# 整体架构-jQuery3.2.1

# 创建jQuery对象----(L94)

# jQuery原型添加方法与属性

# 核心函数

# 选择器sizzle引擎

# 回调对象

# Deferred对象(异步队列)

# 数据缓存

# 队列操作

# 事件系统

# DOM操作

# 样式操作

## 概述

### 给一个HTML元素设置css属性

var head= document.getElementById("head");  
head.style.width = "20px";  
head.style.height = "10px";  
head.style.display = "block";

### 传统js设置的存在的问题

* + - 1. 单一的设置太麻烦，而且每次style一次就等于浏览器要绘制一次，当然高级的浏览器是可能会合并style的次数的
      2. style只能针对行类样式，对于link引入的样式，无法获取
      3. 样式属性名的兼容问题，比如驼峰，保留字；如background-image =>要写成 backgroundImage；float是保留字，一般浏览器是cssFloat，IE为styleFloat

### cssText使用

* + - 1. 针对style重复渲染，使用cssText可以避免

var head= document.getElementById("head");  
head.style.cssText="width:20px;height:10px;display:bolck";

* + - 1. cssText 返回值

在某些浏览器中（比如 Chrome），你给他赋什么值，它就返回什么值。在 IE 中则比较痛苦，它会格式化输出、会把属性大写、会改变属性顺序、会去掉最后一个分号

* + - 1. 主要问题：会把原有的cssText清掉

比如原来的style中有’display:none;’，那么执行完element.style.cssText=”width:20px;height: 20px;border:solid 1px red;”;后，display就被删掉了。

* + - 1. 解决cssText覆盖问题
         1. 利用累加的方式

Element.style.cssText += ‘width:100px;height:100px;top:100px;left:100px;’

* + - * 1. Ie使用开头加分号；方式

Element.style.cssText += ‘**;**width:100px;height:100px;top:100px;left:100px;’

### getComputedStyle

* + - 1. 用法

document.defaultView.getComputedStyle(element[,pseudo-element]);

首先是有两个参数，元素和伪类。第二个参数不是必须的，当不查询伪类元素的时候可以忽略或者传入 null。

* + - 1. 调用方式

在许多在线的演示代码中, getComputedStyle 是通过 document.defaultView 对象来调用的。 大部分情况下，这是不需要的， 因为可以直接通过window对象调用

而且除了在 IE8 浏览器中 **document.defaultView === window** 返回的是 false 外，其他的浏览器（包括 IE9 ）返回的都是 true。

* + - 1. 和 style 的异同
         1. 相同点：

二者返回的都是 CSSStyleDeclaration 对象，取相应属性值得时候都是采用的 CSS 驼峰式写法，均需要注意 float 属性。

* + - * 1. 不同点：

element.style 读取的只是元素的“内联样式”，即 写在元素的 style 属性上的样式；

getComputedStyle 读取的样式是最终样式，包括了“内联样式”、“嵌入样式”和“外部样式”。

element.style 既支持读也支持写

getComputedStyle 仅支持读并不支持写入

## css自定义属性

### 定义

用这样的方式来声明一个变量：--variable-name: variable-value;（变量名是大小写敏感的）。变量的值可以是颜色、字符串、多个值的组合等

:root {  
 --main-color: #4d4e53;  
 --main-bg: rgb(255, 255, 255);  
}

可能语法看起来有些丑陋，却是有原因的。比如不能使用像$var这样的变量语法，因为已经被其他的CSS预处理程序使用了。

### 用法

以这样的方式来使用一个变量: some-css-value: var(--variable-name [, declaration-value]);

li {

background-color: #4d4e53;  
 background-color: var(--main-color,#000);  
}

如—main-color未指定颜色，则使用#000，但为了兼容性可能需要加上background-color；这个可能可以通过自动化工具生成

### 作用域

* + - 1. 自定义属性遵循CSS级联规则
      2. 使用:root 作用域来定义全局变量：

:root {  
 --main-color: #4d4e53;  
 --main-bg: rgb(255, 255, 255);  
}

* + - 1. 如果想让某个变量只在部分元素/组件下可见，只需要在特定的元素下定义该变量
         1. 如下，.block与#aaa是兄弟关系，在.block设置的自定义css属性并不能在#aaa中使用

<div class="block"> My block is  
 <div class="block\_\_highlight">awesome</div>  
</div>  
<div id="aaa">asd</div>

.block {  
 --block-background : red;  
}  
  
.block\_\_highlight {  
 background: #fff;  
 color:var(--block-background)  
}  
#aaa{  
 background-color: var(--block-background);  
}

### 在js中使用

* + - 1. 使用CSS样式声明接口,可以在JS中方便的读/写自定义属性(getPropertyValue, setProperty)

// READ   
const rootStyles = getComputedStyle(document.documentElement);   
const varValue = rootStyles.getPropertyValue('--screen-category').trim();   
// WRITE   
document.documentElement.style.setProperty('--screen-category', value);

### Js判断浏览器是否支持

if (window.CSS && window.CSS.supports && window.CSS.supports('--a', 0)) {  
 alert('CSS properties are supported');  
} else {  
 alert('CSS properties are NOT supported');  
}

## 样式操作对外接口：$().css()

### 注意

* + - 1. 此函数jQuery.fn.css()具有两个核心函数，jQuery.css与jQuery.style，为什么会有两个呢？因为样式读写不是同一个方式
      2. 只读：jQuery.css()，依赖于window上的getComputedStyle方法，内联样式与外联都可以查到
      3. 读写操作：jQuery.style，利用elem.style方式，只能读内联样式；但问题是无法查询外联样式
      4. 因此，jQuery.fn.css方法中使用前者的读，后者的写

### var getStyles = function( elem ) --------(L6073)

// 获得元素elem的计算属性，并对getComputedStyle函数有问题浏览器进行兼容处理  
var getStyles = function( elem ) {  
  
 // Support: IE <=11 only, Firefox <=30 (#15098, #14150)  
 // ie会在popups中创建一个元素时抛出异常  
 // FF 同时会在iframe元素上调用defaultView.getComputedStyle抛出异常  
 var view = elem.ownerDocument.defaultView; // 除了在 IE8 浏览器中 document.defaultView === window 返回的是 false 外，其他的浏览器（包括 IE9 ）返回的都是 true。  
 // window.opener是对弹出窗口的母窗口的一个引用  
 // a.html中，通过点击按钮等方式window.open出一个新的窗口b.html。  
 // 那么在b.html中，就可以通过window.opener（省略写为opener）来引用a.html，包括a.html的document等对象，操作a.html的内容。  
 if ( !view || !view.opener ) {  
 view = window;  
 }  
  
 return view.getComputedStyle( elem );  
 };

### 源码分析--------------------(L6627)

// 访问第一个匹配元素的计算后样式值  
// $("p").css({ "color": "#ff0011", "background": "blue" });  
jQuery.fn.extend( {  
 /\*\*  
 \* 调用方式：  
 \* 获取样式值：return String  
 \* 1、string形式：$('li').css('background-color');  
 \* 2、数组形式：$('li').css(['background-color','font-size'])，返回map，{'background-color':'#000','font-size':'12px'}  
 \* 设置值：return jQuery  
 \* 1、obj形式：$('li').css({'background-color':'#000','font-size':'12px'}),实际是通过access内部循环调用access，将值分别设置  
 \* 2、$('li').css(name,value);value为undefined，调用jQuery.css方法，否则调用jQuery.style()方法  
 \*  
 \*/ css: function( name, value ) {  
 // 如name为obj，access中会对obj进行循环调用access，将每个key-value再调用一次access函数  
 // 对于如$('li'),匹配了多个元素，实际access绑定数据的是elems[ 0 ]元素，access最后返回  
 // len ? fn( elems[ 0 ], key ) : emptyGet;len为匹配元素长度  
 return access( this, function( elem, name, value ) {  
 var styles, len,  
 map = {},  
 i = 0;  
 // 如name为数组，类似$('p').css(["paddingTop", "paddingRight", "paddingBottom", "paddingLeft"]);

// 增加一种个性化的 取值 方式  
 if ( Array.isArray( name ) ) {  
 // 获得元素elem的计算属性  
 styles = getStyles( elem );  
 len = name.length;  
  
 for ( ; i < len; i++ ) {

// false参数则只返回未经处理的样式值，给定了styles则从styles对象取样式  
 map[ name[ i ] ] = jQuery.css( elem, name[ i ], false, styles );  
 }  
  
 return map;  
 }  
  
 return value !== undefined ?

// 赋值  
 jQuery.style( elem, name, value ) :

// 取值  
 jQuery.css( elem, name ); // value为undefined时，调用jQuery.css  
 }, name, value, arguments.length > 1 );  
 }

## jQuery.style()

### 处理思路

* + - 1. 对输出参数name进行处理
         1. 对于需要增加浏览器前缀的name，增加前缀；
         2. 需要进行转换驼峰的进行转换；
         3. 需要对关键字如float进行转换的，进行转换
      2. 利用cssHooks对不同浏览器进行兼容

### 注意：

* + - 1. jQuery.css、jQuery.style相似点比较多，之所以设计的较为复杂是考虑了非必要的“易用”、“兼容”、“扩展”。使得设置更灵活、输出更一致、支持累加换算、支持拓展的功能。
      2. 由于jQuery.style并不会对cssNumber里面列表进行添加单位操作，故对于$(‘p’).css(‘line-height’:’+=5’)；如原始p的line-height：10px，结果输出是line-height：15;并无单位；

### function vendorPropName( name )-------(L6256)

// 给一些样式属性增加浏览器前缀  
// 返回带有浏览器前缀的map表  
function vendorPropName( name ) {  
  
 // 快捷方式判断name是否需要增加浏览器前缀  
 // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*提供如何判断哪些样式需要增加浏览器前缀  
 if ( name in emptyStyle ) {  
 return name;  
 }  
  
 // 检测浏览器前缀名  
 // 把name的第一个字符变为大写，如rex变为Rex  
 // 因为浏览器厂商样式都是形如：webkitColumnBreakInside形式  
 var capName = name[ 0 ].toUpperCase() + name.slice( 1 ),  
 i = cssPrefixes.length;  
  
 while ( i-- ) {  
 name = cssPrefixes[ i ] + capName;  
 if ( name in emptyStyle ) {  
 return name;  
 }  
 }  
}

### function finalPropName( name ) ---------(L6276)

// 根据jQuery.cssProps和浏览器前缀属性，返回一个map表  
// 将name转换成style设置时使用的名称  
function finalPropName( name ) {  
 // cssProps主要是针对一些特殊情况，如float，返回style中使用的名称  
 // 如果需要兼容低版本的ie，可以才cssProps中这样返回：  
 // { "float": jQuery.support.cssFloat ? "cssFloat" : "styleFloat"}  
 // 如除了float，又有其他的样式名称需要转换，假如有个样式是rex，需要转换为REX，在cssProps中增加  
 // { "float":"cssFloat","rex":"REX"}  
 var ret = jQuery.cssProps[ name ]; // 如name为float，返回cssFloat  
 if ( !ret ) {  
 // vendorPropName,给一些样式属性增加浏览器前缀

// vendorPropName计算后的带有前缀的样式会添加到cssProps中，这样遇到相同name无需再进行计算增加前缀了  
 ret = jQuery.cssProps[ name ] = vendorPropName( name ) || name;  
 }  
 return ret;  
}

### jQuery.cssHooks.opacity---------------(L6385)

* + - 1. cssHooks对象并不只有opacity一个，下面再看

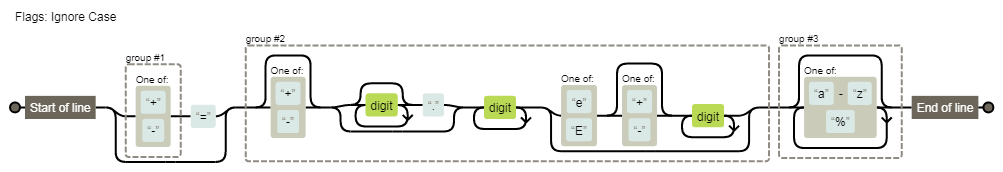
// “添加样式属性”hooks以覆盖获取和设置样式属性的默认行为。  
cssHooks: {  
 opacity: {  
 get: function( elem, computed ) {  
 if ( computed ) {  
  
 // opacity属性，应该总返回一个数值  
 // 如未设置，则返回1

// curCSS获取的是浏览器计算后的最终值

// curCss之后再看，此处只知是获得计算最终值即可  
 var ret = curCSS( elem, "opacity" );  
 return ret === "" ? "1" : ret;  
 }  
 }  
 }  
},

### rcssNum-------------------(L4487)

// source 属性用于返回模式匹配所用的文本。不包括限定符  
var pnum = ( /[+-]?(?:\d\*\.|)\d+(?:[eE][+-]?\d+|)/ ).source;  
// rcssNum:^(?:([+-])=|)([+-]?(?:\d\*\.|)\d+(?:[eE][+-]?\d+|))([a-z%]\*)$  
// 匹配的是+1px,+=1.23em 等，用于匹配非string，数字的样式值，转换为数值的正则  
var rcssNum = new RegExp( "^(?:([+-])=|)(" + pnum + ")([a-z%]\*)$", "i" );



### function adjustCSS( elem, prop, valueParts, tween ) --------------(L4536)

/\*\*  
 \* 调整css设置值，如+=10%,则计算出实际应该设置为多少，  
 \* 对增量计算的换算，并得到最终值。在jq内部，除了css样式会换算，动画处理也支持换算。  
 \* 此函数也可以把动画的tween对象的初始值和增量进行累加换算，得到最终值赋给tween对象  
 \* 1调用、对于jQuery.style()调用，  
 \* elem为匹配的第一个元素，  
 \* prop为需要设置的样式的key如height；  
 \* valueParts为rcssNum.exec( value )  
 \* tween：undefined  
 \* 如HTML为：  
 \* <div>  
 \* <p>asdf</p>  
 \* </div>  
 \* js为：  
 \* $('p').css('height','+=10%')  
 \* 样式：  
 \* div{height:50px;}  
 \* // 当前chrome中，p高度默认为20px，因此，最终.css('height','+=10%'),p上应该设置的是20/50+10 =50%;才对  
 \*  
 \*/  
function adjustCSS( elem, prop, valueParts, tween ) {  
 var adjusted,  
 scale = 1,  
 maxIterations = 20,  
 currentValue = tween ?  
 function() {  
 return tween.cur();  
 } :  
 function() {  
 return jQuery.css( elem, prop, "" );//第3个参数""，则将获得的值转换为数值  
 },  
 //1调用，就是调用jQuery.css获得,注意获得的initial初始值并不带有单位  
 //即使未给p高度，p也会有个默认高度，假如当前chrome，p高度为20px，故initial为20  
 initial = currentValue(),  
 // 匹配单位，若不在cssNumber目录，并且没带单位，则当做px  
 unit = valueParts && valueParts[ 3 ] || ( jQuery.cssNumber[ prop ] ? "" : "px" ),//获取单位  
  
 // 为了避免潜在单位错配，进行初值计算  
 // 由于初始值若匹配到单位，都会是px，单位不是px的在执行css过程中jq也有钩子修正  
 // 所以有可能需要换算的只有cssNumber列表中项目，或者unit不为px且initial有非0数值的（0无需换算）。初始值为字符串如"auto"，则会在下面按照0处理  
 initialInUnit = ( jQuery.cssNumber[ prop ] || unit !== "px" && +initial ) &&  
 rcssNum.exec( jQuery.css( elem, prop ) );// jQuery.css( elem, prop )与jQuery.css( elem, prop ,"")区别就是前面带单位，后面返回不带单位  
 // 只是提供一个算法，计算css('height','+=10%')这样的相对单位，实际应该设置值为多少  
 // 以%为例，css('height','+=10%'),需要先计算出当前p相对于父级div所占百分比，  
 if ( initialInUnit && initialInUnit[ 3 ] !== unit ) {  
  
 // 如unit为undefined，使用jQuery.css获取到的单位  
 unit = unit || initialInUnit[ 3 ];  
  
 // 确保稍后更新tween属性  
 valueParts = valueParts || [];  
  
 // 从非零起点迭代逼近  
 initialInUnit = +initial || 1;  
  
 do {  
  
 // 如上一次迭代scale为0，则加倍直到取得某些东西  
 // 使用字符串设置加倍，这样下面scale就不会不改变  
 scale = scale || ".5";  
  
 // 调整initialInUnit值  
 initialInUnit = initialInUnit / scale;  
 // 对于1调用的例子，实际此处设置的是height:20%  
 jQuery.style( elem, prop, initialInUnit + unit );  
  
 // 更新scale，允许tween.cur()出现NaN与0值  
 // 如scale未改变或者到达最大循环maxIterations次数，跳出循环  
 // 对于调用1，因为父级div为50px，故currentValue() 等于 10  
 } while (  
 scale !== ( scale = currentValue() / initial ) && scale !== 1 && --maxIterations  
 );  
 }  
  
 if ( valueParts ) {  
 // 初始值为字符串，也将按照0处理，  
 initialInUnit = +initialInUnit || +initial || 0;  
  
 // 如指定了+=，-=类似的相对值，则对返回值进行调整  
 // 根据是否为增量运算判断直接赋值还是换算后的初始值与增量相加，css运算中只允许增量运算使用该函数  
 adjusted = valueParts[ 1 ] ?  
 initialInUnit + ( valueParts[ 1 ] + 1 ) \* valueParts[ 2 ] :  
 +valueParts[ 2 ];  
 if ( tween ) { // 如tween存在，则设置tween的属性  
 tween.unit = unit;  
 tween.start = initialInUnit;  
 tween.end = adjusted;  
 }  
 }  
 return adjusted;  
}

### jQuery.style()源码---------------------(L6423)

// 获取并设置DOM Node上的样式属性。  
// 对于$().css调用此函数，value为undefined，extra为undefined  
// 参数：elem是元素，name是样式名称，后面2个是用于把样式值转为数字类型（后面2个参数在$().width()使用再看）  
// 综合取值和设置值操作：  
// 取值操作：1、hooks的get返回值!=undefined，用此值  
// 2、否则使用elem.style[...]获取值（并不是计算后的样式值）  
// 3、注意：$('li').css()方法并不使用jQuery.style()取值，故此函数取值方法并没有对css自定义属性做判断  
// 设置值：1、value为null，不设置  
// 2、$('li').css('height',2);value为数值，则增加px单位  
style: function( elem, name, value, extra ) {  
  
 // 不要在文本和注释节点上设置样式。  
 if ( !elem || elem.nodeType === 3 || elem.nodeType === 8 || !elem.style ) {  
 return;  
 }  
  
 // 确保我们使用的是正确的name值，即name进行驼峰标记法转换，是否为自定义样式判断  
 var ret, type, hooks,  
 // 因为上来调用name转驼峰标记法，因此.css({ "background-color": "#ffe", "border-left": "5px solid #ccc" }) 和  
 // .css({backgroundColor: "#ffe", borderLeft: "5px solid #ccc" })返回相同值  
 // jq支持驼峰和’-‘串联两种写法,这里进行了处理  
 origName = jQuery.camelCase( name ),  
 isCustomProp = rcustomProp.test( name ),//匹配以--开头，判断name是否为CSS自定义属性，Css自定义属性都是以--开头  
 style = elem.style;  
  
 // 确保我们使用正确的名称，如是用用户定义的css自定义属性，我们不想修改他的值  
 if ( !isCustomProp ) {  
 name = finalPropName( origName );  
 }  
  
 // 先使用有前缀的name，然后再用原始name，获得cssHooks  
 // cssHook，即是对不同浏览器，不同属性的兼容  
 // 获取cssHook值  
 // jQuery.cssHooks是样式机制的钩子系统。可以对需要hack的属性，添加钩子，  
 // 在jQuery.css、jQuery.style读取写入之前，都会先看是否存在钩子并调用，  
 // 然后决定是否继续下一步还是直接返回。通过在set、get属性中定义函数，使得行为正确一致。  
 hooks = jQuery.cssHooks[ name ] || jQuery.cssHooks[ origName ];  
  
 // 检查是设置值还是取值  
 if ( value !== undefined ) {  
 type = typeof value; // 判断value的数据类型。  
  
 // 允许增量’+=20px’式写法，由于采用jQuery.css获取的初始值单位有可能不同，  
 // 因此封装了一个自动单位换算并输出增量后最终结果的函数adjustCSS()  
 // 转换 "+=" 或 "-=" 到相关numbers (#7345)  
 // $().css('margin':'+=10'),如之前margin为10，则现在为20  
 if ( type === "string" && ( ret = rcssNum.exec( value ) ) && ret[ 1 ] ) {  
 // 对于类似+=，-=的调用方式如$('p').css('height','+=10%')，进行值计算  
 value = adjustCSS( elem, name, ret );  
  
 // 修改bug #9237  
 type = "number";  
 }  
  
 // 确保value为null或NaN不会设置 (#7116)  
 if ( value == null || value !== value ) {  
 return;  
 }  
  
 // 为了提高易用性，jQuery.style()可以自动为设置值加上默认单位’px’，  
 // 由于有些属性值可以为数字，因此定义了cssNumber的列表，列表中的项目不会加上默认单位。  
 // 如value为数值，则为value增加单位 (除了某些CSS属性)  
 if ( type === "number" ) {  
 // cssNumber里的样式，不自动添加px单位  
 value += ret && ret[ 3 ] || ( jQuery.cssNumber[ origName ] ? "" : "px" );  
 }  
  
 // background-\* 属性会影响原始的克隆值  
 // 如当前浏览器存在background的bug则修复下，修复与判断并不理解  
 if ( !support.clearCloneStyle && value === "" && name.indexOf( "background" ) === 0 ) {  
 style[ name ] = "inherit";  
 }  
  
 // 如果hook提供set方法，并且set结果值不是undefined，使用hooks后的值  
 if ( !hooks || !( "set" in hooks ) ||  
 ( value = hooks.set( elem, value, extra ) ) !== undefined ) {  
 // css自定义属性要使用setProperty设置  
 if ( isCustomProp ) {  
 style.setProperty( name, value );  
 } else {  
 style[ name ] = value;  
 }  
 }  
  
 } else {// 取值操作  
  
 // 如果hook提供了get方法获取非计算值，使用之  
 if ( hooks && "get" in hooks &&  
 ( ret = hooks.get( elem, false, extra ) ) !== undefined ) {  
  
 return ret;  
 }  
  
 // 如无hook的get方法，则直接从style对象中获取值  
 return style[ name ];  
 }  
},

## jQuery.css()--------(L6500)

// 访问匹配元素的样式属性。  
// 注意与style区别，CSS只是获取值（参数并没有value值），并没有设置，并没有进行DOM操作  
// 因为jQuery.css并未对name参数进行处理，jQuery.css不能传入obj，name为obj jQuery.camelCase会报错  
// 调用1：对于 $('p').css('padding')，会调用jQuery.css();  
 // 返回值：  
 // jQuery.css对样式值的读取，可以指定对于带单位字符串和”auto”等字符串如何返回，新增了extra参数。为""（不强制）和true（强制）返回去单位值，false不做特殊处理直接返回。  
 // 功能扩展：jq允许直接通过innerWidth()/innerHeight()、outerWidth()/outerHeight()读取，也支持赋值，直接调整到正确的宽高。这是通过extra指定padding、border、margin等字符串做到的  
 // cssHooks.expand：对于margin、padding、borderWidth等复合属性，通过扩展expand接口，可以得到含有4个分属性值的对象  
css: function( elem, name, extra, styles ) {  
 var val, num, hooks,  
 origName = jQuery.camelCase( name ),  
 isCustomProp = rcustomProp.test( name );// 判断name是否为CSS自定义属性  
  
 // 确保我们使用正确的名称，如是用用户定义的css自定义属性，我们不想修改他的值  
 if ( !isCustomProp ) {  
 // 将name进行转换，对需要的增加浏览器前缀，float改为cssFloat  
 name = finalPropName( origName );  
 }  
  
 // 先使用有前缀的name，然后再用原始name，获得cssHooks  
 // cssHook，即是对不同浏览器，不同属性的兼容  
 hooks = jQuery.cssHooks[ name ] || jQuery.cssHooks[ origName ];  
  
 // 如提供了hooks方法，并有get方法，通过hooks的get获取value值。  
 if ( hooks && "get" in hooks ) {  
 // 此处的hooks.get调用与$.style()不同，true表示获得计算后的样式（具体原因hooks再看）  
 // 但并不是什么时候都可以获得计算后样式的，比如：  
 // 新创建的元素，在没有添加到文档中时，浏览器没有对此元素进行渲染，样式还没有计算，所以只能获得非计算的样式值。  
 val = hooks.get( elem, true, extra );  
 }  
  
 // 如无hooks，那么调用curCss方法获取样式值  
 if ( val === undefined ) {  
 //对于调用1： $('p').css('padding')， elem为匹配的p元素，name为padding，styles为undefined  
 val = curCSS( elem, name, styles );  
 }  
  
 // 将normal值转换为计算值  
 // 如fontWeigth默认是normal，转换为数值  
 if ( val === "normal" && name in cssNormalTransform ) {  
 val = cssNormalTransform[ name ];  
 }  
  
 // extra === "" 去单位处理，若为"normal"、"auto"等字符串，原样返回  
 // extra === true 强制去单位，若为parseFloat后为NaN的字符串，返回0  
 // extra === false/undefined 不特殊处理

if ( extra === "" || extra ) {  
 num = parseFloat( val );  
 return extra === true || isFinite( num ) ? num || 0 : val;  
 }  
  
 return val;  
}

## jQuery.cssHooks.width与 jQuery.cssHooks.height

### 注意：

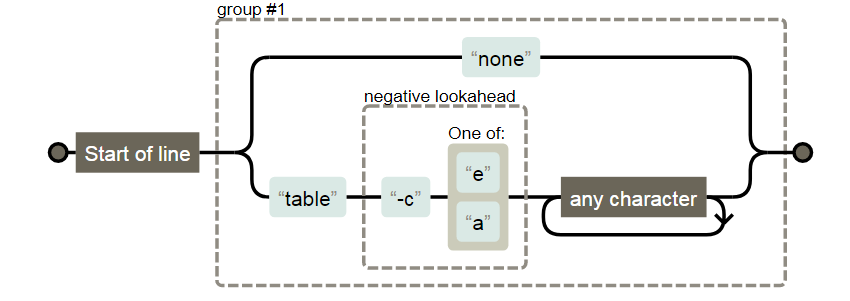
* + - 1. 因为浏览器缺陷并不了解那么深，所以hook可能并不理解为什么这么做

### rdisplayswap--------(L6242)

// 如display为none或以table开头（但除了"table", "table-cell", or "table-caption"），需要包裹一下  
// 这是查看display的值//developer.mozilla.org/en-US/docs/CSS/display

// 这个主要是针对display这个属性的某些属性值进行处理  
rdisplayswap = /^(none|table(?!-c[ea]).+)/,

* + - * 1. (?!exp) 匹配后面跟的不是exp的位置，则table-ce不匹配
        2. (?=exp) 匹配exp前面的位置，则table-ce匹配



### var swap = function( elem, options, callback, args )--------(L4513)

/\*\*  
 \* 主要功能是将元素旧的属性存储，然后根据options设置新的样式属性，然后利用callback计算新样式属性下的结果值；  
 \* 最后将原有样式原有  
 \* 调用1：jQuery.cssHooks.width与jQuery.cssHooks.height中调用， 其中  
 \* elem:匹配到的元素，  
 \* options为{ position: "absolute", visibility: "hidden", display: "block" }；  
 \* callback为getWidthOrHeight返回值  
 \*/  
var swap = function( elem, options, callback, args ) {  
 var ret, name,  
 old = {};  
  
 // 记录旧样式值，插入新值  
 for ( name in options ) {  
 old[ name ] = elem.style[ name ];  
 elem.style[ name ] = options[ name ];  
 }  
  
 ret = callback.apply( elem, args || [] );  
  
 // 将旧值重新设置回elem  
 for ( name in options ) {  
 elem.style[ name ] = old[ name ];  
 }  
  
 return ret;  
};

### function curCSS( elem, name, computed )

* + - 1. rnumnonpx---------------(L6071)

// /^([+-]?(?:\d\*\.|)\d+(?:[eE][+-]?\d+|))(?!px)[a-z%]+$/i  
// 单位不是px  
var rnumnonpx = new RegExp( "^(" + pnum + ")(?!px)[a-z%]+$", "i" );

* + - 1. curCSS源码分析-----------(L6167)

/\*\*  
 \* curCSS获取的是浏览器计算后的最终值  
 \* 通过"getComputedStyle"和“currentStyle”功能来获取样式计算后的值；  
 \* 因为elem.style获取的只是内联样式  
 \* @param elem 当前要处理的元素  
 \* @param name 需要获得哪个属性的计算样式值  
 \* @param computed 可以通过computed指定样式对象替代内部的getStyle  
 \* @return {string}  
 \*/  
function curCSS( elem, name, computed ) {  
 var width, minWidth, maxWidth, ret,  
  
 // Support: Firefox 51+  
 // 在计算样式时，先调用elem.style用于修复某些元素get到错误值的问题  
 style = elem.style;  
 // 获得元素elem的计算样式值，并对getComputedStyle函数有问题浏览器进行兼容处理  
 computed = computed || getStyles( elem );  
  
 // getPropertyValue is needed for:  
 // .css('filter') (IE 9 only, #12537)  
 // .css('--customProperty) (#3144)  
 // 能获取到elem的计算样式属性  
 if ( computed ) {  
 // getPropertyValue方法可以获取CSS样式申明对象上的属性值（直接属性名称）,必须-连接书写，否则返回""  
 // name无需驼峰标记法，也不用对float进行转换，直接是样式表的值即可，如backgroun-color  
 // 先使用getPropertyValue方法获取样式值是为了兼容IE9下获取filter样式。没有这个方法，才使用computed[name]获取样式值。  
 ret = computed.getPropertyValue( name ) || computed[ name ];  
  
 if ( ret === "" && !jQuery.contains( elem.ownerDocument, elem ) ) {  
 ret = jQuery.style( elem, name );  
 }  
  
 // 此方法来源于Dean Edwards--并不十分理解？？？？？？？？？？？  
 // 为了兼容有的浏览器margin相关方法返回百分比等非px值的情况，由于width输出是px，并且margin的百分比是按照width计算的，  
 // 因此可以直接赋值width。设置minWidth/maxWidth是为了保证设置的width不会因为超出限制失效  
 // https://drafts.csswg.org/cssom/#resolved-values  
 if ( !support.pixelMarginRight() && rnumnonpx.test( ret ) && rmargin.test( name ) ) {  
  
 // 记录原始值  
 width = style.width;  
 minWidth = style.minWidth;  
 maxWidth = style.maxWidth;  
  
 // 传入新值后，再计算width  
 style.minWidth = style.maxWidth = style.width = ret;  
 ret = computed.width;  
  
 // 回复值  
 style.width = width;  
 style.minWidth = minWidth;  
 style.maxWidth = maxWidth;  
 }  
 }  
  
 return ret !== undefined ?  
  
 // Support: IE <=9 - 11 only  
 // 处理IE对zIndex返回integer类型的bug  
 ret + "" :  
 ret;  
}

### function getWidthOrHeight( elem, name, extra )--------(L6345)

* + - 1. 因为$().css调用方式extra为undefined，故对于此函数的作用理解不深刻，之后再细看，此函数主要是针对box-sizing进行处理

// 获取宽高  
function getWidthOrHeight( elem, name, extra ) {  
  
 // 先获得计算样式  
 var valueIsBorderBox,  
 styles = getStyles( elem ),  
 val = curCSS( elem, name, styles ),  
 // boxSizing:用于计算元素宽度和高度的默认的 CSS 盒子模型  
 // content-box标准盒子模型  
 // border-box:width与height包括内容，内边距和边框，但不包括外边距  
 isBorderBox = jQuery.css( elem, "boxSizing", false, styles ) === "border-box";  
  
 // 单位不是px，则直接返回  
 if ( rnumnonpx.test( val ) ) {  
 return val;  
 }  
  
 // 如是border-box，再检查下，防止有些浏览器getComputedStyle 返回不可靠值，出现错误的box-sizing值  
 // 如是border-box，elem内联无height/width信息，则对于chrome会返回false  
 valueIsBorderBox = isBorderBox &&  
 ( support.boxSizingReliable() || val === elem.style[ name ] );  
  
 // 内联元素没有height/width的为auto时，使用offsetWidth(gh-3571)  
 // offsetWidth:返回元素的可见宽度，宽度的单位为画素（pixels)。该宽度包括 内缘（padding）、线框（border） 和滚动条尺寸。  
 if ( val === "auto" ) {  
 val = elem[ "offset" + name[ 0 ].toUpperCase() + name.slice( 1 ) ];  
 }  
  
 // 通过parseFloat将val转换成纯数值或0  
 val = parseFloat( val ) || 0;  
  
 // 使用合适的 box-sizing 模型增加或减少相关的样式值（因为box-sizing不同，width，height是不同的）  
 return ( val +  
 // 对于$().css()函数调用，extra传过来的都是undefined或false  
 augmentWidthOrHeight(  
 elem,  
 name,  
 extra || ( isBorderBox ? "border" : "content" ),  
 valueIsBorderBox,  
 styles  
 )  
 ) + "px";  
}

### jQuery.cssHooks[name].get------(L6542)

get: function( elem, computed, extra ) {  
 if ( computed ) {  
  
 // 一些元素虽然不可见，但是是可以有宽高的，但是最好是有一个具体的样式属性值  
 return rdisplayswap.test( jQuery.css( elem, "display" ) ) &&  
  
 // Support: Safari 8+  
 // Table columns在Safari浏览器具有非0的offsetWidth值  
 // 并且，如display不改变，getBoundingClientRect().width 一直为0  
 // Support: IE <=11 only  
 // 调用 getBoundingClientRect 在一个非DOM节点元素时，在IE会抛出错误  
 // elem.getClientRects()返回元素相关的CSS边框信息，如bottom，left，width等  
 // getBoundingClientRect()返回元素左，上，右和下分别相对浏览器视窗的位置，里面right是指元素右边界距窗口最左边的距离，bottom是指元素下边界距窗口最上面的距离。  
 // 对于最简单的display:none,则都为0；但实际是有宽高的，处理方式是将隐藏的elem元素，设置cssShow属性，得到宽高，然后再将设置的属性还原  
 ( !elem.getClientRects().length || !elem.getBoundingClientRect().width ) ?  
 swap( elem, cssShow, function() {  
 return getWidthOrHeight( elem, name, extra );  
 } ) :  
 getWidthOrHeight( elem, name, extra );  
 }  
},

### jQuery.cssHooks[name].set------(L6563)

// 对于类似 $('#a').css('width','200px')调用，extra传入为undefined  
set: function( elem, value, extra ) {  
 var matches,  
 styles = extra && getStyles( elem ),//因此styles为undefined  
 subtract = extra && augmentWidthOrHeight(  
 elem,  
 name,  
 extra,  
 jQuery.css( elem, "boxSizing", false, styles ) === "border-box",  
 styles  
 );  
  
 // 如需要值调整，则转换为px  
 if ( subtract && ( matches = rcssNum.exec( value ) ) &&  
 ( matches[ 3 ] || "px" ) !== "px" ) {  
  
 elem.style[ name ] = value;  
 value = jQuery.css( elem, name );  
 }  
 // 对value值进行调整，并设置有效值  
 return setPositiveNumber( elem, value, subtract );  
}

### function setPositiveNumber( elem, value, subtract )--------------(L6287)

// 对value值进行调整，并设置有效值

// 保证非负值，保留单位，subtract可以指定需要减去的值  
function setPositiveNumber( elem, value, subtract ) {  
  
 // 任何+123或-123都再这进行处理  
 var matches = rcssNum.exec( value );  
 return matches ?  
  
 // ( subtract || 0 )避免cssHooks使用时，subtract出现undefined值  
 // Math.max是保证不让有些属性值为负  
 Math.max( 0, matches[ 2 ] - ( subtract || 0 ) ) + ( matches[ 3 ] || "px" ) :  
 value;  
}

## jQuery.cssHooks.marginLeft

### function addGetHookIf( conditionFn, hookFn )------(L6221)

// 根据conditionFn()结果判定是否需要hook，如需要则将hook.get=hookFn  
function addGetHookIf( conditionFn, hookFn ) {  
  
 // 定义这个hook，如果真的需要，jQuery会在第一次运行时检查  
 return {  
 get: function() {  
 if ( conditionFn() ) {  
  
 // 如不需要hook，或者由于依赖不能使用hook，删除get方法  
 delete this.get;  
 return;  
 }  
  
 // 如需要hook，则重新定义hook，避免support测试不再执行  
 return ( this.get = hookFn ).apply( this, arguments );  
 }  
 };  
}

### 源码分析--------------(L6587)

// support.reliableMarginLeft,浏览器都返回true，有些老的浏览器返回false  
// 正常浏览器，width=0，marginleft=0，获取的marginleft应为0  
jQuery.cssHooks.marginLeft = addGetHookIf( support.reliableMarginLeft,  
 function( elem, computed ) {  
 if ( computed ) {  
 return ( parseFloat( curCSS( elem, "marginLeft" ) ) ||  
 elem.getBoundingClientRect().left -  
 swap( elem, { marginLeft: 0 }, function() {  
 return elem.getBoundingClientRect().left;  
 } )  
 ) + "px";  
 }  
 }  
);

### jQuery.cssHooks.margin/padding/border-----------(L6600)

* + - 1. animate再补充分析

// 这些hooks为了在animate中扩展属性使用的  
jQuery.each( {  
 margin: "",  
 padding: "",  
 border: "Width"  
}, function( prefix, suffix ) {  
 jQuery.cssHooks[ prefix + suffix ] = {  
 expand: function( value ) {  
 var i = 0,  
 expanded = {},  
  
 // Assumes a single number if not a string  
 parts = typeof value === "string" ? value.split( " " ) : [ value ];  
  
 for ( ; i < 4; i++ ) {  
 expanded[ prefix + cssExpand[ i ] + suffix ] =  
 parts[ i ] || parts[ i - 2 ] || parts[ 0 ];  
 }  
  
 return expanded;  
 }  
 };  
  
 if ( !rmargin.test( prefix ) ) {  
 jQuery.cssHooks[ prefix + suffix ].set = setPositiveNumber;  
 }  
} );

## jQuery.cssHooks.left，jQuery.cssHooks.top

* + - 1. 此hooks是根据jQuery.fn.postion构建的， 因此在（章节：位置尺寸操作3.5分析）