Package managers

Nodejs

- npm
    - default package installer
    - npm init -> creates package.json file
    - package-lock.json
- npx
- Yarn
    - Much better than npm in terms of performance
    - Provides better security
    - lock.json file for depedencies

Monolithic and micro services

While a monolithic application is a single unified unit, a microservices architecture breaks it down into a collection of smaller independent units. These units carry out

every application process as a separate service. So, all the services have their own logic and databases, as well as perform specific functions.

System Design

Systems design is the process of defining elements of a system like modules, architecture, components and their interfaces and data for a system based on the specified requirements.

Highest paid job

Raectjs

Creating react app and running

```
npx creact-react-app hello-world
npm start
```

React Comments

```
{
  /* <p>This is commened p</p> */
}
```

Adding bootstrap in to the app

Tutorail

```
npm install bootstrap
# OR
```

```
yarn add bootstrap
```

```
// Bootstrap CSS
import "bootstrap/dist/css/bootstrap.min.css";
// Bootstrap Bundle JS
import "bootstrap/dist/js/bootstrap.bundle.min";
```

React bootstrap or reactstrap

```
npm install react-bootstrap bootstrap
```

```
import { Button } from "react-bootstrap";
```

Separation of Concerns

- Different files for different components
- Each file has its own html (jsx), css, and js

Exports

Types of Exports in React

Default Export

- ```
  Only one default export per file
  ```
- ```
  Can be imported with any name
  ```

- Can be imported without curly braces
- Can be imported with or without curly braces

```
export default function Avatar() {
  return (
    <img
      src="https://picsum.photos/20/30"
      width={20}
      height={20}
      alt="Avatar"
    />
  );
}
```

```
import Avatar from "./Components/Avatar";
```

Named Export

- Can be imported with curly braces

```
export function Avatar2() {
  return (
    <>
      <img src="https://picsum.photos/300/200" alt="Avatar" />
      <p>{PI}</p>
    </>
  );
}
```

```
import { Avatar2 } from "./Components/Avatar";
```

Conditional Rendering

```
import "./App.css";
import Avatar, { Avatar2 } from "./Components/Avatar";

function App() {
  let isLogged = true;
  return <div className="App">{isLogged ? <Avatar /> : <Avatar2 />}</div>;
}

export default App;
```

Props

```
function Avatar(props) {
  const { url, alt } = props;
  const style = {
    margin: "10px",
  };
  return (
    <>
      <img src={url} alt={alt} style={style} />
    </>
  );
}

function App() {
  let isLogged = false;

  return (
    <div className="App">
      {isLogged ? (
        <Avatar
          url="https://cdn.pixabay.com/photo/2015/04/23/22/00/tree-736885__480.jpg"
          alt="Sun set image"
        />
      ) : (
        <Avatar
          url="https://cdn.pixabay.com/photo/2017/03/19/20/19/ball-
2157465__340.png"
          alt="Logo image"
        />
      )}
    </div>
  );
}
```

Default Props

In React, default props are used to provide default values for a component's props in case they are not specified by the parent component. This can be useful when you want to provide sensible default values for optional props, or when you want to avoid errors caused by missing props.

To define default props for a component, you can use the "defaultProps" property. Here's an example:

```
class MyComponent extends React.Component {
  static defaultProps = {
    color: "red",
    size: 10,
    disabled: false,
  };

  render() {
    return (
      <div>
        <p>Color: {this.props.color}</p>
        <p>Size: {this.props.size}</p>
        <p>Disabled: {this.props.disabled.toString()}</p>
      </div>
    );
  }
}
```

In this example, we define default props for the "color", "size", and "disabled" props. If these props are not specified by the parent component, the default values will be used instead.

If a prop is specified by the parent component, its value will override the default value. For example:

```
<MyComponent size={20} />
```

In this case, the "size" prop is specified with a value of 20, so the default value of 10 is ignored.

Default props can also be defined for functional components using the "defaultProps" property:

```
function MyFunctionalComponent(props) {
  return (
```

```
    <div>
      <p>Color: {props.color}</p>
      <p>Size: {props.size}</p>
      <p>Disabled: {props.disabled.toString()}</p>
    </div>
  );
}

MyFunctionalComponent.defaultProps = {
  color: "red",
  size: 10,
  disabled: false,
};
```

In this example, we define default props for a functional component called "MyFunctionalComponent". These default props will be used if the parent component does not specify a value for a particular prop.