Signals

Dynamic

Representative

- Solid

Logistic

- 声明signal更新数组
- setter时遍历更新数组

Pseudo Code

```
const [getArr, setArr] = createSignal([0, 1, 2])
const [getCount, setCount] = createSignal(0)

getArr().forEach(i => {
  const div = document.createElement("div")
  addSignalFunc(idOrTag, setCount, () => {
    div.textContent = getCount()
  })
})

// When setCount
coutSignalFunc.forEach(func => func())

// When arr change and some elements need to unmount
removeSignalFunc(idOrTag)
```

Pro

- 自由添加
- 逻辑清晰
- 组件级递归简单

Con

- Arrow Function占内存（有针对性优化办法）
- If 或者For这些control flow在对应元素卸载时需要手动起初更新数组中对应的更新项

Static

Representative

- Svelte4

- DLight

Logistic

- 通过compiler静态生成所有的更新函数
- 不再维护数组动态添加

Pseudo Code

```
const [getArr, setArr] = createSignal([0, 1, 2])
const [getCount, setCount] = createSignal(0)

forNode.update = (div) => {
  div.textContent = getCount()
}

const update = () => {
  forNode.update()
}

// When setCount
update()

// When arr change and some elements need to unmount
// Nothing to do
```

Pro

- 内存占用小，无更新数组维护
- 不需要手动cleanup

Con

- compiler实现较难
- 编译出的代码和原始代码差别很大
- 超出scope的情况会很复杂，更适合于template和class语法