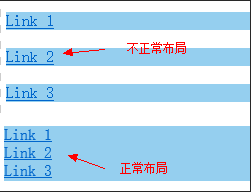
浏览器兼容性，是指网页在各种浏览器上的显示效果可能不一致而产生浏览器和网页间的兼容问题，如下图所示。



# 兼容性问题分类

网页兼容性问题的原因主要来自以下几方面：

* 浏览器引擎bug
* 网页标准更迭
* 不同浏览器不同标准

## 浏览器引擎bug

因为IE浏览器占用市场份额大，涌现比较多的是IE低版本的bug。到目前为止，已发现如下所示的bug列表。

### Bug列表

**宽高bug**

1. IE6-浏览器下子元素能撑开父级设置好的宽高
2. IE6-浏览器下最小高度问题，设置(0-15px)高度小于等于15px的元素，在IE6下会被当作15px来处理
3. 在IE6下，如果要给元素定义100%高度，必须要明确定义它的父级元素的高度，如果你需要给元素定义满屏的高度，就得先给html和body定义 height:100%。

**边框bug**

1. IE6-浏览器下1px的点线边框，点线会变成虚线
2. 标准下背景会延伸到边框区，而IE7-浏览器下背景只延伸到padding区
3. 在IE10-浏览器下被<a>标签包含的<img>元素会产生边框

**盒模型bug**

1. IE7-浏览器下父级有边框，无法阻止子元素的上下margin值传递
2. IE7-浏览器下不设置文档声明会导致怪异盒模型解析。在怪异盒模型下内容宽=width-2\*padding-2\*borderWidth
3. IE6-浏览器下使用margin负值，使元素移出父级，移出部分会被父级裁掉

**列表项bug**

1. (li的4px空隙bug)IE7-浏览器下，li本身没浮动，但内容有浮动，li下边会多出4px的空隙
2. (li下的4px间隙和最小高度共存的问题)IE7浏览器下，当li下的4px间隙问题和最小高度问题共存的时候，设置垂直对齐方式无效
3. (li的3px空隙bug)IE7-浏览器下li有高度或宽度或zoom:1，且仅包含内联元素，且内联元素被设置为display:block，li下会多出3px的垂直间距

**浮动bug**

1. (3pxbug)在IE6-浏览器下浮动元素和非浮动元素相邻时，会出现3px像素的空隙
2. IE6-下父元素浮动后，且子元素设置了高度，如果父元素不设置宽度，宽度会撑满整行
3. (浮动折行)在IE7-浏览器下，如果两个元素一个右浮动，一个不浮动。浮动元素会折到下一行
4. (双边距bug)IE6-浏览器下块元素有浮动，且有横向的margin值。若仅有左margin，最左边的浮动的块元素的左margin会放大成两倍。若仅有右margin，最右边的浮动的块元素的右margin会放大成两倍。若左右都有，最左边的左margin和最右边的右margin会放大成两倍
5. (margin-bottomBUG)在IE7-浏览器下父级宽度和每行元素的宽度之和相差超过3px时，或者有不满行的情况，最后一行的margin-bottom失效
6. (文字溢出bug)IE6-浏览器下两个浮动元素(浮动元素不能是li)一个左浮无宽度，另一个右浮动宽度与父级宽度相差不超过3px，浮动元素中间有注释或者内联元素，文字就被复制

**定位bug**

1. 在IE7-浏览器下子元素有相对定位，父级的overflow无效
2. 在IE6-浏览器下浮动元素和绝对定位元素是并列关系，且浮动元素设置margin-left和width的和正好等于父元素的宽度，这时绝对定位元素会消失
3. 在IE6-浏览器下绝对定位元素的父级元素的宽度为奇数时，元素的right会有1px的偏差；高度为奇数时，元素的bottom会有1px的偏差

**表单bug**

1. IE6-浏览器下label标签只支持for属性，不支持仅仅包含<input>的写法
2. (input空隙问题)当input元素被div包围时，IE6-浏览器下它们之间上下会各多出1px的空隙；而IE7、8浏览器下它们之间的上边会多出1px的空隙
3. IE6-浏览器下当input元素被div包围时，在已经给input设置浮动的情况下，设置border:none无法得到理想效果
4. IE7-浏览器下输入类型表单控件如<textarea>、<input>等输入文字时，背景图像会跟着文字一起滚动
5. IE6-浏览器中select控件无法被<div>覆盖，因为在IE6中select控件是处于最顶层的
6. 如果页面中多个元素的ID值相同，getElementById（）只会返回文档中第一个出现的元素。假如IE7及较低版本中的name特性与给定ID匹配，对应的表单元素（<button>、<textarea>、<select>）也会被该方法返回。如果有哪个表单元素的name特性等于指定的ID，而且该元素在文档中位于带有给定ID的元素前面，那么IE就会返回那个表单元素。

**其他类型bug**

1. 在IE6下，浮动层之间有注释文字的话，之前那个浮动层的内容文字就有可能遭遇一个“隐形”的复制，但是代码里查看文字可并没有多出来。
2. IE6下使用DXImageTransform.Microsoft.AlphaImageLoader滤镜（用于PNG32 Alpha透明）后链接不能点击的BUG
3. 在<select>上设置了一个静态的宽度时，当其<option>选项内容超过设置的宽度时，其内容也将直接被截

### 解决方法

2) 设置font-size为0，但最小高度为2px;设置overflow:hidden，但最小高度为1px;要想实现最小高度为0，只能是不设置高度

5) 给图片设置{border: none}

6) 触发父级的haslayout

9) 给子级加相对定位relative

10) 给li加浮动;设置vertical-align

11) 给li加浮动

12) 触发li中子元素的haslayout

13) 使用CSShack，给浮动元素设置相反方向的-3px的margin值，将非浮动元素的相应方向的margin设为0(加IE6前缀);去掉非浮动元素的margin值，加浮动。

14) 给浮动的父元素设置宽度;给子元素设置宽度;给子元素设置浮动

15) 给不浮动的元素也加浮动;在HTML中先放右浮动的元素

16) 给块元素设置display:inline

17) 尽量不要用margin-bottom，而用margin-top代替

18) 将注释去掉;将内联元素变成块元素;内联元素及注释整个用<div>包起来

19) 给父级也设置相对定位

20) 给定位元素外面包一个div

22) 使用for属性

23) 给input加浮动

24) 设置border:0;重置input的背景

25) 把背景加给父级，并清掉自身背景

26) iframe比select优先级高，把iframe嵌套在<div>里面，并设置为不可见

27) 统一使用id

28) 不放置注释

29) 在使用滤镜的容器外面再加上一个容器，这个容器加上绝对定位。a标签上加相对定位的属性（position:relative）

30) 使用了jQuery方法来解决,当元素在静态下（未触发select中的option选项时）使用有一个宽度值来设置<select>，在IE下时，鼠标移动到select的option选项中，宽度变成自动，以适合内容字段的宽度，当鼠标移出时又回到表态宽度的设置

## 前端网页标准更迭

该类型bug是由于技术规范更新换代，无论是先前还是往后，书写的代码并未考虑向前向后的兼容，导致在不同版本浏览器上使用引发了兼容性。标准的更迭主要有html、css、js这三方面。

### Html版本

Html版本从1993年h1版本到目前2014年h5版本，主流是h4、4.01版本。各版本定义的html标签有增有减，如下图所示；另一方面，低版本浏览器版本并无法感知新的h5标准；因此，书写html代码应当考虑浏览器对新版本的支持情况。

#### 语义标签

【新增】

h5版本新增了 'abbr', 'article', 'aside', 'audio', 'bdi', 'canvas', 'data', 'datalist','details', 'dialog', 'figcaption', 'figure', 'footer', 'header', 'hgroup','main', 'mark', 'meter', 'nav', 'output', 'picture', 'progress', 'section','summary', 'template', 'time', 'video'等标签。

【解决方法】

引入polyfill，使得旧浏览器支持解析新增元素；但并不能实现对应标签的功能，需要额外引入对应标签功能的polyfill。

【polyfill列表】

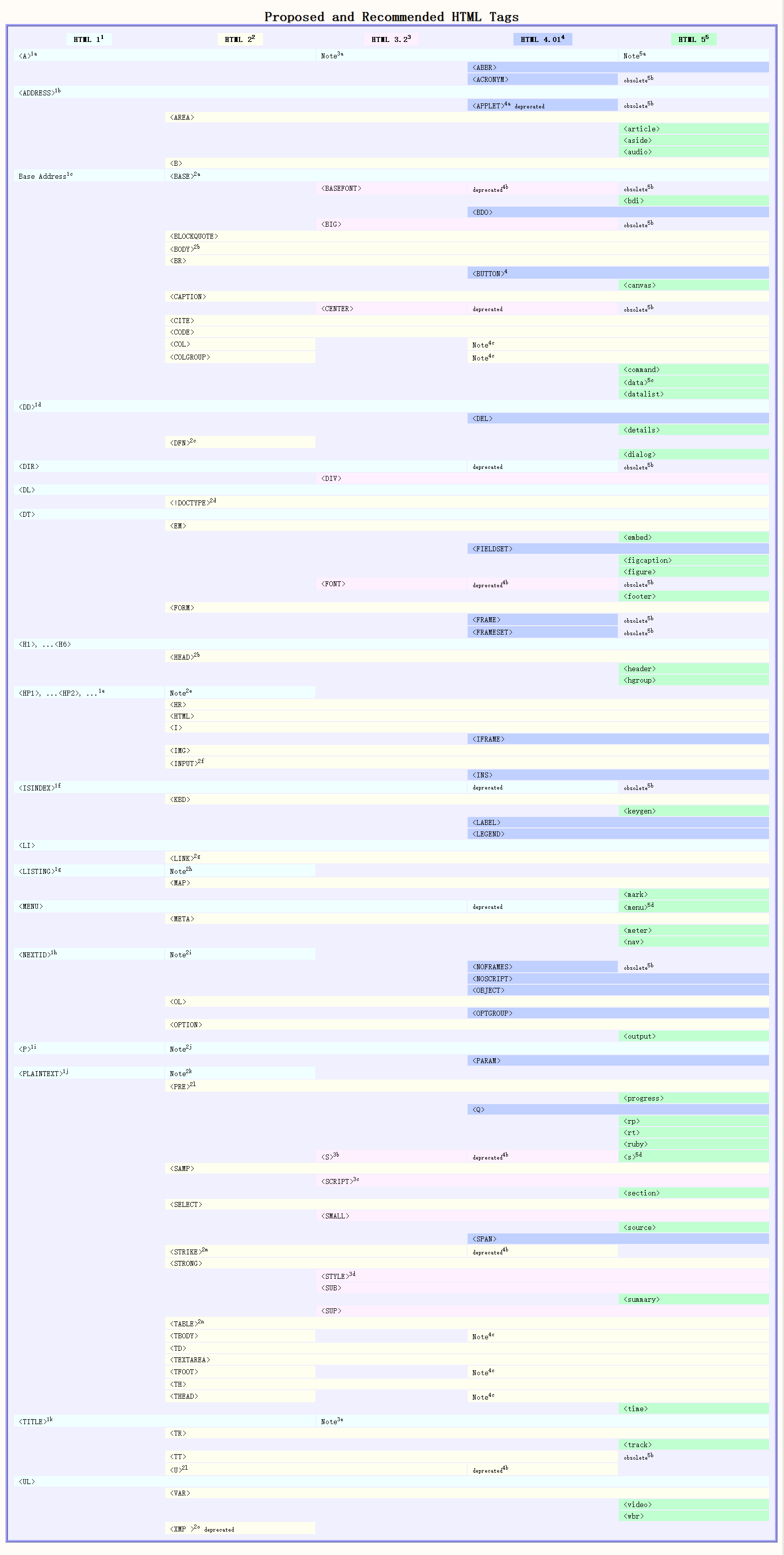
，支持浏览器解析新增标签

，支持details标签功能

，支持dialog标签功能

，支持progress标签功能

，支持picture标签功能



#### 表单

【新增】

在h5版本中，既有对原有的表单元素扩展，又有加入新的特性，如以下几种所示。

**<input> 元素**

扩展type的属性值

* search: 这个元素呈现为一个搜索框。除了换行符会自动从输入中移除，无其他强制性语法。
* tel: 这个元素可现为一个编辑电话号码的输入控件。因为电话号码国际化差异非常明显，所以除了换行符会自动从输入中移除，无其他强制性语法。你可以使用如 pattern 与 maxlength 等属性来限制输入到控件中的值。
* url: 这个元素呈现为一个编辑URL 的输入控件。换行符与首尾的空格将会被自动去除。
* email: 这个元素呈现为一个邮件地址。
* …

新的特性

* list： <datalist> 元素的 ID，该元素的内容，<option> 元素被用作提示信息，会在 input 的建议区域作为提议显示出来。
* pattern： 一个正则表达式，用于检查控件的值，能够作用于 type 值是 text， tel， search， url， 和 email 的 input 元素。
* form： 一个字符串，用于表明该 input 属于哪个 <form> 元素。一个 input 只能存在于一个表单中。
* formmethod：一个字符串，用于表明表单提交时会使用哪个 HTTP 方法 (GET 或 POST)；如果定义了它，则可以覆盖 <form> 元素上的 method 特性。只有当 type 值为 image 或 submit，并且 form 特性被设置的情况下， formmethod 才能生效。

**<form> 元素**

<form> 元素有了一个新特性：novalidate：设置了该特性不会在表单提交之前对其进行验证

**<datalist> 元素**

<datalist> 元素会在填写 <input> 字段时，显示一列 <option> 作为提示。你可以使用 <input> 元素上的 list 特性来将一个特定的 input 与特定的 <datalist> 元素做关联。

**<output> 元素**

<output> 元素表示计算的结果。使用 for 特性来在 <output> 元素与文档内其他能够影响运算的元素(例如，input 或参数)建立关联。 for 特性的值是以空格做分隔的其他元素的 ID 列表。

**placeholder 特性**

placeholder 特性作用于 <input> 与 <textarea> 元素上，提示用户此域内能够输入什么内容。placeholder 中的文本不能包含回车与换行

**autofocus 特性**

autofocus 特性让你能够指定一个表单控件，当页面载入后该表单自动获得焦点，除非用户覆盖它，例如在另一个控件中输入值。一个文档内只有一个表单能够拥有 autofocus 特性，它是一个 Boolean 值。这个特性适用于 <input>， <button>， <select>，与 <textarea> 元素。例外情况是，如果一个 autofocus 元素的 type 特性值设置成了 hidden，则 autofocus 无法生效(就是说，你无法让一个隐藏控件自动获得焦点)

**label.control DOM 属性**

HTMLLabelElement DOM 接口除了为 <label> 元素提供了对应的特性外，还提供了一个额外的属性。 control 属性返回被打上标签的控件，就是 label 适用的控件，由 for 特性(如果定义的话) 或是第一个后代元素控件来确定

**约束验证**

HTML5 为客户端表单的验证提供了语法与 API。如果 <input> 元素设置了 title 特性，当验证失败时，特性值会显示在提示信息中。如果 title 设置为空字符串，则不会显示提示信息。如果没有设置 title 特性，会使用标准验证信息(例如通过 x-moz-errormessage 特性指定，或调用 setCustomValidity() 方法) 代为显示

**约束验证的 HTML 语法**

* <input>, <select>， 和 <textarea> 元素上的 required 特性，指定必须提供该元素的值。(在 <input> 元素上， required 只能与特定的 type 特性值结合起来生效。)
* <input> 元素上的 pattern 特性用于限定元素值必须匹配一个特定的正则表达式。
* <input> 元素上的 min 与 max 特性限定了能够输入元素的最大与最小值。
* <input> 元素的 step 特性(与 min 与 max 特性结合使用) 限定了输入值的间隔。如果一个值与允许值的梯级不相符，则它无法通过验证。
* <input> 与 <textarea> 元素的 maxlength 特性限制了用户能够输入的最大字符数(在 Unicode 代码点内)。
* type 的 url 与 email 值分别用于限制输入值是否为有效的 URL 或电子邮件地址。

此外，若要阻止对表单进行约束验证，你可以在 <form> 上设置 novalidate 特性，或在 <button> 与 <input> 元素(当 type 是 submit 或 image)上设置 formnovalidate 特性。这些特性指定了当表单提交时不做验证

**约束验证 API**

HTMLFormElement 对象上的 checkValidity() 方法，当表单的相关元素都通过了它们的约束验证时返回 true，否则返回 false。

在 表单相关元素上：

* willValidate 属性，如果元素的约束没有被符合则值为 false。
* validity 属性，是一个 ValidityState 对象，表示元素当前所处的验证状态(就是说约束成功或是失败)。
* validationMessage 属性，用于描述与元素相关约束的失败信息。
* checkValidity() 方法，如果元素没有满足它的任意约束，返回false，其他情况返回 true。
* setCustomValidity() 方法，设置自定义验证信息，用于即将实施与验证的约束来覆盖预定义的信息

【解决方法】

上述多种的表单元素，若是在低版本提供支持，其实现原理类似开发jquery插件，扩展元素属性。可参考jquery组件开发、UI组件开源框架。

【polyfill列表】



#### 通信

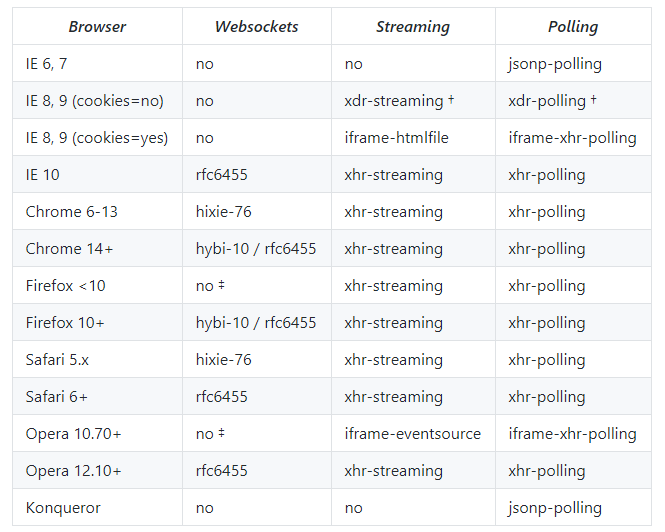
【新增】

Web Sockets，允许在页面和服务器之间建立持久连接并通过这种方法来交换非 HTML 数据。

WebRTC，即时通信，允许连接到其他人，直接在浏览器中控制视频会议。

【解决方法】

Web Sockets可使用sockjs-client，在低版本使用iframe实现功能，如下图所示。



WebRTC需要使用插件，Temasys WebRTC Plugin

【polyfill列表】



#### 离线、存储

【新增】

在线、离线事件，WEB是否离线状态事件

浏览器历史，允许对浏览器历史记录进行操作。

IndexedDB，是一个为了能够在浏览器中存储大量结构化数据，并且能够在这些数据上使用索引进行高性能检索的 Web 标准

【解决方法】

在线、离线事件，可定时使用异步ajax检测网站首页的响应

引入 history.js

引入Indexeddbshim.js

【polyfill列表】

#### 多媒体

Video，Audio ie6-8不支持

#### 图像处理

【新增】

Canvas，绘制图像和其他对象

Svg，一个基于 XML 的可以直接嵌入到 HTML 中的矢量图像格式

Webgl，WebGL 通过引入了一套非常地符合 OpenGL ES 2.0 并且可以用在 HTML5 <canvas> 元素中的 API 给 Web 带来了 3D 图像功能。

【解决方法】

Canvas 使用Silverlight HTML 5 Canvas支持IE6~IE8

Svg IE6-9不支持，但可使用VML(一种XML语言用于绘制向量图形（vector graphics）。1998年VML建议书由微软、Macromedia等向W3C提出审核。VML遭到拒绝，因为Adobe、Sun等提出了PGML计划书。这两套标准后来合并成更具潜力的SVG)

Webgl,无

#### 其他

【新增】

Visibility网页可见状态管理

拖拽，ie6-11没有解决方案

requestAnimationFrame，ie6-9没有解决方案

【解决方法】

Visibility.js，遵守<https://www.w3.org/TR/page-visibility/>规范

【polyfill列表】



#### 设备访问

【新增】

使用摄像头，允许使用和操作设备的摄像头，并从中存取照片

地理位置，浏览器使用地理位置服务定位用户的位置

触摸事件，对用户按下触控屏的事件做出反应的处理程序

检测设备方向，用户在运行浏览器的设备变更方向时能够得到信息

指针锁定，允许锁定到内容的指针，这样游戏或者类似的应用程序在指针到达窗口限制时也不会失去焦点

【解决方法】

地址位置，使用ip判断位置

其他功能，无方案

【polyfill列表】

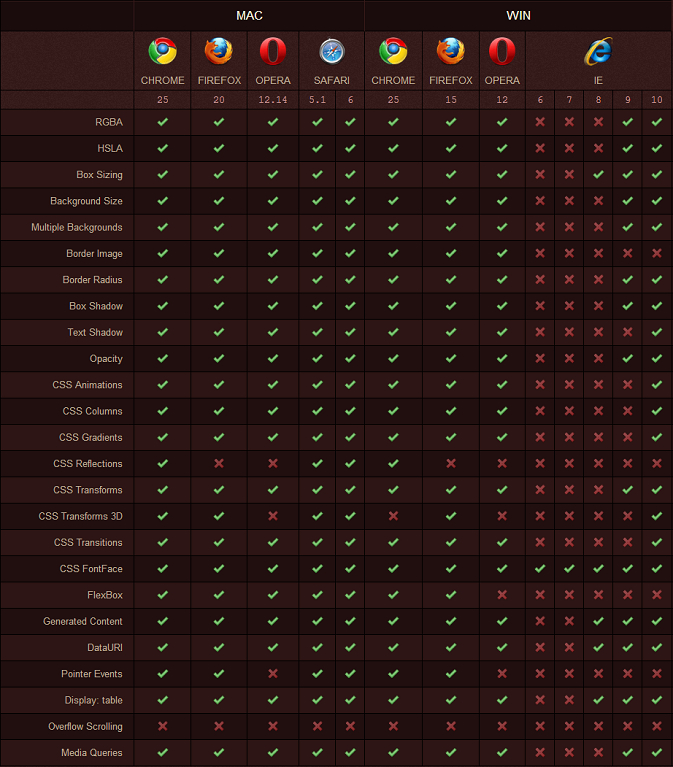


### css版本

css版本从1996年1.0版本到目前2011年3.0版本，4.0版本正在研制中。

【新增】

浏览器对css3.0支持情况，如下图所示。





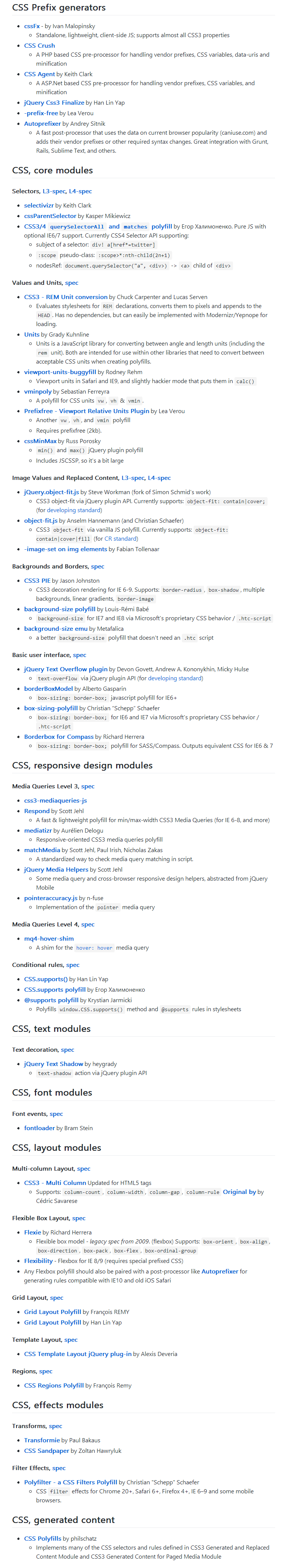
如上图所示，旧版本浏览器对css3.0的新特性并不支持。

【解决方法】

其中，能够兼容解决的有以下几种：

* Values and Units模块中的“vw, vh & vmin”
* Image Values and Replaced Content模块中的“object-fit”
* Backgrounds and Borders模块中的“border-radius, box-shadow, multiple backgrounds, linear gradients, border-image, background-size”
* Basic user interface模块中的“text-overflow, box-sizing”
* Media Queries Level 3模块中的” Media Queries, min/max-width”
* Text decoration模块中的” text-shadow”
* Multi-column Layout模块中的” column-count, column-width, column-gap, column-rule”
* Flexible Box Layout模块中的” box-orient, box-align, box-direction, box-pack, box-flex, box-ordinal-group”
* Transforms

【polyfill列表】



### js版本

js版本从1995年js1.0到2017年ECMA-262 8th，经历多次版本的迭代。

【新增】

截止到ECMAScript 2015版本，js语言新增不少特性，如下文档所示。



【解决方法】

新增特性较多，且不可能完全在低版本实现兼容，只能做到以下几点：

**Fundamental Objects**

* Object: assign(), is(), setPrototypeOf()
* Symbol:Symbol(description),Symbol.for(),Symbol.keyFor(),Symbol.iterator, Symbol.toStringTag
* Not supported: Function.prototype.toMethod()

**Numbers and Dates**

* Number: EPILON, isFinite(), isInteger(), isNaN(), isSafeInteger(), MAX\_SAFE\_INTEGER, MIN\_SAFE\_INTEGER, parseFloat(), parseInt()
* Math: acosh(), asinh(), atanh(), cbrt(), clz32(), cosh(), expm1(), fround, hypot(), imul(), log1p(), log10(), log2(), sign(), sinh(), tanh(), trunc()

**Text Processing**

* String: fromCodePoint(), raw
* String prototype: codePointAt(), endsWith(), includes(), repeat(), startsWith(), [@@iterator]()
* Not supported: String.prototype.normalize() - see https://github.com/walling/unorm/
* RegExp prototype: @@match(), @@replace(), @@search(), @@split(), flags
* String.prototype match(), replace(), search(), and split() dispatch through RegExp symbol methods

**Indexed Collections**

* Array: from(), of()
* Array prototype: copyWithin(), entries(), fill(), find(), findIndex(), keys(), values(), [@@iterator]()
* TypedArray - for browsers without native support (IE9-) include typedarray.js
* %TypedArray% prototype: from(), of()
* %TypedArray% prototype: copyWithin(), entries(), every(), fill(), filter(), find(), findIndex(), forEach(), indexOf(), join(), keys(), lastIndexOf(), map(), reduce(), reduceRight(), reverse(), slice(), some(), sort(), values(), [@@iterator]()

**Keyed Collections**

* Map: clear(), delete(), entries(), forEach(), get(), keys(), has(), set(), size, values(), [@@iterator]()
* Set: add(), clear(), delete(), entries(), forEach(), has(), size, values(), [@@iterator]()
* WeakMap: clear(), delete(), get(), has(), set()
* WeakSet: add(), clear(), delete(), has()
* WeakMap and WeakSet are intrusive and modify the valueOf property of keys

**Asynchronous Programming**

* Promise: p = new Promise(), Promise.resolve(), Promise.reject(), Promise.cast(), Promise.race(), Promise.all(), p.then(), p.catch()

**Typed Arrays**

* ArrayBuffer
* Uint8Array, Int8Array, Uint16Array, Int16Array, Uint32Array, Int32Array, Float32Array, Float64Array
* DataView

【Polyfill列表】



## 不同浏览器不同标准

### DOM&EVENT&API

这部分内容是指浏览器的window、document等对象的变化。由于不用浏览器对window、document的实现有所不同，因此，在使用上存在兼容性问题。

【问题点】

**DOM&EVENT**

* Selectors (for IE7-) - adapted from Paul Young
* element = document.querySelector(selector)
* elementArray = document.querySelectorAll(selector)
* elem.matches(selector) (for IE, Firefox 3.6, early Webkit and Opera 15.0)
* elementArray = document.getElementsByClassName(classNames) (for IE8-)
* e = element.nextElementSibling, e = element.previousElementSibling (for IE8)
* Node constants: Node.ELEMENT\_NODE, etc (for IE8-)
* DOMException constants: DOMException.INDEX\_SIZE\_ERR (for IE8-)
* Events (for IE8)

1. Where EventTarget is window, document, or any element:

* EventTarget.addEventListener(event, handler) - for IE8+
* EventTarget.removeEventListener(event, handler) - for IE8+

1. Event: target, currentTarget, eventPhase, bubbles, cancelable, timeStamp, defaultPrevented, stopPropagation(), cancelBubble()

* QuirksMode
* window.addEvent(EventTarget, event, handler)
* window.removeEvent(EventTarget, event, handler)
* DOMTokenList - classList, relList
* DOMTokenList: length, item(index), contains(token), add(token), remove(token), toggle(token)
* tokenList = elem.classList - for IE8+
* tokenList = elem.relList - for IE8+
* Non-standard helpers for IE7-:
* tokenList = window.getClassList(element)
* tokenList = window.getRelList(element)
* ParentNode: node.prepend(nodes...), node.append(nodes...)
* ChildNode: node.before(nodes...) , node.after(nodes...) , node.replaceWith(nodes...) , node.remove()
* document.defaultView, window.getComputedStyle

**API**

Fetch\ URL\ Notification

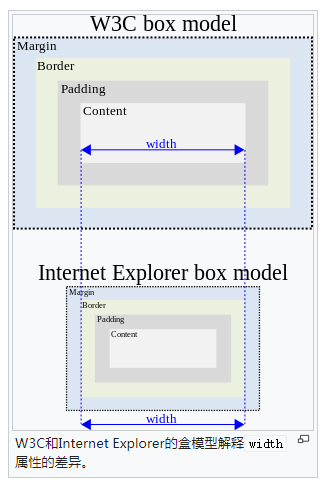
【polyfill列表】



### 低版本IE浏览器问题

#### 盒子缺陷

万维网联盟（W3C）指定的，当任意一个块级元素的宽度或高度被显式指定，它应当只确定这个可见元素自身的宽度或高度，而padding, border和margin随后被应用。Internet Explorer则把内容，内边距（padding）和边框（border）全部包括在一个指定的宽度或高度之内；这导致它呈现出一个比遵从标准行为的结果更窄或者更短的盒子，如下图所示。



#### hasLayout

hasLayout可以简单看作是IE5.5/6/7中的BFC(Block Formatting Context)。也就是一个元素要么自己对自身内容进行组织和尺寸计算(即可通过width/height来设置自身的宽高)，要么由其containing block来组织和尺寸计算。而IFC（即没有拥有布局）而言，则是元素无法对自身内容进行组织和尺寸计算，而是由自身内容来决定其尺寸（即仅能通过line-height设置内容行距，通过行距来支撑元素的高度；也无法通过width设置元素宽度，仅能由内容来决定而已）

当hasLayout为true时(就是所谓的"拥有布局")，相当于元素产生新BFC，元素自己对自身内容进行组织和尺寸计算;

当hasLayout为false时(就是所谓的"不拥有布局")，相当于元素不产生新BFC，元素由其所属的containing block进行组织和尺寸计算。

和产生新BFC的特性一样，hasLayout无法通过CSS属性直接设置，而是通过某些CSS属性间接开启这一特性。不同的是某些CSS属性是以不可逆方式间接开启hasLayout为true。并且默认产生新BFC的只有html元素，而默认hasLayout为true的元素就不只有html元素了。

另外我们可以通过object.currentStyle.hasLayout属性来判断元素是否开启了hasLayout特性。

**默认hasLayout==true的元素**

<html>, <body>

<table>, <tr>, <th>, <td>

<img>,<hr>

<input>, <button>, <select>, <textarea>, <fieldset>, <legend>

<iframe>, <embed>, <object>, <applet>,<marquee>

**触发hasLayout==true的方式**

display: inline-block

height: (除 auto 外任何值)

width: (除 auto 外任何值)

float: (left 或 right)

position: absolute

writing-mode: tb-rl

zoom: (除 normal 外任意值)

**IE7 还有一些额外的属性(不完全列表)可以触发 hasLayout**

min-height: (任意值)

min-width: (任意值)

max-height: (除 none 外任意值)

max-width: (除 none 外任意值)

overflow: (除 visible 外任意值，仅用于块级元素)

overflow-x: (除 visible 外任意值，仅用于块级元素)

overflow-y: (除 visible 外任意值，仅用于块级元素)

position: fixed

**其中通过display:inline-block或min-width:0或min-height:0将不可逆地启用hasLayout特性。而在没有其他属性启用hasLayout时，可通过以下方式关闭hasLayout**

max-width, max-height (设为 "none")(在IE7中)

position (设为 "static")

float (设为 "none")

overflow (设为 "visible") (在IE7中)

zoom (设为 "normal")

writing-mode (从 "tb-rl" 设为 "lr-t")

#### quirks mode

一些网页浏览器为了维持对较旧的网页设计的向后兼容性，有别于严格遵循万维网联盟（W3C）与互联网工程任务组（IETF）标准而设计的“标准模式”。

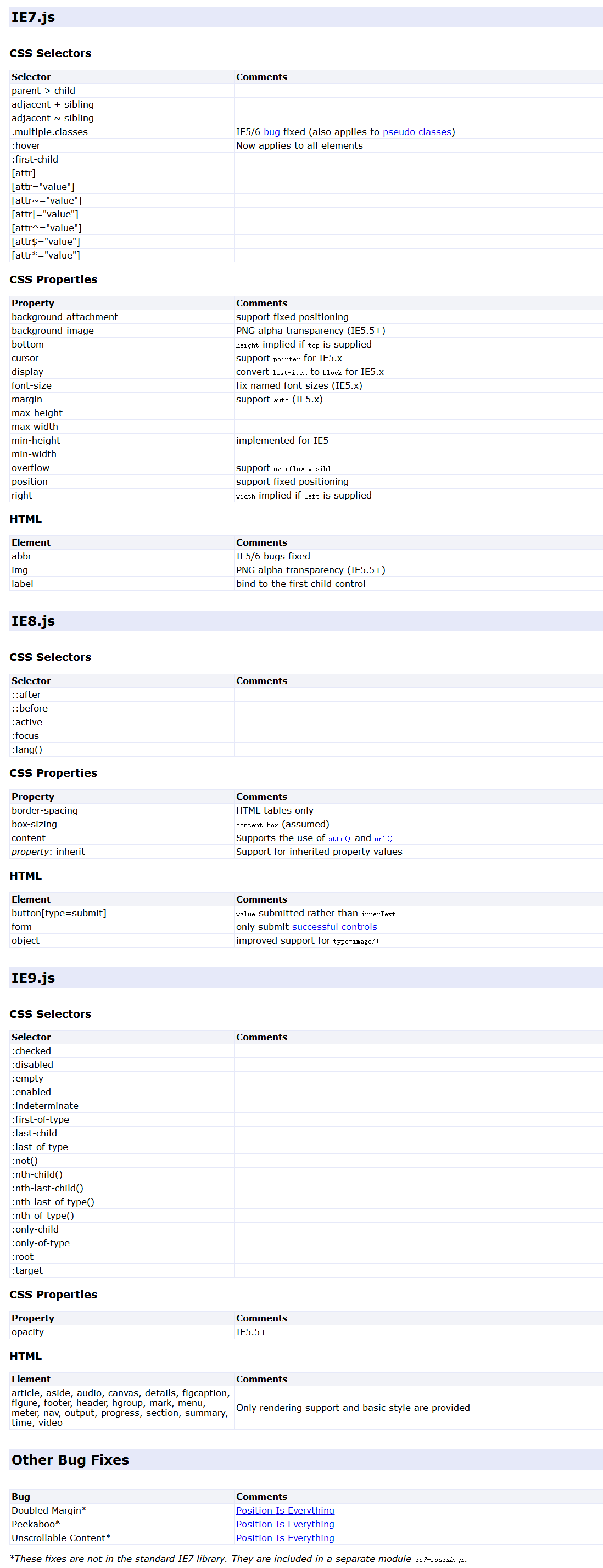
通常，浏览器基于页面中文件类型描述的存在以决定采用哪种渲染模式；如果存在一个完整的DOCTYPE则浏览器将会采用标准模式，而如果它缺失则浏览器将会采用怪异模式，如下图所示。



#### 问题汇总

结合上述三方面的描述，以下几种兼容性问题可归类到此类型。

* 双空白边浮动bug
* 3像素文本偏移bug
* IE 6躲躲猫bug
* 相对容器中的绝对定位



### 其他跨浏览器问题

1. li标签样式
2. IE下,可以使用()或[]获取集合类对象;非IE,用[]获取集合类对象
3. 不同浏览器换行符高度问题
4. 统一使用cursor:pointer 少用cursor:hand ，因为firefox不支持hand，但ie支持pointer
5. innerText在IE中能正常工作，但在FireFox中却不行
6. ul和ol列表缩进问题
7. firefox与IE的父元素(parentElement)的区别
8. input.type属性问题，IE下input.type属性为只读；但是Firefox下input.type属性为读写
9. 元素水平居中问题
10. 链接(a标签)的边框与背景，a链接加边框和背景色，需设置 display: block, 同时设置 float: left 保证不换行
11. 超链接访问过后hover样式就不出现的问题，被点击访问过的超链接样式不在具有hover和active了,解决技巧是改变CSS属性的排列顺序: L-V-H-A
12. FORM标签，标签在IE中,将会自动margin一些边距,而在FF中margin则是0,因此,如果想显示一致,所以最好在css 中指定margin和 padding,针对上面两个问题,我的css中一般首先都使用这样的样式ul,form{margin:0;padding:0;}
13. …

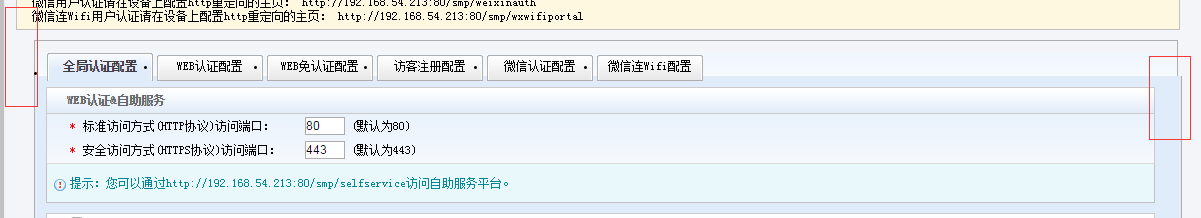
# 项目常见兼容性问题

## Smp\ipc\ess项目

1. 菜单、登录页面高度问题，如bug列表描述的第3条



1. 获取不到元素节点，如bug列表描述的第27条
2. 边距对齐，如IE盒子缺陷描述



1. 判断enter按键事件不兼容，如dom&event&api章节描述
2. 如问题汇总的第3条描述

## Eportal项目

1. 输入框提示文字，如表单中描述的Placeholder特性。
2. Js新语法特性在低版本浏览器报错，“{liner:liner,slowfast,slowfast}” –> “{ liner,slowfast}”
3. 图片边框问题，如问题汇总的第2条
4. Dom节点删除方法，如dom&event&api章节描述
5. 居中使用换行符，如问题汇总的第4条
6. Canvas的使用，如图像处理描述
7. Css3的使用，“圆角、阴影、透明…”，如css版本描述
8. 轮播图片中的object.create、document.getElementsByClassName、document.defaultView.getComputedStyle函数
9. Addeventlistener兼容性
10. 按键事件兼容性
11. 表单控件的兼容性

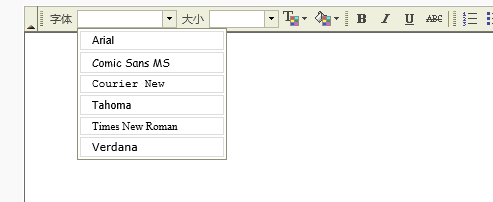
# 兼容性问题指导

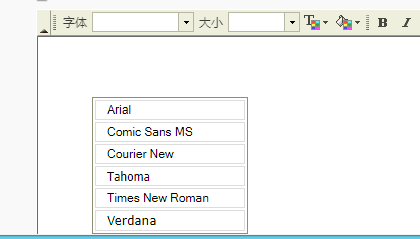
## 排查问题

1. 根据bug现象，判断兼容性问题类型
2. 针对类型采取不同措施
   1. 浏览器bug，一般采用规避方法
   2. 网页标准更迭，引用兼容库
   3. 不同浏览器不同标准，引用兼容库，统一css处理

**【实例】**

Sam自助配置页面通知公告的配置，在IE浏览器下弹出菜单正常；而在chrome浏览器下，菜单则偏移且跟随滚动条移动。初步判断是不同浏览器不同标准类型，引起的原因还需要看具体的代码，从以下几点步骤进行。

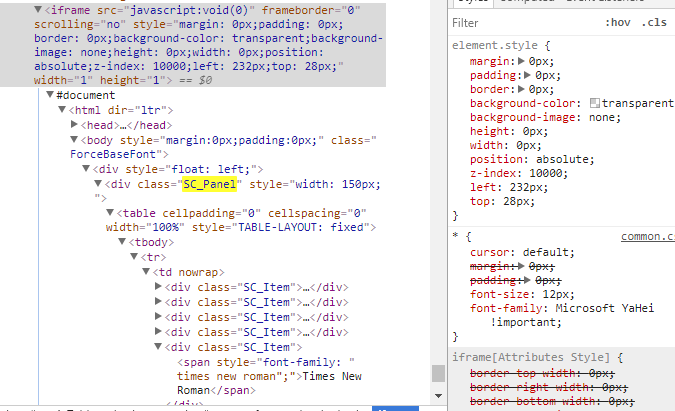




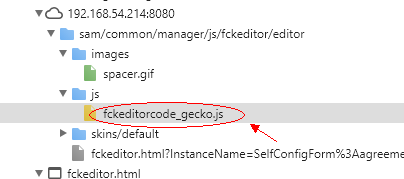
步骤1，按F12,打开浏览器调试工具

步骤2，鼠标停留在偏移的菜单项，(在谷歌浏览器)shift+ctrl+I选择元素

步骤3，查看元素的css样式，判断菜单项布局的css实现方式



步骤4，根据步骤3的css代码，判断菜单是使用iframe标签实现，利用绝对布局控制位置，长宽控制是否显示与否；追踪js代码实现部分，如下图所示。由于该js代码被压缩，无法得到源码实现逻辑。



经过上述的步骤分析，该bug是由于代码兼容不同浏览器引起，需要修改其实现方式；但因代码被压缩，无法进行修改。因此，建议将此控件换成新版本，或者找到该控件的源码，修复bug代码。

## 解决问题思路

所有的兼容性问题并不能都被解决。一般地，低版本兼容高版本特性，尽力而为，其解决方法是利用低版本提供的能力去实现高版本的特性功能。W3C定义的标准在不同浏览器实现而产生的问题，则可以解决，其解决方法是判断浏览器类型编写适配浏览器的代码。

【实例】

1. 低版本兼容高版本特性

在ie8中并无document.getElementsByClassName函数，利用IE8现有的API实现该函数功能，如下代码所示。

document.getElementsByClassName = function (className,element) {

var children = (element || document).getElementsByTagName('\*');

var elements = new Array();

for (var i = 0; i < children.length; i++) {

var child = children[i];

var classNames = child.className.split(' ');

for (var j = 0; j < classNames.length; j++) {

if (classNames[j] == className) {

elements.push(child);

break;

}

}

}

return elements;

};

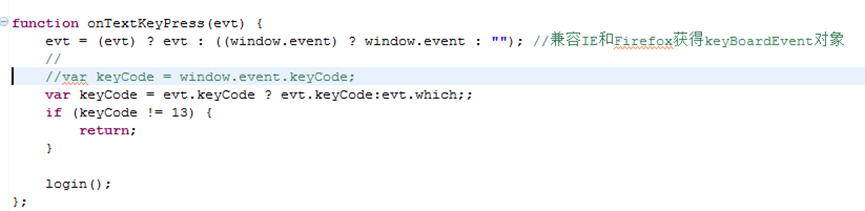
if(!Element.getElementsByClassName){

Element.prototype.getElementsByClassName = document.getElementsByClassName;

}

1. 代码兼容不同浏览器

Keypress兼容



## 编码规范

经过上述的讨论，新增的html、js、css代码应当考虑以下几点注意事项：

1. css样式是否存在hack情况，否则需要书写对应浏览器的css代码
2. js api接口是否兼容，否则引入polyfill解决
3. html标签是否兼容，否则引入标签polyfill及对应功能polyfill解决
4. html页面类型是否需要兼容，正确指定页面类型

# 参考链接

<http://www.w3cplus.com/solution/iebug/iebug.html>

<https://zh.wikipedia.org/wiki/%E6%80%AA%E5%BC%82%E6%A8%A1%E5%BC%8F>

<http://www.positioniseverything.net/explorer.html>

<http://www.positioniseverything.net/index.php>

<http://adamghost.com/2009/03/ie-has-layout-and-bugs-zh/>

<http://www.cnblogs.com/xiaohuochai/p/4845342.html#anchor1>