

STATE VS. PROPS

STATE VS. PROPS

STATE

- 👉 **Internal data**, owned by component

STATE VS. PROPS

STATE

- 👉 Internal data, owned by component

```
function Question() {  
  const [upvotes, setUpvotes] = useState(0);  
  
  return (  
    <div>  
      {/* ... */}  
      <Button upvotes={upvotes} bgColor="blue" />  
    </div>  
  );  
}
```

```
function Button({ upvotes, bgColor }) {  
  const [hovered, setHovered] = useState(false);  
  
  return (  
    <div>  
      {/* ... */}  
      <button  
        onMouseEnter={() => setHovered(true)}  
        onMouseLeave={() => setHovered(false)}  
        style={{ background: bgColor }}  
      >  
        {hovered ? "Upvote" : `👍 ${upvotes}`}  
      </button>  
    </div>  
  );  
}
```

STATE VS. PROPS

STATE

- 👉 Internal data, owned by component

```
function Question() {  
  const [upvotes, setUpvotes] = useState(0);  
  
  return (  
    <div>  
      { /* ... */ }  
      <Button upvotes={upvotes} bgColor="blue" />  
    </div>  
  );  
}
```

```
function Button({ upvotes, bgColor }) {  
  const [hovered, setHovered] = useState(false);  
  
  return (  
    <div>  
      { /* ... */ }  
      <button  
        onMouseEnter={() => setHovered(true)}  
        onMouseLeave={() => setHovered(false)}  
        style={{ background: bgColor }}  
      >  
        {hovered ? "Upvote" : `👍 ${upvotes}`}  
      </button>  
    </div>  
  );  
}
```

STATE VS. PROPS

STATE

👉 Internal data, owned by component

```
function Question() {  
  const [upvotes, setUpvotes] = useState(0);  
  
  return (  
    <div>  
      {/* ... */}  
      <Button upvotes={upvotes} bgColor="blue" />  
    </div>  
  );  
}
```

PROPS

👉 External data, owned by parent component

```
function Button({ upvotes, bgColor }) {  
  const [hovered, setHovered] = useState(false);  
  
  return (  
    <div>  
      {/* ... */}  
      <button  
        onMouseEnter={() => setHovered(true)}  
        onMouseLeave={() => setHovered(false)}  
        style={{ background: bgColor }}  
      >  
        {hovered ? "Upvote" : `👍 ${upvotes}`}  
      </button>  
    </div>  
  );  
}
```

STATE VS. PROPS

STATE

- 👉 Internal data, owned by component

```
function Question() {  
  const [upvotes, setUpvotes] = useState(0);  
  
  return (  
    <div>  
      { /* ... */ }  
      <Button upvotes={upvotes} bgColor="blue" />  
    </div>  
  );  
}
```

```
function Button( upvotes, bgColor ) {  
  const [hovered, setHovered] = useState(false);  
  
  return (  
    <div>  
      { /* ... */ }  
      <button  
        onMouseEnter={() => setHovered(true)}  
        onMouseLeave={() => setHovered(false)}  
        style={{ background: bgColor }}  
      >  
        {hovered ? "Upvote" : `👍 ${upvotes}`}  
      </button>  
    </div>  
  );  
}
```

PROPS

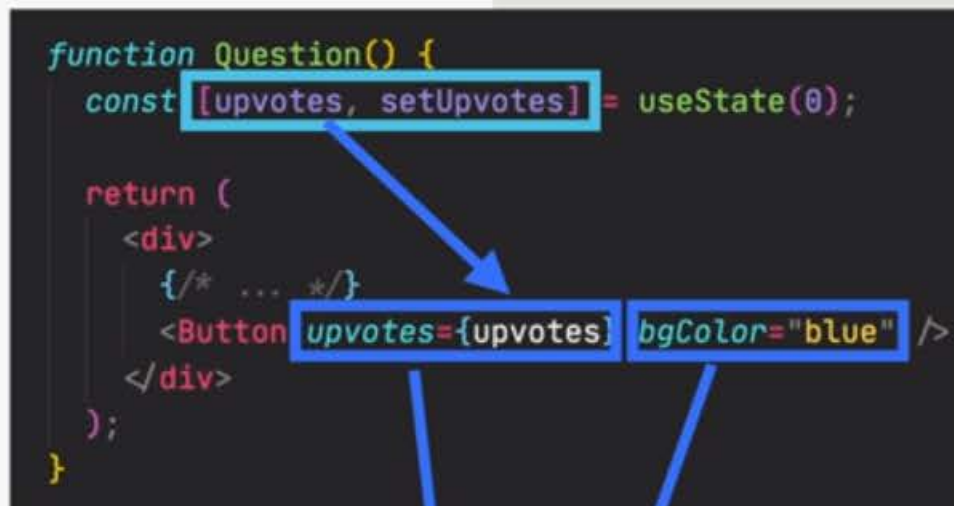
- 👉 External data, owned by parent component

STATE VS. PROPS

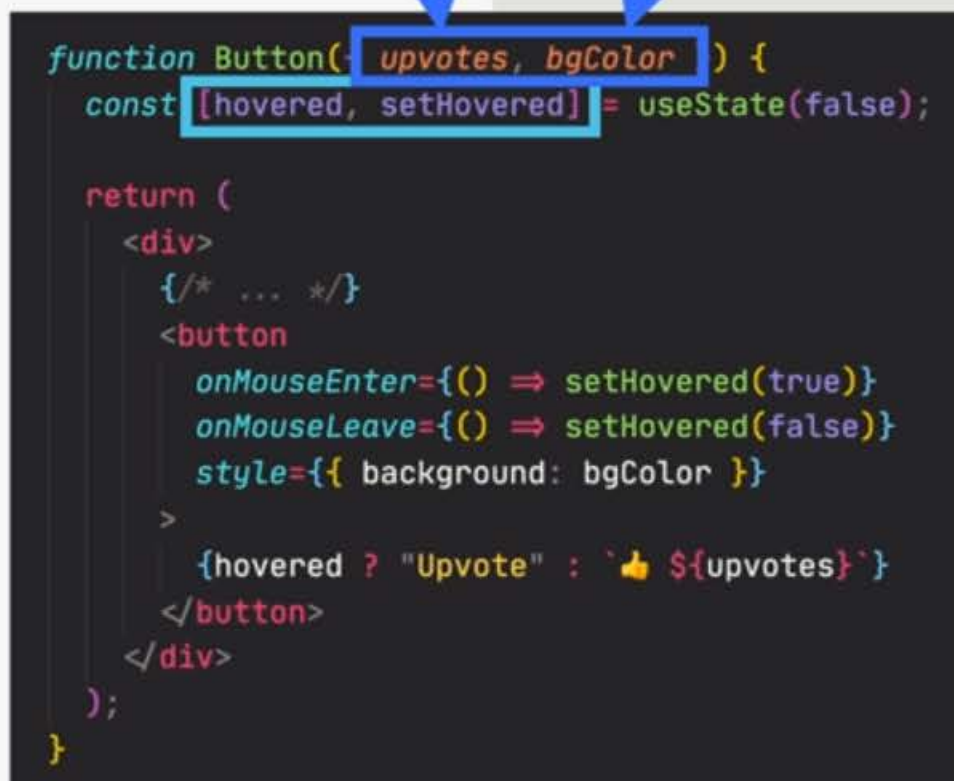
STATE

- 👉 Internal data, owned by component

```
function Question() {  
  const [upvotes, setUpvotes] = useState(0);  
  
  return (  
    <div>  
      { /* ... */ }  
      <Button upvotes={upvotes} bgColor="blue" />  
    </div>  
  );  
}
```

A diagram illustrating the flow of state. In the `Question` component, the `upvotes` state is managed by `useState(0)`. This state is passed to the `Button` component as a prop `upvotes={upvotes}`. The `bgColor="blue"` prop is also shown. Arrows indicate the flow from the state in `Question` to the `upvotes` prop in `Button`.

```
function Button( upvotes, bgColor ) {  
  const [hovered, setHovered] = useState(false);  
  
  return (  
    <div>  
      { /* ... */ }  
      <button  
        onMouseEnter={() => setHovered(true)}  
        onMouseLeave={() => setHovered(false)}  
        style={{ background: bgColor }}  
      >  
        {hovered ? "Upvote" : `👍 ${upvotes}`}  
      </button>  
    </div>  
  );  
}
```

A diagram illustrating the flow of props. The `upvotes` prop from the `Question` component is passed to the `Button` component. The `bgColor` prop is also passed. Arrows indicate the flow from the `upvotes` prop in `Question` to the `upvotes` parameter in `Button`, and from the `bgColor` prop in `Question` to the `bgColor` parameter in `Button`.

PROPS

- 👉 External data, owned by parent component
- 👉 Similar to function parameters

STATE VS. PROPS

STATE

- 👉 Internal data, owned by component
- 👉 Component “memory”

```
function Question() {  
  const [upvotes, setUpvotes] = useState(0);  
  
  return (  
    <div>  
      { /* ... */ }  
      <Button upvotes={upvotes} bgColor="blue" />  
    </div>  
  );  
}
```

```
function Button( upvotes, bgColor ) {  
  const [hovered, setHovered] = useState(false);  
  
  return (  
    <div>  
      { /* ... */ }  
      <button  
        onMouseEnter={() => setHovered(true)}  
        onMouseLeave={() => setHovered(false)}  
        style={{ background: bgColor }}  
      >  
        {hovered ? "Upvote" : `👍 ${upvotes}`}  
      </button>  
    </div>  
  );  
}
```

PROPS

- 👉 External data, owned by parent component
- 👉 Similar to function parameters

STATE VS. PROPS

STATE

- 👉 Internal data, owned by component
- 👉 Component "memory"
- 👉 Can be updated by the component itself

```
function Question() {  
  const [upvotes, setUpvotes] = useState(0);  
  
  return (  
    <div>  
      { /* ... */ }  
      <Button upvotes={upvotes} bgColor="blue" />  
    </div>  
  );  
}
```

```
function Button( upvotes, bgColor ) {  
  const [hovered, setHovered] = useState(false);  
  
  return (  
    <div>  
      { /* ... */ }  
      <button  
        onMouseEnter={() => setHovered(true)}  
        onMouseLeave={() => setHovered(false)}  
        style={{ background: bgColor }}  
      >  
        { hovered ? "Upvote" : `👍 ${upvotes}` }  
      </button>  
    </div>  
  );  
}
```

PROPS

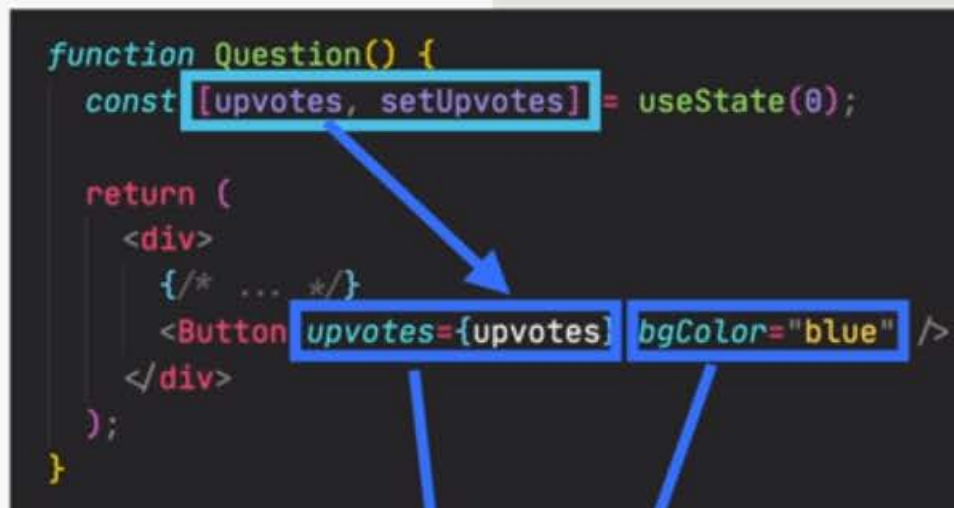
- 👉 External data, owned by parent component
- 👉 Similar to function parameters

STATE VS. PROPS

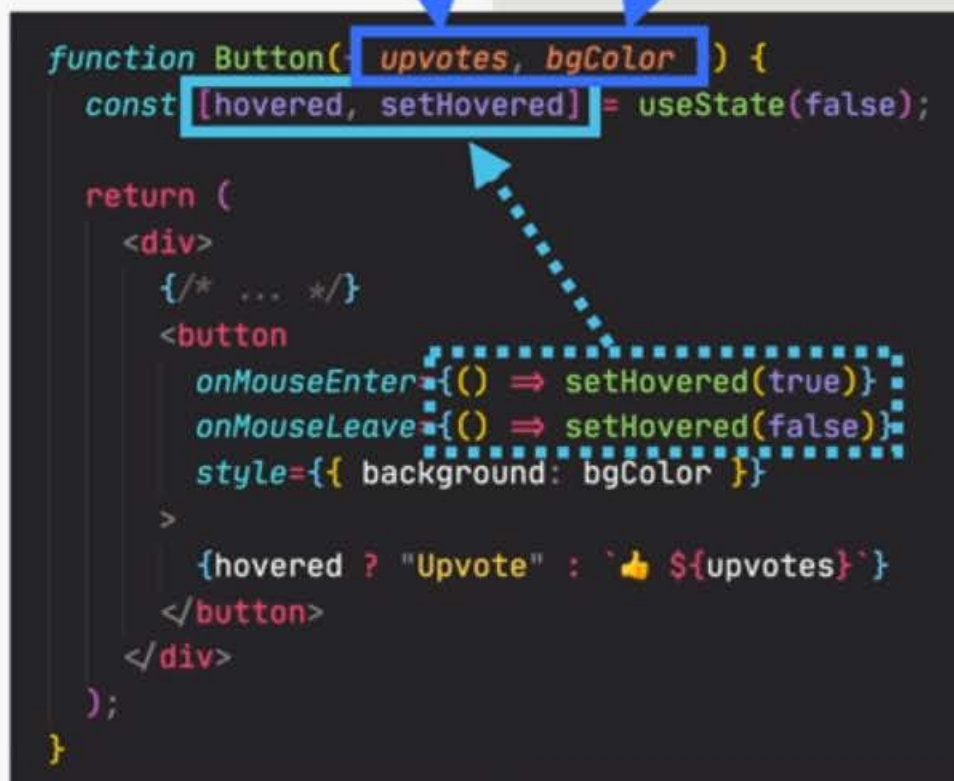
STATE

- 👉 Internal data, owned by component
- 👉 Component "memory"
- 👉 Can be updated by the component itself

```
function Question() {  
  const [upvotes, setUpvotes] = useState(0);  
  
  return (  
    <div>  
      { /* ... */ }  
      <Button upvotes={upvotes} bgColor="blue" />  
    </div>  
  );  
}
```

A diagram illustrating the flow of state. In the `Question` component, the `upvotes` state is defined. A solid blue arrow points from the `upvotes` variable to the `upvotes={upvotes}` prop in the `Button` component. Another solid blue arrow points from the `bgColor="blue"` prop to the `bgColor` parameter in the `Button` component's function signature.

```
function Button( upvotes, bgColor ) {  
  const [hovered, setHovered] = useState(false);  
  
  return (  
    <div>  
      { /* ... */ }  
      <button  
        onMouseEnter={() => setHovered(true)}  
        onMouseLeave={() => setHovered(false)}  
        style={{ background: bgColor }}  
      >  
        {hovered ? "Upvote" : `👍 ${upvotes}`}  
      </button>  
    </div>  
  );  
}
```

A diagram illustrating the flow of props. A solid blue arrow points from the `upvotes` prop in the `Question` component to the `upvotes` parameter in the `Button` component's function signature. A dashed blue arrow points from the `onMouseEnter` and `onMouseLeave` event handlers in the `Button` component back to the `upvotes` state in the `Question` component, indicating that the state is updated based on these events.

PROPS

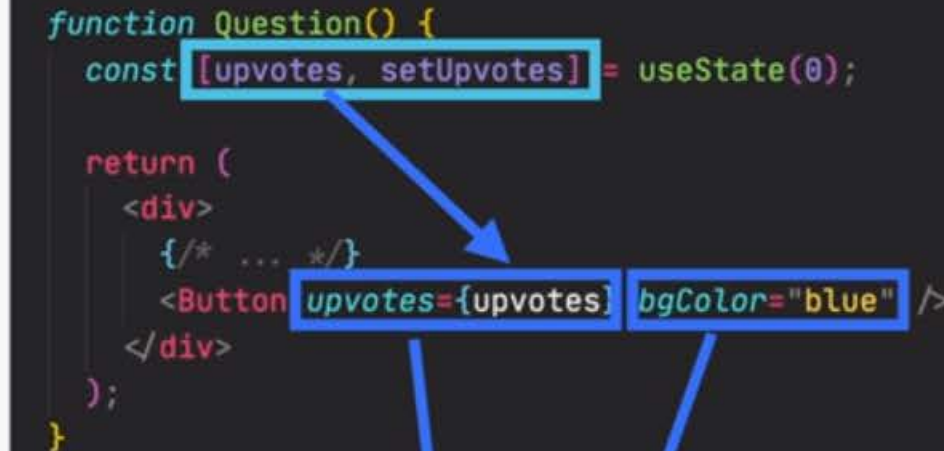
- 👉 External data, owned by parent component
- 👉 Similar to function parameters

STATE VS. PROPS

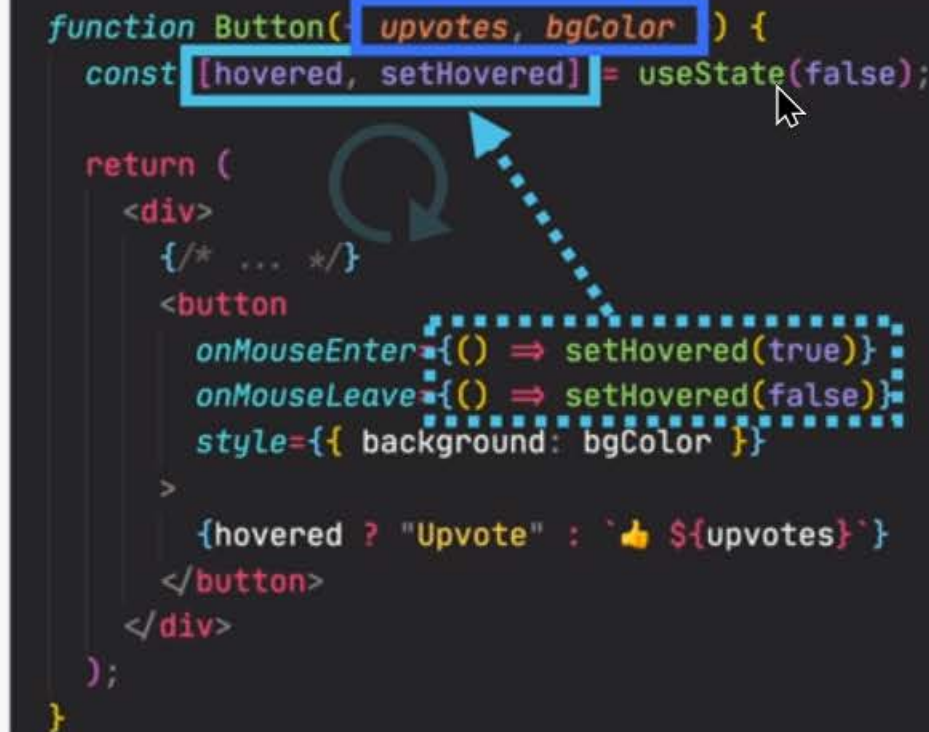
STATE

- 👉 Internal data, owned by component
- 👉 Component "memory"
- 👉 Can be updated by the component itself
- 👉 Updating state causes component to re-render

```
function Question() {  
  const [upvotes, setUpvotes] = useState(0);  
  
  return (  
    <div>  
      { /* ... */ }  
      <Button upvotes={upvotes} bgColor="blue" />  
    </div>  
  );  
}
```

A diagram illustrating the flow of state. In the `Question` component, the `upvotes` state is defined. A solid blue arrow points from the `upvotes` variable to the `upvotes={upvotes}` prop in the `Button` component. Another solid blue arrow points from the `bgColor="blue"` prop to the `bgColor` parameter in the `Button` component's function signature.

```
function Button( upvotes, bgColor ) {  
  const [hovered, setHovered] = useState(false);  
  
  return (  
    <div>  
      { /* ... */ }  
      <button  
        onMouseEnter={() => setHovered(true)}  
        onMouseLeave={() => setHovered(false)}  
        style={{ background: bgColor }}  
      >  
        {hovered ? "Upvote" : `👍 ${upvotes}`}  
      </button>  
    </div>  
  );  
}
```

A diagram illustrating state flow and self-update within the `Button` component. The `upvotes` and `bgColor` props are shown as parameters in the function signature. A solid blue arrow points from the `upvotes` prop to the `upvotes` parameter. A solid blue arrow points from the `bgColor` prop to the `bgColor` parameter. A dashed blue arrow points from the `setHovered` function back to the `hovered` state variable, indicating a self-update. A circular arrow icon is also present near the `setHovered` function.

PROPS

- 👉 External data, owned by parent component
- 👉 Similar to function parameters

STATE VS. PROPS

STATE

- 👉 Internal data, owned by component
- 👉 Component "memory"
- 👉 Can be updated by the component itself
- 👉 Updating state causes component to re-render
- 👉 Used to make components interactive

```
function Question() {  
  const [upvotes, setUpvotes] = useState(0);  
  
  return (  
    <div>  
      { /* ... */ }  
      <Button upvotes={upvotes} bgColor="blue" />  
    </div>  
  );  
}
```

```
function Button( upvotes, bgColor ) {  
  const [hovered, setHovered] = useState(false);  
  
  return (  
    <div>  
      { /* ... */ }  
      <button  
        onMouseEnter={() => setHovered(true)}  
        onMouseLeave={() => setHovered(false)}  
        style={{ background: bgColor }}  
      >  
        {hovered ? "Upvote" : `👍 ${upvotes}`}  
      </button>  
    </div>  
  );  
}
```

PROPS

- 👉 External data, owned by parent component
- 👉 Similar to function parameters

STATE VS. PROPS

STATE

- 👉 Internal data, owned by component
- 👉 Component "memory"
- 👉 Can be updated by the component itself
- 👉 Updating state causes component to re-render
- 👉 Used to make components interactive

```
function Question() {  
  const [upvotes, setUpvotes] = useState(0);  
  
  return (  
    <div>  
      { /* ... */ }  
      <Button upvotes={upvotes} bgColor="blue" />  
    </div>  
  );  
}
```

```
function Button( upvotes, bgColor ) {  
  const [hovered, setHovered] = useState(false);  
  
  return (  
    <div>  
      { /* ... */ }  
      <button  
        onMouseEnter={() => setHovered(true)}  
        onMouseLeave={() => setHovered(false)}  
        style={{ background: bgColor }}  
      >  
        {hovered ? "Upvote" : `👍 ${upvotes}`}  
      </button>  
    </div>  
  );  
}
```

PROPS

- 👉 External data, owned by parent component
- 👉 Similar to function parameters
- 👉 Read-only

STATE VS. PROPS

STATE

- 👉 Internal data, owned by component
- 👉 Component "memory"
- 👉 Can be updated by the component itself
- 👉 Updating state causes component to re-render
- 👉 Used to make components interactive

```
function Question() {  
  const [upvotes, setUpvotes] = useState(0);  
  
  return (  
    <div>  
      { /* ... */ }  
      <Button upvotes={upvotes} bgColor="blue" />  
    </div>  
  );  
}
```

```
function Button( upvotes, bgColor ) {  
  const [hovered, setHovered] = useState(false);  
  
  return (  
    <div>  
      { /* ... */ }  
      <button  
        onMouseEnter={() => setHovered(true)}  
        onMouseLeave={() => setHovered(false)}  
        style={{ background: bgColor }}  
      >  
        {hovered ? "Upvote" : `👍 ${upvotes}`}  
      </button>  
    </div>  
  );  
}
```

PROPS

- 👉 External data, owned by parent component
- 👉 Similar to function parameters
- 👉 Read-only
- 👉 Receiving new props causes component to re-render. Usually when the parent's state has been updated

STATE VS. PROPS

STATE

- 👉 Internal data, owned by component
- 👉 Component "memory"
- 👉 Can be updated by the component itself
- 👉 Updating state causes component to re-render

```
function Question() {  
  const [upvotes, setUpvotes] = useState(0);  
  
  return (  
    <div>  
      { /* ... */ }  
      <Button upvotes={upvotes} bgColor="blue" />  
    </div>  
  );  
}
```

```
function Button( upvotes, bgColor ) {  
  const [hovered, setHovered] = useState(false);  
  
  return (  
    <div>  
      { /* ... */ }  
      <button  
        onMouseEnter={() => setHovered(true)}  
        onMouseLeave={() => setHovered(false)}  
        style={{ background: bgColor }}  
      >  
        {hovered ? "Upvote" : `👍 ${upvotes}`}  
      </button>  
    </div>  
  );  
}
```

PROPS

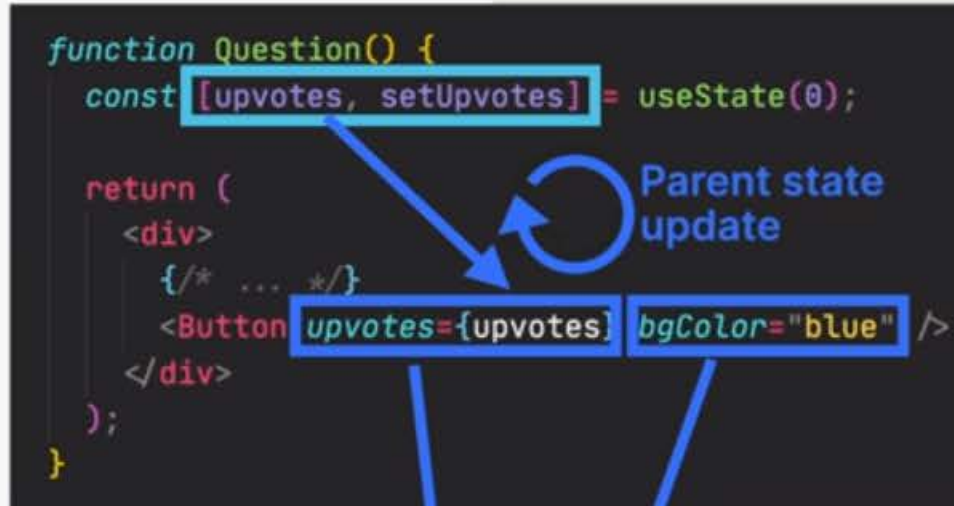
- 👉 External data, owned by parent component
- 👉 Similar to function parameters

STATE VS. PROPS

STATE

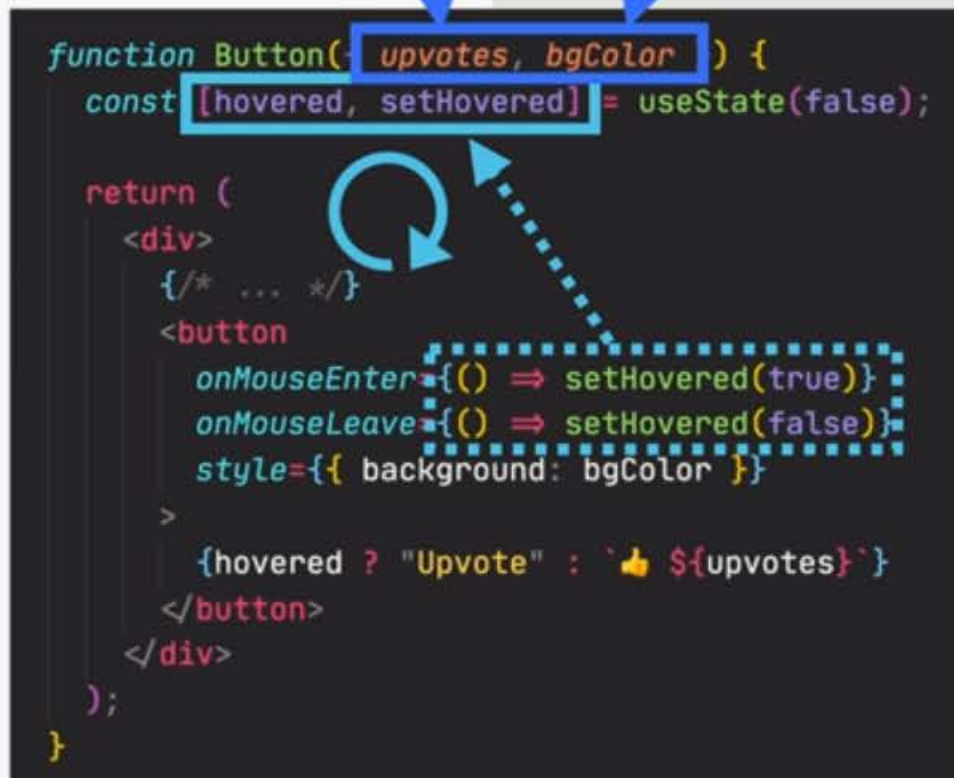
- 👉 Internal data, owned by component
- 👉 Component "memory"
- 👉 Can be updated by the component itself
- 👉 Updating state causes component to re-render
- 👉 Used to make components interactive

```
function Question() {  
  const [upvotes, setUpvotes] = useState(0);  
  
  return (  
    <div>  
      { /* ... */ }  
      <Button upvotes={upvotes} bgColor="blue" />  
    </div>  
  );  
}
```



A blue circular arrow labeled "Parent state update" points to the `upvotes` prop in the `<Button>` component, indicating that the parent component's state update triggers a re-render of the child component.

```
function Button( upvotes, bgColor ) {  
  const [hovered, setHovered] = useState(false);  
  
  return (  
    <div>  
      { /* ... */ }  
      <button  
        onMouseEnter={() => setHovered(true)}  
        onMouseLeave={() => setHovered(false)}  
        style={{ background: bgColor }}  
      >  
        {hovered ? "Upvote" : `👍 ${upvotes}`}  
      </button>  
    </div>  
  );  
}
```



A blue circular arrow indicates a self-update of the `hovered` state within the `Button` component. A dashed blue arrow points from the `upvotes` prop to the `setHovered` function, showing that the parent's state update triggers a re-render of the child component.

PROPS

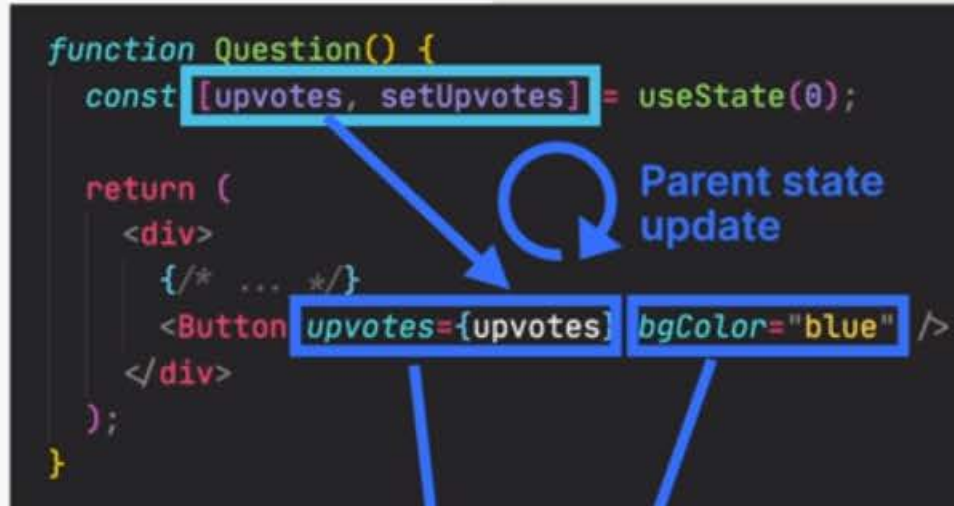
- 👉 External data, owned by parent component
- 👉 Similar to function parameters
- 👉 Read-only
- 👉 Receiving new props causes component to re-render. Usually when the parent's state has been updated

STATE VS. PROPS

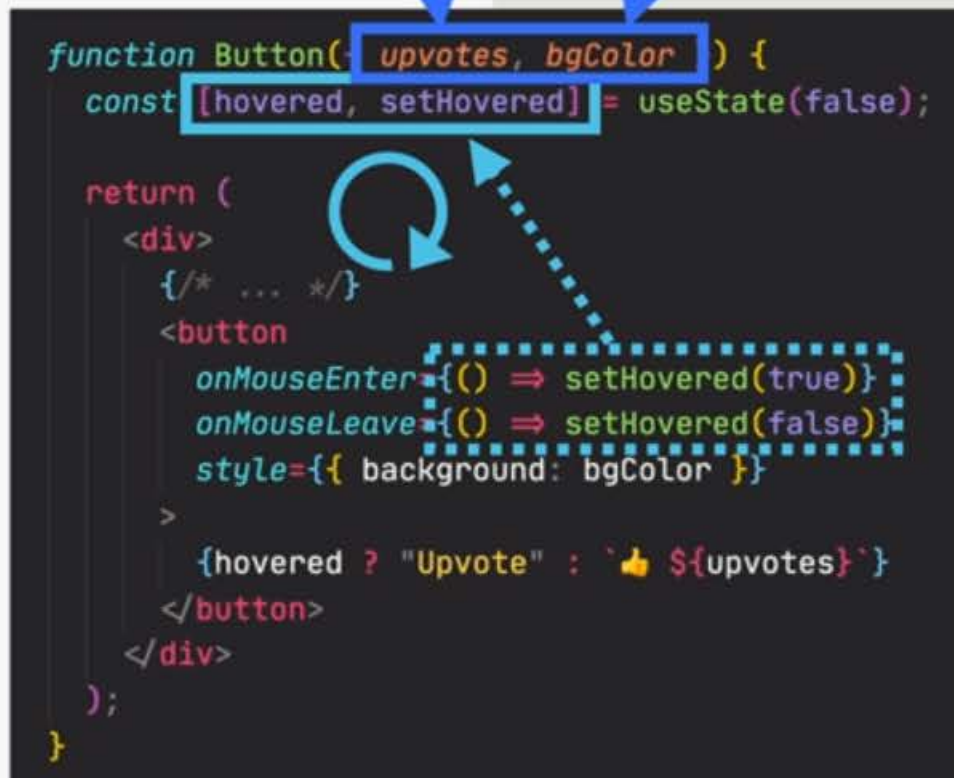
STATE

- 👉 Internal data, owned by component
- 👉 Component "memory"
- 👉 Can be updated by the component itself
- 👉 Updating state causes component to re-render
- 👉 Used to make components interactive

```
function Question() {  
  const [upvotes, setUpvotes] = useState(0);  
  
  return (  
    <div>  
      { /* ... */ }  
      <Button upvotes={upvotes} bgColor="blue" />  
    </div>  
  );  
}
```



```
function Button( upvotes, bgColor ) {  
  const [hovered, setHovered] = useState(false);  
  
  return (  
    <div>  
      { /* ... */ }  
      <button  
        onMouseEnter={() => setHovered(true)}  
        onMouseLeave={() => setHovered(false)}  
        style={{ background: bgColor }}  
      >  
        {hovered ? "Upvote" : `👍 ${upvotes}`}  
      </button>  
    </div>  
  );  
}
```



PROPS

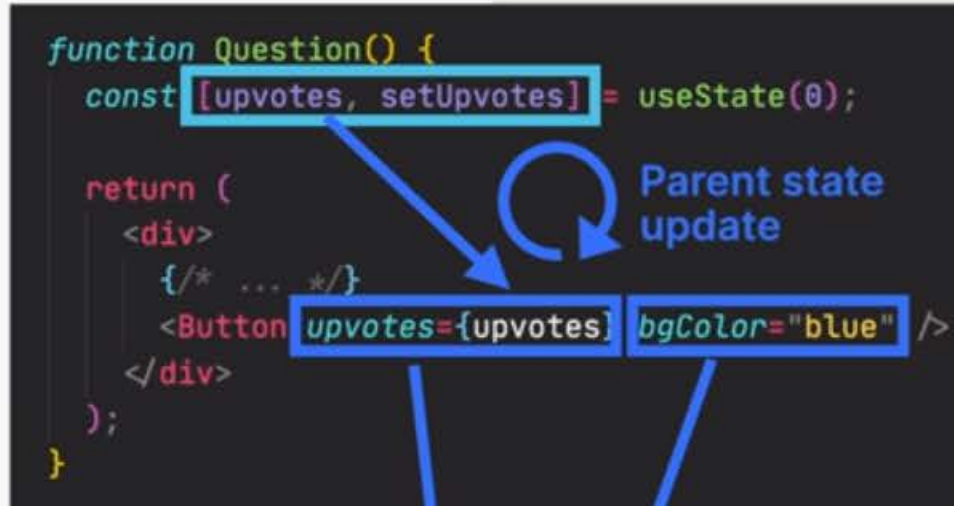
- 👉 External data, owned by parent component
- 👉 Similar to function parameters
- 👉 Read-only
- 👉 Receiving new props causes component to re-render. Usually when the parent's state has been updated

STATE VS. PROPS

STATE

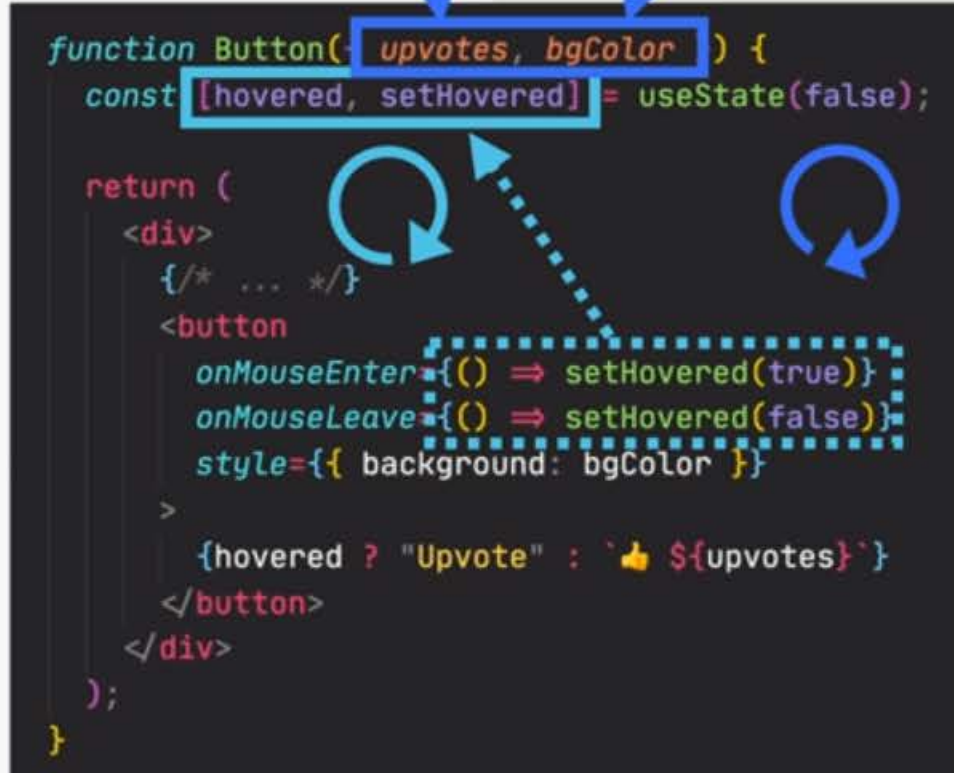
- 👉 Internal data, owned by component
- 👉 Component "memory"
- 👉 Can be updated by the component itself
- 👉 Updating state causes component to re-render
- 👉 Used to make components interactive

```
function Question() {  
  const [upvotes, setUpvotes] = useState(0);  
  
  return (  
    <div>  
      { /* ... */ }  
      <Button upvotes={upvotes} bgColor="blue" />  
    </div>  
  );  
}
```



A blue arrow points from the `upvotes` state variable to the `upvotes={upvotes}` prop in the `<Button>` component. A circular blue arrow next to the `setUpvotes` setter function is labeled "Parent state update".

```
function Button( upvotes, bgColor ) {  
  const [hovered, setHovered] = useState(false);  
  
  return (  
    <div>  
      { /* ... */ }  
      <button  
        onMouseEnter={() => setHovered(true)}  
        onMouseLeave={() => setHovered(false)}  
        style={{ background: bgColor }}  
      >  
        {hovered ? "Upvote" : `👍 ${upvotes}`}  
      </button>  
    </div>  
  );  
}
```



Two circular blue arrows are shown: one next to the `setHovered` setter function and another next to the `upvotes` prop. A dashed blue arrow points from the `upvotes` prop to the `setHovered` setter function.

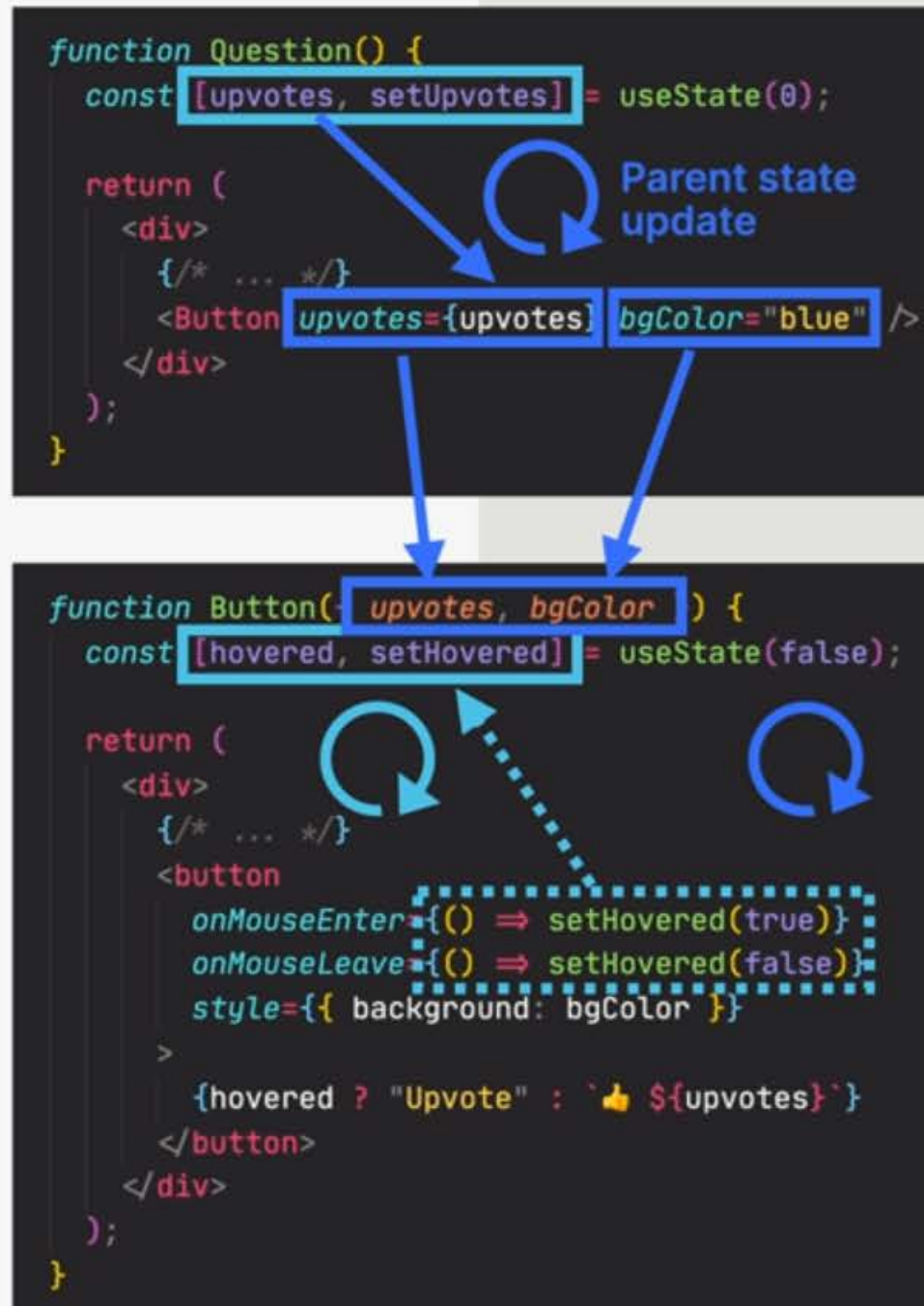
PROPS

- 👉 External data, owned by parent component
- 👉 Similar to function parameters
- 👉 Read-only
- 👉 Receiving new props causes component to re-render. Usually when the parent's state has been updated

STATE VS. PROPS

STATE

- 👉 Internal data, owned by component
- 👉 Component "memory"
- 👉 Can be updated by the component itself
- 👉 Updating state causes component to re-render
- 👉 Used to make components interactive



PROPS

- 👉 External data, owned by parent component
- 👉 Similar to function parameters
- 👉 Read-only
- 👉 Receiving new props causes component to re-render. Usually when the parent's state has been updated

STATE VS. PROPS

STATE

- 👉 Internal data, owned by component
- 👉 Component "memory"
- 👉 Can be updated by the component itself
- 👉 Updating state causes component to re-render
- 👉 Used to make components interactive

```
function Question() {  
  const [upvotes, setUpvotes] = useState(0);  
  
  return (  
    <div>  
      { /* ... */ }  
      <Button upvotes={upvotes} bgColor="blue" />  
    </div>  
  );  
}
```

```
function Button( upvotes, bgColor ) {  
  const [hovered, setHovered] = useState(false);  
  
  return (  
    <div>  
      { /* ... */ }  
      <button  
        onMouseEnter={() => setHovered(true)}  
        onMouseLeave={() => setHovered(false)}  
        style={{ background: bgColor }}  
      >  
        { hovered ? "Upvote" : `👍 ${upvotes}` }  
      </button>  
    </div>  
  );  
}
```

PROPS

- 👉 External data, owned by parent component
- 👉 Similar to function parameters
- 👉 Read-only
- 👉 Receiving new props causes component to re-render. Usually when the parent's state has been updated

STATE VS. PROPS

STATE

- 👉 Internal data, owned by component
- 👉 Component “memory”
- 👉 Can be updated by the component itself
- 👉 Updating state causes component to re-render
- 👉 Used to make components interactive

```
function Question() {  
  const [upvotes, setUpvotes] = useState(0);  
  
  return (  
    <div>  
      { /* ... */ }  
      <Button upvotes={upvotes} bgColor="blue" />  
    </div>  
  );  
}
```

```
function Button( upvotes, bgColor ) {  
  const [hovered, setHovered] = useState(false);  
  
  return (  
    <div>  
      { /* ... */ }  
      <button  
        onMouseEnter={() => setHovered(true)}  
        onMouseLeave={() => setHovered(false)}  
        style={{ background: bgColor }}  
      >  
        { hovered ? "Upvote" : `👍 ${upvotes}` }  
      </button>  
    </div>  
  );  
}
```

PROPS

- 👉 External data, owned by parent component
- 👉 Similar to function parameters
- 👉 Read-only
- 👉 Receiving new props causes component to re-render. Usually when the parent's state has been updated
- 👉 Used by parent to configure child component (“settings”)