**任务一：表单（一）单个表单项的检验**

## 一. 基础知识

1. 在js中字符串使用unicode来表示的，并且js使用的是[UTF-16](https://zh.wikipedia.org/wiki/UTF-16)（UTF-16是Unicode字符编码五层次模型的第三层）。
2. 这样来说大部分字符都是16bit编码。
3. 但是也有较少的部分是32bit，这是由于完整的unicode字符集是需要21bit来标识的。

## 二. 如何判断中英文字符

按照题目的要求，只考虑中文与英文即可，不考虑其他语言。这样编码在 0x00-0xFF, 范围为英文，其他的认作中文。

常见的几种方式是：  
1） 在循环中使用 String.charCodeAt() 拿到字符编码值， 如果值在 0x00-0xFF 区间就len++， 其他则 len += 2。  
2） 通过正则表达式将0x00-0xFF区间以外的替换为任意两个英文字符，例如：String.replace(/[^x00-xff]/g,'AB')，这样后再取String.length 即可得到结果。

## 三. 如何判断32bit编码的中文字符

以上的判断均没有考虑 32bit 编码的中文字符， 其实使用 32bit 编码的中文字符属于不常用字符。

要能够正确识别，需对以上的方法做改进。但首先要知道 UTF-16 的编码规则，详细的内容可以看 [一. 基础知识] 中的链接。**简单讲就是使用32bit编码的， 前16bit 一定在0xD800-0xDBFF 区间， 后16bit 一定在 0xDC00-0xDFFF 区间。**

知道了原理后我们对 [ 二 ] 中的方法改进：  
1）只需要在循环内增加判断 0xD800-0xDBFF 区间的 len += 2; i++ [注：使用的是 for 循环: for (let i = 0; i < String.length; i++) { ... }]; 即跳过下一个16bit 。  
2）0xD800-0xDBFF 加上 0xDC00-0xDFFF， 得到 0xD800-0xDFFF。我们可以先替换 32bit 字符， 即先执行 String.replace(/[xd800-xffff]/g, 'A');然后再执行String.replace(/[^x00-xff]/g,'AB') ，再用 String.length 取得结果。

另外，ES6中对字符串有扩展，可以参考 [ECMAScript6入门-字符串的扩展](http://es6.ruanyifeng.com/#docs/string) 。根据新语法我们又可以这样来做：

**function** **getStrLen** (str)

**var** **enLen** = 0;

**var** zhLen = 0;

**for** (**let** ch of str) {

**if** (isASCII(ch))

enLen++;

**else**

zhLen++;

}

**return** enLen + zhLen \* 2;

}

**function** **isASCII** (c) {

**return** c.codePointAt(0) <= 0xFF;

}

备注：对于 32bit 字符判断，我有写一个[demo](http://codepen.io/alvinhuang/pen/gmarKx?editors=1010), 用来返回真正的字符串长度(String.length 会将32bit 字符当作两个字符)。

赞48

## 任务四：听指令的小方块（一）

# 从任务看 Javascript 获取 CSS 样式

一点小总结

## 问题

在[任务四 听指令的小方块（一）](http://ife.baidu.com/course/detail/id/109)中，要求使用输入的指令来移动小方块。一看题目就觉得很简单，不就是把小方块用 position: absolute; 定位，再用 js 来改变 left | top 属性的值就可以了吗？

有了想法就开始行动，我写了如下代码：

HTML：

<td>

<**div** class="block" id="block">

</**div**>

</td>

CSS：

.block {

position: absolute;

left: 0;

top: 0;

}

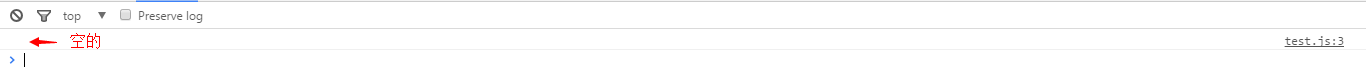
为了避免出错，我又写了下面的 js 看看效果（没错，我就是这么细心~~）

window.onload = **function** () {

**var** block = document.getElementById("block");

console.log(block.style.left); *// 输出空字符串*

};

结果，打开控制台，结果发现是这个样子。  


what？浏览器怎么不按套路出牌？不应该输出 0px 吗？一定是浏览器坏了。。。嗯。。。

## 原因

其实原因 hin 简单，在这里浏览器只输出了一个空字符串，于是就变成图中的样子咯。（别问我是怎么知道他是个空字符串的，这是一个很长的故事 ┭┮﹏┭┮）

这个现象的根本原因在于在上面的 js 代码里面我们用到了 block.style.left 。

正如我们都知道的，我们在 HTML 中可以通过 <link> 标签、<style> 标签和元素的 style 属性来给元素添加样式。而我们平时访问元素的 CSS 样式时使用的 element.style 其实访问的是元素的 style 属性，也就是元素的**行内样式**。而上面的代码中，我是将样式写在 <link> 标签引入的样式表文件里的，于是 style 属性自然访问不到，因此 console.log(block.style.left) 自然就输出了空字符串。

## 解决方法

那么怎么获取小方块通过样式表层叠而来的 CSS 样式信息呢？我们可以利用 DOM2 级提供的 getComputedStyle() 方法。

在“DOM2 级样式”中提供了 getComputedStyle() 方法。这个方法接受两个参数：要取得计算样式的元素和一个伪元素字符串（例如":after"）。如果不需要伪元素信息，第二个参数可以是 null。getComputedStyle() 方法返回一个 CSSStyleDeclaration 对象（与 style 属性的类型相同），其中包含当前元素的所有计算的样式。———— 以上来自《JavaScript 高级程序设计》

简言之，就是要获取元素来自层叠样式表的 CSS 样式的话，可以用 window.getComputedStyle(目标元素, null) 来代替我们上面使用的 目标元素.style 。

于是我把代码改成了这样：

window.onload = **function** () {

**var** block = document.getElementById("block");

console.log(window.getComputedStyle(block, null).left); *// 输出 “0px”*

};

打开浏览器查看结果：  


值得注意的是，IE7、8、9 不支持 getComputedStyle() 方法，但它有一种类似的概念。在 IE 中，每个具有 style 属性的元素还有一个 currentStyle 属性。这个属性是 CSSStyleDeclaration 的实例，包含当前元素全部计算后的样式。可以按 目标元素.currentStyle 来使用。

window.onload = **function** () {

**var** block = document.getElementById("block");

console.log(block.currentStyle.left); *// 输出“0px”*

};

兼容的写法，用一个函数来屏蔽浏览器差异：

*// 兼容的写法*

**function** **getStyle** (obj,attr) {

**return** obj.currentStyle ? obj.currentStyle[attr]:getComputedStyle(obj)[attr];

}

window.onload = **function** () {

**var** block = document.getElementById("block");

console.log(getStyle(block, "left")); *// 输出“0px”*

};

## 最后

忘了说一句，无论在哪个浏览器中，最重要的一条是要记住所有计算的样式都是**只读的**。

**只读的**。  
**读的**。。  
**的**。。。

**w(ﾟДﾟ)w 啊啊啊啊!!!!!**  
**只读我该怎么改变它的位置啊！！！！**

还好，DOM2 级拓展还提供了操作样式的 API。不过，**相 当 繁 杂**。限于篇幅，我就不提了（其实是懒得写了）。

**Σ(っ °Д °;)っ 等等！那怎么改变位置啊？**

很简单，机智的我把样式写在 style 属性里了。

<td>

<**div** class="block" id="block" style="position: absolute;left: 0;top: 0;">

</**div**>

</td>

这样就可以用原来的 js 访问 left 了。**目标元素.style 是可读可写的**。

赞9

## 任务五：听指令的小方块（二）

# 问题描述

[任务五：听指令的小方块（二）](http://ife.baidu.com/course/detail/id/112)中有实现转动过渡效果的要求。如直接转动小方块，小方块的四个角在转动过程中必定会突出棋盘格，影响美观。若能在转动前将给小方块加个圆角 border-radius: 50%，旋转动画后再改回 border-radius: 0，岂不妙哉~

## 一. 用Animation实现

既然圆角需要先从0到50%，再从50%到0，首先想到的就是可以实现往返效果的animation了。代码如下  
CSS

*//朝上的情况下左转的动画*

@keyframes tl {

0% {

border-radius: 0;

transform: rotate(0deg);

}

33% {

border-radius: 50%;

transform: rotate(0deg);

}

67% {

border-radius: 50%;

transform: rotate(-90deg);

}

100% {

border-radius: 0;

transform: rotate(-90deg);

}

}

*//设置通用的动画CSS*

animation-duration: 3s;

animation-fill-mode: forwards;

* 其他朝向不同转动方向的动画就不列举了，可以通过修改rotate的角度实现
* 当然，也可以只设置一个rotate动画，然后在JS中修改动画效果

JS

*//取得小方块*

**var** **box** = …

*//当小方块朝上，向左转时*

style.animationName = “tl”;

* 其他朝向不同转动方向的animation赋值就不列举了
* forwards是必须的，表示转动后显示动画100%的状态。如果不加forwards，则需要在给box加一个transform: rotate(N deg)的CSS，否则box会在转动后变为朝上。

## 二. 用Transition实现

Trasnsition，可以显示某个元素从状态1到状态2的过渡动画，而且与Animation相比，需要的代码量**更少**。但如果需要再从状态2返回状态1，单纯用CSS好像无法实现，需要借助JS。  
先来实现从状态1到状态2吧，从方角到圆角，然后再转个向。代码如下  
CSS

transition-property: transform, border-radius;

transition-**time**: 1s, 1s;

transition-delay: 1s, 0s;

JS

//取得小方块

var box = …

//设置CSS

box.style.transform = "rotate(" + N\*90 + "deg)";

box.style.borderRadius = “50%”;

至此，状态1到状态2已经实现了。接下来就是状态2到状态1了。  
从状态1到状态2设置了1s+1s的过渡时间，那么在2s末，需要将border-radius的值修改为0。延迟2s运行某些代码，很自然的就能想到 setTimeout。代码就不列了。

其实，我不是来推销setTimeout的，我是来安利transitionend事件的——  
从字面意思来理解，就可以知道transitionend事件是在过渡动画完成时触发的。transitionend事件不仅拥有target、type等一般事件属性，还拥有3个特有的属性：propertyName、elapsedTime、pseudoElement，分别表示过渡动画的属性名称(CSS中设置的transition-property)、过渡动画经历的时长以及伪元素名称(如果该过渡动画显示在伪元素上)，具体可以参考[这里](https://developer.mozilla.org/en-US/docs/Web/Events/transitionend)。  
用transitionend事件实现目标的原理也很简单：监听transitionend事件，如果该事件是由transform引起的，则改变borderRadius的值。代码如下

box.addEventListener("transitionend", **function**(e) {

**if** (e.propertyName === "transform"){

box.style.borderRadius = "0";

}

});

延伸一下：既然有transitionend事件，是否会有transitionstar事件呢？答案是，有的，而且不仅仅只有transitionstar，还会有transitionrun呢！！！更令人惊讶的是，这两个事件会出现在Firefox 53中，嗯，或许你看到这篇文章的时候，53版本还没有发布...

个人理解，从时效上讲，用transitionend事件还是比用setTimeout靠谱一些的。

## 三. Animation和Transition的异同

我也不是很清楚，查阅了[MDN](https://developer.mozilla.org/)和[stackOverflow](http://stackoverflow.com/)，也没找出个所以然来。我就参考别人总结的和自己理解的，谈一谈两者在表象上的异同吧。

#### 两者的相似之处

之所以要谈两者的异同，是因为两者比较相近，那么就先来看看两者的相近之处吧。  
1． 两者能产生动画效果的property相同，而且都要事先定义需要产生效果的property  
2． 两者能接收的timing-function相同，默认值也相同  
3． 两者都能在动画前产生延迟  
4． 两者都有相应的开始和结束事件  
5． 最大的相似之处，两者都能产生动画，且动画效果在视觉上相同

#### 两者的不同之处

1. 两者的动画的触发形式不同，并由此导致Animation动画播放后，会默认回到动画前的状态
2. Animation可以轻松实现循环播放、倒序播放、暂停播放等功能，Transition不能
3. Animation可以轻松实现多步走，既从状态1到状态2再到状态3等等，Transition则需借助于JS

小结：两者有较多的相似之处，且仅用比较简单的JS语句就可以消除两者的不同之处(Animation的暂停播放貌似是个例外)，怪不得网上有人说Animation∈Transition，又有人说Transition∈Animation。我都开始怀疑在底层这两者是不是同一个东西。  
关于异同，这篇[文章](https://www.kirupa.com/html5/css3_animations_vs_transitions.htm)写得不错，告诉大家哪种情况下用哪个代码少比较好，大家可以看看。

赞6