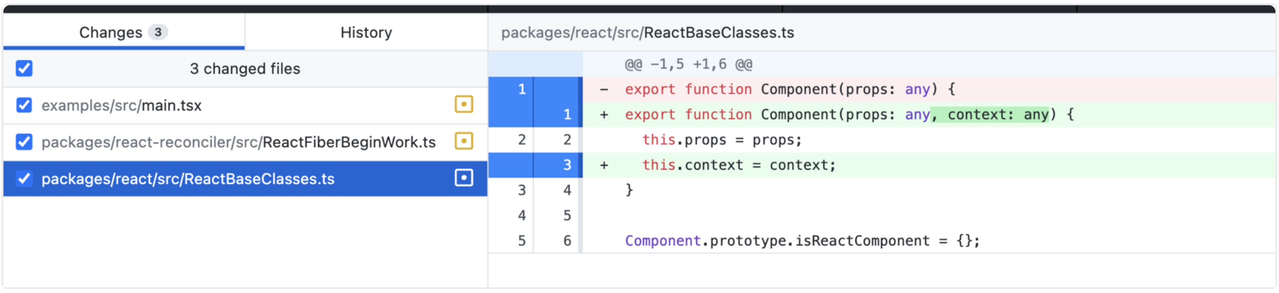
15-6 实现 contextType，掌握类组件对于 Context 消费方式的原理

示例

React TSX

```tsx
// !1. 创建context对象
const CountContext = createContext(100); // 默认值
const ThemeContext = createContext("red"); // 默认值

// !2. 创建Provider组件，用于向后代组件传递value
function FunctionComponent() {
  const [count, setCount] = useReducer((x) => x + 1, 0);

  return (
    <div className="border">
      <h1>函数组件</h1>
      <button onClick={() => setCount()}>{count}</button>

      {/* [green, count,count+1] */}
      <ThemeContext.Provider value="green">
        <CountContext.Provider value={count}>
          <CountContext.Provider value={count + 1}>
            <Child />
          </CountContext.Provider>
          <Child />
        </CountContext.Provider>
      </ThemeContext.Provider>
    </div>
  );
}

function Child() {
  // !3. 后代组件消费value，寻找的最近的匹配的Provider组件的value
  const count = useContext(CountContext);
  const theme = useContext(ThemeContext);
  return (
    <div className={"border " + theme}>
      <h1>Child</h1>

      <p>第一种消费方式：useContext</p>
      <p>{count}</p>

      <p>第二种消费方式：Consumer</p>
      <ThemeContext.Consumer>
        {(theme) => (
          <div className={theme}>
            <CountContext.Consumer>
              {(value) => <p>{value}</p>}
            </CountContext.Consumer>
          </div>
        )}
      </ThemeContext.Consumer>

      <p>第三种消费方式：contextType，只能消费单一的context来源</p>
      <ClassComponent />
    </div>
  );
}

class ClassComponent extends Component {
  static contextType = CountContext;
  render() {
    console.log("ClassComponent render");
    return (
        <div className="border">
          <h1>类组件</h1>
          <p>{this.context as number}</p>
        </div>
      );
    }
  }
```

实现



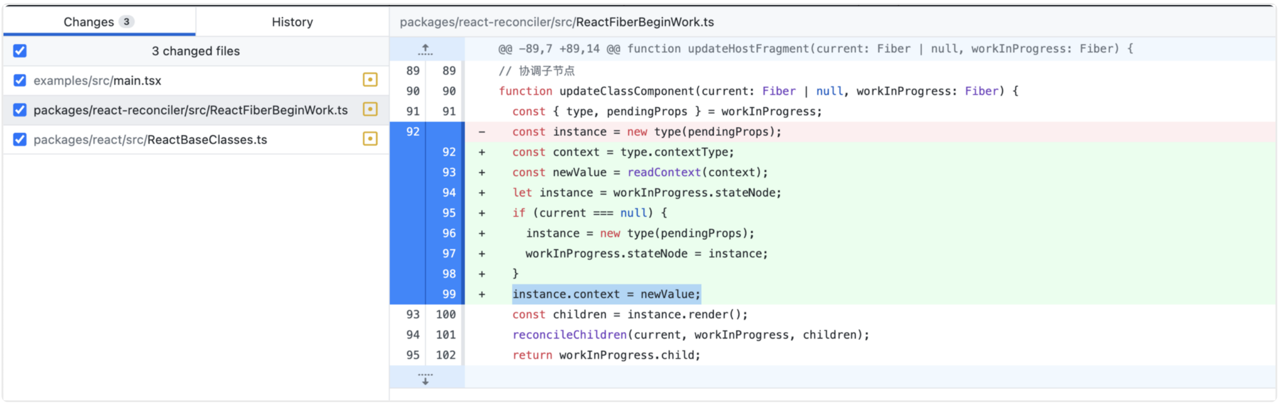
Changes 3 | History

3 changed files

- examples/src/main.tsx
- packages/react-reconciler/src/ReactFiberBeginWork.ts
- packages/react/src/ReactBaseClasses.ts

packages/react/src/ReactBaseClasses.ts

```
@@ -1,5 +1,6 @@
- export function Component(props: any) {
+ export function Component(props: any, context: any) {
    this.props = props;
+   this.context = context;
  }

  Component.prototype.isReactComponent = {};
```



Changes 3 | History

3 changed files

- examples/src/main.tsx
- packages/react-reconciler/src/ReactFiberBeginWork.ts
- packages/react/src/ReactBaseClasses.ts

packages/react-reconciler/src/ReactFiberBeginWork.ts

```
@@ -89,7 +89,14 @@ function updateHostFragment(current: Fiber | null, workInProgress: Fiber) {
  // 协调子节点
  function updateClassComponent(current: Fiber | null, workInProgress: Fiber) {
    const { type, pendingProps } = workInProgress;
-   const instance = new type(pendingProps);
+   const context = type.contextType;
+   const newValue = readContext(context);
+   let instance = workInProgress.stateNode;
+   if (current === null) {
+     instance = new type(pendingProps);
+     workInProgress.stateNode = instance;
+   }
+   instance.context = newValue;
    const children = instance.render();
    reconcileChildren(current, workInProgress, children);
    return workInProgress.child;
```