HTMLUnknownElement与HTML5自定义元素的故事

这篇文章发布于 2018年03月20日，星期二，23:47，归类于 HTML相关。 阅读 6299 次, 今日 6 次 5 条评论

一、了解HTMLUnknownElement元素

在网页中，随便写一个标签，例如：

```
<username>zhangxinxu</username>
```

这个 `<username>` 就是一个HTMLUnknownElement元素。

简单测试下：

```
document.querySelector('username') instanceof HTMLUnknownElement;    // 返回值是true
```

在HTML规范中，HTMLUnknownElement元素是一个被认可的合法的元素，CSS可以无障碍使用，例如：

```
username {
    text-transform: uppercase;
}
```

则实时效果如下（用户名大写）：

ZHANGXINXU

HTMLUnknownElement继承HTMLElement中的方法，因此，基本上，常用的HTML方法都是可以畅快使用的，例如，文字变红色，可以直接：

```
document.querySelector('username').style.color = 'red';
```

我们不一样

在HTML世界中，HTMLUnknownElement和HTMLDivElement，HTMLSpanElement等等都是平级的，平起平坐，都是HTMLElement的子集。那其中有没有什么不一样的地方呢？

区别在于，规范中的一部分HTML元素自己带有一些特殊的属性或者方法，例如，表单元素HTMLFormElement元素有 `reset()` 方法， `novalidate` 属性。然而，HTMLUnknownElement自己是没有携带任何属性和方法。

这使其是一件好事，我们就可以为HTMLUnknownElement扩展非常私有的方法，而不用担心会影响其他元素。例如来说，默认所有HTMLUnknownElement元素的 `display` 计算值都是 `inline` ，我们可以扩展了 `block()` 方法使其块状化。

```
HTMLUnknownElement.prototype.block = function () {
    this.style.display = 'block';
};
```

此时执行`document.querySelector('username').block()`就可以让`<username>`元素块状化了。

从实用角度讲，基于原型扩展的方法，还不算太智能，要是可以针对不同标签类型进行扩展就更好了，这个后面会介绍。

二、HTMLUnknownElement与自定义元素（Custom Elements）

我之前以为写一个规范以外的标签元素就是自定义元素，后来发现不是的。HTMLUnknownElement与自定义元素并不能直接相等，甚至可以说是陌路两人。

W3规范中，对自定义元素的定义是中间必须要有短横线（就是键盘上的减号）连接，并且浏览器也是这么认为的，例如：

```
document.createElement('username') instanceof HTMLUnknownElement;    // 返回值是true
document.createElement('user-name') instanceof HTMLUnknownElement;    // 返回值是false
```

```
> document.createElement('user-name') instanceof HTMLUnknownElement
< false
> document.createElement('username') instanceof HTMLUnknownElement;
< true
```

从这一点看，HTMLUnknownElement一定不是自定义元素，换句通俗的话解释就是“自定义元素不等于随便定义元素”。

虽然都是自己命名的标签，多一个短横和没有短横去区别之大，远不是外面看上去的那点区别。

规范和浏览器为有短横的自定义元素开了很多很棒的特权，可以让我们实现很多很棒的事情！

三、自定义元素（Custom Elements）专场

自定义元素有哪些特权呢？我们先从目前可以在实际项目中应用的特性说起。

1. ES6下的继承与自定义HTML元素类型

对于自定义元素，规范提供了一套各种HTML特性可继承可扩展的机制，通常使用套路如下：

1. ES6 class继承；
2. customElements定义元素；

先说说目前支持相对较好的匿名自定义元素（Autonomous custom elements），也就是继承自HTMElement的用法。

例如，我们实现一个基于`rows`属性多行打点效果的小组件。

注意，下面的演示代码基本上可以作为各类自定义元素（甚至Web Components）使用的模板，很有用，例如，对于学习类似Vue的实时刷新很有帮助，对于学习Shadow DOM和Web Components也是非常好的案例。以后要实现类似功能，代码拷贝过去，修改修改即可！

```
class HTMLEllElement extends HTMLElement {
  // 指定观察的属性，这样attributeChangedCallback才会起作用
  static get observedAttributes() { return ['rows']; }
```

```
constructor() {
  // constructor中首先第一件事情就是调用 super
  // super指代了整个prototype或者__proto__指向的对象
  // 这一步免不了的
  super();

  // 创建shadow元素，实际上，从本例要实现的效果讲，
  // 直接元素上设置也可以，就是HTML丑了点，CSS要放在外部
  // 且目前火狐并不支持shadow dom可以不用，
  // 但一切为了学习，还是展现下现代web组件的实现方式
  var shadow = this.attachShadow({
      // open外部可访问（通过element.shadowRoot），closed则不能
      mode: 'open'
  });

  // 文本内容移动到shadow dom元素中
  var div = document.createElement('div');
  div.innerHTML = this.innerHTML;
  this.innerHTML = '';
  var style = document.createElement('style');
  shadow.appendChild(style);
  shadow.appendChild(div);
}

// 下面4个方法为常用生命周期
connectedCallback() {
  console.log('自定义元素加入页面');
  // 执行渲染更新
  this._updateRendering();
}
disconnectedCallback() {
  // 本例子该生命周期未使用，占位示意
  console.log('自定义元素从页面移除');
}
adoptedCallback() {
  // 本例子该生命周期未使用，占位示意
  console.log('自定义元素转移到新页面');
}
attributeChangedCallback(name, oldValue, newValue) {
  console.log('自定义元素属性发生变化');
  this._rows = newValue;
  // 执行渲染更新
  this._updateRendering();
}

// 设置直接get/set rows属性的方法
get rows() {
  return this._rows;
}
set rows(v) {
  this.setAttribute('rows', v);
}

_updateRendering() {
  // 根据变化的属性，改变组件的UI
  var shadow = this.shadowRoot;
  var childNodes = shadow.childNodes;
  var rows = this._rows;
```

```
    for (var i = 0; i < childNodes.length; i++) {
      if (childNodes[i].nodeName === 'STYLE') {
        childNodes[i].textContent = `div {
          display: -webkit-box;
          -webkit-line-clamp: ${rows};
          -webkit-box-orient: vertical;
          overflow: hidden;
        }`;
      }
    }
  }
}
// 定义x-ell标签元素为多行打点元素
customElements.define('x-ell', HTMLEllElement);
```

上面代码看上去很长，实际上，也就2部分，一个`class`继承，一个`customElements.define`注册。个中细节参见注释，是非常好的学习案例。

那实现了什么效果呢?

很棒的效果，我们直接在页面中写入如下CSS和HTML：

```
x-ell {
  display: block;
}
```

```
<x-ell rows="2">对于现代浏览器，例如webkit内核的浏...组合如下。</x-ell>
```

这段文字如果超过2行，就会自动在末尾打点。可以看到，页面上没有任何关于打点相关CSS代码的设置，因为全部在自定义HTMLEllElement的时候写在Shadow DOM中了。

如果想要3行打点，也非常简单，直接设置`rows`为`3`即可。我们可以直接手动修改：

```
<x-ell rows="3">
```

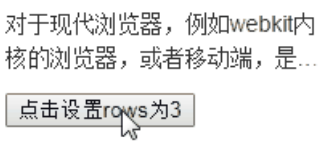
也可以直接一行JS直接修改属性：

```
document.querySelector('x-ell').rows = '3';
```

浏览器会自动渲染成3行文字打点，这就是现代Web Components组件的模样。

眼见为实，您可以狠狠地点击这里：HTML5自定义元素与rows属性直接控制几行打点demo

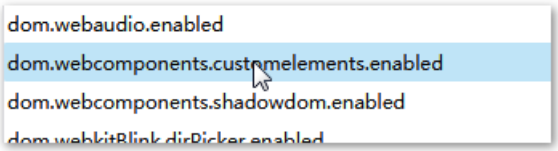
效果如下Gif截屏所示，点击按钮设置`rows`为`3`，然后3行打点：



`rows`属性的表现就好像`<textarea>`元素的`rows`属性一样，修改后直接触发了元素本身UI的变化，而且是实时的。这就是自定义元素，自己定义一个元素，可以和原生元素一样的行为和特性，是不是很酷！

然而兼容性问题不容忽视，如果只考虑ES6继承特性，Firefox，Safari等浏览器也是可以用的，也就是移

动端冒进下也是可以使用的。但是自定义元素注册，以及Shadow DOM等特性目前就Chrome，Android以及UC等浏览器支持，因此，只能用在一些内部产品（如中后台管理系统、内部工具）上。Firefox目前想要支持可以开启实验功能，about:config，然后设置`dom.webcomponents.customelements.enabled`为`true`，以及`dom.webcomponents.shadowdom.enabled`为`true`。



不过，目前Firefox在开发，Edge在考虑，相信一统江山的时候很快就会到来。

小疑问

`customElements.define`自定义元素的时候，标签可否没有短横线，例如：

```
customElements.define('ell', HTMLEllElement);
```

答案是，不可以，非短横线自定义元素不认为是合法元素，于是会报错。

下面再看来下目前还没有浏览器支持，但即将支持的**定制内置元素**（Customized built-in elements）。

什么意思呢？上面自定义元素我们都是继承于HTMLElement，实际上，HTMLElement还有非常非常都的内置子集元素，例如上面提到的HTMLDivElement，HTMLSpanElement，HTMLFormElement元素等，每一种类型标签几乎都对应一种内置元素。

所谓“定制内置元素”，指的就是我们的自定义元素继承自这些内置元素。

举个例子，我们希望定义一个继承`<form>`元素的自定义元素，名为`custom-form`，可以这么处理：

```
class HTMLCustomFormElement extends HTMLFormElement { /* 略 */ }
customElements.define('custom-form', HTMLCustomFormElement);
```

此时，元素`<custom-form>`就有了原生`<form>`的各种属性和方法，例如直接可以使用`reset()`方法重置内部表单元素的值。

这种特性在我们实际开发的时候有什么用呢？

举例来说，在HTML标准中，`<form>`元素是不能相互嵌套的，这就导致一个问题，当我们需要局部重置表单内的某些属性值的时候，就不能使用`reset()`方法，因为会误伤其他可能已经输入的值。

例如表单中有很多输入信息，外加一个图片上传。需求是图片选择即上传完毕，此时需要在图片Ajax上传完毕后重置`file`类型`input`的值，IE下值重置不像Chrome，可以直接设置`value`为空，最佳做法直接`<form>`元素的`reset()`方法，此时，我们就可以在`file`类型`input`外面包一层`<custom-form>`标签，这样，HTML解析时候既没有嵌套问题，又可以使用`reset()`方法对表单元素进行重置。

当然，上面的分析只是理论上的判断，由于目前没有浏览器支持定制内置元素，因此无法断定是否真的如此，等回头浏览器支持了，我们再看一看究竟。

2. 自定义元素与HTML import引入

自定义元素还可以在HTML模块中使用，目前仅Chrome支持。

大致套路这样的：

1. HTML模块注册与构建自定义元素；
2. 母页面引入模块；
3. 母页面自定义标签自动组件呈现；

例如下面HTML：

```
<link rel="import" href="module.html">
<zxx-info/>
```

此时 `<zxx-info>` 这个元素在页面上呈现出来的效果就是：

那这个module.html究竟做了什么事情呢？就是自定义 `<zxx-info>` 这个元素。

完整代码如下：

```
<template id="tpl">
  <style>
  .scope {
    contain: content;
  }
  .scope > img {
    float: left;
    margin-right: 10px;
  }
  .scope > p {
    margin: 0;
    overflow: hidden;
  }
  </style>
  <div class="scope">
    <img src="zxx.jpg">
    <p>帅哥一枚！</p>
  </div>
</template>
<script>
  // 定义<zxx-info>
  class HTMLZxxInfoElement extends HTMLElement {
    constructor() {
      super();
      // 内部显示信息
      this.innerHTML = document.currentScript.ownerDocument.querySelector('#tpl').innerHTM
L;
    }
  };
  // 注册
  customElements.define('zxx-info', HTMLZxxInfoElement);
</script>
```

两部分，一部分模板，一部分自定义元素定义和注册。都是Web Components中的概念。不展开，大家感受感受即可。

眼见为实，您可以狠狠地点击这里：HTML5自定义元素与HTML import模块引入demo

四、结束语

如果我们平时就使用一些自定义标签定义样式，HTMLUnknownElement用用足矣，简单方便又灵活，什么不符合W3C规范之类的，完全不用在意，规范的哈，你看都有特别的身份标识了，没必要像张灵玉那样过分拘泥，可以学学张楚岚，那才是真智慧。

但是如果你是希望使用自定义标签来开发Web Components组件，得了，标签的短横线还是要加上的。考虑到目前业界几乎没有大规模使用的案例，什么命名规范之类的，其实也不用太在意，怎么开心怎么来就好，你就是先驱者，别人按照你的来，久而久之，也就成了约定俗成的规范了。

好，就这些吧。

抛砖引玉，欢迎交流，感谢阅读！

> 《CSS世界》签名版独家发售，包邮，可指定寄语，点击显示购买码

（本篇完） // 想要打赏？点击这里。有话要说？点击这里。

« 小tips: 在canvas上实现元素图片镜像翻转动画效果

玩转HTML5 Video视频WebVTT字幕使用样式与制作 »

猜你喜欢

- HTML5 <template>标签元素简介
- 伪元素表单控件默认样式重置与自定义大全
- DOM元素querySelectorAll可能让你意外的特性表现
- ES6 JavaScript Promise的感性认知
- 简单了解ES6/ES2015 Symbol() 方法
- JS字符串补全方法padStart()和padEnd()简介
- 万岁，浏览器原生支持ES6 export和import模块啦！
- 看，for..in和for..of在那里吵架！

标签： ES6, HTMLUnknownElement, Shadow DOM, template, Web Components, 自定义元素

发表评论（目前5条评论）

名称 (必须)

邮件地址(不会被公开) (必须)

网站

提交评论

1. **TechQuery**说道：
2018年11月14日 09:22

HTML import 草案已从标准规范中移除，官方现在推荐 ES module，新的 Web 组件导入技术还在制定。

我自己封装了一套更方便的 Web components API，欢迎大家指教 —— https://web-cell.tk/

回复

2. **nikolausliu**说道：
2018年06月12日 17:33

长大居然也在看一人之下

回复

3. **cshenger**说道：
2018年03月23日 17:37

最近正在补一人之下

回复

4. 涂山苏苏说道：
2018年03月21日 10:30

感觉这个自定义组件是现在各大流行框架都喜欢干的事情，好像就是在浏览器已经封装好各种html标签和css声明的基础上，加一些自己喜欢的标签和相对应的css声明，再添加一些脚本方法，我们好像是不满足于浏览器厂商按照规范给出的这些浏览器特性，偏偏要自己作一堆自定义的东西。浏览器提供的面向大众，基本可以满足日常需求，一些特殊需求可以折中满足（因为特殊需求来源于特殊客户），自定义的方式确实是多样性，更随性，但是对团队长每个成员的要求都很高，就不是像我这种转行培训毕业的页面仔可以流畅驾驭的了，一旦掌握这种自定义的技能，公司的成本必然提升（因为公司要招懂这方面的人，懂这方面的人不可能10K左右就拿下）。到底是好，还是坏

回复

5. **ycy**说道：
2018年03月21日 09:45

三的1下面，继承自HTMelement,是不是少了个L?

回复

最新文章
» 常见的CSS图形绘制合集
» 粉丝群第1期CSS小测点评与答疑
» 分享三个纯CSS实现26个英文字母的案例
» 小tips: 纯CSS实现打字动画效果
» CSS/CSS3 box-decoration-break属性简介
» CSS :placeholder-shown伪类实现Material Design占位符交互效果

» 从天猫某活动视频不必要的3次请求说起
» CSS vector-effect与SVG stroke描边缩放
» CSS ::backdrop伪元素是干嘛用的?
» 周知：CSS -webkit-伪元素选择器不再导致整行无效

今日热门

» 常见的CSS图形绘制合集 (178)
» 粉丝群第1期CSS小测点评与答疑 (112)
» 未来必热：SVG Sprite技术介绍 (111)
» HTML5终极备忘大全（图片版+文字版） (85)
» 让所有浏览器支持HTML5 video视频标签 (83)
» Selectivizr-让IE6~8支持CSS3伪类和属性选择器 (80)
» CSS3下的147个颜色名称及对应颜色值 (78)
» 小tips: 纯CSS实现打字动画效果 (72)
» 写给自己看的display: flex布局教程 (69)
» 分享三个纯CSS实现26个英文字母的案例 (69)

今年热议

» 《CSS世界》女主角诚寻靠谱一起奋斗之人 (76)
» 不借助Echarts等图形框架原生JS快速实现折线图效果 (64)
» 看，for..in和for..of在那里吵架！ (60)
» 是时候好好安利下LuLu UI框架了！ (47)
» 原来浏览器原生支持JS Base64编码解码 (35)
» 妙法攻略：渐变虚框及边框滚动动画的纯CSS实现 (33)
» 炫酷H5中序列图片视频化播放的高性能实现 (31)
» CSS scroll-behavior和JS scrollIntoView让页面滚动平滑 (30)
» windows系统下批量删除OS X系统.DS_Store文件 (26)
» 写给自己看的display: flex布局教程 (26)

猜你喜欢

- HTML5 <template>标签元素简介
- 伪元素表单控件默认样式重置与自定义大全
- DOM元素querySelectorAll可能让你意外的特性表现
- ES6 JavaScript Promise的感性认知
- 简单了解ES6/ES2015 Symbol() 方法
- JS字符串补全方法padStart()和padEnd()简介
- 万岁，浏览器原生支持ES6 export和import模块啦！
- 看，for..in和for..of在那里吵架！