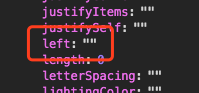
**从一个样式不生效简单探讨jQuery.cssHooks**

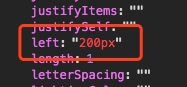
曾经在做一个定制的时候，需要调整页面布局，但是通过id找到元素，通过jQuery赋值后，样式尽然不起作用，**问题抽象如下**

$('#' + id).css({"left": "200"})

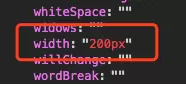
用上述语句设置left值，后打印元素属性如下



发现left为空，刚开始以为id弄错了或者新增js代码没有执行，打断点定位元素，都没发现明显问题，为了把问题单一话，排除其他可能的干扰，折腾了一下，发现如果写成$('#' + id).css({"left": 200})却可以生效



看来字符串和数字的区别引起的了，出于好奇，对width也测试了一次，发现width设置为字符串尽然是可以的，$('#' + id).css({"width": "200"})如下图所示



那么为什么width设定为字符串就可以添加px后缀，而left却不可以，此时问题就集中到**jQuery.fn.css方法（都是通过该方法实现的）设定样式时，为什么string类型的值在某些属性上无法生效？此时我们找到源码，通过查看源码来尝试寻找线索**

先找一份没有压缩过的jQuery库，（附件有一份可以参考）：

先找到我们本次设定样式的方法jQuery.fn.css：（附件源码6656行）

jQuery.fn.extend( {

css: function( name, value ) {

return access( this, function( elem, name, value ) {

var styles, len,

map = {},

i = 0;

if ( Array.isArray( name ) ) {

styles = getStyles( elem );

len = name.length;

for ( ; i < len; i++ ) {

map[ name[ i ] ] = jQuery.css( elem, name[ i ], false, styles );

}

return map;

}

return value !== undefined ?

**jQuery.style( elem, name, value ) :**

jQuery.css( elem, name );

}, name, value, arguments.length > 1 );

}

} );

此时我们通过Chrome浏览器来调试源码（直接看源码太繁琐，直接打断点看清晰一些）

进入jQuery.style之后就会来到最终产生区别的地方：

附件源码6441行

style: function( elem, name, value, extra ) {

...

hooks = jQuery.cssHooks[ name ] || jQuery.cssHooks[ origName ];

if ( value !== undefined ) {

type = typeof value;

if ( type === "string" && ( ret = rcssNum.exec( value ) ) && ret[ 1 ] ) {

value = adjustCSS( elem, name, ret );

type = "number";

}

...

if ( type === "number" ) {

**value += ret && ret[ 3 ] || ( jQuery.cssNumber[ origName ] ? "" : "px" );**

}

...

if ( !hooks || !( "set" in hooks ) ||( value = hooks.set( elem, value, extra ) ) !== undefined ) {

//此时的value到底是200还是200px;只有添加了后缀才能赋值成功

if ( isCustomProp ) {

style.setProperty( name, value );

} else {

style[ name ] = value;

}

}

}

...

},

从源码中可以看到对string和number做了区分，number类型，会自动添加px后缀；

**但是为什么left和width均传入string一个生效一个不生效哪？**我们继续看源码：

hooks = jQuery.cssHooks[ name ] || jQuery.cssHooks[ origName ];

...

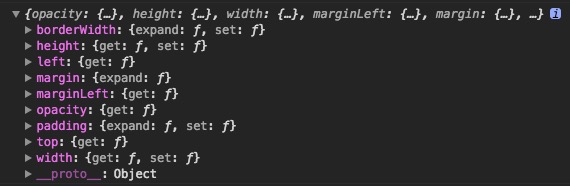
if ( !hooks || !( "set" in hooks ) ||

( value = **hooks.set( elem, value, extra )** ) !== undefined ) {

...

}

可以看出value是string类型，到最终赋值之前，还要通过value = hooks.set( elem, value, extra ) ) !== undefined这一层的判断，也就是说如果hooks.set方法存在，还有机会通过该方法将string类型的value进行后缀补全。此时有必要看看hooks，而hooks是由jQuery.cssHooks得到的



从上图可以看出left只有get；width有get和set。此时可以推断由于width存在了set方法，在其方法中对string类型的value完成了后缀的补齐，而left没有。

**为什么width要对其设定**set**函数？**

答案可以从其set方法来窥探一下（附件源码6579行）：

set: function( elem, value, extra ) {

var matches,

styles = getStyles( elem ),

isBorderBox = jQuery.css( elem, "boxSizing", false, styles ) === "border-box",

subtract = extra && boxModelAdjustment(

elem,

dimension,

extra,

isBorderBox,

styles

);

if ( isBorderBox && support.scrollboxSize() === styles.position ) {

subtract -= Math.ceil(

elem[ "offset" + dimension[ 0 ].toUpperCase() + dimension.slice( 1 ) ] -

parseFloat( styles[ dimension ] ) -

boxModelAdjustment( elem, dimension, "border", false, styles ) -

0.5

);

}

if ( subtract && ( matches = rcssNum.exec( value ) ) &&

( matches[ 3 ] || "px" ) !== "px" ) {

elem.style[ dimension ] = value;

value = jQuery.css( elem, dimension );

}

return setPositiveNumber( elem, value, subtract );

}

可见对width做特殊处理是因为css的盒模型有content-box|border-box|inherit，故对外为了统一，从而增加了set方法。顺带着把px补全了。而left没需要兼容，也就没有设定set方法，自然也不会去补全。