Lightning Web Component（一）

Web Component是一组新的Api，它建立在被浏览器广泛采用的Web标准之上(请参阅webcomponents.org上的浏览器支持)。它们允许开发人员制作灵活的自定义组件——但是这种灵活性带来了责任。在这篇由两部分组成的博客中，我们将概述web组件是什么以及它们具有的特定的可访问性考虑因素，这样您就可以将web组件集成到自己的产品中，同时考虑到所有用户。您可以阅读第二篇博客，全部关于Web组件的可访问性。

Web Component允许开发人员使用本地HTML和JavaScript制作自己的定制组件。它们由三部分组成：

- Custom elements
- Html templates
- Shadow DOM

Salesforce的Lightning Web Component(LWC)组件框架构建在Web Component之上，使创建快速、轻量级组件变得容易。让我们研究一个示例web Component，看看如何最好地利用它们。

Custom elements

Lightning Web Component提供一个很棒的自定义标签，它拓展了现有的html元素，可以直接使用在html模板中。

```
class CustomButton extends HTMLElement {
  constructor() {
    super();
  }
}

window.customElements.define('custom-button', CustomButton);
```

HTML templates

有两种形式，分别是`templates`和 `slots`元素组件

▼ Templates

默认情况下，模板的display= " none "可以被JavaScript引用，这使得它们很适合HTML，这些HTML将在你的组件中重用。

为了开始构建组件，在DOM(文档对象模型)中添加了一个模板标记'template'，并在其中添加了一个按钮。然后，在构造函数中，将模板的内容附加到定制元素本身。

```
let myTemplate = document.createElement('template');
myTemplate.innerHTML = `<button>
  <span>Default text</span>
</button>`;

class CustomButton extends HTMLElement {
  constructor() {
    super();

    this.appendChild(myTemplate.content.cloneNode(true));
    this.button = this.querySelector('button');
    this.updateText = this.updateText.bind(this);//this:
  }

  connectedCallback() {
    if (this.hasAttribute('text') this.updateText();
  }

  updateText() {
    let buttonSpan = this.button.querySelector('span');
    buttonSpan.innerText = this.getAttribute('text');
  }
```

```
}

window.customElements.define('custom-button', CustomButton);
```

按钮模板内部有一个带有默认文本的span，用户可以通过将字符串传递给带有text属性的自定义元素来替换该文本。

添加了一个connectedCallback函数，这是一个web组件生命周期函数，在组件连接到DOM时发生。在该函数中，我将按钮的innerText设置为自定义组件传递的值。即：

```
<custom-button text="Click me!"></custom-button>
```

在浏览器中的代码便可以写成这样：

```
<custom-button text="Click me!">
  <button>Click me!</button>
</custom-button>
```

▼ slots

`slots`允许灵活性，因为它们允许你在里面放任何东西。如果您需要允许组件的使用者添加自定义HTML，这一点特别有用。要记住的一件事是`slots`需要`shadow DOM`能够正确工作。

对于以上自定义按钮组件，可能想要添加一个icon这样我可以使用`slots`，即：

```
<button>
  <slot name="icon"></slot>
  <span>Default text</span>
</button>
```

Shadow DOM

shadow DOM是一个限定了作用域的document对象模型树，它只存在于自定义元素中。如果在你的组件中启用了shadow DOM，那么组件的元素就会在与页面其他部分分

开的树中。

- **No Shadow vs Open vs Closed**

Web Component不需要启用shadow DOM，但如果启用了shadow DOM，它就可以打开或关闭。

如果不打开Shadow DOM：该组件位于主DOM中。页面上的JavaScript和CSS会影响组件的内容。

```
<custom-button>
  <button>Default text</button>
</custom-button>
```

如果打开Shadow DOM：主DOM不能以传统的方式访问子树，但是您仍然可以使用Element.shadowRoot访问子树。文档。getElementById、其他查询选择器和来自组件外部的CSS不会影响它。

```
<custom-button>
  #shadow-root (open)
   <button>Default text</button>
</custom-button>
```

如果关闭Shadow DOM： 主DOM根本不能访问组件内部的元素。组件外部的JavaScript和CSS不会影响它。

为了查看定制按钮示例的源代码，像这样启用了打开Shadow DOM:

```
let myTemplate = document.createElement('template');
myTemplate.innerHTML = `
  <button>
    <slot name="icon"></slot>
    <span>Default text</span>
  </button>
`;

class CustomButton extends HTMLElement {
  constructor() {
```

```
    super();
    let shadowRoot = this.attachShadow({ 'mode': 'open' });
    shadowRoot.appendChild(myTemplate.content.cloneNode(true));
  }
}

window.customElements.define('custom-button', CustomButton);
```

模板的内容现在被添加到shadow root，而不是直接添加到自定义元素。

最终完成的自定义按钮：

当鼠标点击按钮时，切换aria-pressed属性，以便用户知道是否按下了它。

```
let myTemplate = document.createElement('template');
myTemplate.innerHTML = `
  <button>
    <slot name="icon"></slot>
    <span>Default text</span>
  </button>
`;

class CustomButton extends HTMLElement {
  constructor() {
    super();
    let shadowRoot = this.attachShadow({ 'mode': 'open' });
    shadowRoot.appendChild(myTemplate.content.cloneNode(true));

    this.button = this.shadowRoot.querySelector('button');
    this.handleClick = this.handleClick.bind(this);
    this.updateText = this.updateText.bind(this);
  }

  get ariaPressed() {
    const value = this.button.getAttribute('aria-pressed');
    return (value === 'true');
  }

  set ariaPressed(value) {
    this.button.setAttribute('aria-pressed', value);
  }

  connectedCallback() {
    this.button.addEventListener('click', this.handleClick);
    if (this.hasAttribute('text')) this.updateText();
  }

  handleClick() {
```

```
    this.ariaPressed = !this.ariaPressed;
  }

  updateText() {
    let buttonSpan = this.button.querySelector('span');
    buttonSpan.innerText = this.getAttribute('text');
  }
}

window.customElements.define('custom-button', CustomButton);
```

最终功能展示及说明：

完整代码： https://codepen.io/shleewhite/pen/PowqZzM?editors=1010

- get ariapress:返回自定义按钮元素内的按钮上aria-pressed属性的值。
- set ariappressed:设置自定义按钮元素内的按钮的aria按下属性的值。
- connectedCallback:在组件连接到DOM时添加一个onClick监听器。
- handleClick:切换按钮被点击时的ariappressed值。

然后我们可以用以下代码，添加该自定义按钮到html

```
<custom-button id="important-button" text="Click me!"></custom-button>
```

可以编程设置ariappressed属性像这样:

```
document.getElementById('important-button').ariaPressed = true;
```

现在我们有了一个带有ariapress属性的按钮组件，可以用JavaScript进行设置。该组件将自定义元素、HTML模板和阴影DOM都与纯JavaScript和HTML结合在一起，不需要框架