NextJS tutorial

**This tutorial is based on the React tutorial. Please visit https://matthiasbrat.dev/js/react first.*

Introduction

Lexicon

NAME	DESCRIPTION	NOTE	USAGE
CSR	Client-side rendering	Take longer to reach first contentful paint.	
SSG	Server-side generation (static generation)	Render all pages at build time.	Blogs, or apps where data doesn't change often.
SSR	Server-side rendering	Built the html page each time it's requested by the user.	Apps where data change often.
ISR	Incremental static regeneration	Re-generate single pages in the background.	

Files tree structure

NAME	DESCRIPTION	NOTE
PAGES/	All the pages and routes for the application	
PAGES/_app.js	Main entry point into the app.	
PAGES/index.js	Root component of your project.	
STYLES/	Contains your CSS files.	Next support CSS modules.
API/	Server-only routes	For setting up routes that will only apply to the server. Won't increase the JS client-side bundle. **Use it to do stuff on the backend.**

Routes & Pages

Each React component has its own route, which is the component name.

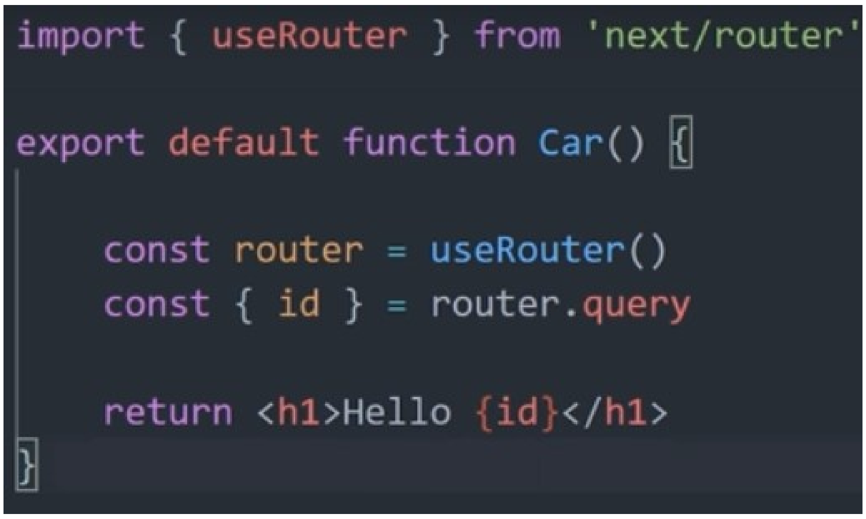
For example:

- The component pages/cars/Cars.js can be accessed at route localhost:3000/cars/cars.

Dynamic components

The component pages/cars/[id].js can be accessed at route localhost:3000/cars/tesla.

We need to use the hook useRouter to access the query parameter from the URL.



```
import { useRouter } from 'next/router'

export default function Car() {

    const router = useRouter()
    const { id } = router.query

    return <h1>Hello {id}</h1>
}
```

Server-side data fetching

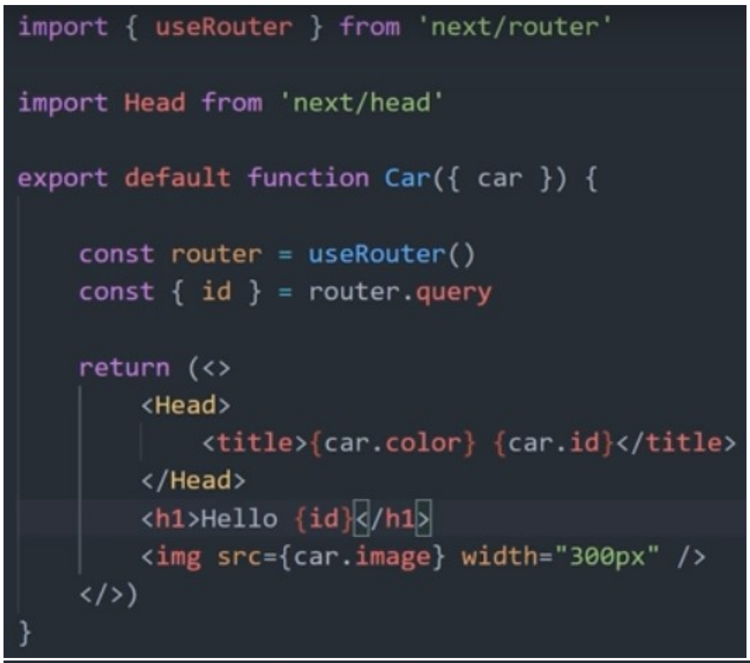
Next allow us to fetch data and render html in the server.
The user get rendered content quicker.

SSG

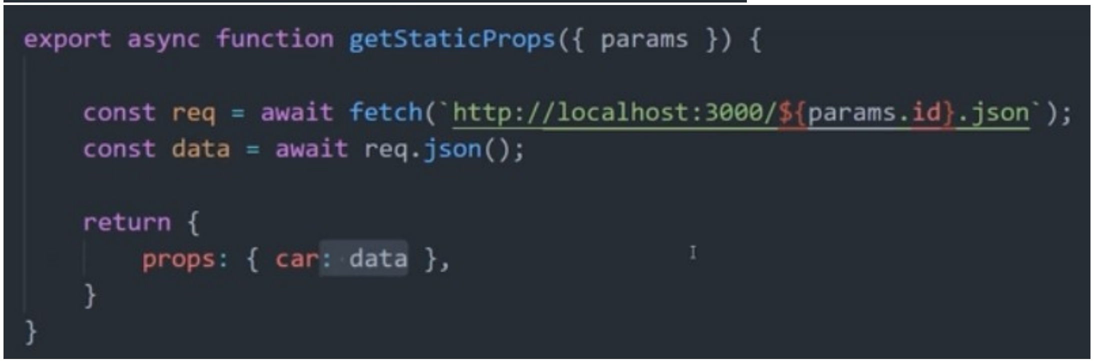
Render all pages at **built time**.

Each component can implement the function getStaticProps.

1. Fetch data.
2. Return props.
3. Use props.



```
import { useRouter } from 'next/router'

import Head from 'next/head'

export default function Car({ car }) {

    const router = useRouter()
    const { id } = router.query

    return (<>
        <Head>
            <title>{car.color} {car.id}</title>
        </Head>
        <h1>Hello {id}</h1>
        <img src={car.image} width="300px" />
    </>)
}
```



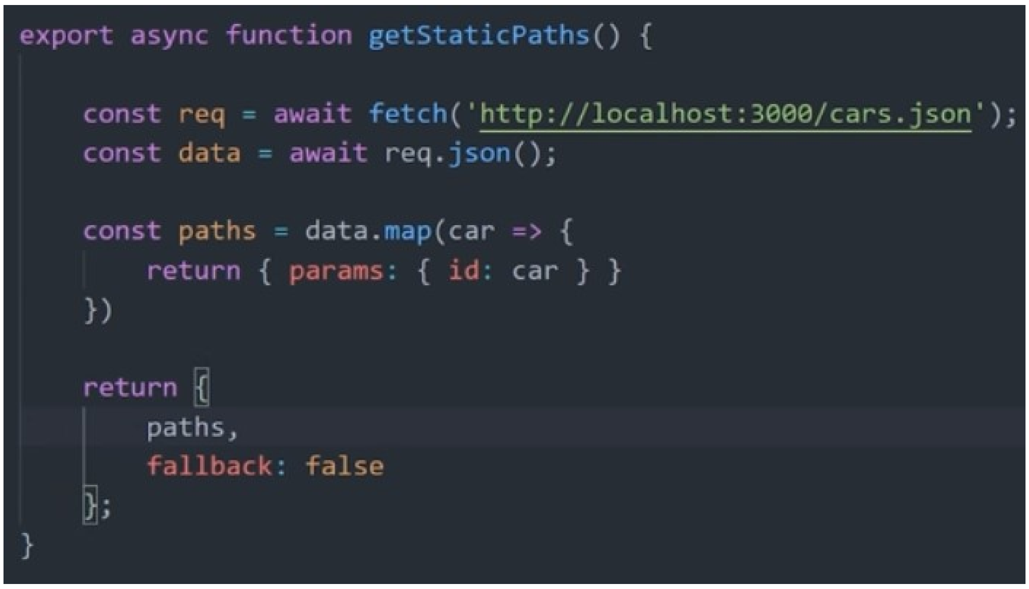
```
export async function getStaticProps({ params }) {

    const req = await fetch(`http://localhost:3000/${params.id}.json`);
    const data = await req.json();

    return {
        props: { car: data },
    }
}
```

CONS

- Your data may become stale.
- Prerendering a million pages is slow.

When working with dynamic routes, we must specify which dynamic page we want to render, in order to prerender all of the car ids, next needs to know these ids in advance.

The getStaticPaths function, pre-renders all the paths specified.



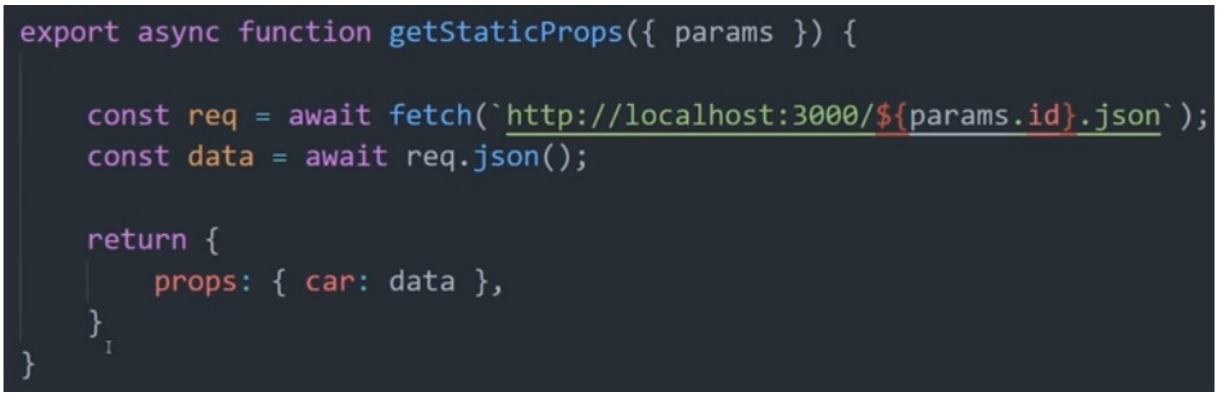
```
export async function getStaticPaths() {

    const req = await fetch('http://localhost:3000/cars.json');
    const data = await req.json();

    const paths = data.map(car => {
        return { params: { id: car } }
    })

    return {
        paths,
        fallback: false
    };
}
```

SSR

Built the html page each time it’s requested by the user.

Renders the content in advance on the server, so the first thing that the users see is the fully rendered html.
After the loading of the initial page, CSR takes over and works just like a traditional react app.

Each component can implement the function getServerSideProps.
Instead of running at built time this function runs at **request time**.



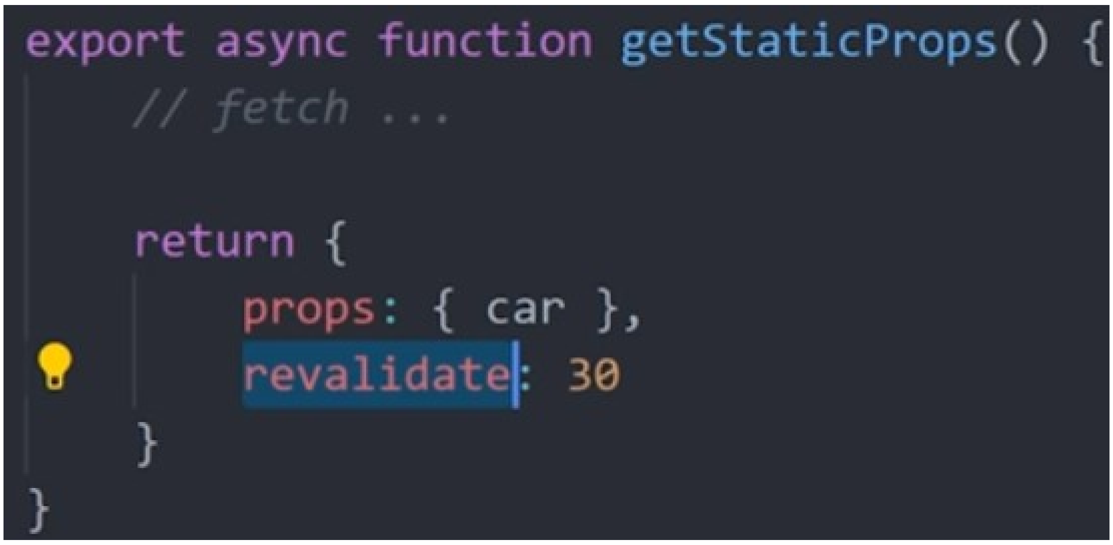
```
export async function getStaticProps({ params }) {

    const req = await fetch(`http://localhost:3000/${params.id}.json`);
    const data = await req.json();

    return {
        props: { car: data },
    }
}
```

CONS

- Slower, the server must load the data first.
- Less data caching.

ISR

Regenerate a page whenever a new request comes in within a certain **time interval.** Here, we rebuild the app at most every 30 seconds.



```
export async function getStaticProps() {
    // fetch ...

    return {
        props: { car },
        revalidate: 30
    }
}
```

Commands

Create

```
$> npx create-next-app
```

Start

```
$> npm run dev
```

Made by :

Matthias Brat

Socials:

github.com/matthiasbrat

stackoverflow.com/users/17921879/matthias-b

matthiasbrat.dev