目录

[DOM选取 1](#_Toc496028380)

[网页性能管理详解 2](#_Toc496028381)

[Jquery事件on 2](#_Toc496028382)

[one 3](#_Toc496028383)

[$.type v1.0+ 3](#_Toc496028384)

[css 3](#_Toc496028385)

[XMLHttpRequest 2进度信息 4](#_Toc496028386)

[<script>标签打开defer或async属性 5](#_Toc496028387)

[css3不定宽高水平垂直居中 5](#_Toc496028388)

[运用margin:auto进行垂直居中 5](#_Toc496028389)

[固定宽高 5](#_Toc496028390)

[不固定宽高 6](#_Toc496028391)

[加载进度条 6](#_Toc496028392)

[三角按钮 7](#_Toc496028393)

[不用jq的隐藏和显示 12](#_Toc496028394)

# DOM选取

function query(selector) {

return Array.from(document.querySelectorAll(selector));

}

var query = document.querySelector.bind(document);

var queryAll = document.querySelectorAll.bind(document);

var $ = document.querySelectorAll.bind(document);

var fromId = document.getElementById.bind(document);

var fromClass = document.getElementsByClassName.bind(document);

var fromTag = document.getElementsByTagName.bind(document);

Array.prototype.map.call(document.querySelectorAll('button'),function(element,index){

element.onclick = function(){

}

})

**边框宽度不允许使用百分比值，width：100%**

当父容器里有绝对定位的子元素时，子元素设置width:100%实际上指的是相对于父容器的padding+content的宽度。当子元素是非绝对定位的元素时width:100%才是指子元素的 content ，其等于父元素的 content宽度。

**ex和 ch单位**

ex：取当前作用效果的字体的x的高度，在无法确定x高度的情况下以0.5em计算；

ch:以节点所使用字体中的“0”字符为基准，找不到时为0.5em；

样式的写操作之后，如果有下面这些属性的读操作，都会引发浏览器立即重新渲染。

offsetTop/offsetLeft/offsetWidth/offsetHeight

scrollTop/scrollLeft/scrollWidth/scrollHeight

clientTop/clientLeft/clientWidth/clientHeight

getComputedStyle()

# 网页性能管理详解

"生成布局"（flow）和"绘制"（paint）这两步，合称为"渲染"（render）。

window.requestAnimationFrame()

还有一个函数window.requestIdleCallback()，也可以用来调节重新渲染。

它指定只有当一帧的末尾有空闲时间，才会执行回调函数。

# Jquery事件on

on(type, [selector], function(e){ ... }) ⇒ self

on(type, [selector], [data], function(e){ ... }) ⇒ self v1.1+

on({ type: handler, type2: handler2, ... }, [selector]) ⇒ self

on({ type: handler, type2: handler2, ... }, [selector], [data]) ⇒ self v1.1+

添加事件处理程序到对象集合中得元素上。多个事件可以通过空格的字符串方式添加，或者以事件类型为键、以函数为值的对象 方式。如果给定css选择器，当事件在匹配该选择器的元素上发起时，事件才会被触发（愚人码头注：即事件委派，或者说事件代理）。

如果false 在回调函数的位置上作为参数传递给这个方法， 它相当于传递一个函数，这个函数直接返回false。（愚人码头注：即将 false 当作 function(e){ ... } 的参数，作为 function(){ return false; } 的简写形式，例如：$("a.disabled").on("click", false);这相当于$("a.disabled").on("click", function(){ return false; } );）

**var** elem = $('#content')

// observe all clicks inside #content:

elem.on('click', **function**(e){ ... })

// observe clicks inside navigation links in #content

elem.on('click', 'nav a', **function**(e){ ... })

// all clicks inside links in the document

$(document).on('click', 'a', **function**(e){ ... })

// disable following any navigation link on the page

$(document).on('click', 'nav a', **false**)

### one

one(type, [selector], function(e){ ... }) ⇒ self

one(type, [selector], [data], function(e){ ... }) ⇒ self v1.1+

one({ type: handler, type2: handler2, ... }, [selector]) ⇒ self

one({ type: handler, type2: handler2, ... }, [selector], [data]) ⇒ self v1.1+

添加一个处理事件到元素，当第一次执行事件以后，该事件将自动解除绑定，保证处理函数在每个元素上最多执行一次。selector 和 data 等参数说明请查看[.on()](http://www.runoob.com/manual/zeptojs.html#on)。

# $.type v1.0+

$.type(object) ⇒ string

获取JavaScript 对象的类型。可能的类型有： null undefined boolean number string function array dateregexp object error

# css

css(property) ⇒ value

css([property1, property2, ...]) ⇒ object v1.1+

css(property, value) ⇒ self

css({ property: value, property2: value2, ... }) ⇒ self

读取或设置DOM元素的css属性。当value参数不存在的时候，返回对象集合中第一个元素的css属性。当value参数存在时，设置对象集合中每一个元素的对应css属性。

多个属性可以通过传递一个属性名组成的数组一次性获取。多个属性可以利用对象键值对的方式进行设置。

当value为空(空字符串，null 或 undefined)，那个css属性将会被移出。当value参数为一个无单位的数字，如果该css属性需要单位，“px”将会自动添加到该属性上。

# XMLHttpRequest 2进度信息

新版本的XMLHttpRequest对象，传送数据的时候，有一个progress事件，用来返回进度信息。

它分成上传和下载两种情况。下载的progress事件属于XMLHttpRequest对象，上传的progress事件属于XMLHttpRequest.upload对象。

我们先定义progress事件的回调函数。

　　xhr.onprogress = updateProgress;

xhr.upload.onprogress = updateProgress;

el.insertAdjacentHTML('beforebegin', htmlString);  
el.insertAdjacentHTML('afterend', htmlString);

$(el).clone(); el.cloneNode(true);

$.contains(el, child); el!== child && el.contains(child);

document.querySelectorAll('.my #awesome selector');

getComputedStyle(el)[ruleName];

el.textContent || el.innerText

if (el.classList)

el.classList.contains(className);

else

new RegExp('(^| )' + className + '( |$)', 'gi').test(el.className);

$(el).is('.my-class');

$(el).replaceWith(string); el.outerHTML = string;

if (el.textContent !== undefined)

el.textContent = string;

else

el.innerText = string;

# <script>标签打开defer或async属性

<script src="path/to/myModule.js" defer></script>

<script src="path/to/myModule.js" async></script>

上面代码中，<script>标签打开defer或async属性，脚本就会异步加载。渲染引擎遇到这一行命令，就会开始下载外部脚本，但不会等它下载和执行，而是直接执行后面的命令。

defer与async的区别是：前者要等到整个页面正常渲染结束，才会执行；后者一旦下载完，渲染引擎就会中断渲染，执行这个脚本以后，再继续渲染。一句话，defer是“渲染完再执行”，async是“下载完就执行”。另外，如果有多个defer脚本，会按照它们在页面出现的顺序加载，而多个async脚本是不能保证加载顺序的。

# css3不定宽高水平垂直居中

只要三句话就可以实现不定宽高水平垂直居中。

justify-content:center;//子元素水平居中

align-items:center;//子元素垂直居中

display:-webkit-flex;

## 运用margin:auto进行垂直居中

## 固定宽高

position: absolute;

left: 50%;

top: 50%;

width:200px;

height:100px;

margin-left:-100px;//元素一半

margin-top:-50px;

**借助table**

display: table-cell;

text-align: center;

vertical-align: middle;

## 不固定宽高

<style>

.center {

position: fixed;

top: 50%;

left: 50%;

background-color: #000;

width:50%;

height: 50%;

-webkit-transform: translateX(-50%) translateY(-50%);

}

</style>

# 加载进度条

**最外面遮罩**

Position:fixed;top:0px;left:0px;background:#fff;height:100%;width:100%;z-index:999

**document.readyState**:判断文档是否加载完成。firefox不支持。

这个属性是只读的，传回值有以下的可能：

0-UNINITIALIZED：XML 对象被产生，但没有任何文件被加载。

1-LOADING：加载程序进行中，但文件尚未开始解析。

2-LOADED：部分的文件已经加载且进行解析，但对象模型尚未生效。

3-INTERACTIVE：仅对已加载的部分文件有效，在此情况下，对象模型是有效但只读的。

**4-COMPLETED：文件已完全加载，代表加载成功。//主要是这个**

**右键单击事件：document.oncontextmenu**

# 三角按钮

我们的网页因为 CSS 而呈现千变万化的风格。这一看似简单的样式语言在使用中非常灵活，只要你发挥创意就能实现很多比人想象不到的效果。特别是随着 CSS3 的广泛使用，更多新奇的 CSS 作品涌现出来。  
  
今天给大家带来 CSS 三角形绘制方法  


复制代码 代码如下:

#triangle-up {  
    width: 0;  
    height: 0;  
    border-left: 50px solid transparent;  
    border-right: 50px solid transparent;  
    border-bottom: 100px solid red;  
}



复制代码 代码如下:

#triangle-down {  
    width: 0;  
    height: 0;  
    border-left: 50px solid transparent;  
    border-right: 50px solid transparent;  
    border-top: 100px solid red;  
}



复制代码 代码如下:

#triangle-left {  
    width: 0;  
    height: 0;  
    border-top: 50px solid transparent;  
    border-right: 100px solid red;  
    border-bottom: 50px solid transparent;  
}



复制代码 代码如下:

#triangle-right {  
    width: 0;  
    height: 0;  
    border-top: 50px solid transparent;  
    border-left: 100px solid red;  
    border-bottom: 50px solid transparent;  
}



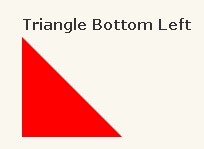
复制代码 代码如下:

#triangle-topleft {  
    width: 0;  
    height: 0;  
    border-top: 100px solid red;  
    border-right: 100px solid transparent;  
}



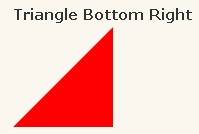
复制代码 代码如下:

#triangle-topright {  
    width: 0;  
    height: 0;  
    border-top: 100px solid red;  
    border-left: 100px solid transparent;   
}



复制代码 代码如下:

#triangle-bottomleft {  
    width: 0;  
    height: 0;  
    border-bottom: 100px solid red;  
    border-right: 100px solid transparent;  
}



复制代码 代码如下:

#triangle-bottomright {  
    width: 0;  
    height: 0;  
    border-bottom: 100px solid red;  
    border-left: 100px solid transparent;  
}

# 不用jq的隐藏和显示

例如将元素隐藏，你或许会用display:none。

（这里插一句题外话，用display:none隐藏容易显示难，如果你用的是JQuery等插件，你或许会疑惑，直接用show/hide API不就行了，难在哪里？其中一个难点就是保存隐藏前元素的display属性值。例如A隐藏前display:block，B隐藏前display:inline，A和B都改成none隐藏后，要显示出来时，你必须事先保存元素的display属性值，否则做不到显示后display仍旧是原先的值。而这些工作JQuery插件都替你做好了，才让你产生了隐藏显示很容易的错觉。）

其实我更推荐的是absolute控制隐藏和显示。方法当然相当简单，如absolute+ top:-9999em，或absolute + visibility:hidden。

优点是absolute由于层级的关系，隐藏和显示只会重绘，但不会回流（其实我对absolute不会导致回流这一观点是持怀疑态度的，说不会回流显得过于武断，应该是absolute的回流开销小于正常DOM流中回流的开销。但我战斗力不够，尚未找到靠谱合理的检测回流的方法）。而用display:none会导致render tree重绘和回流。

另外，考虑到重绘和回流的开销，可以将动画效果放到absolute元素中，避免浏览器将render tree回流。

套接字（socket）是通信的基石，是支持TCP/IP协议的网络通信的基本操作单元。它是网络通信过程中端点的抽象表示，包含进行网络通信必须的五种信息：连接使用的协议，本地主机的IP地址，本地进程的协议端口，远地主机的IP地址，远地进程的协议端口。

应用层通过传输层进行数据通信时，TCP会遇到同时为多个应用程序进程提供并发服务的问题。多个TCP连接或多个应用程序进程可能需要通过同一个 TCP协议端口传输数据。为了区别不同的应用程序进程和连接，许多计算机操作系统为应用程序与TCP／IP协议交互提供了套接字(Socket)接口。应 用层可以和传输层通过Socket接口，区分来自不同应用程序进程或网络连接的通信，实现数据传输的并发服务。

套接字之间的连接过程分为三个步骤：服务器监听，客户端请求，连接确认。