DOM 本质

DOM 的本质是一颗 DOM 树，树形结构。

property 和 attr(attribute) 的区别?

两则都有可能引起 DOM 的重新渲染。

1. 数据类型：Attribute是字符串类型，主要用于描述元素的特性，比如id、class、name等。而Property可以是任何数据类型，如数值、布尔值、对象等，主要用于表示元素的状态和行为。
2. 读写方式：`Attribute` 可以通过 `Element` 的 `getAttribute` 方法读取，通过 `Element` 的 `setAttribute` 方法写入。而 `Property` 可以通过 `Element` 的属性直接读取和写入，如 `element.innerHTML`、`element.disabled` 等。
3. 浏览器识别：浏览器解析 `HTML` 代码时，会识别并处理`Attribute`，但对于 `Property`，浏览器需要使用 `JavaScript` 来获取和设置。

```
// property
const pList = document.querySelectorAll('p');
const p = pList[0];

console.log(p.style.color); // 获取样式
p.style.color = "#333";  // 修改样式

console.log(p.className); // 获取 class
p.className = "p1";  // 修改 class

console.log(p.nodeName); // "P"
console.log(p.nodeType); // 1
```

```
// attribute
const pList = document.querySelectorAll('p');
const p = pList[0];

p.getAttribute("data-name"); // 获取第一个 p 元素的 data-name 属性
p.setAttribute("data-name", "CSS");  // 设置第一个 p 元素的 data-name 属性

p.getAttribute("style"); // 获取第一个 p 元素的 style 属性
p.setAttribute("style", "font-size: 30px");  // 设置第一个 p 元素的 font-size 属性
```

DOM 操作性能优化

DOM 查询和 DOM 操作很耗费性能。

- 将 DOM 查询做缓存。浏览器自身不能缓存 DOM 查询，因为 JS 可能在查询 DOM 后又修改了 DOM，导致缓存的 DOM 不准确。

- 将多次操作 DOM 改为一次操作。

```
// 将 DOM 查询做缓存
const pList = document.getElementsByTagName("p");
const len = pList.length;

for (let i = 0; i < len; i++) {
    // TODO
}
```

上面代码，将获取 DOM 操作放在循环外面，而不放在函数循环体中，避免多次获取 DOM 节点，造成性能浪费。

```
// 将多次操作 DOM 改为一次操作

// 先定义一个文档片段，这个片段还没被放入 DOM 结构中，且这时候不会被渲染
const frag = document.createDocumentFragment();

// 往 frag 中插入一些节点
for (let i = 0; i < 10; i++) {
    const item = document.createElement("li");
    item.innerHTML = `item ${i}`;
    frag.appendChild(item);
}

// 最后将 frag 插入 DOM 结构中，渲染 DOM ，完成一次操作
document.body.appendChild(frag);
```