7-3 掌握不同类型组件的 Fiber：查看并调试

fiber 上有个属性 tag，用于标记组件类型，即描述的组件类型，如原生标签、函数组件、类组件、Fragment 等。

WorkTag

react/packages/react-reconciler/src/ReactWorkTags.js

```
Flow
export type WorkTag =
  | 0
  | 1
  | 2
  | 3
  | 4
  | 5
  | 6
  | 7
  | 8
  | 9
  | 10
  | 11
  | 12
  | 13
  | 14
  | 15
  | 16
  | 17
  | 18
  | 19
  | 20
  | 21
  | 22
  | 23
  | 24
  | 25
  | 26
  | 27;

export const FunctionComponent = 0;
export const ClassComponent = 1;
export const IndeterminateComponent = 2; // Before we know whether it
export const HostRoot = 3; // Root of a host tree. Could be nested ins
export const HostPortal = 4; // A subtree. Could be an entry point to
export const HostComponent = 5;
export const HostText = 6;
export const Fragment = 7;
export const Mode = 8;
export const ContextConsumer = 9;
export const ContextProvider = 10;
export const ForwardRef = 11;
export const Profiler = 12;
export const SuspenseComponent = 13;
export const MemoComponent = 14;
export const SimpleMemoComponent = 15;
export const LazyComponent = 16;
export const IncompleteClassComponent = 17;
export const DehydratedFragment = 18;
export const SuspenseListComponent = 19;
export const ScopeComponent = 21;
export const OffscreenComponent = 22;
export const LegacyHiddenComponent = 23;
export const CacheComponent = 24;
export const TracingMarkerComponent = 25;
export const HostHoistable = 26;
export const HostSingleton = 27;
```

ExamplePage

React JSX

```jsx
class ClassComponent extends Component {
  componentDidUpdate() {
    console.log("update"); // sy-log
  }

  componentWillUnmount() {
    console.log("unmount"); // sy-log
  }

  handle = () => {
    const { count } = this.state;
    this.setState({ count: count + 1 });
    this.setState({ count: count + 100 });
  };
  render() {
    return <div className="class border">{this.props.name}</div>;
  }
}

function Child() {
  useEffect(() => {
    return () => {
      console.log("unmount"); // sy-log
    };
  }, []);
  console.log("Child render"); // sy-log
  return <div>Child</div>;
}

function FunctionComponent(props) {
  const [count1, setCount1] = useReducer((x) => x + 1, 0);
  return (
    <div className="border">
      <p>{props.name}</p>
      {count1 > 3 ? (
        <div>
          <Child />
        </div>
      ) : (
        <section>
          <Child />
        </section>
      )}

      {count1 > 3 ? (
        <div>
          <ClassComponent name="函数内的类组件" />
        </div>
      ) : (
        <section>
          <ClassComponent name="函数内的类组件" />
        </section>
      )}

      <button
        onClick={() => {
          setCount1();
        }}
      >
        {count1}
      </button>
    </div>
  );
}

const jsx = (
  <div className="box border">
    <h1 className="border">omg</h1>
    123
    {/* <h1 className="border">omg2</h1>

    <h2>ooo</h2> */}
    {/* omg
    <div>
      bubucuo <a href="https://github.com/bubucuo"></a>
    </div> */}
    <FunctionComponent name="函数组件" />
    <ClassComponent name="class组件" />
    <>
      <h1>1</h1>
      <h1>2</h1>
    </>
  </div>
);

export default jsx;
```