# 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>
  );
}
```

```
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>
  );
}
```

```
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>
  );
}
```

```
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"

```
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

```react
function Question() {
  const [upvotes, setUpvotes] = useState(0);

  return (
    <div>
      {/* ... */}
      <Button upvotes={upvotes} bgColor="blue" />
    </div>
  );
}
```

```react
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
- 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>
  );
}
```

```
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>
  );
}
```

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>
  );
}
```

## 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>
  );
}
```

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>
  );
}
```

## 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>
  );
}
```

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>
  );
}
```

## 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>
  );
}
```

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>
  );
}
```

## 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>
  );
}
```

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>
  );
}
```

## 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>
  );
}
```

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>
  );
}
```

## 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")