

# Fullstack application with Next.js, tRPC and DrizzleORM

# Agenda

- Setup
- The Stack
- Single Page Applications
- Important Concepts
- React
- Next.js
- tRPC
- DrizzleORM
- Practical Work

# Please start building the docker container now :)

It may take some time

1. Clone our GitHub repo: `git clone https://github.com/sebivenlo/ESD-2023-Full-Stack`
2. cd into the cloned repo: `cd ESD-2023-Full-Stack`
3. Run `docker-compose up -d`
4. Meanwhile please open a new VSCode window and search for the extension called `Dev Containers`
5. CMD/CTRL + SHIFT + P -> 'Developer reload windows'
6. CMD/CTRL + SHIFT + P -> 'Dev Containers: Attach to Running Container...'
7. Choose Path `/usr/src/app`

# The STACK

- Next.js
- tRPC
- DrizzleORM

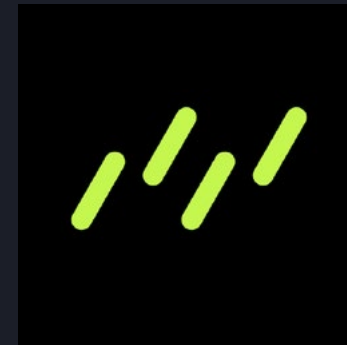


Image source(s): <https://nextjs.org/>, <https://trpc.io/>, <https://orm.drizzle.team/>

# Single Page Application

- Only loads the document once and then updates it dynamically without loading new pages
- Has to download all the html, CSS and JS in the beginning
- High initial load time
- Can be laggy when the app becomes more complex

# Important concepts

## Client-Side

- Rendering happens in the user's browser
- Requires more client-based processing power
- Search engines does not see full content -> not perfect for SEO

## Server-Side

- Rendering occurs on the server before page is sent to the user
- Reduces client-side processing
- Better for SEO, because search engines can crawl the full content

# React

- Built by Facebook
- One of the biggest web frameworks (~ 42% market share <sup>[2]</sup>)
- Open-source frontend JavaScript framework for building interactive user interfaces out of components
- Large ecosystem

```
function MyOwnComponent() {  
  return (  
    <>  
      <h1 style={{ color: "green" }}>Hello World!</h1>  
      <button onClick={() => console.log("Button clicked")}>  
        Increase count  
      </button>  
    </>  
  );  
}
```



```
type MyOwnComponentProps = {
  color: "red" | "blue" | "green";
};

function MyOwnComponent({ color }: MyOwnComponentProps) {
  return (
    <>
      <h1 style={{ color: color }}>Hello World!</h1>
      <button onClick={() => console.log("Button clicked")}>
        Increase count
      </button>
    </>
  );
}
```

```

type MyOwnComponentProps = {
  color: "red" | "blue" | "green";
};

function MyOwnComponent({ color }: MyOwnComponentProps) {
  return (
    <>
      <h1 style={{ color: color }}>Hello World!</h1>
      <button onClick={() => console.log("Button clicked")}>
        Increase count
      </button>
    </>
  );
}

function MainPage() {
  return (
    <>
      <MyOwnComponent color="red" />;
    </>
  );
}

```

```
import { useState } from "react";

function MyOwnComponent() {
  const [count, setCount] = useState(0);

  return (
    <>
      <h1>Current count {count}</h1>
      <button onClick={() => setCount(count + 1)}>
        Increase count
      </button>
    </>
  );
}
```

```
import { useEffect, useState } from "react";

function MyOwnComponent() {
  const [count, setCount] = useState(0);

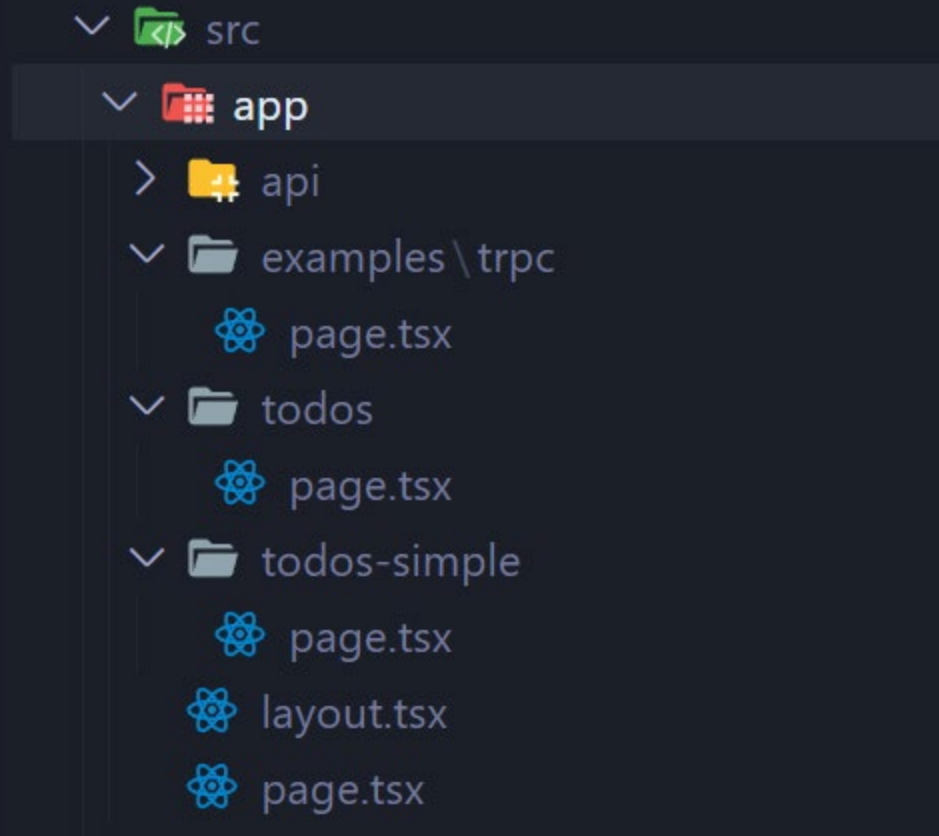
  useEffect(() => {
    console.log("Count has been increased");
  }, [count]);

  return (
    <>
      <h1>Current count {count}</h1>
      <button onClick={() => setCount(count + 1)}>
        Increase count
      </button>
    </>
  );
}
```

# Next.JS

- Built on top of react
- Enables easy Client- and Server-side Rendering (SSR)
- Build in routing (app router)

# App router



# Next.JS

- Built on top of react
- Enables easy Client- and Server-side Rendering (SSR)
- Build in router
- Dynamic HTML streaming
- Built in optimizations
- Can run on the edge

# tRPC

- Ensures type safety
- Automatically generate TypeScript types
- Automatic Serialization/Deserialization
- React integration (Hooks for the API using Tanstack Query)



# DrizzleORM

- SQL-like query APIs
- Lightweight and Performant
- Type Safe
- Flexible and Serverless ready
- Support for multiple databases

```
import { boolean, pgTable, text, timestamp, uuid } from "drizzle-orm/pg-core";

export const todos = pgTable("todo", {
  id: uuid("id").primaryKey().notNull().defaultRandom(),
  description: text("description"),
  isComplete: boolean("is_complete").notNull().default(false),
  createdAt: timestamp("created_at").defaultNow().notNull(),
  updatedAt: timestamp("updated_at").defaultNow().notNull(),
});
```

```
const todosResult = await ctx.db
  .select()
  .from(todos)
  .orderBy(desc(todos.createdAt))
  .execute();
```

*// TypeScript Inferred Type*

```
const todosResult: {
  id: string;
  description: string | null;
  isComplete: boolean;
  createdAt: Date;
  updatedAt: Date;
}[]
```

# The STACK

- Next.js
- tRPC
- DrizzleORM

NEXT.JS

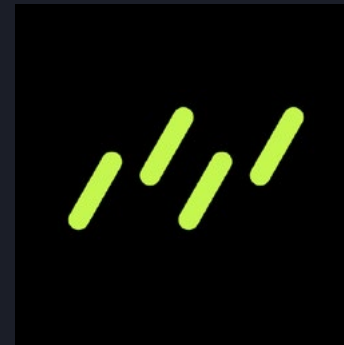


Image source(s): <https://nextjs.org/>, <https://trpc.io/>, <https://orm.drizzle.team/>

# Differences between JavaScript Frameworks

Framework	Next.js	Angular	Vue
Language	Javascript/Typescript, JSX/TSX	Typescript	Javascript (Can use Typescript)
Features	Hot Reload; Automatic Code Splitting	Powerful CLI	
Rendering	Server-Side Rendering, Static-Site Generation	Normally Client-Side Rendering	Client-Side Rendering
Community	React created by Facebook – Next.js backed by Vercel	By Google	Vue Foundation

# Practical uses cases

- If Server-Side Rendering is required (SEO)
- Faster initial page load
- Next.js is gaining popularity
- Companies that use NEXT.JS

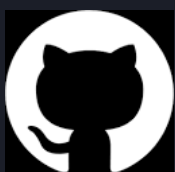


Image source(s): <https://www.netflix.com/>, <https://binance.com/>, <https://github.com/>, <https://www.airbnb.de/>

# Practical work – TODO App

# Practical work – TODO App

1. Basic project setup and structure
2. Creating a react component
3. Routing in Nextjs
4. Using tRPC and drizzle to create a backend API
5. Implement the web UI



# Sources

[1]

[2] <https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/>

[3] <https://nextjs.org/showcase>

[4] <https://medium.com/@devsenior2000/stop-using-next-js-on-production-12cc60201525#:~:text=One%20of%20the%20biggest%20drawbacks,render%20the%20pages%20from%20scratch>