Props in React

- **What is Props?**
  - In React, props (short for **properties**) are a way to pass data from a parent
  - component to its child components, **like function arguments** in JS and **attributes** in HTML.
  - Props are an important concept in React as they enable the **flow of data** and behavior between components.
  - Props should be treated as **read-only** data, **cannot be modified**.

- **Defining Props**
  - Props are **defined as attributes** on a component when it is used in JSX.
  - For example, consider a **parent** component called Parent that renders a **child** component called Child

```
// Parent component
const Parent = () => {
  const name = "Creative Design";
  return <Child name={name} />;
};

// Child component
const Child = (props) => {
  return <h1>Hello, {props.name}!</h1>;
};
```

- **Passing Props**
  - Props are passed from a **parent component to its child component** by specifying them as attributes on the child component's JSX tag.
  - In the **previous example**: The **name** prop is passed from the Parent component to the Child component.

```
// Parent component
    …
  const name = "Creative Design";
  <Child name={name} />;


// Child component
    …
  <h1>Hello, {props.name}!</h1>;
```

- **Receiving Props**
  - In the child component, props are received as a **single parameter.**
  - The parameter can be **named anything**, but conventionally it is named props.
  - You can access individual props using **dot notation**, such as **props.name**.

```
// Child component
    …
  <h1>Hello, {props.name}!</h1>;
```

- **Dynamic Props :**
  - Props can be dynamic, meaning their values can change based on the **component's state** or **other factors**.
  - When a prop's value changes, React will **automatically re-render** the component and update the rendered output based on the new prop values.

- **Default Props :**
  - Default props provide a **fallback value** and help ensure that the component behaves as expected even if certain props are not explicitly passed.
  - In **functional** components, **default props** can be defined using the **default Props** property outside the component function.

```
const MyComponent = (props) => {
  return (
    <div>
      {props.greeting}, {props.name}!
    </div>
  );
};

MyComponent.defaultProps = {
  greeting: "Hello",
  name: "Guest"
};
```

- **Children Props :**
  - In React, the children prop is a special prop that allows you to pass **content** or **components** between the opening and closing tags of a component.
  - It provides a way to nest and compose components, **enabling more flexible** and dynamic component structures
- Passing Content as Children:
- Passing Components as Children

```jsx
// 01. Passing Content as Children:
const MyComponent = (props) => {
  return (
    <div>
      <h1>{props.title}</h1>
      {props.children}
    </div>
  );
};

const App = () => {
  return (
    <MyComponent title="Parent Component">
      <p>This is the content passed as children.</p>
    </MyComponent>
  );
};

// 2. Passing Components as Children:
const Button = (props) => {
  return <button>{props.children}</button>;
};

const App = () => {
  return (
    <Button>
      <span>Click me!</span>{" "}
    </Button>
  );
};
```