🚀 React Class Component & `super(props)` Full Guide

📘 What Is `super(props)` and Why It Matters?

🧠 In JavaScript Classes:

- `super()` refers to the **parent class's constructor**.
- In React, your class component extends from `React.Component`, so `super()` refers to `React.Component`.

🧠 Why Use `super(props)`?

☑️ To **correctly initialize** `this.props` in the constructor of a class component.

🔎 Without `super(props)`:

```
constructor(props) {
  super(); // ❌ Missing props
  console.log(this.props); // ❌ undefined
}
```

☑️ With `super(props)`:

```
constructor(props) {
  super(props); // ☑️
  console.log(this.props); // ☑️ accessible
}
```

📢 Without it, you'll see errors like:

> ❌ **"Must call super constructor in derived class before accessing 'this'"**

📦 Real-World Example (Your Code)

1️⃣ `UserClass` — Class Component

```
// 📦 import React
import React from 'react';

// 🏫 Class Component
class UserClass extends React.Component {
```

```
    constructor(props) {
        super(props); // ☑ Required to access this.props inside constructor
    }

    render() {
        return (
            <div className="user-card">
                <h1>👤 User: {this.props.name}</h1>
                <h2>🧑 Name : Darshan Vasani</h2>
                <h2>🎂 Age : 22</h2>
                <h2>📍 Location : Surat, Gujarat</h2>
            </div>
        );
    }
}

export default UserClass;
```

---

2 User — Functional Component (⚠ Doesn't need super(props))

```
// 📦 import React
import React from 'react';

// 🔧 Function-based component
const User = ({ name }) => {
  return (
    <div className="user-card">
      <h1>👤 User: {name}</h1>
      <h2>🧑 Name : Darshan Vasani</h2>
      <h2>🎂 Age : 22</h2>
      <h2>📍 Location : Surat, Gujarat</h2>
    </div>
  );
}

export default User;
```

---

3 About — How You Use Both Components

```
import User from "./User";
import UserClass from "./UserClass";

const About = () => {
    return (
        <div className="about-page">
            <h1>ℹ About</h1>
            <p>This is the about page of our application.</p>
```

```
            {/* 🔧 Function Component */}
            <User name={"Darshan Vasani From Function"} />

            {/* 🏛 Class Component */}
            <UserClass name={"Darshan Vasani From Class"} />
        </div>
    );
};

export default About;
```

💡 Summary Table

🧩 Concept	🔍 Why Needed?
`super()`	Calls `React.Component` constructor (required in child class)
`super(props)`	Initializes `this.props`, lets you use `this.props` in constructor
Omit `props` ❌	`this.props` will be `undefined`, errors on usage
Functional Comp ☑️	Doesn't need `super()` or constructor

☑️ Best Practices

- Always use `super(props)` in a constructor of class-based components if you need `this.props`.
- In modern React, **functional components + hooks** are preferred over class components — but class components still show up in many legacy codebases.

🔚 Final Thought:

🎯 If you're learning React or working with legacy apps, **understanding `super(props)` is essential**. But for new projects, prefer **functional components with hooks** — they're simpler, cleaner, and more powerful!
