Introduction to NextJs

Course Code: CSC 4182 Course Title: Advanced Programming In Web Technologies

Dept. of Computer Science
Faculty of Science and Technology

Lecture No:	**1**	**Week No:**	**09**	**Semester:**	
Lecturer:	*Sazzad Hossain; sazzad@aiub.edu*				

Lecture Outline

- ✓ What is NextJS
- ✓ NextJS Features
- ✓ File-based Routing
- ✓ Dynamic Routes
- ✓ 404 file
- ✓ What is rendering?
- ✓ Types of Rendering
- ✓ Client-Side Rendering(CSR)
- ✓ Pre-rendering
- ✓ Server-side Rendering
- ✓ Static Site Generation (SSG)
- ✓ Static Site Generation (SSG) VS Server-Side Rendering (SSR)

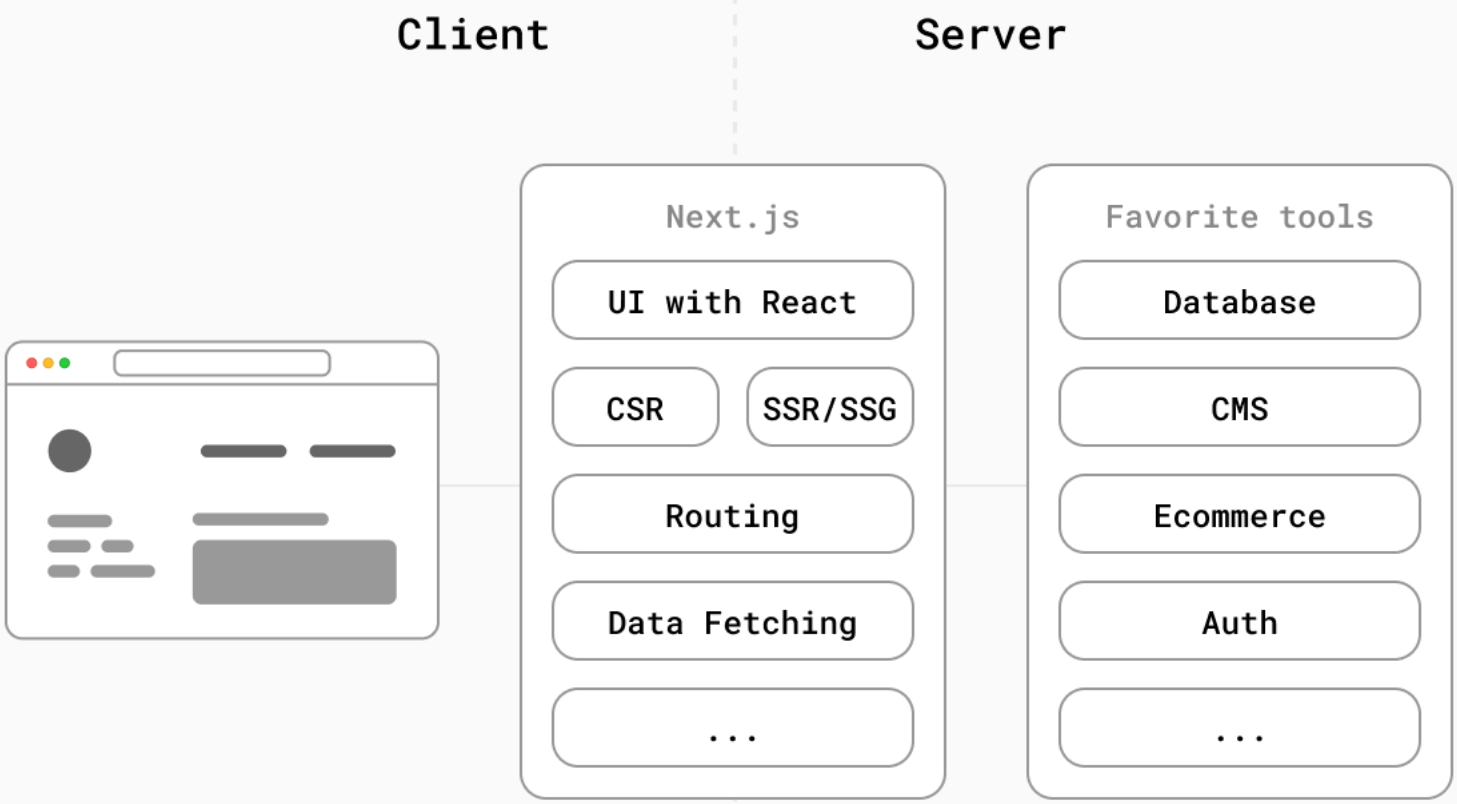
What is NextJS

Next.js is a flexible React framework that gives you building blocks to create fast web applications.

Next.js is a **React framework** designed to facilitate the development of web applications

- Provides building blocks and tools for creating web apps
- Handles tooling and configuration needed for React
- Offers additional structure, features, and optimizations

What is NextJS



Building Blocks of a Web Application

There are a few things you need to consider when building modern applications. Such as:

- **User Interface** - how users will consume and interact with your application.
- **Routing** - how users navigate between different parts of your application.
- **Data Fetching** - where your data lives and how to get it.
- **Rendering** - when and where you render static or dynamic content.
- **Integrations** - what third-party services you use (CMS, auth, payments, etc) and how you connect to them.
- **Infrastructure** - where you deploy, store, and run your application code (Serverless, CDN, Edge, etc).
- **Performance** - how to optimize your application for end-users.
- **Scalability** - how your application adapts as your team, data, and traffic grow.
- **Developer Experience** - your team's experience building and maintaining your application.

NextJS Features

- In the development stage, Next.js optimizes for the developer and their experience building the application. It comes with features that aim to improve the **Developer Experience** such the TypeScript and ESLint integration, Fast Refresh, and more.

- Since each environment has different considerations and goals, there is a lot that needs to be done to move an application from development to production. For instance, the application code needs to be compiled, bundled, minified, and code split.

NextJS Features

- File based routing and dynamic routing
- Pre-rendering
  - Sever-side rendering
  - Static generation

File-based Routing

File-based routing involves creating a separate file for each page or route in the application. For example, if your application has a homepage and a contact page, you would create two separate files: "HomePage.js" and "ContactPage.js". Each file would contain the code for the corresponding page, and the **routing is handled by importing and rendering the appropriate file based on the URL.**

File-based Routing

Homepage.js

```
import React from "react";

function HomePage() {
  return (
    <div>
      <h1>Welcome to my website!</h1>
      <p>This is the home page.</p>
    </div>
  );
}

export default HomePage;
```

Dynamic Routes

dynamic routes allow you to create pages with **dynamic content based on a parameter in the URL**.

For example, you could create a blog website with dynamic routes to display individual blog posts based on the post ID in the URL.

Create a dynamic page file In your **pages** directory, create a new file called **[postId].js**. The square brackets around "**postId**" indicate that it is a dynamic parameter that can be any value. This file will handle the dynamic routing and display the individual blog post based on the **postId** in the **URL**.

Dynamic Routes

```
import { useRouter } from "next/router";

function BlogPost() {
  const router = useRouter();
  const { postId } = router.query;

  // fetch blog post data using the postId

  return (
    <div>
      {/* display the blog post content */}
    </div>
  );
}

export default BlogPost;
```

The **useRouter** hook allows you to access the current route and its parameters. In this case, we access the postId parameter using router.query.postId. We can then use this ID to fetch the corresponding blog post content and display it on the page.

404 file

- React hooks are functions introduced in React 16.8 that allow functional components to have **state** and use **lifecycle methods** without the need for writing class components.

Hooks provide a Next.js 13, the 404 file is a special page that is displayed when a user tries to access a page that doesn't exist. By default, Next.js will show a generic 404 error page when this happens. However, you can customize the 404 page by creating a file called 404.js in your pages directory.

Here's an example of how to create a custom 404 page in Next.js 13:

- Create a 404.js file In your pages directory, create a new file called 404.js.

- Write the code for your custom 404 page In the 404.js file, you can write the code for your custom 404 page. This could be a simple message or a more complex page with links to other parts of your site. Here's an example:
- more **straightforward and reusable way to manage state and perform side effects** in functional components.

404 file

```
function Custom404() {
    return <h1>404 - Page Not Found</h1>;
}

export default Custom404;
```

Test your custom 404 page

To test your custom 404 page, try accessing a page on your site that doesn't exist. Next.js should display your custom 404 page instead of the default one.

What is rendering?

Rendering is the process that **turns the code** you write **an application** in into something that users can interact with on a web page.

Types of Rendering

Pre-rendering

- Static Site Generation (SSG)
- Server Side Rendering (SSR)

No Pre-rendering

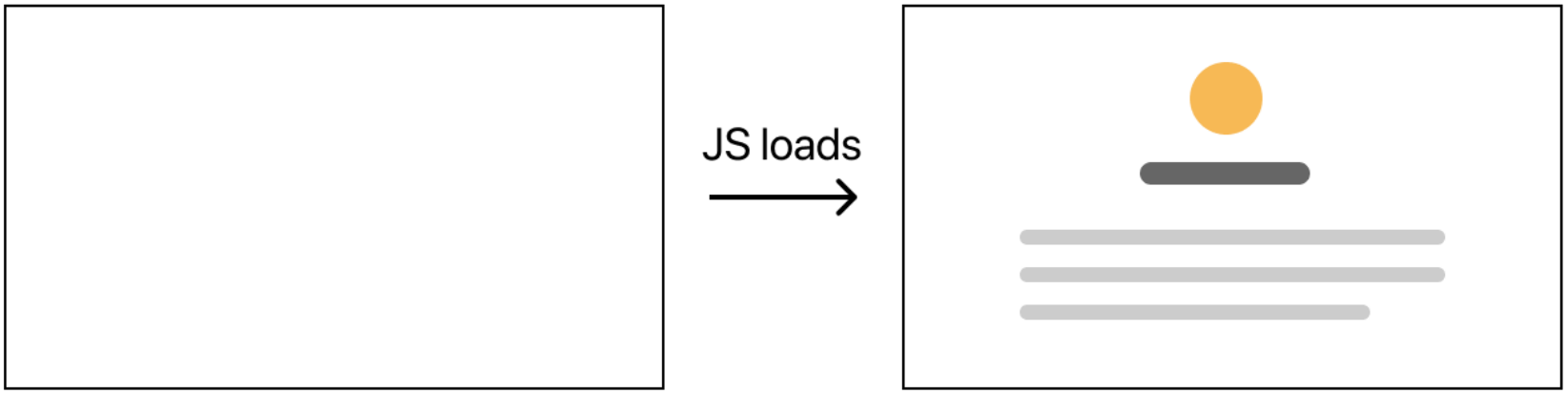
- Client-Side Rendering(CSR)

Client-Side Rendering(CSR)

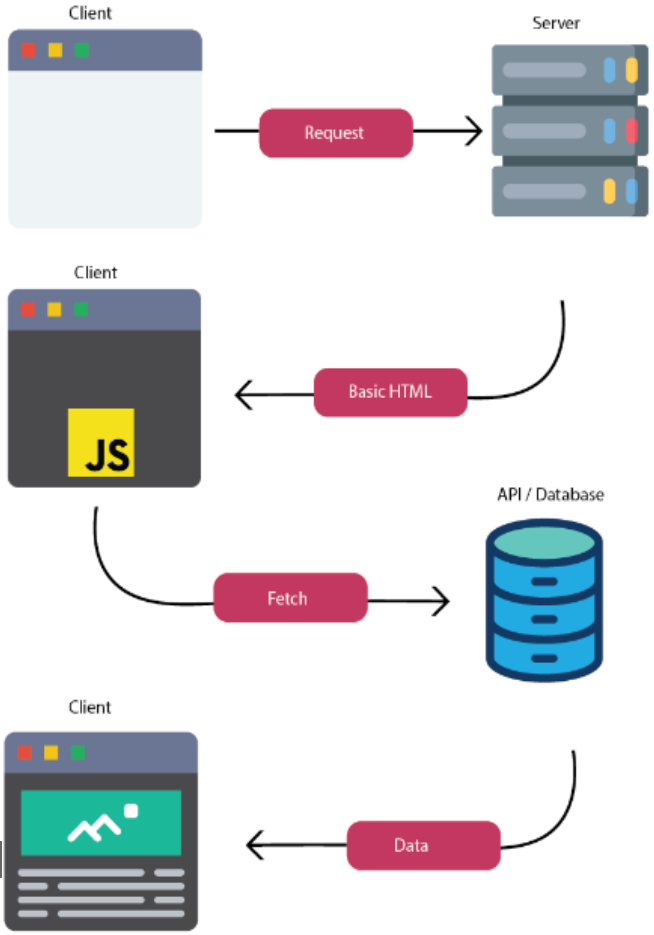
No Pre-rendering (Plain React.js app)

Initial Load:
App is not rendered

Hydration: React components are initialized and App becomes interactive



Client-Side Rendering(CSR)



Pre-rendering

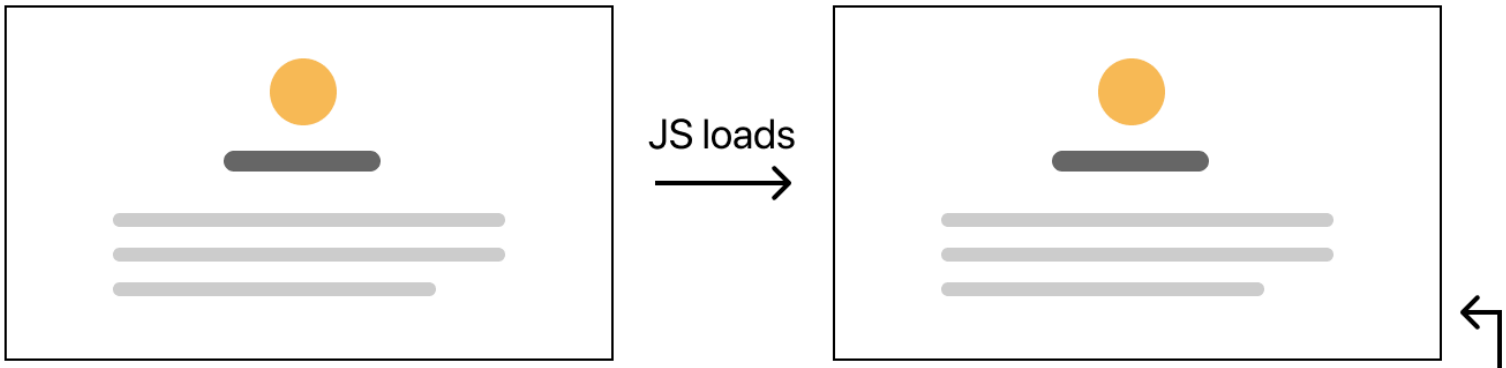
- By default, Next.js pre-renders every page.
- Next.js *generates HTML for each page in advance*, instead of having it all done by client-side JavaScript.
- Pre-rendering can result in better performance and SEO.
- Each generated HTML is associated with minimal JavaScript code necessary for that page.
- When a page is loaded by the browser, its JavaScript code runs and makes the page fully interactive. (This process is called **hydration**.)

Pre-rendering

Pre-rendering (Using Next.js)

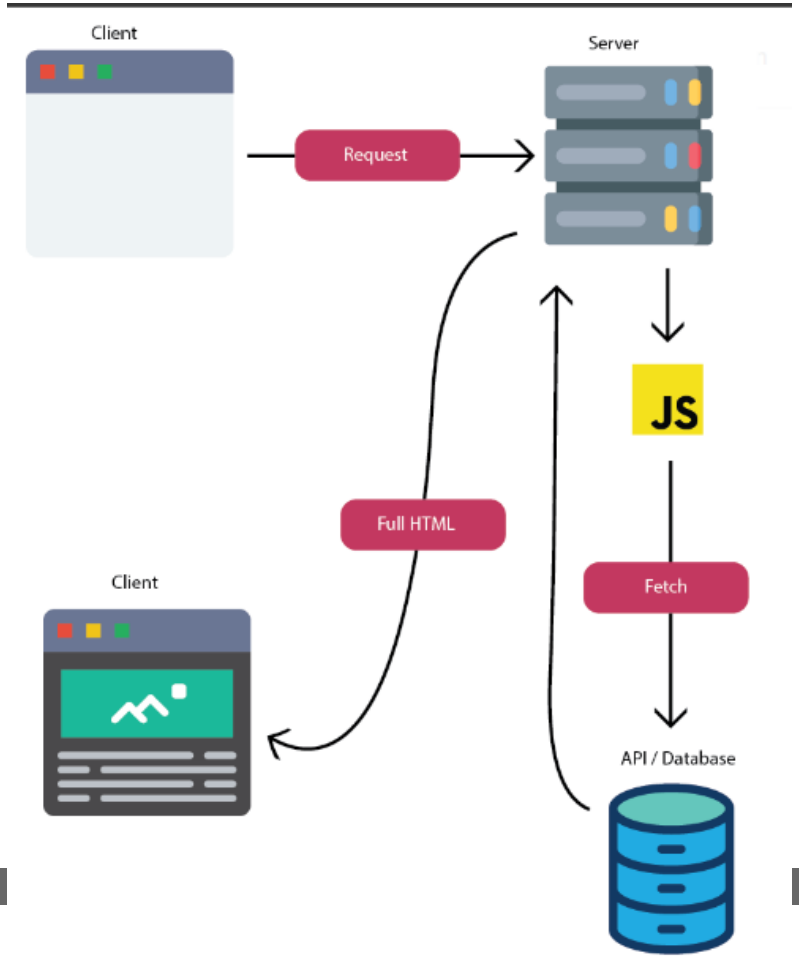
Initial Load:
Pre-rendered HTML is displayed

Hydration: React components are initialized and App becomes interactive



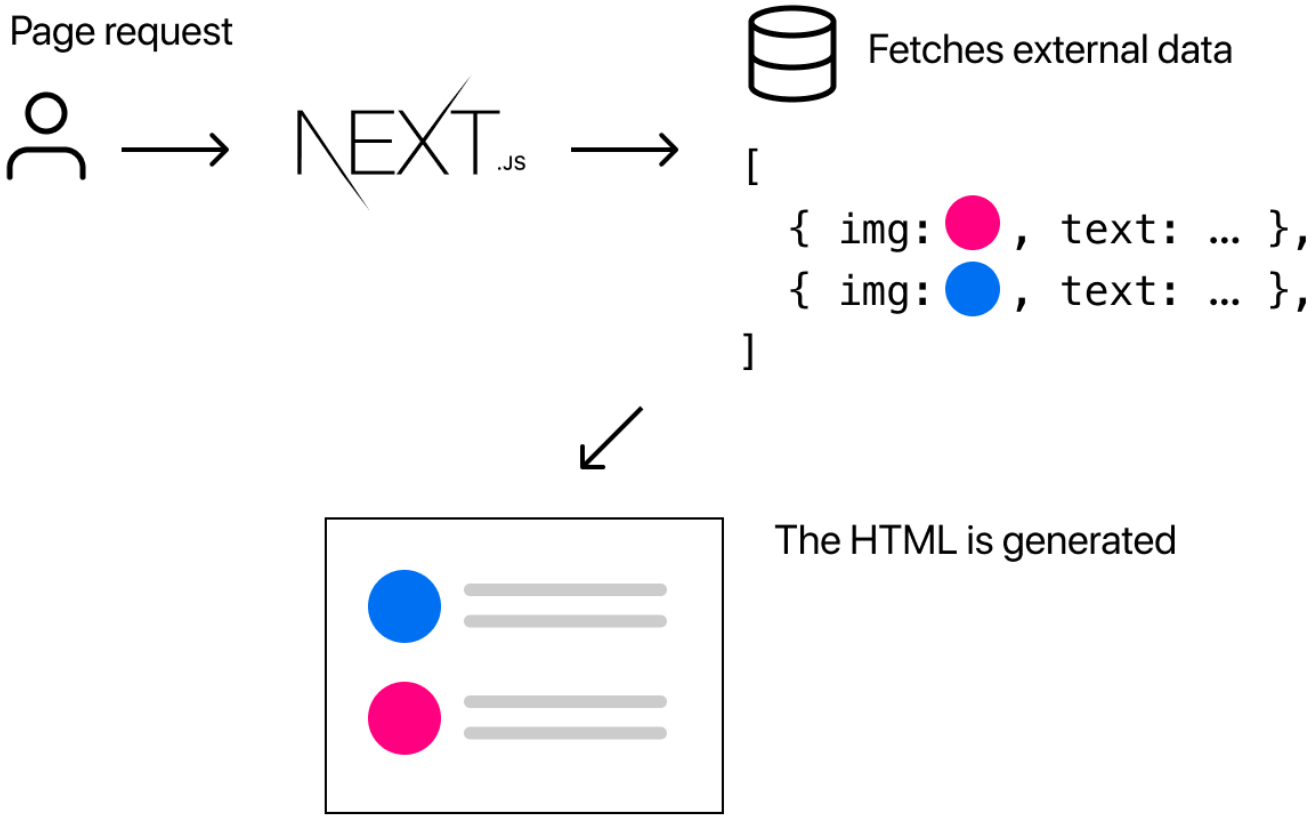
If your app has interactive components like `<Link />`, they'll be active after JS loads

Pre-rendering



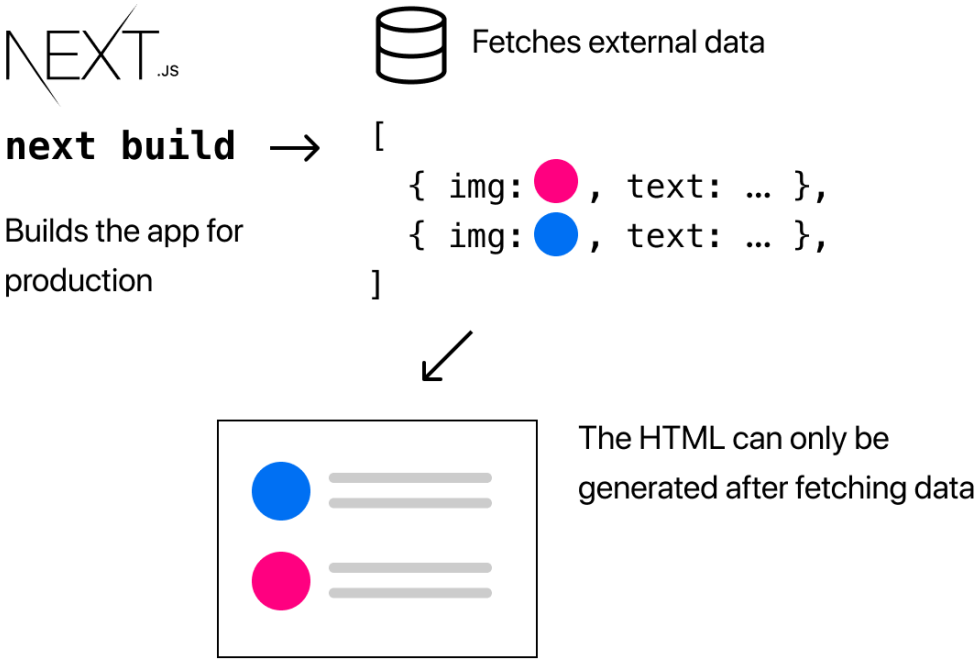
Server-side Rendering

Server-side Rendering is the pre-rendering method **that generates the HTML on each request.**

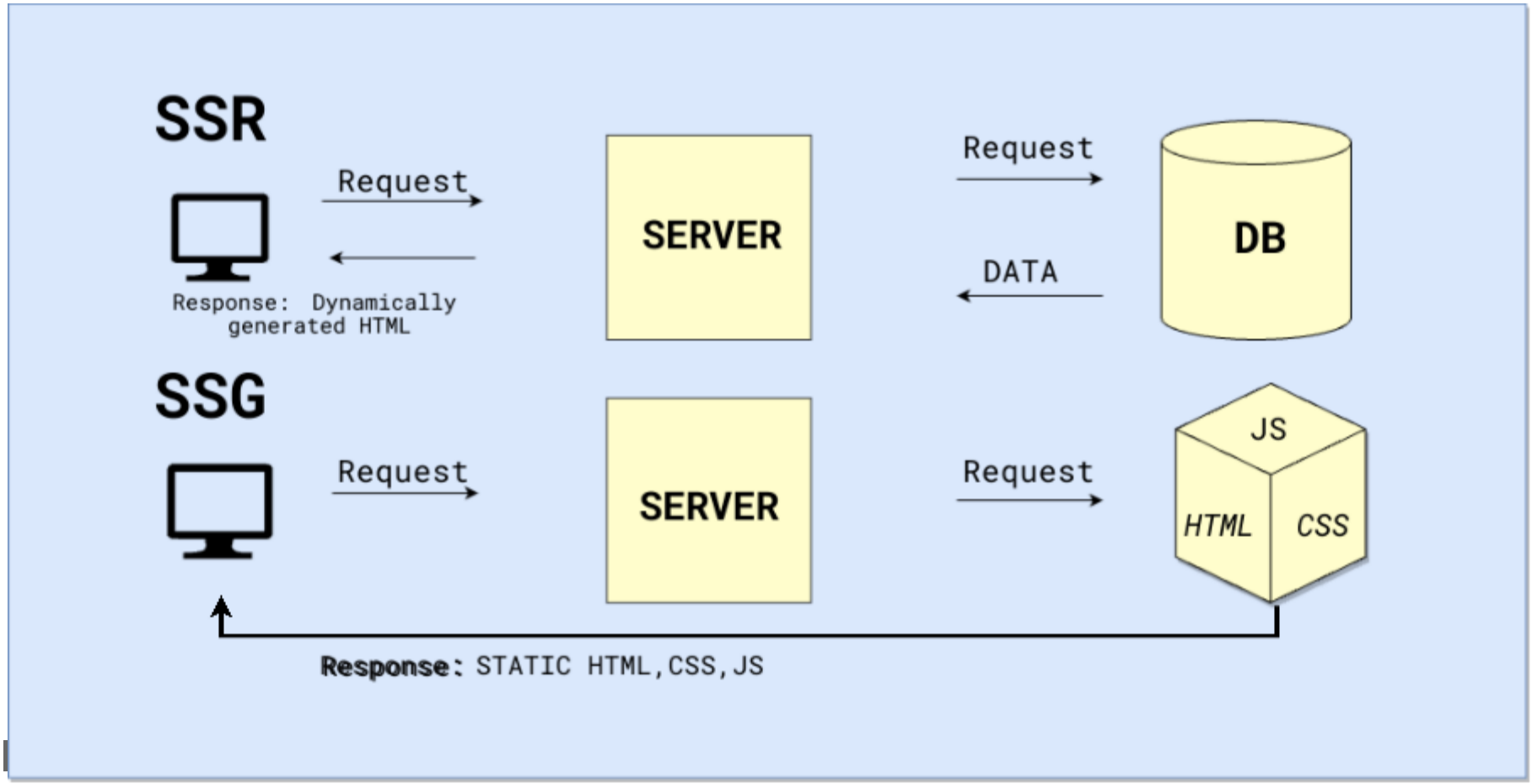


Static Site Generation (SSG)

- Static Generation is the pre-rendering method that generates the HTML at build time.
- The pre-rendered HTML is then reused on each request.



Static Site Generation (SSG) VS Server-Side Rendering (SSR)



References

1. **W3Schools Online Web Tutorials, URL: http://www.w3schools.com**
2. **Next.js, URL:** https://nextjs.org/
3. **Mozilla Developer Networks, URL:** https://developer.mozilla.org/


AMERICAN INTERNATIONAL UNIVERSITY
PRAESIDIUM
DISCIPLINA
CIVITATIS
1994
BANGLADESH

Thank You!