# React Basics

- Virtual Dom (VDOM) is a tree data structure (of HTML elements) that represents the DOM.
- VDOM reflects changes only in the changed parts of the tree (partial update).
- React implements the virtual DOM and updates the DOM based on the virtual DOM.
- React is a library that allows us to create components.
- React uses an algorithm called diffing to update the DOM (tracks the changes)
- Whatever changes are made to the virtual DOM compared to the actual DOM, React updates the DOM using the diffing algorithm.

## Components

- React is a library that allows us to create components, and is component based library.
- Components are the building blocks of React, we can breakdown website into multiple components.
- For example, we can create a component that renders a button, and another component that renders a list of buttons.

## JSX

- JSX is a syntax extension to JavaScript that allows us to write HTML inside of JavaScript.
- Shortcommings of JSX is that it cant run any loops.
- Alternative to loops we can use map and filter.
- New functions can not be defined in JSX.
- Rather while using JSX we can use newly created functions or methords in JSX.

## Getting hands dirty with DOM Manipulation basics

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
    <title>Document</title>
  </head>
  <body>
    <div class="root"></div>
  </body>
  <script>
    // add h1 element hello to root div
    let h1 = document.createElement("h1");
    h1.innerHTML = "Hello";
    let root = document.querySelector(".root");
    root.appendChild(h1);
  </script>
</html>
```

- Just to explain basic react code we are using cdn, which in production is not recommended.
- Installation using npm will be discussed in next bit by bit.
- Same as above, but using JSX syntax.

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
    <title>Document</title>
    <script
      crossorigin
      src="https://unpkg.com/react@18/umd/react.development.js"
    ></script>
    <script
      crossorigin
      src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"
    ></script>
    <script src="https://unpkg.com/babel-standalone@6/babel.min.js">
</script>
  </head>

  <body>
    <div class="root"></div>

    <script type="text/babel">
      // add h1 element hello to root div
      // let h1 = document.createElement('h1');
      // h1.innerHTML = 'Hello';
      // let root = document.querySelector('.root');
      // root.appendChild(h1);

      // let ele = React.createElement('h1', {
      //     className: 'hello',
      //     id: 'hello',
      // }, 'Hello');

      // Syntax : ReactDOM.render(<element to render>, <where to render>);
      // ReactDOM.render(ele, document.querySelector('.root'));

      // Functional Component
      function Element() {
        return <h1>Hello</h1>;
      }
      ReactDOM.render(<Element />, document.querySelector(".root"));
    </script>
  </body>
</html>
```

- In order to use react we need react and react-dom cdn.

```
<script
  crossorigin
  src="https://unpkg.com/react@18/umd/react.development.js"
></script>
<script
  crossorigin
  src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"
></script>
```

- Inorder to render anything using react we use, `ReactDOM.render(<element to render>, <where to render>);`
- Inorder to use JSX Syntax we need to transpile it using babel.
- To use babel we use cdn links in the head section.

```
<script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
```

- Babel is a compiler that transforms JavaScript code into a format that can be run by a JavaScript engine.
- In order for the browser to use the babel we need to tell that script type is `text/babel`.

```
<script type="text/babel">
  function Element() {
    return <h1>Hello</h1>;
  }
  ReactDOM.render(<Element />, document.querySelector(".root"));
</script>
```

- Babel is a transpiler that takes JSX and converts it to JavaScript.
- Alternatives to it are SWC transpiler and Webpack.
- Webpack is a module bundler.

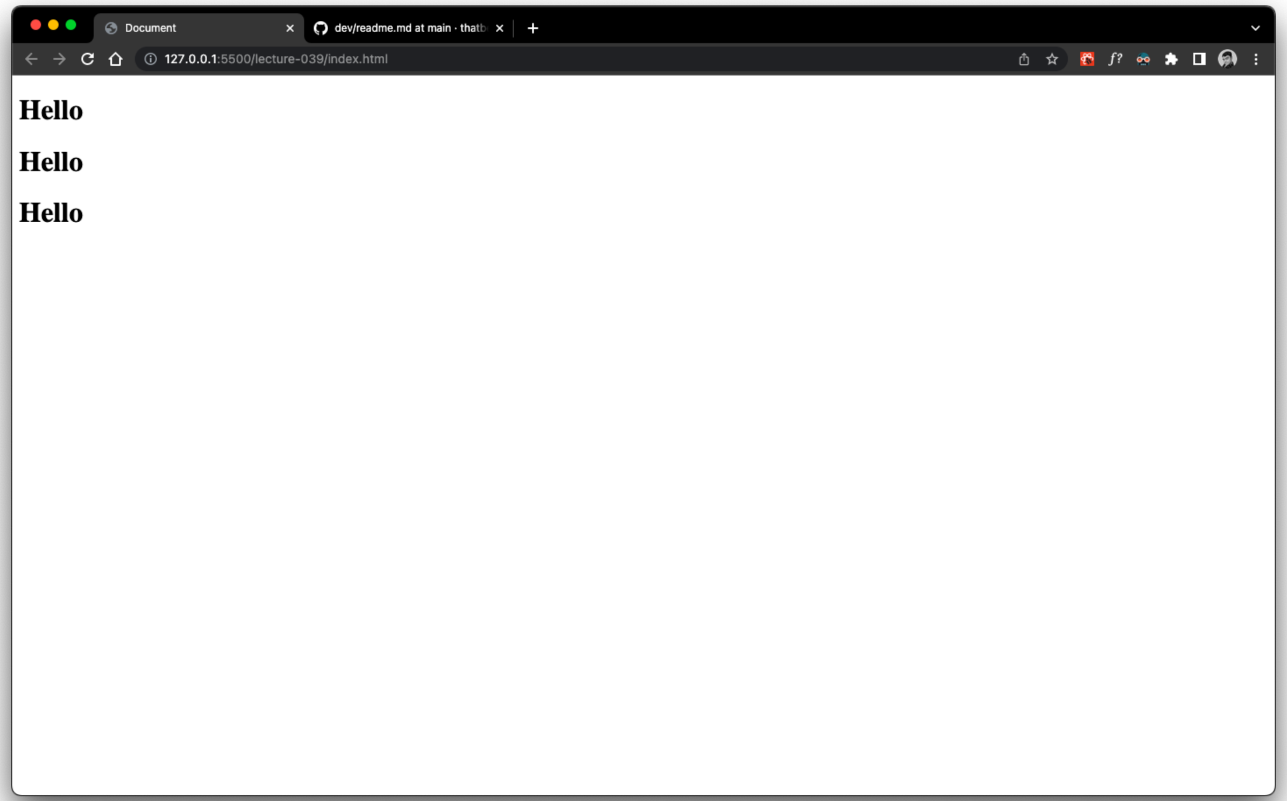
## Child and Parent Components

```
function Child() {
  return <h1>Hello</h1>;
}
function Parent() {
  return <Child />;
}
ReactDOM.render(<Parent />, document.querySelector(".root"));
```

- Note in order to use multiple child components we need to wrap them in a parent component and then render it.
- Common convention is to wrap them inside a div or an empty element `<> </>` popularly known as react fragment.

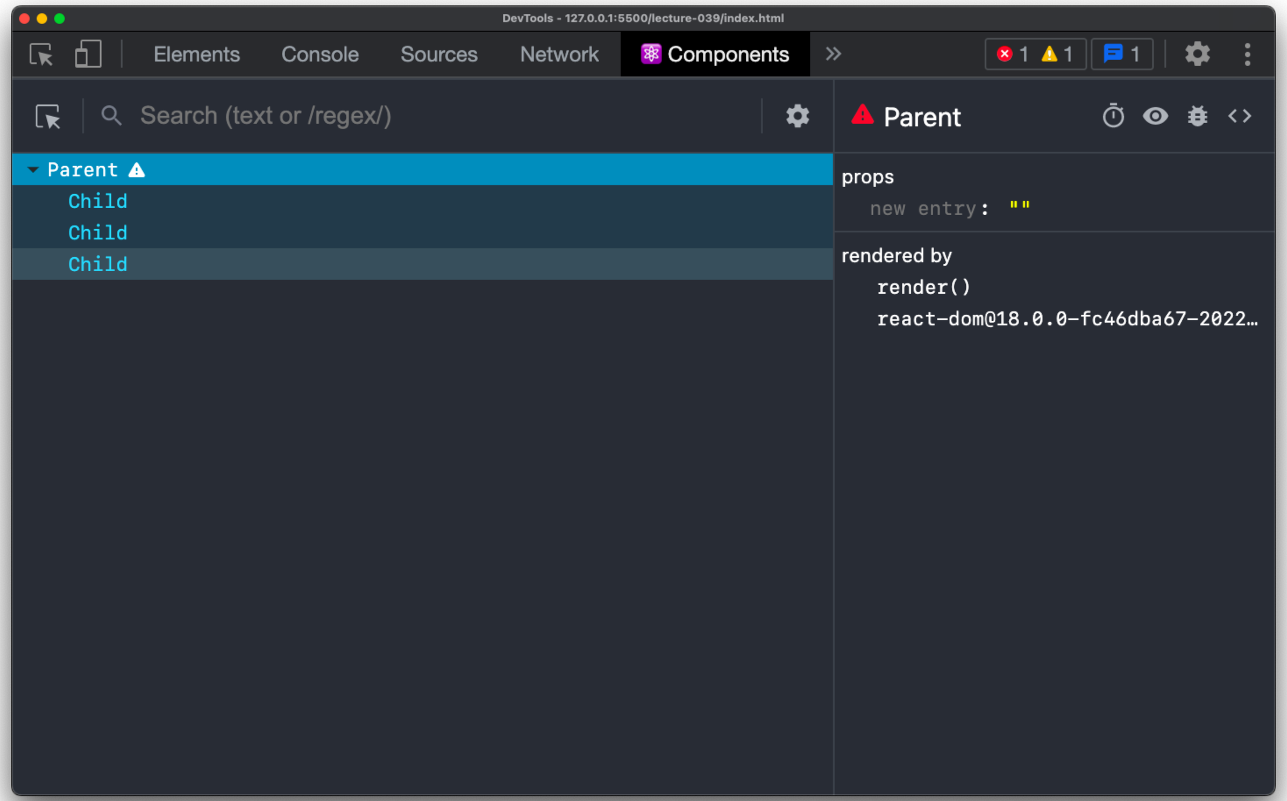
```
function Child() {
  return <h1>Hello</h1>;
}
function Parent() {
  return (
    <>
      <Child />
      <Child />
      <Child />
    </>
  );
}
ReactDOM.render(<Parent />, document.querySelector(".root"));
```

- This will show 3 child components with same Hello text.


Document
dev/readme.md at main · thatb
127.0.0.1:5500/lecture-039/index.html

```
Hello
Hello
Hello
```

- Inorder to see components we can use react developer tools, in the development build.



```
Parent
  Child
  Child
  Child
```

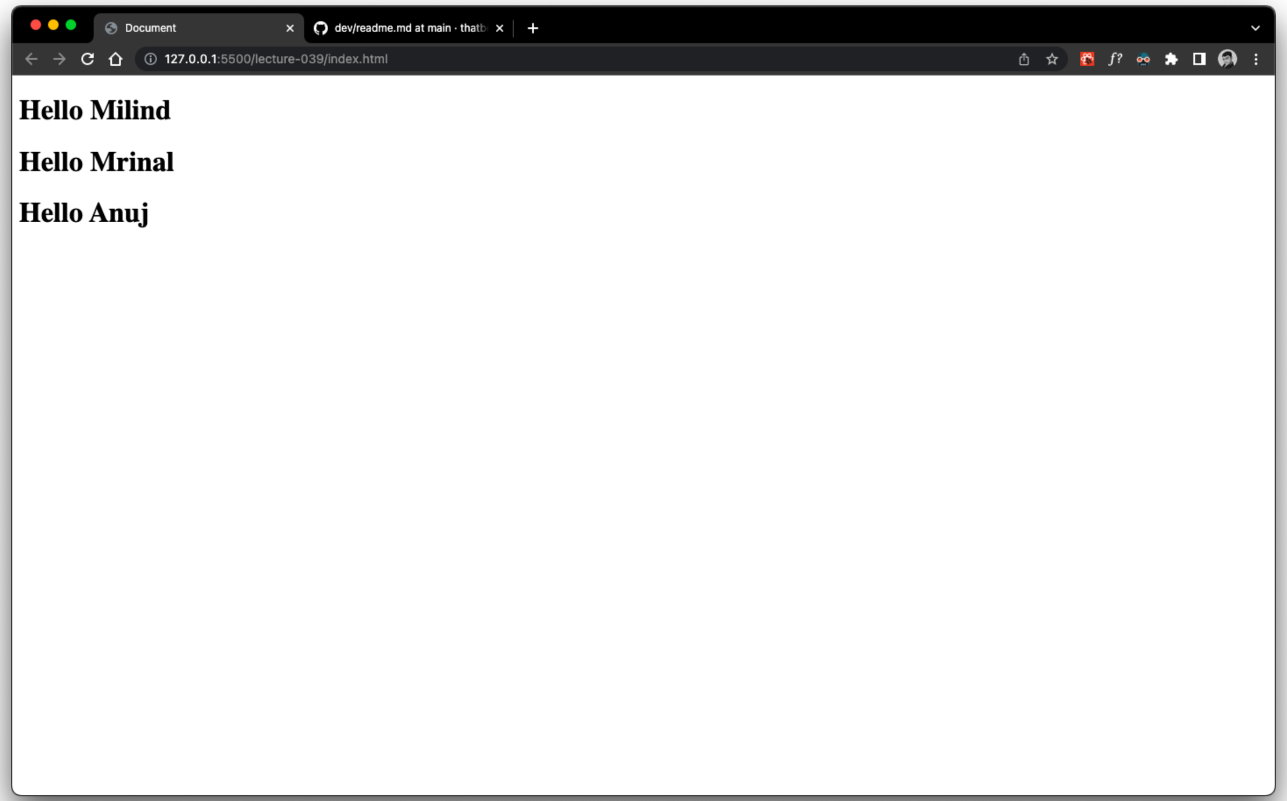
```
props
  new entry: ""
rendered by
  render()
  react-dom@18.0.0-fc46dba67-2022…
```

## Props

- Props are the properties of a component that are passed to it.
- Props are used to pass diffrent data to a component.
- Props can be passed to a component using `props` keyword.
- Independently we can pass props using the {`prop-name`} syntax.

```
function Child({ name }) {
  return <h1>Hello {name}</h1>;
}
function Parent() {
  return (
    <div>
      <Child name="Milind" />
      <Child name="Mrinal" />
      <Child name="Anuj" />
    </div>
  );
}
ReactDOM.render(<Parent />, document.querySelector(".root"));
```

- Displaying usage of props in the child component.

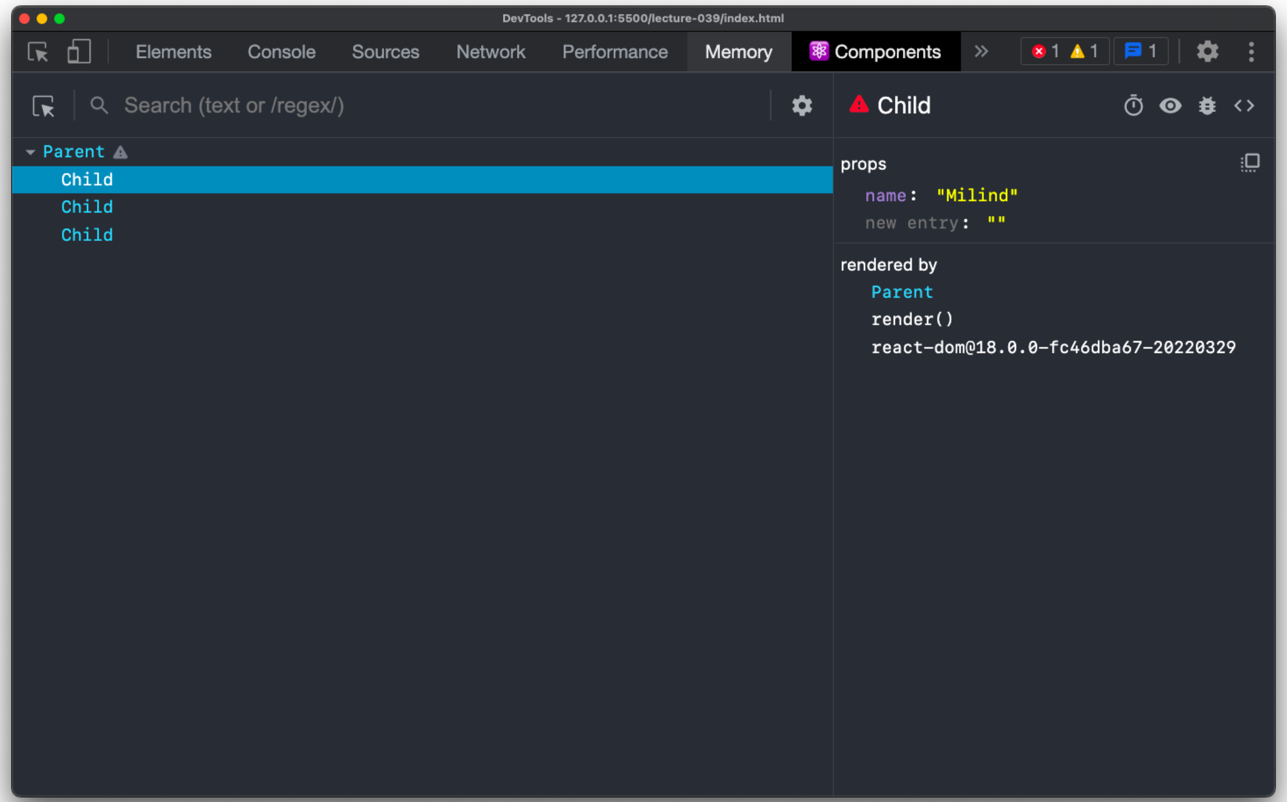

Document
dev/readme.md at main · thatb
127.0.0.1:5500/lecture-039/index.html

Hello Milind

Hello Mrinal

Hello Anuj

- Props in dev tools are shown as `props` tab.



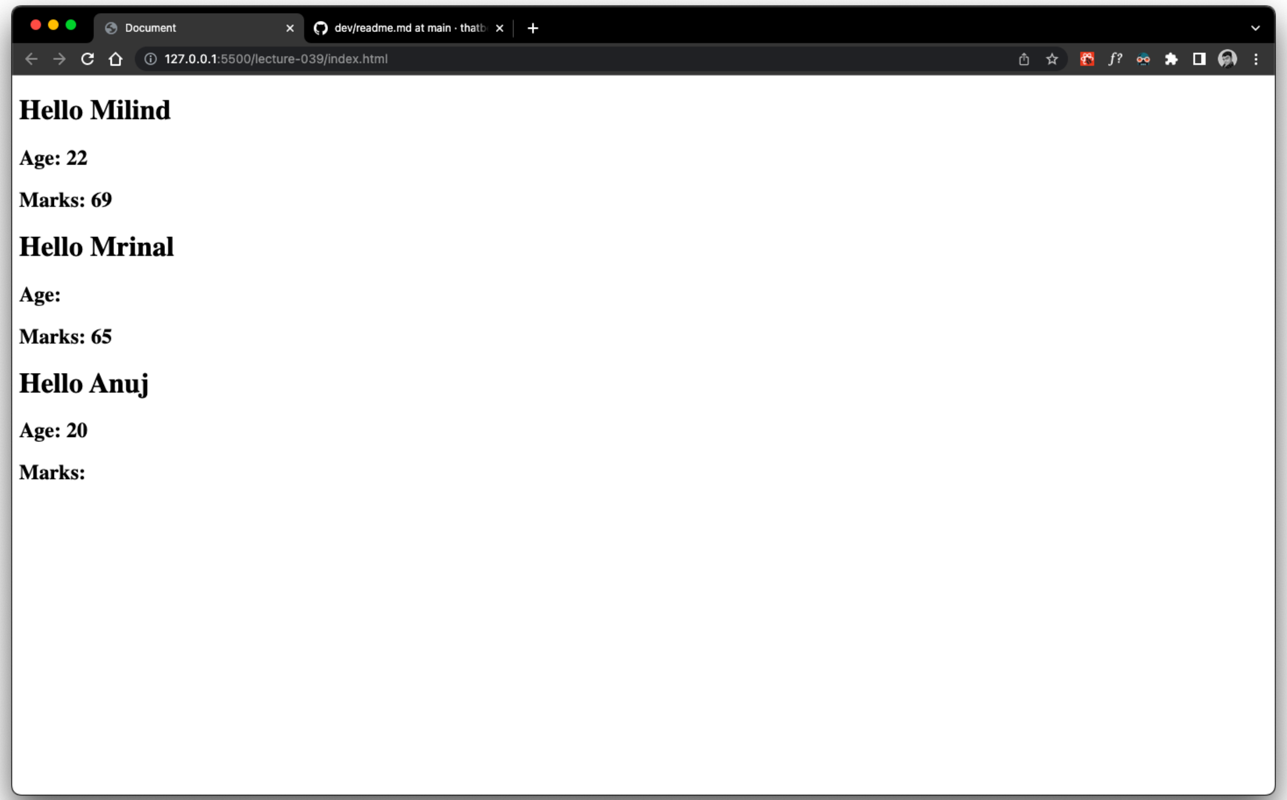
```
Parent
  Child
  Child
  Child
```

```
Child
props
  name: "Milind"
  new entry: ""
rendered by
  Parent
  render()
  react-dom@18.0.0-fc46dba67-20220329
```

- Passing multiple props to a component.

```
function Child({ name, age, marks }) {
  return (
    <div>
      <h1>Hello {name}</h1>
      <h2>Age: {age}</h2>
      <h2>Marks: {marks}</h2>
    </div>
  );
}
function Parent() {
  return (
    <div>
      <Child name="Milind" age="22" marks="69" />
      <Child name="Mrinal" marks="65" />
      <Child name="Anuj" age="20" />
    </div>
  );
}
ReactDOM.render(<Parent />, document.querySelector(".root"));
```

- Showing usage of multiple props in the child component.


Document
dev/readme.md at main · thatb
127.0.0.1:5500/lecture-039/index.html

Hello Milind

Age: 22

Marks: 69

Hello Mrinal

Age:

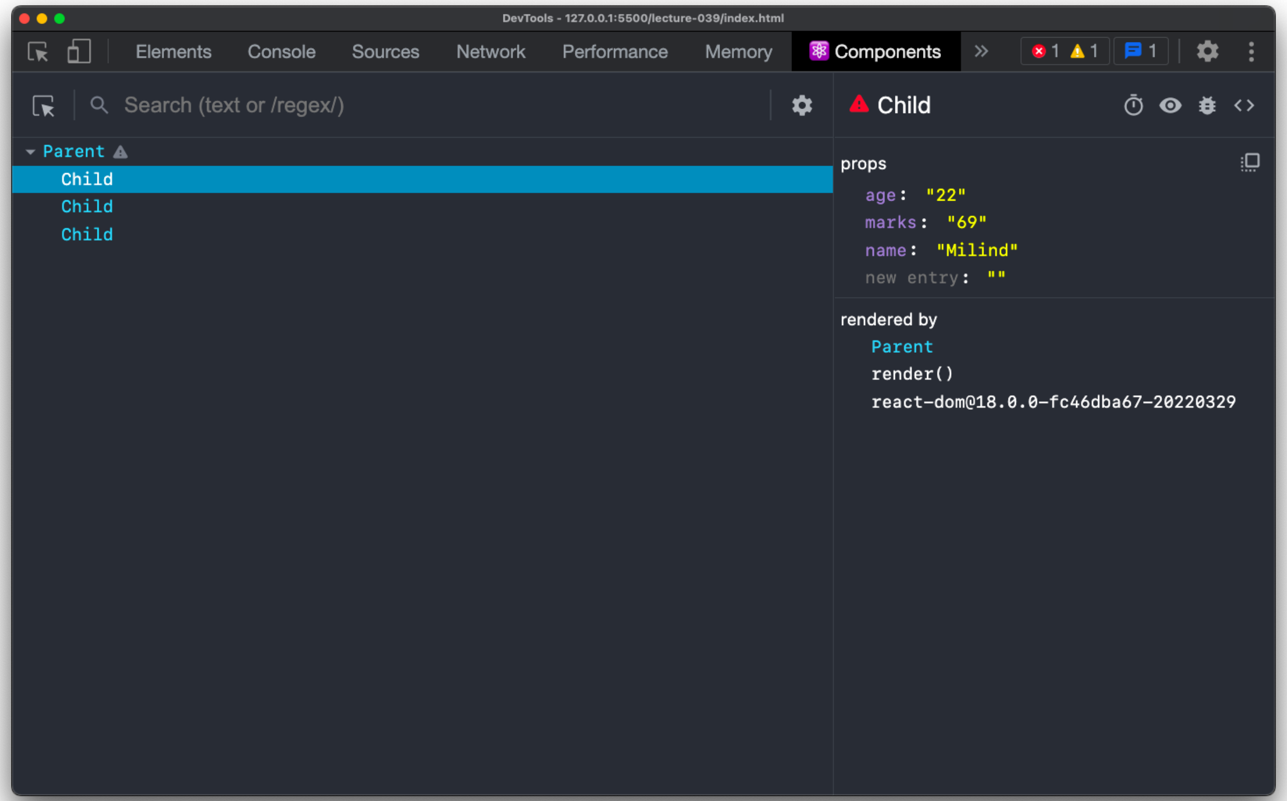
Marks: 65

Hello Anuj

Age: 20

Marks:

- Dev tools shows multiple props as `props` tab.



```
Parent
  Child
  Child
  Child
```

Child

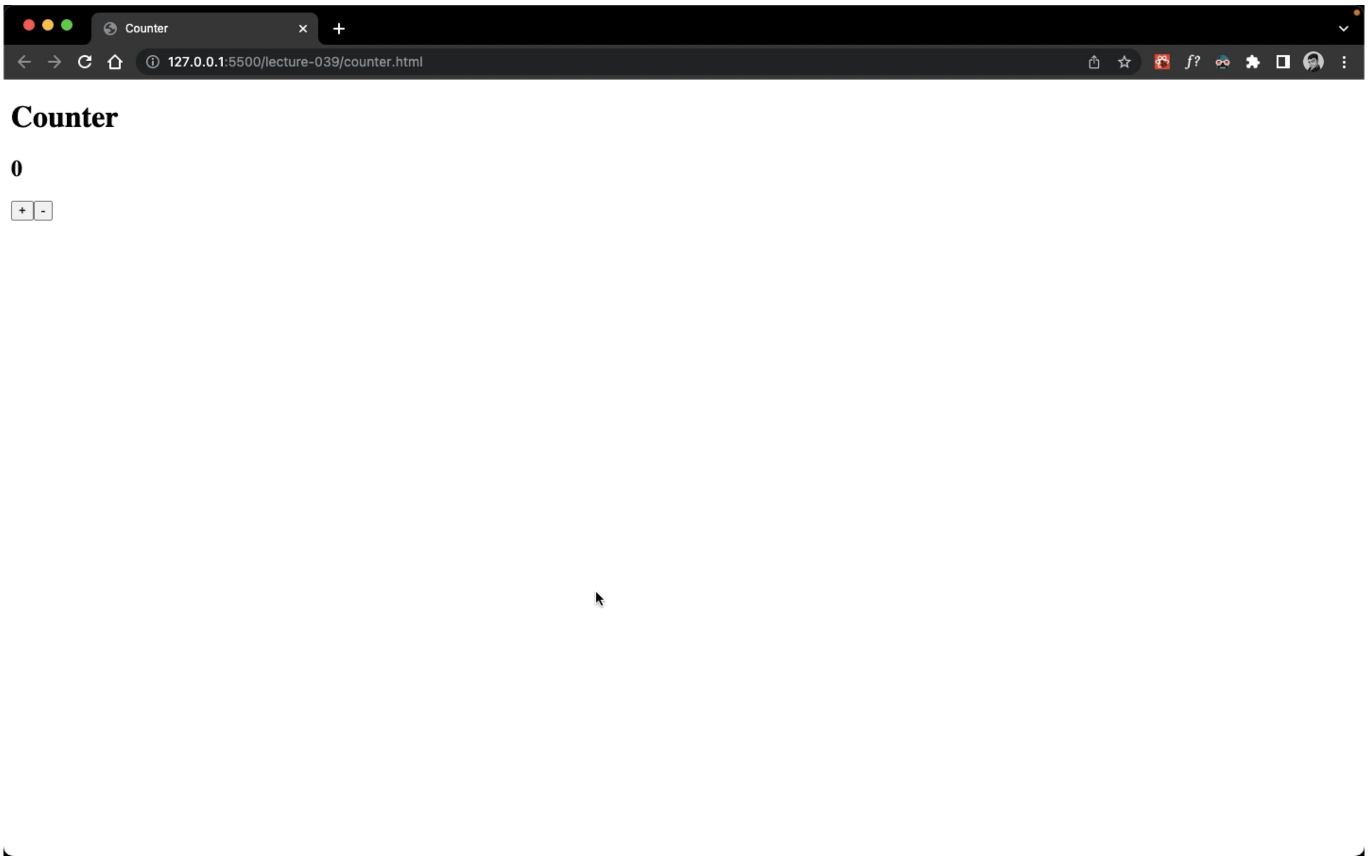
```
props
  age: "22"
  marks: "69"
  name: "Milind"
  new entry: ""

rendered by
  Parent
  render()
  react-dom@18.0.0-fc46dba67-20220329
```

## Counter in React with class based component

- Implementing counter in React using class component.
- Using class based component we can use state.
- React.Component is a base class for all React components when we extend it we can use all the methods of React.Component.

```
class Counter extends React.Component {
  constructor() {
    super();
    this.state = {
      count: 0,
    };
  }
  render() {
    return (
      <div>
        <h1>Counter</h1>
        <h2>{this.state.count}</h2>
        <button onClick={() => this.setState({ count: this.state.count + 1
})}>
          +
        </button>
        <button onClick={() => this.setState({ count: this.state.count - 1
})}>
          -
        </button>
      </div>
    );
  }
}
ReactDOM.render(<Counter />, document.querySelector(".root"));
```

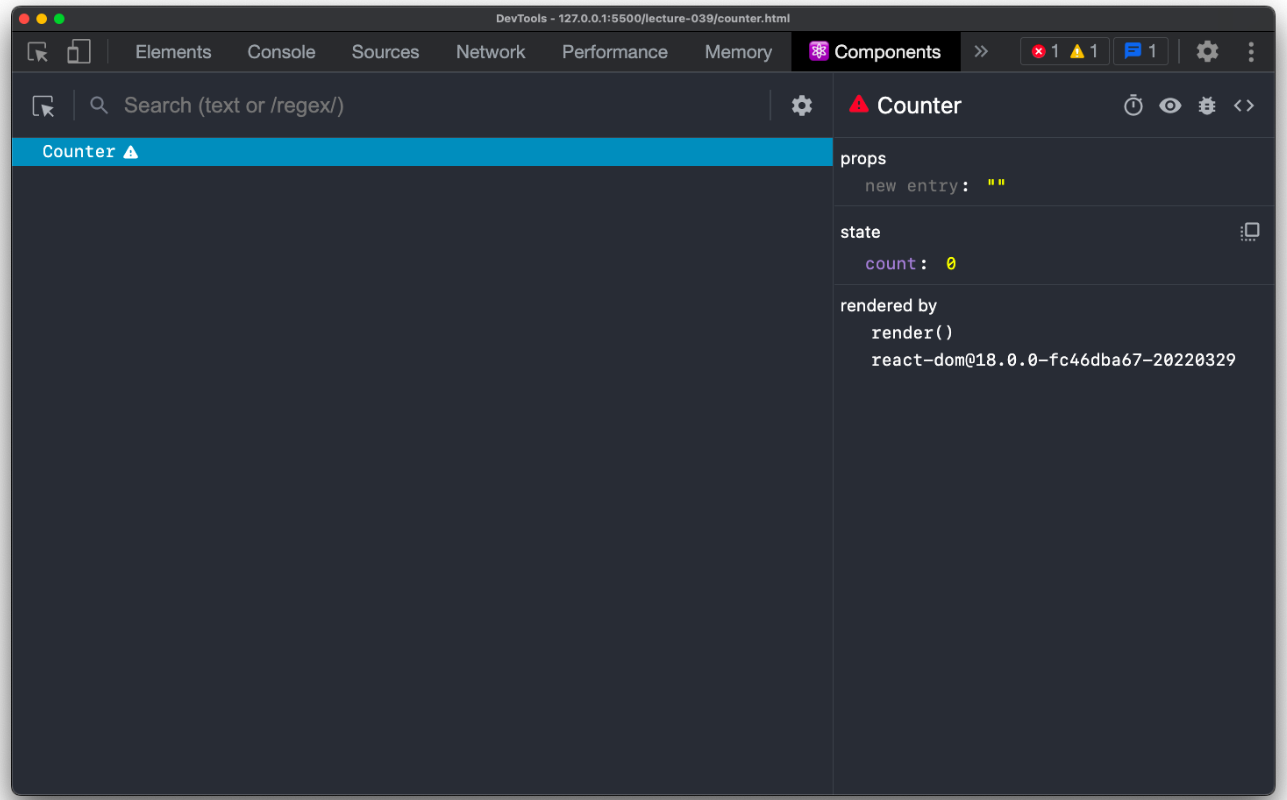
- Counter Demo.



## Counter

**0**

- Displaying counter in dev tools.



```
Counter ⚠
```

Counter

```
props
  new entry: ""
state
  count: 0
rendered by
  render()
  react-dom@18.0.0-fc46dba67-20220329
```

- Understanding the class based component, the React.Component class consists of all the important methods of React.Component like render, componentDidMount, componentDidUpdate, componentWillUnmount... etc.
- render() is the method that is called to render the component which takes in JSX input and returns a React element.
- At the end we still need to call the ReactDOM.render() method to render the component and pass in the class based component as an argument in simmilar to the functional component.
- The class based component has a structure same as that of a class in js.

```
class Component extends React.Component {
  constructor() {
    super();
  }
}
```

- constructor() is a special method that is called when the class based component is created.
- super() is a special method that is called when the class based component is created which is used to call the constructor of the parent class ans pass the props to the parent class.

## State

- State is a plain JavaScript object that is used to record and react to user events.
- React gives a special methord to change properties of a component called setState() which is used to change the state of a component.
- this.setState() is a method that is called to change the state of a component.