ParentNode 接口和 ChildNode 接口

节点对象除了继承 Node 接口以外，还拥有其他接口。ParentNode 接口表示当前节点是一个父节点，提供一些处理子节点的方法。ChildNode 接口表示当前节点是一个子节点，提供一些相关方法。

1. ParentNode 接口

如果当前节点是父节点，就会混入了（mixin）ParentNode 接口。由于只有元素节点（element）、文档节点（document）和文档片段节点（documentFragment）拥有子节点，因此只有这三类节点会拥有 ParentNode 接口。

1.1. ParentNode.children

children 属性返回一个 HTMLCollection 实例，成员是当前节点的所有元素节点。该属性只读。

```
let children = document.body.children;
children  // HTMLCollection(12) [...]

for(let i = 0, len = children.length; i < len; i++) {
    console.log(children[i]);
}
// <script>...</script>
// ...

children[10] // <script>...</script>
typeof children[10]  // "object"
children[10] instanceof HTMLScriptElement  // true
HTMLScriptElement.__proto__ === HTMLElement  // true
```

- <script>...</script> 是 HTMLScriptElement 的实例。
- HTMLScriptElement 的原型对象是 HTMLElement。

1.2. ParentNode.firstElementChild

firstElementChild 属性返回当前节点的第一个元素子节点。如果没有任何元素子节点，则返回 null。

```
document.firstElementChild.nodeName
// "HTML"
```

1.3. ParentNode.lastElementChild

lastElementChild 属性返回当前节点的最后一个元素子节点。如果没有任何元素子节点，则返回 null。

```
document.lastElementChild  // <html>...</html>
document.firstElementChild.firstElementChild  // <head>...</head>
```

```
document.firstElementChild.lastElementChild  // <body>...</body>
```

1.4. ParentNode.childElementCount

`childElementCount` 属性返回一个整数，表示当前节点的所有元素子节点的数目。如果不包含任何元素子节点，则返回0。

```
document.body.childElementCount  // 13
document.body.lastElementChild.lastElementChild.childElementCount  // 0

document.body.childElementCount === document.body.children.length // true
```

1.5. ParentNode.append(), ParentNode.prepend()

(1) ParentNode.append()

`append()` 方法为当前节点追加一个或多个子节点，位置是最后一个元素子节点的后面。

该方法不仅可以添加元素子节点（参数为元素节点），还可以添加文本子节点（参数为字符串）。

```
let parent = document.body;

// 添加元素子节点
let p = document.createElement('p');
parent.append(p);

// 添加文本子节点
parent.append('Hello');
```

上面代码先向 `body` 元素节点添加了 `p` 元素节点，然后向 `body` 元素节点添加了 `Hello` 文本节点。DOM 结构如下所示：


```
// 添加多个元素子节点
let p1 = document.createElement('p');
let p2 = document.createElement('p');
parent.append(p1, p2);

// 添加元素子节点和文本子节点
let p = document.createElement('p');
parent.append('Hello', p);
```

该方法没有返回值。

该方法与 `Node.prototype.appendChild()` 方法有三点不同。

- `append()` 允许字符串作为参数，`appendChild()` 只允许子节点作为参数。
- `append()` 没有返回值，而 `appendChild()` 返回添加的子节点。
- `append()` 可以添加多个子节点和字符串（即允许多个参数），`appendChild()` 只能添加一个节点（即只允许一个参数）。

(2) ParentNode.prepend()

`prepend()` 方法为当前节点追加一个或多个子节点，位置是第一个元素子节点的前面。它的用法与 `append()` 方法完全一致，也是没有返回值。

2. ChildNode 接口

如果一个节点有父节点，那么该节点就拥有了 `ChildNode` 接口。

2.1. ChildNode.remove()

`remove()` 方法用于从父节点移除当前节点。

```
document.body.lastElementChild.remove()  // 删除 body 节点中最后一个元素节点
document.body.lastChild.remove()  // 删除 body 节点中最后一个子节点（可能是文本节点、注释节点）
```

2.2. ChildNode.before(), ChildNode.after()

(1) ChildeNode.before()

`before()` 方法用于在当前节点的前面，插入一个或多个同级节点。两者拥有相同的父节点。

该方法不仅可以插入元素节点，还可以插入文本节点。

```
let footer = document.getElementsByClassName("footer")[0];
let p1 = document.createElement("p");
let div1 = document.createElement("div");
let span1 = document.createElement("span");
let section = document.createElement("section");
let str1 = "Hello";
let str2 = "World";
let str3 = "DOM";
let str4 = "ChildNode.before";

footer.before(p1);
footer.before(div1, span1);
footer.before(str1);
footer.before(str2, str3);
footer.before(section, str4);
```

上面代码步骤如下：

1. 取出类名为 `footer` 的第一个元素。
2. 在其前面插入 `p` 元素节点。
3. 在其前面插入 `div` 和 `span` 元素节点。
4. 在其前面插入 `Hello` 文本节点。
5. 在其前面插入 `World` 文本节点和 `DOM` 文本节点。
6. 在其前面插入 `section` 元素节点和 `ChildNode.before` 文本节点。

最后结果如下：

(2) ChildNode.after()

`after()` 方法用于在当前节点的后面，插入一个或多个同级节点，两者拥有相同的父节点。用法与 `before()` 方法完全相同。

ChildNode.replaceWith()

`replaceWith()` 方法使用参数节点，替换当前节点。参数可以是元素节点，也可以是文本节点。

```
let footer = document.getElementsByClassName("footer")[0];
let aside = document.createElement('aside');
footer.replaceWith(aside)
```

上面代码使用 `aside` 元素节点替换了 `footer` 元素节点。

结果如下：

上一步在 `footer` 元素节点前面插入的那些节点下面，变成了 `aside` 节点。这说明 `footer` 元素节点被替换了。