12-8 子节点为 null、undefined、布尔值

TypeScript

```
function FunctionComponent() {
  const [count1, setCount1] = useReducer((x) => x + 1, 1);
  const [count2, setCount2] = useState(1);

  // const arr = count1 % 2 === 0 ? [0, 1, 2, 3, 4] : [0, 1, 2, 3];
  // const arr = count1 % 2 === 0 ? [0, 1, 2, 3, 4] : [0, 1, 2, 4];

  const arr = count1 % 2 === 0 ? [0, 1, 2, 3, 4] : [3, 2, 0, 4, 1];

  // old 0, 1, 2, 4
  // new 0, 1, 2, 3, 4
  // 1个before 4

  // old 3, 2, 0, 4, 1
  // new 0, 1, 2, 3, 4
  // 3个before null

  const _cls = count1 % 2 === 0 ? "red green_bg" : "green red_bg";

  // 0 删除
```

```
  return (
    <div className="border">
      <h3 className={_cls}>函数组件</h3>
      <button
        onClick={() => {
          setCount1();
        }}
      >
        {count1}
      </button>
      <button
        onClick={() => {
          setCount2(count2 + 1);
        }}
      >
        {count2}
      </button>
      <ul>
        {arr.map((item) => (
          <li key={"li" + item}>{item}</li>
        ))}
      </ul>

      {count1 % 2 === 0 ? <h1>null</h1> : null}
      {count1 % 2 === 0 ? <h1>undefined</h1> : undefined}
      {count1 % 2 === 0 && <h1>boolean</h1>}

      {/* {count1 % 2 === 0 ? (
        <button
          onClick={() => {
            setCount1();
          }}
        >
          {count1}
        </button>
      ) : (
        <span
          onClick={() => {
            setCount1();
          }}
        >
```

```
          react
        </span>
      )} */}
    </div>
  );
}
```

updateFromMap

TypeScript

```
function updateFromMap(
  existingChildren: Map<string | number, Fiber>,
  returnFiber: Fiber,
  newIdx: number,
  newChild: any
): Fiber | null {
  if (isText(newChild)) {
    const matchedFiber = existingChildren.get(newIdx) || null;
    return updateTextNode(returnFiber, matchedFiber, newChild + "");
  } else if (typeof newChild === "object" && newChild !== null) {
    const matchedFiber =
      existingChildren.get(newChild.key === null ? newIdx : newChild
      null;
    return updateElement(returnFiber, matchedFiber, newChild);
  }
  return null;
}
```

updateSlot

TypeScript

```
function updateSlot(
  returnFiber: Fiber,
  oldFiber: Fiber | null,
  newChild: any
```

```
) {
  // 判断节点是否可以复用
  const key = oldFiber !== null ? oldFiber.key : null;
  if (isText(newChild)) {
    if (key !== null) {
      // 新节点是文本，老节点不是文本
      return null;
    }
    // 有可能可以复用
    return updateTextNode(returnFiber, oldFiber, newChild + "");
  }

  if (typeof newChild === "object" && newChild !== null) {
    if (newChild.key === key) {
      return updateElement(returnFiber, oldFiber, newChild);
    }
  }
  return null;
}
```