让过时浏览器支持新技术 **JavaScript Polyfill**

Published on Mar 5, 2022 UTC by Wang Ye <https://wangye.org>

互联网上有很多关于IE（Internet Explorer）的梗，确实IE已经是一款很老的浏览器了，刚才我打开Windows 10自带的IE，除了推荐我使用Edge外也在明示IE将在6月正式退役，所以现在开发的网站基本也不会在考虑IE，同样的考虑到本站的访客大部分应该是有一定技术背景的，用IE的可能性也不大，所以网站也不兼容IE，尤其是淘汰了TLS 1.1加密套件后，基本上也就目前最高版本的IE还能打开，我也稍微调整了一下JavaScript和CSS，让IE勉强可以浏览内容，但是交互的部分就无法保证了，等IE彻底退役后将不再考虑IE的可访问性。

实际上对于类似IE这类老旧浏览器最大的问题就是新的前端特性支持不全，导致脚本报错，除了可以使用jQuery替代外，也能使用一些Polyfill进行对应的Hack，其实随着新的前端技术的引入，我们对于jQuery的依赖越来越小，以后大的方向应该是抛弃沉重的jQuery，所以我的策略是尽量采用新的前端标准去编写JavaScript，所以Polyfill进行Hack是尽量让老浏览器支持访问的主要手段之一，有关什么是Polyfill，可以参考MDN关于Polyfill的解释。

1 传统的**Polyfill**方式

1.1 引入第三方**Polyfill**库

一个最简单明了的办法就是直接引入第三方的Polyfill库，但是这些库往往为了保持兼容性会做的大而全，使用淘汰过时浏览器的访客毕竟是少数，如果为了照顾少数人而背负一个重量级的JavaScript库，不仅增加带宽流量消耗，还会增加页面渲染的性能开销。

虽然仅对于IE浏览器来说可以通过条件注释的方式，将Polyfill库仅提供给IE浏览器，但是微软在最新版本的IE已经自信的取消了条件注释这个特性，也就是说最新的IE11浏览器完全无视条件注释包含的Polyfill库，所以直接引入Polyfill库是偷懒但不

完美的一种方式。有关条件注释最简单的应用举例如下：

```
<!--[if lt IE 9]>
<script type="text/javascript" src="polyfill.js"></script>
<![endif]-->
```

上面的条件注释指示小于IE9的浏览器将继续引入JavaScript脚本文件polyfill，其他浏览器忽略。

1.2 定制自己的Polyfill库

如果对当前网页所使用的新特性有个了解的话，可以自己定制特定的Polyfill Hacks，这里可以参考Can I use和MDN，比如对于`Math.trunc`这个Math下的方法，我们可以参考Can i use关于Math.trunc的支持情况，对于不支持的浏览器可以在网页加载的最开始引入脚本如下（来源：Polyfills and transpilers）：

```
if (!Math.trunc) { // if no such function
  // implement it
  Math.trunc = function(number) {
    // Math.ceil and Math.floor exist even in ancient JavaScript
engines
    // they are covered later in the tutorial
    return number < 0 ? Math.ceil(number) : Math.floor(number);
  };
}
```

这种方式仍然存在着为使用新浏览器访客增加不必要的带宽和流量消耗的问题，另外还需要查询并手动配置所有的Hack，虽然这些Hack可以通过MDN查询到，但仍然不是很方便。

2 使用Polyfill.io

Polyfill.io是在线可选择提供的polyfill服务网站，其会根据开发者的需求提供对应的Polyfill代码，其支持的最低版本的浏览器列表可以看出已经满足绝大部分的过时浏览器访客的需求。

2.1 定制并引入**Polyfill.io**服务

开发者可以通过Create a Polyfill Bundle定制自己所需要的Polyfill代码，无需关注代码本身实现，例如1.2节关于`Math.trunc`的Polyfill可以引入下面的脚本：

```
<script src="https://polyfill.io/v3/polyfill.min.js?
features=Math.trunc"></script>
```

可能有人会好奇的点开脚本所引用的URL地址，结果发现除了两行注释，什么代码都没有，感觉上当受骗之嫌，这里要说明的是，如果你看到的是这种情况，那么说明你使用的是无需Polyfill的新版本浏览器，Polyfill.io会判断浏览器版本，并对支持新特性的浏览器移除Polyfill的代码以解决传输带宽和流量，提高页面渲染效率。

2.2 可选择动态引入**Polyfill.io**服务

2.1节的配置方式还是存在可以改进的地方，比如对于支持某个特性的浏览器，完全可以在页面加载时判断出来从而不必再请求Polyfill.io服务，比如下面的代码：

```
if (typeof Math.trunc === 'undefined') {
    var url = "https://polyfill.io/v3/polyfill.min.js?
features=Math.trunc";
    document.write('<scr' + 'ipt type="text/javascript" src="' + url
            + '" crossorigin="anonymous"><\/scri' + 'pt>');
}
```

上述代码建议放在所有JavaScript脚本之前，比如`<head></head>`标签内靠前位置，对于不支持`Math.trunc`的浏览器（也就是`undefined`）直接输出Polyfill.io的脚本引用，否则忽略相关代码，对于支持新特性的浏览器也就无需多一次访问Polyfill.io的开销。

对于少量新特性的Polyfill我们可以修改上述代码，在`if`条件里判断，但如果要Polyfill判断的特性较多，那么对于上述代码的维护就显得不太友好，为此我为大家提供了一种新的方式，示例代码如下：

```
var polyfill_features = [
  "MediaQueryList.prototype.addEventListener",
  "String.prototype.startsWith",
  "String.prototype.trim",
  "Element.prototype.classList",
  "document.querySelector",
  "pageYOffset",
  "Array.prototype.forEach",
  "matchMedia",
  "localStorage",
  "URLSearchParams",
  "URL",
  "JSON",
  "Object.entries"
];

var cache_version = "v1";

// minimum browser version requirements
var min_browsers = {
    "Firefox": 80,
    "Edg": 80,
    "Chrome": 80,
    "Safari": 15
};

; (function(window, document, js_features, jsf_version, browsers) {
    var cacheKey = "js-features";

    var JsPolyfill = function(js_features, jsf_version) {
        function getPreDetectedResult() {
            var jsf_a_unsupports = js_features.join("%2C");
            if (window.localStorage) {
                var jsf = window.localStorage.getItem(cacheKey);
                if (jsf !== null && jsf.indexOf("^") > 0) {
                    var jsf_a = jsf.split("^");
                    var jsf_a_version = jsf_a[0];
```

```
                    if (jsf_a_version === jsf_version) {
                        if (jsf_a.length > 1) {
                            jsf_a_unsupports = jsf_a[1];
                        } else {
                            jsf_a_unsupports = "";
                        }
                    }
                }
            }

            return jsf_a_unsupports.split("%2C");
        }

        function getUnsupportedJsFeatures() {
            var unsupports = [];
            var jsf_a_features = getPreDetectedResult();
            var jsf_a_prev_features = "#" + js_features.join("#") +
"#";

            for (var i = 0; i < jsf_a_features.length; i++) {
                if (jsf_a_prev_features.indexOf("#" +
jsf_a_features[i] + "#") === -1) {
                    console.warn("Pollyfill: Unknown or suspicious
code detected," +
                                                    " skipped and
continue.");
                    continue;
                }

                if ((function() {
                    if
(jsf_a_features[i].indexOf("Element.prototype.") === 0) {
                        return (typeof Element === 'undefined' ||
                                !
(jsf_a_features[i].substr("Element.prototype.".length)
                                        in Element.prototype));
```

```
                    } else {
                        return eval("typeof " + jsf_a_features[i])
=== 'undefined';
                    }
                })()) {
                    unsupports.push(jsf_a_features[i]);
                }
            }

            return (unsupports.length > 0) ? unsupports.join("%2C") :
"";
        }

        var params = getUnsupportedJsFeatures();
        if (params !== "") {
            window.localStorage &&
                window.localStorage.setItem(cacheKey, jsf_version +
"^" + params);

            var url = "https://polyfill.io/v3/polyfill.min.js?
features=" + params;
            document.write('<scr' + 'ipt type="text/javascript"
src="' + url
                                                + '"
crossorigin="anonymous"><\/scri' + 'pt>');
        } else {
            window.localStorage &&
                window.localStorage.setItem(cacheKey, jsf_version +
"^");
        }
    };

    function isBrowserCompatibleWith() {
        if (window.localStorage &&
                window.localStorage.getItem(cacheKey) ===
(jsf_version + "^")) {
            return true;
```

```
        }

        if (typeof browsers === 'object' &&
               window.navigator && window.navigator.userAgent) {
            return (function() {
                var userAgent = window.navigator.userAgent;
                for (key in browsers) {
                    if (userAgent.indexOf(key + '/') !== -1) {
                        var version = userAgent.split(key + '/')[1];
                        if (key == 'Safari') {
                            version = userAgent.split('Version/')[1];
                        }
                        version_a = version.split('.');
                        version_s = parseInt(version_a[0], 10);
                        if (!isNaN(version_s) && version_s >=
browsers[key]) {
                            window.localStorage.setItem(cacheKey,
jsf_version + "^");
                            return true;
                        }

                        break;
                    }
                }
                return false;
            })();
        } else {
            return false;
        }
    }

    if (!isBrowserCompatibleWith())
        JsPolyfill(js_features, jsf_version);
})(window, document,
       polyfill_features, cache_version, min_browsers);
```

上述代码中可配置部分为变量polyfill_features表示需要polyfill的特性列表，cache_version表示特性列表的缓存版本，如果修改了polyfill_features列表内容则需要同步更新缓存版本，否则访客对于是否使用polyfill特性会受到缓存影响，min_browsers则表示支持所有特性的浏览器最小版本，代码基本处理逻辑如下：

检查浏览器是否在min_browsers列表最小需求版本里，如果在则写入localStorage缓存记住，下次则直接读取缓存避免再次判断，如果不在min_browsers列表最小需求版本里，那么则依次检查polyfill_features特性列表是否支持，并将不支持的特性构建查询字符串（Query String）并引入Polyfill.io脚本服务，同时记入缓存，避免下次访问重复检查拖慢渲染效率。

另外如果修改了polyfill_features列表内容则需要同步更新缓存版本cache_version，因为在读取缓存的时候会判断缓存版本，如果版本不匹配，则会忽略缓存。

3 总结

随着先进浏览器的越来越普及，相信这类Polyfill的操作必将被淘汰于历史的长河中，但是现阶段为了保障小部分人依然可以享受互联网的精彩，我们可以选择做一定的妥协。

External References (Links)

MDN关于Polyfill的解释
https://developer.mozilla.org/docs/Glossary/Polyfill
Can I use
https://caniuse.com/
MDN
https://developer.mozilla.org/
Can i use关于Math.trunc的支持情况

https://caniuse.com/?search=Math.trunc

Polyfills and transpilers

https://javascript.info/polyfills

Polyfill.io

https://polyfill.io/

支持的最低版本的浏览器列表

https://polyfill.io/v3/supported-browsers/

Create a Polyfill Bundle

https://polyfill.io/v3/url-builder/