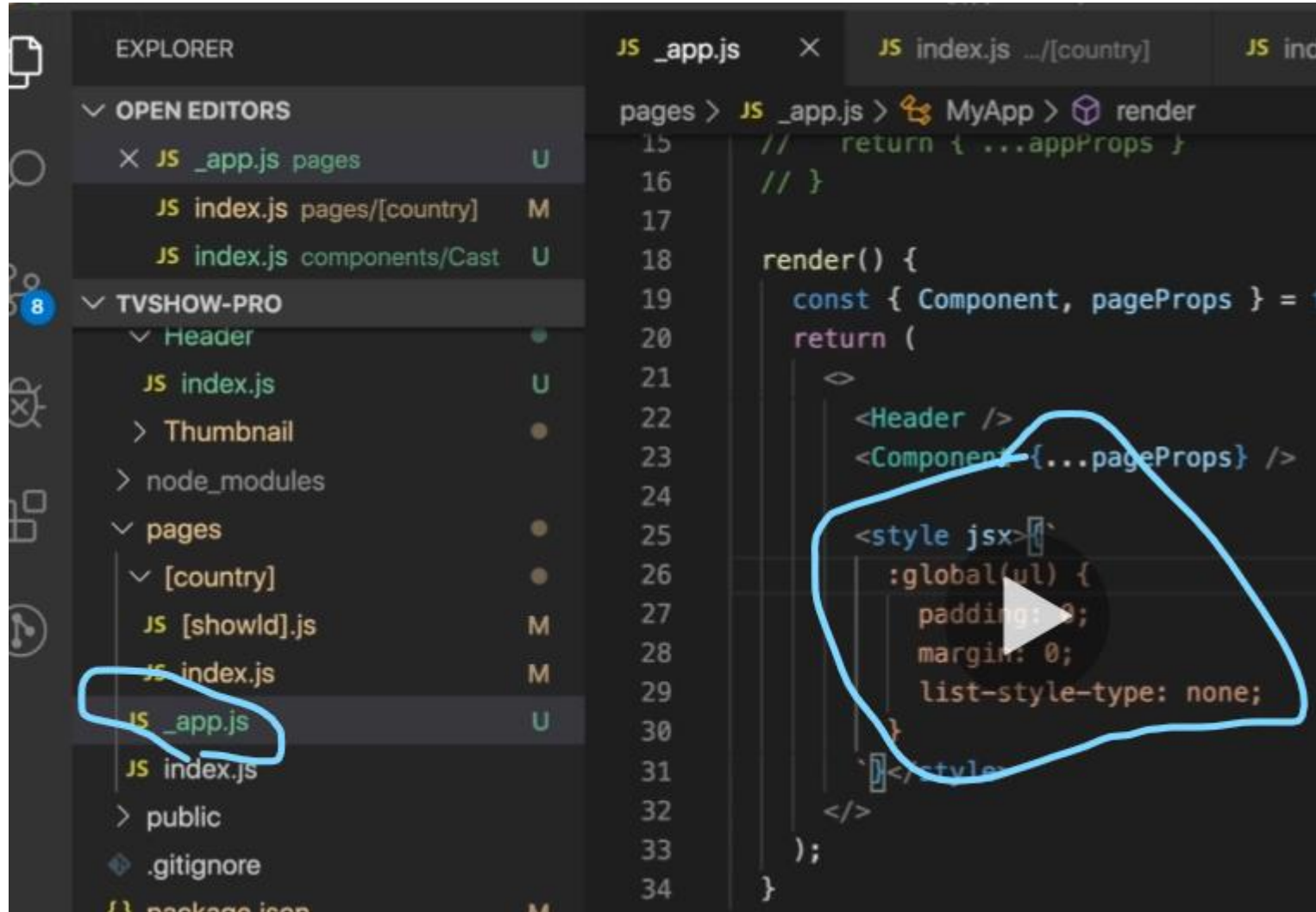
JS index.js JS \_app.js

pages > JS \_app.js > MyApp > render

```
1 import React from 'react' 8.4K (gzipped: 3.4K)
2 import App from 'next/app' 54.7K (gzipped: 19.7K)
3 import Header from '../components/Header';
4
5 class MyApp extends App {
6   // Only uncomment this method if you have blocking
7   // every single page in your application. This disa
8   // perform automatic static optimization, causing e
9   // be server-side rendered.
10  //
11  // static async getInitialProps(appContext) {
12  //   // calls page's `getInitialProps` and fills `a
13  //   const appProps = await App.getInitialProps(app
14  //
15  //   return { ...appProps }
16  // }
17
18  render() {
19    const { Component, pageProps } = this.props
20    return (
21      <Header />
22      <Component {...pageProps} />
23    )
24  }
```

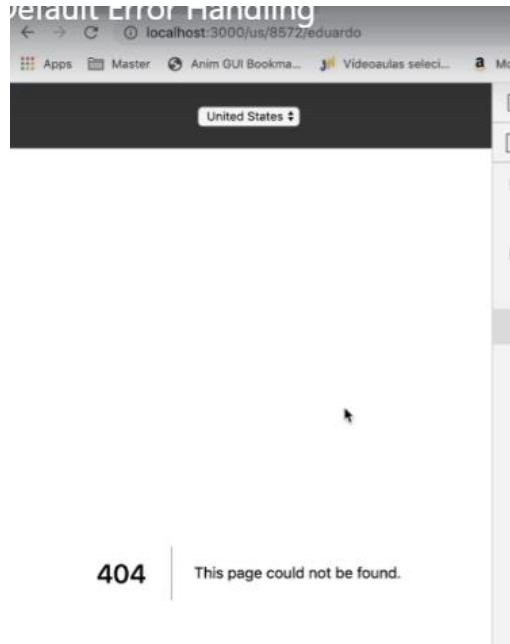
## Global Styles

Add style to \_app.js - use global

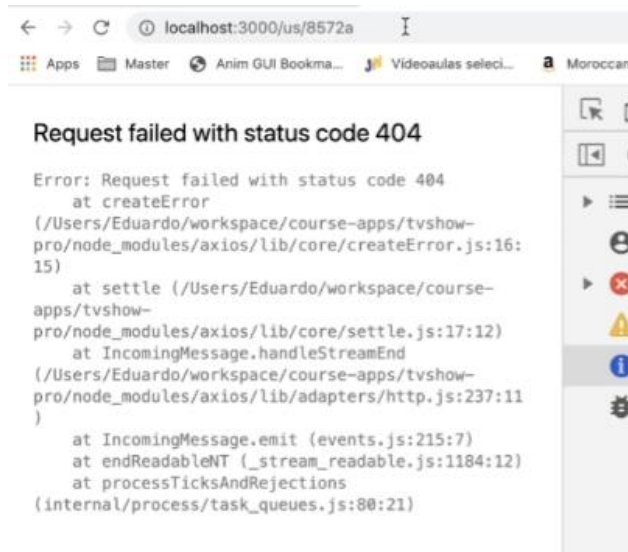


**ERROR Handling** => use Error from next/Error

Bad url → default error handling



## Error handling from a specific page (ex: showDetail page)



Good url but bad data



The api that we work with returns a 404 because show id 8572a does not exist

<pre>import Cast from '../components/cast'; import Error from 'next/error'; 5.8K (gzipped: 2.2K)  const ShowDetails = ({ show = {}, statusCode }) =&gt; {   const { name, image, summary, _embedded } = show;    if (statusCode) {     return &lt;Error statusCode={statusCode} title="Oops! There was a problem here" /&gt;;   } }</pre>	<div>404</div> <div>Oops! There was a problem here....</div>
---	--



```
ShowDetails.getInitialProps = async ({ query }) => {
  try {
    const { showId } = query;
    const response = await axios.get(
      `http://api.tvmaze.com/shows/${showId}?embed=cast`
    );
    return {
      show: response.data
    };
  } catch (error) {
    return {
      statusCode: error.response ? error.response.status : 500
    };
  }
};
```

## Custom Error handling

Create a file `_error.js` in `pages` directory

```
pages > JS _error.js > CustomError
1  const CustomError = ({ statusCode }) => {
2    console.log('TCL: CustomError -> statusCode', statusCode);
3    if (statusCode === 404)
4      return <h1>The resource was not found...</h1>
5    }
6
7    return <div>This is an error</div>;
8  };
9
10 CustomError.getInitialProps = ({ err, res }) => {
11   return { statusCode: res ? res.statusCode : err ? err.statusCode : 404 };
12 };
13
14 export default CustomError;
15
```



```
import CustomError from '../_error';

const ShowDetails = ({ show = {}, statusCode }) => {
  const { name, image, summary, _embedded } = show;

  if (statusCode) {
    return (
      <CustomError
        statusCode={statusCode}
        title="Oops! There was a problem here..."
      />
    );
  }
}
```

Handle error from the home page

## Custom Error handling

### Request failed with status code 422

```
Error: Request failed with status code 422
    at createError
    (/Users/Eduardo/workspace/course-apps/tvshow-
    pro/node_modules/axios/lib/core/createError.js:16:
    15)
    at settle (/Users/Eduardo/workspace/course-
    apps/tvshow-
    pro/node_modules/axios/lib/core/settle.js:17:12)
    at IncomingMessage.handleStreamEnd
    (/Users/Eduardo/workspace/course-apps/tvshow-
    pro/node_modules/axios/lib/adapters/http.js:237:11
    )
    at IncomingMessage.emit (events.js:215:7)
    at endReadableNT (_stream_readable.js:1184:12)
    at processTicksAndRejections
    (internal/process/task_queues.js:80:21)
```

Same that showDetail → we can use our CustomError or the default Error

```
import Error from 'next/error'; 5.8K (gzipped: 2.2K)
import Thumbnail from '../components/Thumbnail';

const Home = ({ shows, country, statusCode }) => {
  if (statusCode) {
    return <Error statusCode={statusCode} />
  }

  const renderShows = () => {
```

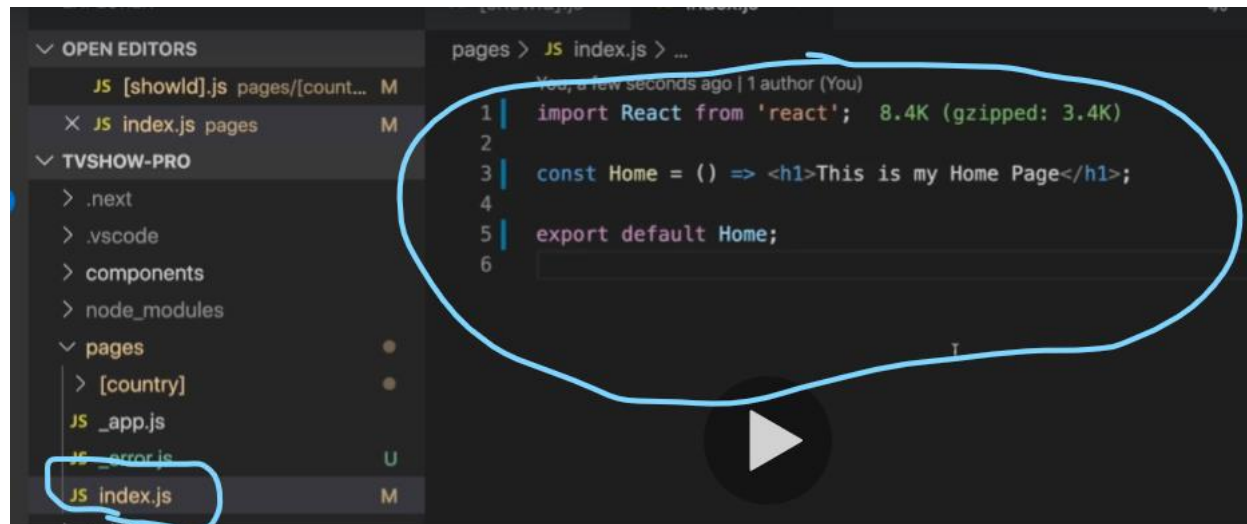
```
Home.getInitialProps = async context => {
  try {
    const country = context.query.country || 'us';

    const response = await axios.get(
      `http://api.tvmaze.com/schedule?country=${country}&date=2014-12-01`
    );

    return {
      shows: response.data,
      country
    };
  } catch(error) {
    return {
      statusCode: error.response ? error.response.status : 500
    };
  }
};
```

## Redirect to the default page

`browser.process` tells us if the component was rendered in the browser side (true) or server side (false)



In our case we want redirect the first time to the default language ( server side )

```
howld].js  JS index.js  ●  ↺  ↻  🔍

s > JS index.js > 🏠 Home.getInitialProps
Unsaved changes (cannot determine recent change or authors)
import Router from 'next/router'; 46.7K (gzipped: 16.8K)

const Home = () => <h1>This is my Home Page {process.browser}</h1>;

Home.getInitialProps = (context) => {
  const country = context.query.country || 'us';

  process.browser ? CLIENT SIDE
    Router.replace('/[country]', `${country}`) :
    context.res.writeHead(302, { Location: `${country}`});
  SERVER SIDE
  context.res.end();
};

export default Home;
```

## Nookies => Cookies for Next.js



### Write cooki

```
useEffect(() => {  
  cookies.set(null, 'defaultCountry', selectedCountry, {  
    maxAge: 30 * 24 * 60 * 60,  
    path: '/'  
  })  
}, [selectedCountry])
```

### Read cookie

```

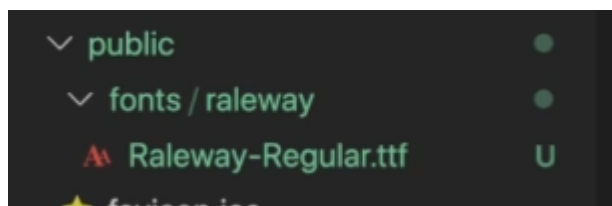
3
4  const Home = () => null;
5
6  Home.getInitialProps = context => {
7    const { defaultCountry } = cookies.get(context);
8    const country = context.query.country || defaultCountry || 'us';
9
10     process.browser
11       ? Router.replace('/[country]', `${country}`)
12       : context.res.writeHead(302, { Location: `/${country}` });
13
14     context.res.end();
15   };
16
17   export default Home;
18

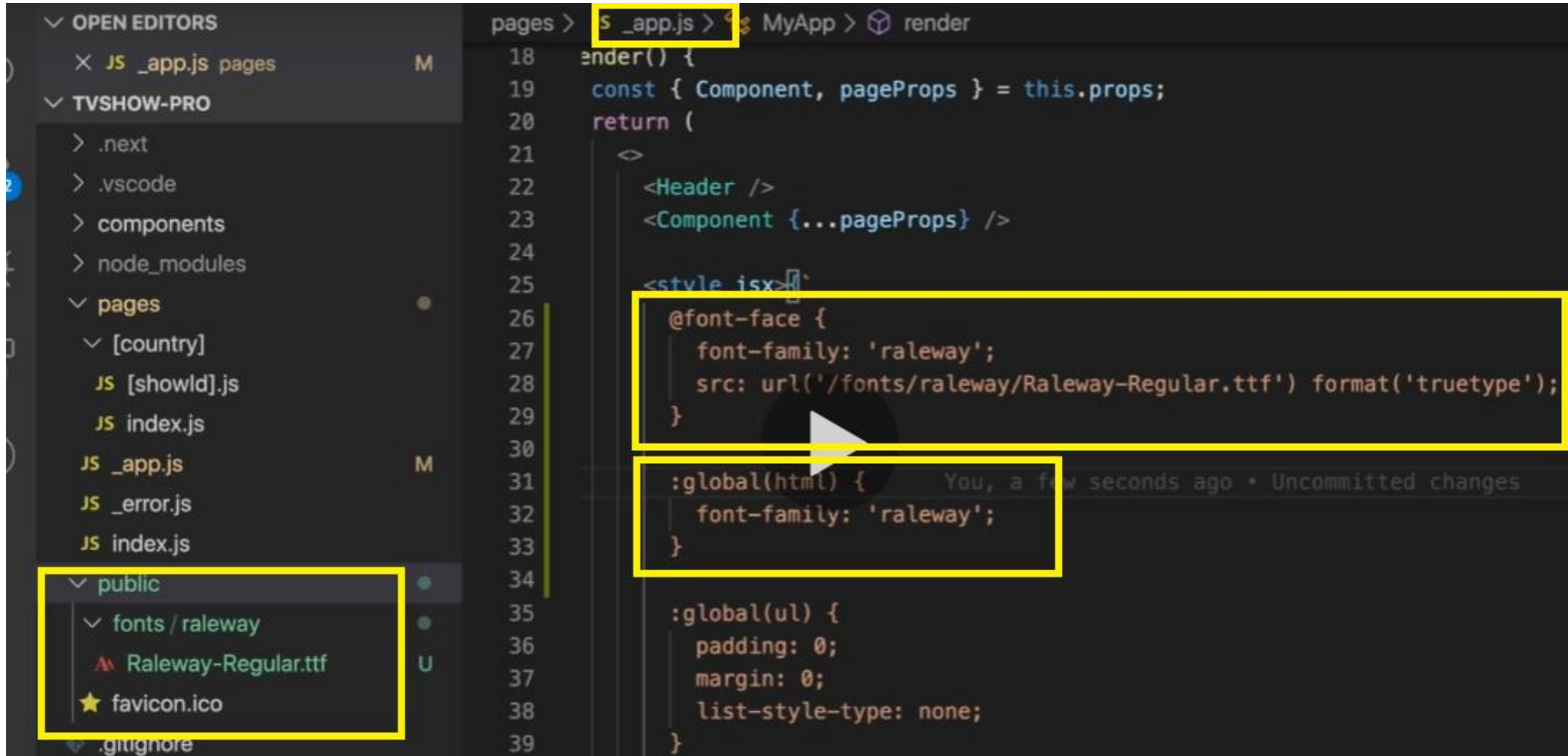
```

## Static file serving

Under public folder

Ex: Font







## Authentication Example

