

React Context



[Download Demo Code](#)

Goals

- Explain what context is
- Use the Context API to provide and consume context

Motivation

What is Context?

- Universal data across your application
- Data accessible across all components

Why is it useful?

- Prop drilling / tunneling
- Less repetition
- Useful for global themes, shared data

Creating context

demo/counter/src/countContext.js

```
import React from "react";

const CountContext = React.createContext();

export default CountContext;
```

This gives us a component:

- `<UserContext.Provider>` - allows you to provide a value to the context

Provider

demo/counter/src/CounterReadOnly.js

```
import React, { useState } from "react";
import Child from "../Child";
import CountContext from "../countContext";

function CounterReadOnly() {
  const [num, setNum] = useState(0);
  function up(evt) {
    setNum(oldNum => oldNum + 1);
  }

  return (
    <CountContext.Provider value={num}>
      <button onClick={up}>>+1 (from parent)</button>
      <Child />
    </CountContext.Provider>
  );
}
```

- Any component inside of a Provider can subscribe to context value.
- In order to subscribe to the value, we need the **useContext** hook.
- Without explicitly subscribing, the value isn't available to components farther down in the hierarchy.

useContext

demo/counter/src/GreatGrandReadOnly.js

```
import React, { useContext } from "react";
import CountContext from "../countContext";

function GreatGrandReadOnly() {
  const num = useContext(CountContext);

  return (
    <div>
      <p>I'm a great-grandchild!</p>
      <p>Here's the count: {num}</p>
    </div>
  );
}
```

- **useContext** looks for the nearest matching context, and reads its value.
- When the value inside of context changes, components subscribing to that context will re-render.
- Components that read from context with **useContext** are sometimes called *consumers* (as opposed to providers).

Setting State from a Consumer

We can also pass state-setting functions into providers, so that any component using context can potentially set state on an ancestor.

Example

demo/counter/src/CounterReadWrite.js

```
import React, { useState } from "react";
import Child from "../Child";
import CountContext from "../countContext";

function CounterReadWrite() {
  const [num, setNum] = useState(0);
  function up(evt) {
    setNum(oldNum => oldNum + 1);
  }

  return (
    <CountContext.Provider value={{ num, up }}>
      <Child />
    </CountContext.Provider>
  );
}
```

demo/counter/src/GreatGrandReadWrite.js

```
import React, { useContext } from "react";
import CountContext from "../countContext";

function GreatGrandReadWrite() {
  const { num, up } = useContext(CountContext);

  return (
    <div>
      <p>I'm a great-grandchild!</p>
      <p>Here's the count: {num}</p>
      <button onClick={up}>
        +1 (from great-grandchild)
      </button>
    </div>
  );
}
```

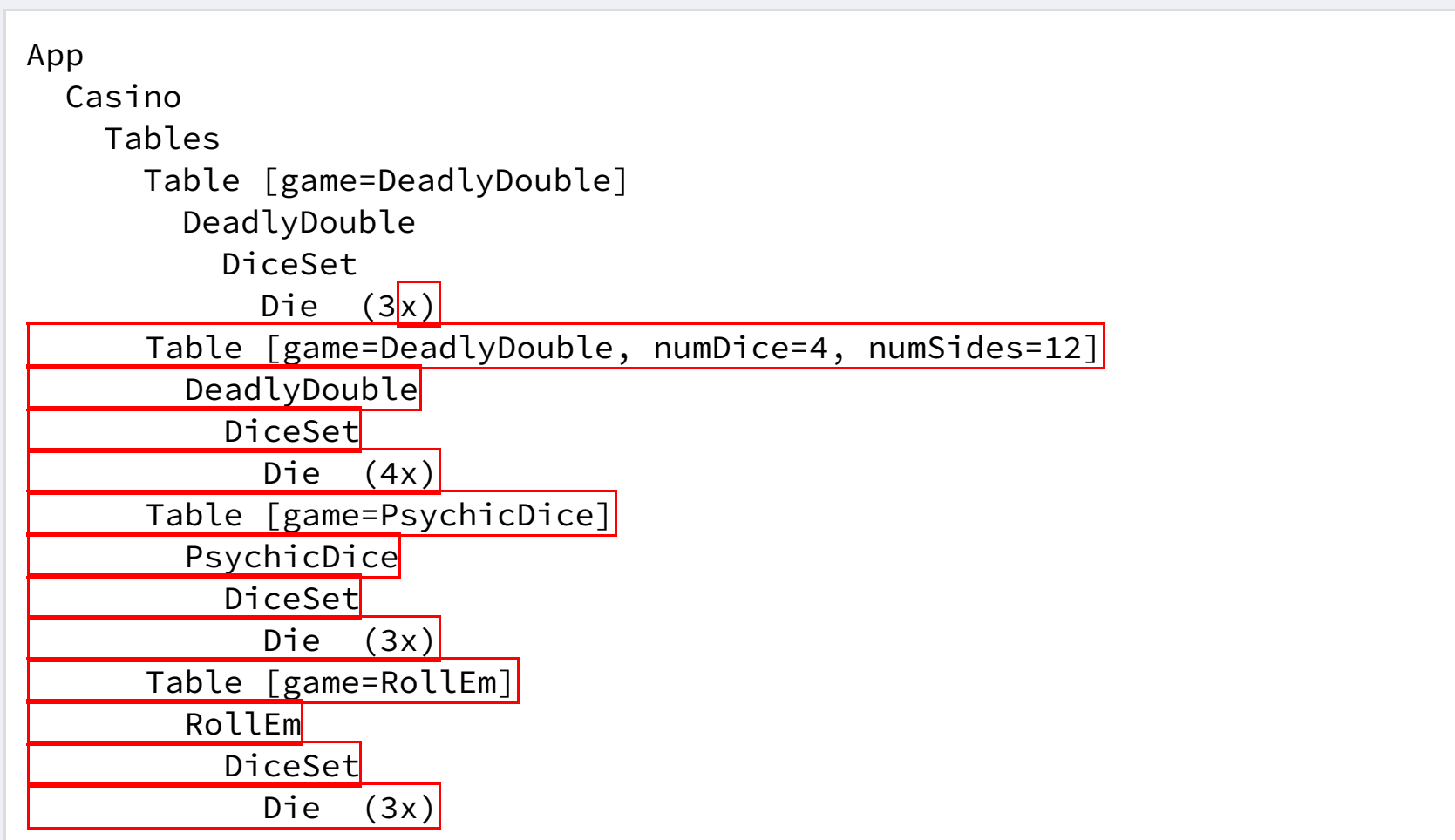
Demo Time

Deadly Doubles

A casino of different dice table games.

Try it out: <http://temp.joelburton.com/casino>

Our Components



React Features

- A generic component, **Table**, which can render different games.
- Another example of polymorphism: you could substitute **AltDie** for a different looking die in **DiceSet**, and everything works
- React's context manager: **Casino** lets you choose a favorite color, and the Die (several layers down) can access the color you chose.

Guidelines for When To Make a Component

- If I didn't, and inlined this in the parent component, would that make the parent state more complex?
 - Mixing together the state in the games [what are values of the dice] with the state in **Table** (how many wins/losses) would make things more complex
- Can I "not repeat myself"
 - Having **Table** lets us reduce repetition in the different game components.
- Might this component be usable elsewhere?
 - The **DiceSet** is useful in all of the games
- Might I want to swap it out?
 - Having the **Die** be a separate component makes it easier to replace it for **DieAlt**

Guidelines for When To Use Context

- Is this something created high-up, but needed far down, and the things in-between don't care about it?
 - The player choose a color once, in the **Casino**, but only the **Die**, far down, cares about it