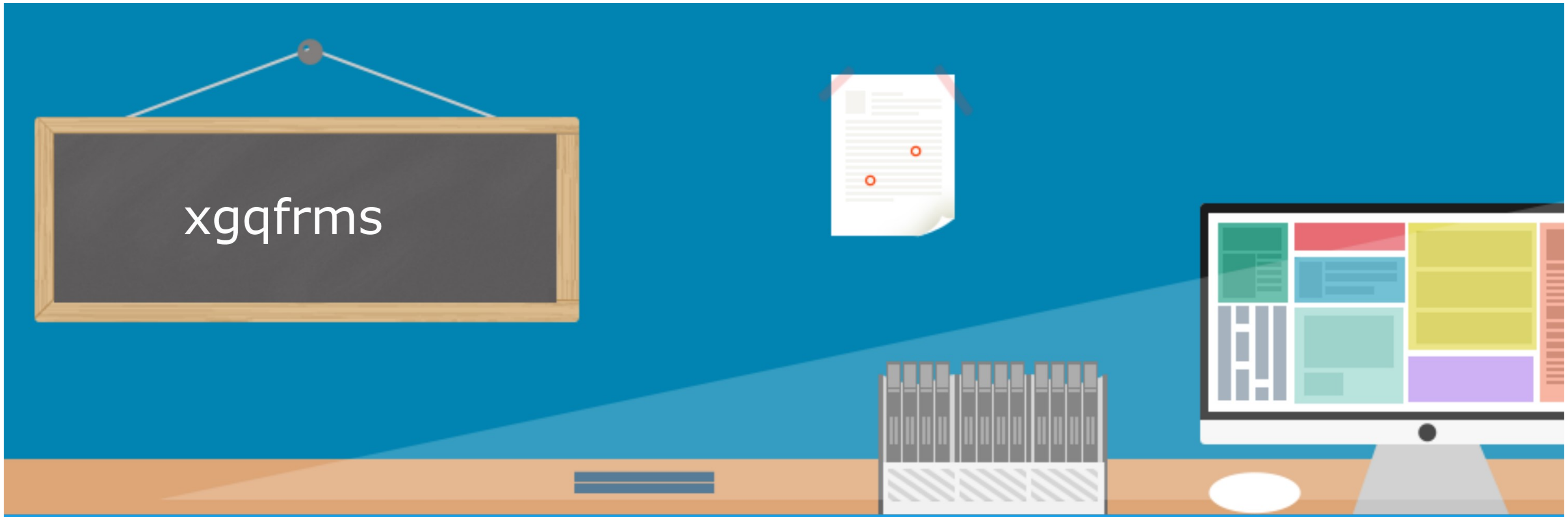


CSS3实现 垂直居中 水平居中 的技巧

1

1

1

How To Center Anything With CSS

Front End posted by Code My Views

1

Recently, we took a dive into the very core concepts behind CSS layout and explored the differences betweenabsolute and relative positioning. We're going to follow that up with another CSS layout talk, this time based around a fundamental question that almost every new developer asks: how do you center something?
There are a bunch of different types of web elements and layout situations, each calling for a unique solution for centering (both vertically and horizontally). Today we'll go over a bunch of these scenarios so you can wrap your mind around how they work and come away with the confidence to center anything!

10
17
24
31
搜索
最新
1. C
2. w
剖析
ute
3. W
===
CSR
n(SE
Appl

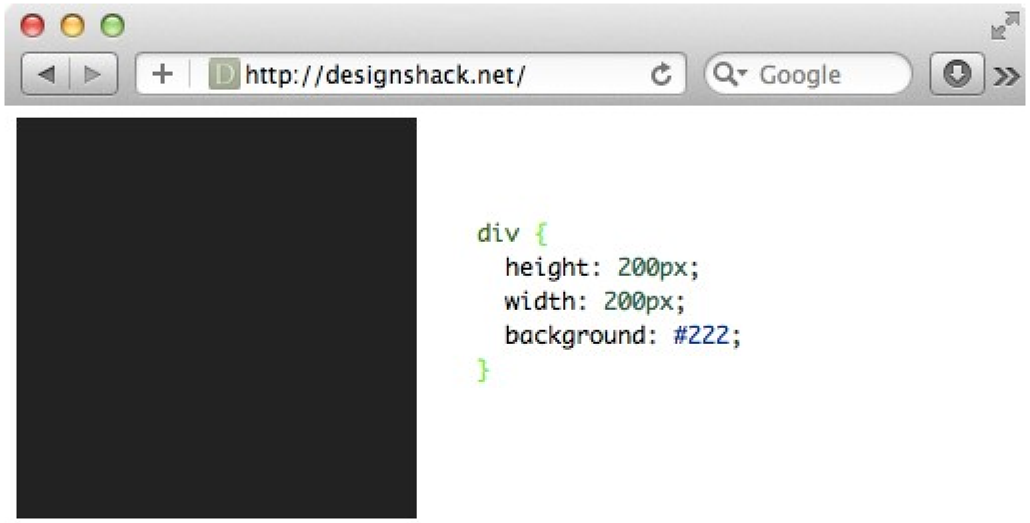
Who’s This For?

I’ve gotten a lot of commenter feedback lately from designers who struggle with the basic methods and concepts of layout in CSS. The general consensus among many of those new to CSS is that they simply “fiddle” with the code u ntil everything finally works.
Having been there quite a few times myself, I know that this is an immensely frustrating period of your professional growth. Knowing that the answer is right there in front of you and not being able to figure it out is excruciating and time consuming.
If this sounds familiar, hopefully I can help ease you out of this period with some solid and practical advice for how t o handle some common layout scenarios. If you’re a CSS ninja who can code a website blindfolded, this article prob ably isn’t for you. If you’re a designer who just wants a better understanding of how to take what’s in your Photosho p file and turn it into CSS, you’re in the right place. Let’s get started.

Horizontally Center an Element

The first scenario that we’ll attack is by far one of the most common: centering an element horizontally in the viewp ort or browser window. To get started, let’s bust out a simple div and give it some basic styling.

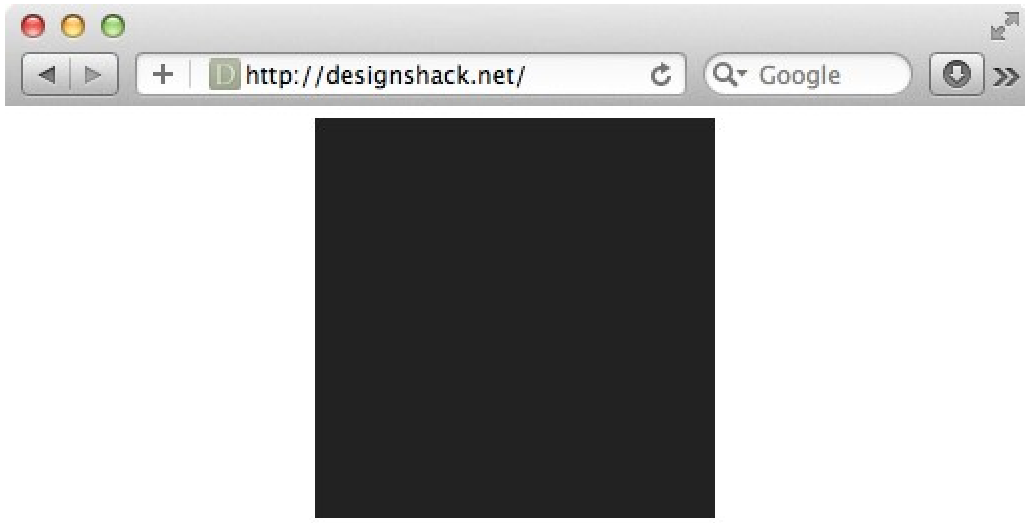
4. W
具 D
s
5. A
6. E
ools
7. C
系统
8. w
面试
9. c
s图
10.
我的
gith



```
div {
  height: 200px;
  width: 200px;
  background: #222;
}
```

As you can see, by default, our div pops up in the top left of the viewport. The trick here is that we need the div to s tay in the center of the window, no matter what the window's size is currently. That means that we can't use absolu te positioning to place it at a specific point because that won't be centered on any other possible window sizes. Instead what we need to do is leverage the "auto" value that can be applied to margins. Here's how this works:

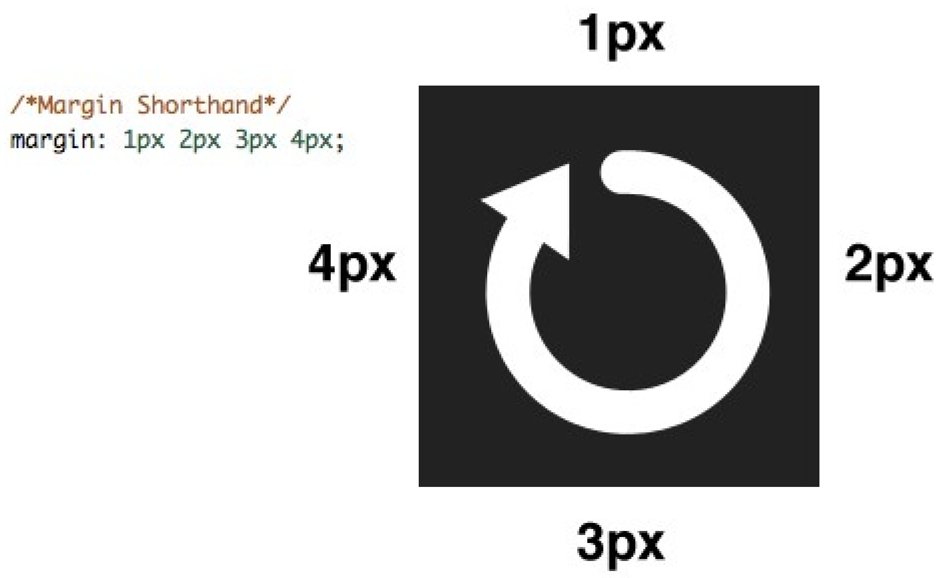
css3
html
linux
goog
php(
free
css(
micr
java
更多
随笔
201



```
div {
  height: 200px;
  width: 200px;
  background: #222;
  margin: 0 auto;
}
```

As you can see, applying a single line of CSS tossed our box straight to the center of its parent, which in this case is the body. To accomplish this, I used a bit of CSS shorthand. If you need a refresher, margin shorthand starts at the top and works its way around clockwise.

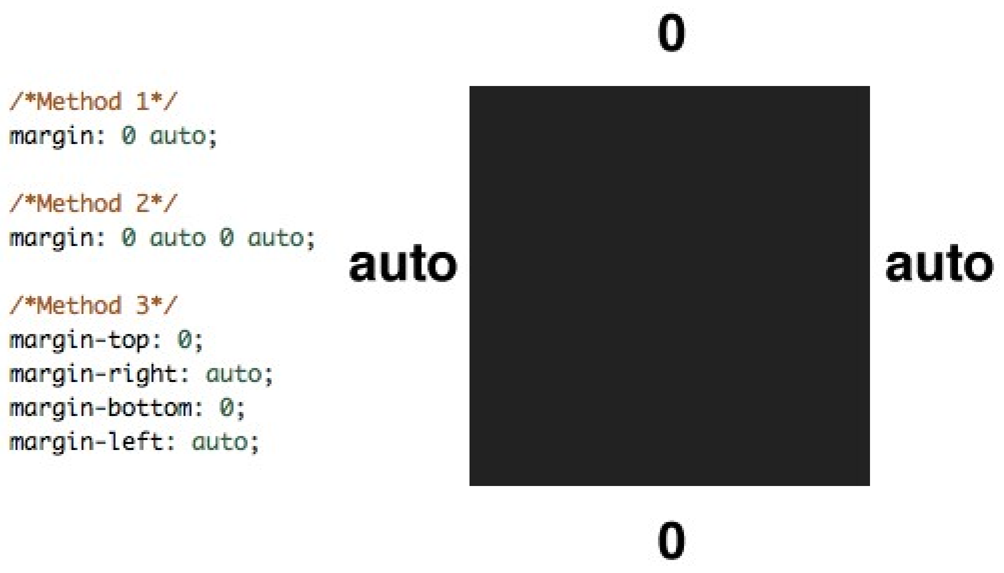
201
201
201
201
201
201
201
201
201
201
201
201



```
/*Margin Shorthand*/
margin: 1px 2px 3px 4px;
```

You can shorten this even further if you only have two values that you need assigned. In this case, the first slot will apply to the top and bottom margins while the second slot will apply to the left and right margins. Here’s another lo ok at our centered div, this time with the margins declared using three separate but perfectly equivalent methods.

201
201
积分
积分
排名
最新
1. R
技巧
lowc
aʊn]
真相
2. R
技巧



```
/*Method 1*/
margin: 0 auto;

/*Method 2*/
margin: 0 auto 0 auto;

/*Method 3*/
margin-top: 0;
margin-right: auto;
margin-bottom: 0;
margin-left: auto;
```

As you can see, our element is void of top and bottom margins, but the left and right are set to *auto*, which keeps the item perfectly centered.

Things To Keep In Mind

There are some important things to remember about using the auto margins trick. First of all, you **must** have a specific width declared for the element that you're centering. The height declaration is not necessary, you can allow the content to determine the height if you wish, which is the default setting, but the width must always be set.

It's important to note that while this trick will work on most block level elements, not just divs, it won't help you out with vertical centering. As an example, let's throw a paragraph inside of a div, then attempt to center that paragraph in the space.

```
div {
  height: 400px;
  width: 400px;
  background: #eee;
  margin: auto;
}

```

The
elat

3. R
开发

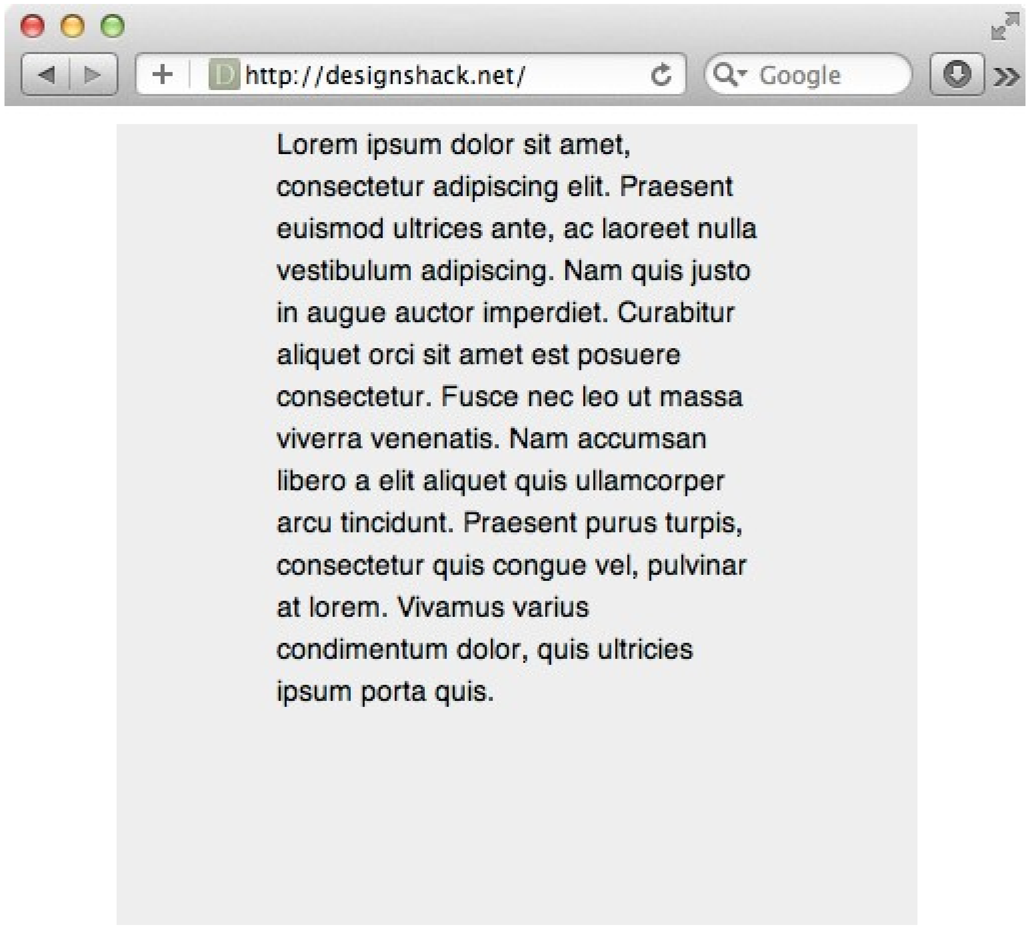
CSS

4. R
开发

CSS

伪类
来实
伪元
种效

```
p {
  width: 60%;
  margin: auto;
  font: 14px/1.5 Helvetica, sans-serif;
}
```



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent euismod ultrices ante, ac laoreet nulla vestibulum adipiscing. Nam quis justo in augue auctor imperdiet. Curabitur aliquet orci sit amet est posuere consectetur. Fusce nec leo ut massa viverra venenatis. Nam accumsan libero a elit aliquet quis ullamcorper arcu tincidunt. Praesent purus turpis, consectetur quis congue vel, pulvinar at lorem. Vivamus varius condimentum dolor, quis ultricies ipsum porta quis.

As you can see, I have auto margins set both on the paragraph and its parent div. This centered everything nicely h

5. R
开发
Typc
line:
men

阅读

1. pl
or 的

2. 关
(单引
178)

3. 栃
母表

orizontally, but it didn’t have any effect on the vertical position.

Center An Absolutely Positioned Element

The method above works to automatically center one item inside another, but the method assumes that you’re usin g the default positioning context: static. If you have absolute positioning applied, this method goes out the window. Using the absolute and relative positioning methods we learned last week, we can apply a simply formula to solve t his issue.

$$\text{left} = (x-y) / 2$$

x = parent width

y = element width

Here we see that on an absolutely positioned item contained inside of a relatively positioned item, we need to set th e *left* property by plugging some numbers into this formula. Here’s a test case:

```
.container {
  height: 300px;
  width: 300px;
  background: #eee;
  margin: 10px auto;
  position: relative;
```

4. C

5. 免
r git
03)

评论

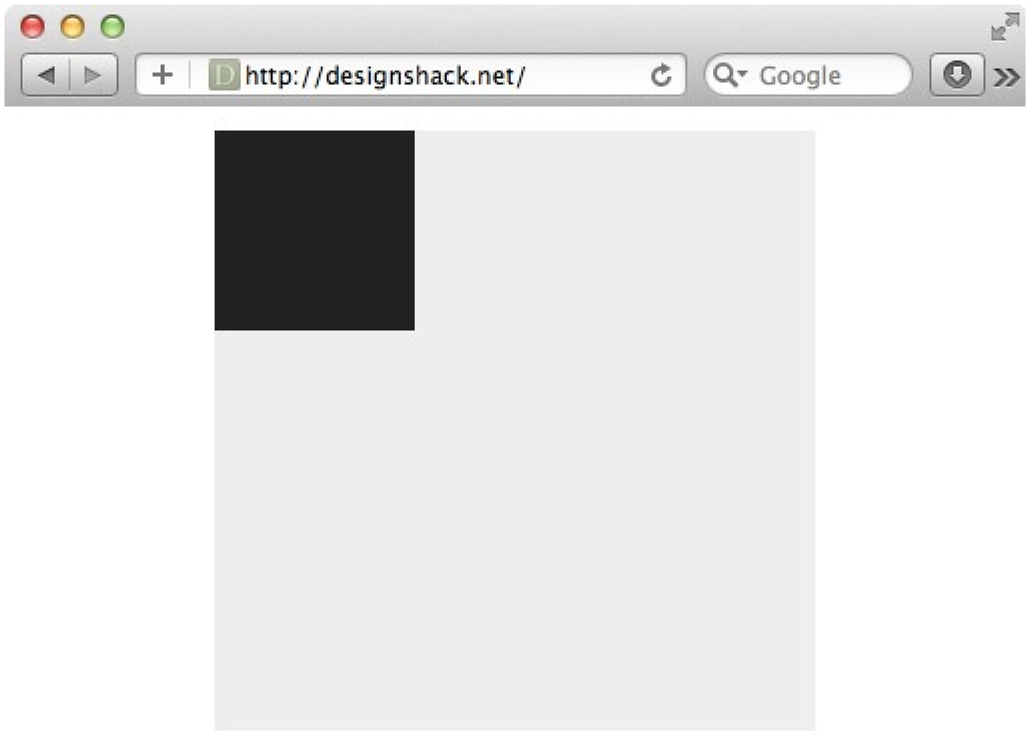
1. li
)

2. 在
各种
le dr
true

3. pl
名访
? ?

4. i1
and
国际

```
}

.box {
  height: 100px;
  width: 100px;
  background: #222;
  position: absolute;
}
```



Let’s see if we can center the black box horizontally. Using our formula, we can see that the *left* property needs to b

司企

5. W
确安

推荐

1. Li
件&
删除

2. W

3. C
(1)

4. 2
3 上
安装

5. 在
机(P
nux
访问

e set to 100px.

Goo

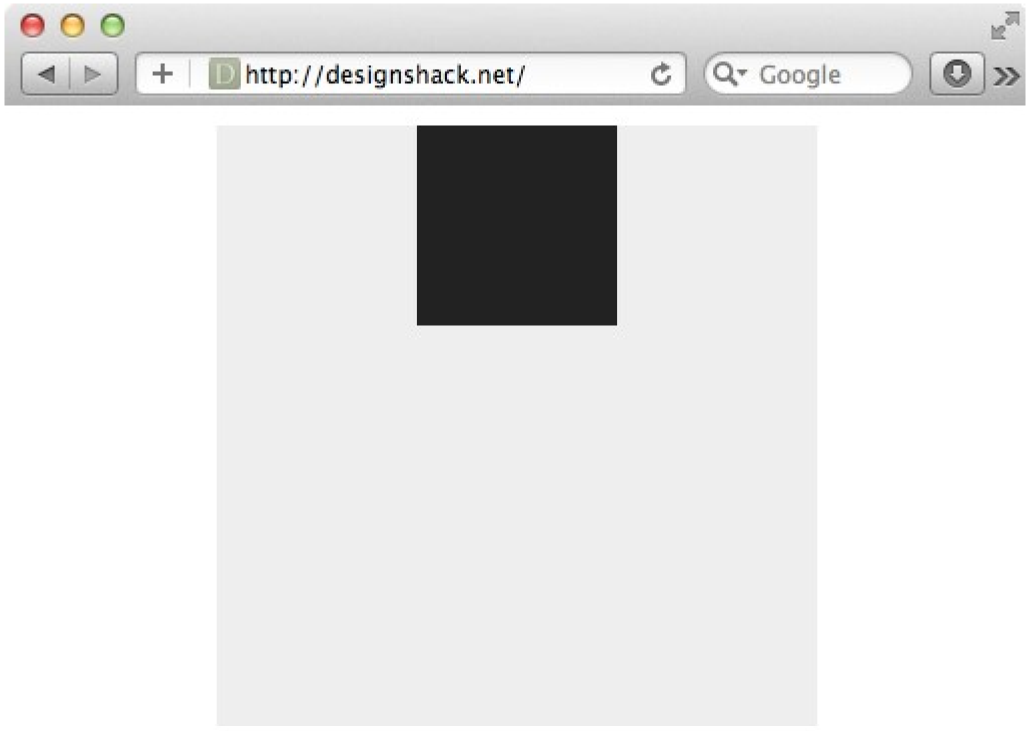
$$(x-y)\ /\ 2 =$$

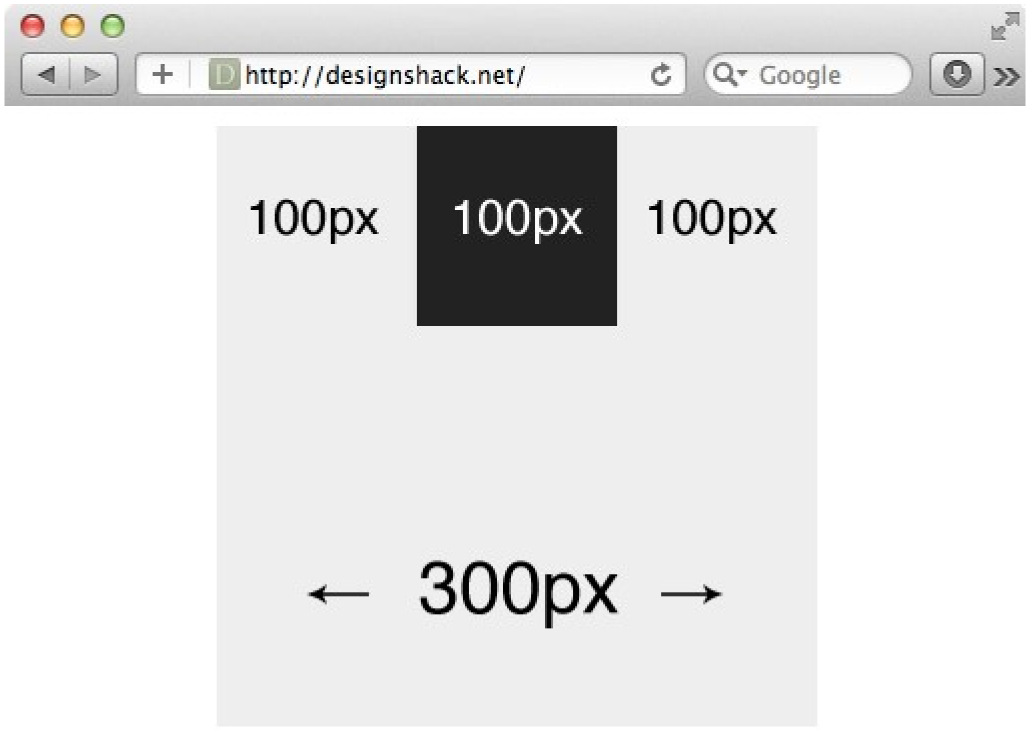
$$(300-100)\ /\ 2 =$$

$$\mathbf{(200)\ /\ 2 = 100}$$

```css
.container {
  height: 300px;
  width: 300px;
  background: #eee;
  margin: 10px auto;
  position: relative;
}

.box {
  height: 100px;
  width: 100px;
  background: #222;
  position: absolute;
  left: 100px;
}
```

With this code, we’ve set the distance between the left side of the box and the edge of its parent container to 100px , which centers it perfectly.

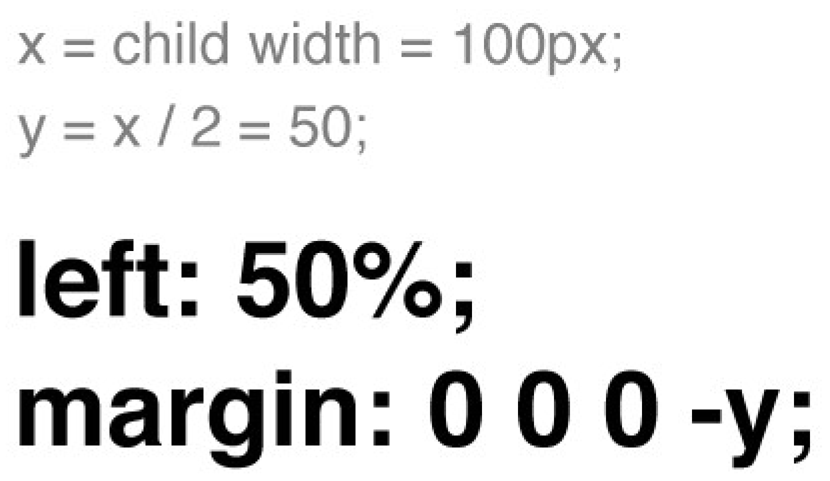




With Fluid Width

The method above only works if the parent container has a static width. Given the popularity of responsive design t hough, more and more containers are going the fluid route lately. This means that we'll need another way to center the child that isn't dependent on the width of the parent.

To accomplish this, we need to use a percentage for the *left* value. The obvious answer is to use 50%, but that won' t really work because you're not accounting for the width of the element that you're centering. To make it work, we need to add in a negative left margin of half the width of the child element.

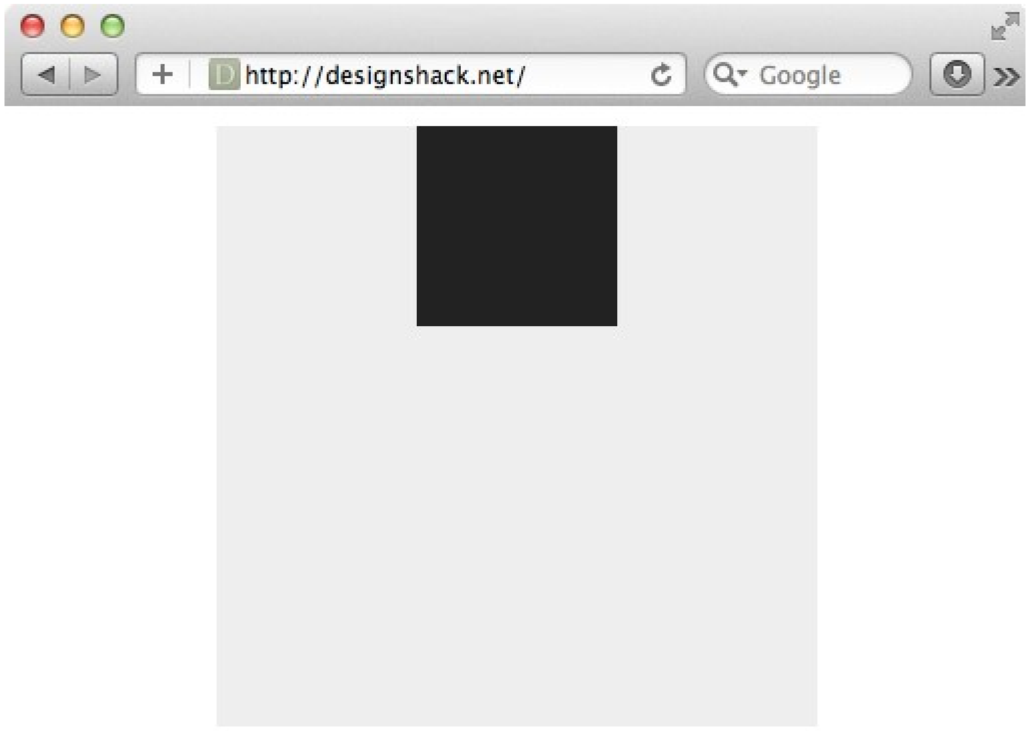


x = child width = 100px;
y = x / 2 = 50;

left: 50%;
margin: 0 0 0 -y;

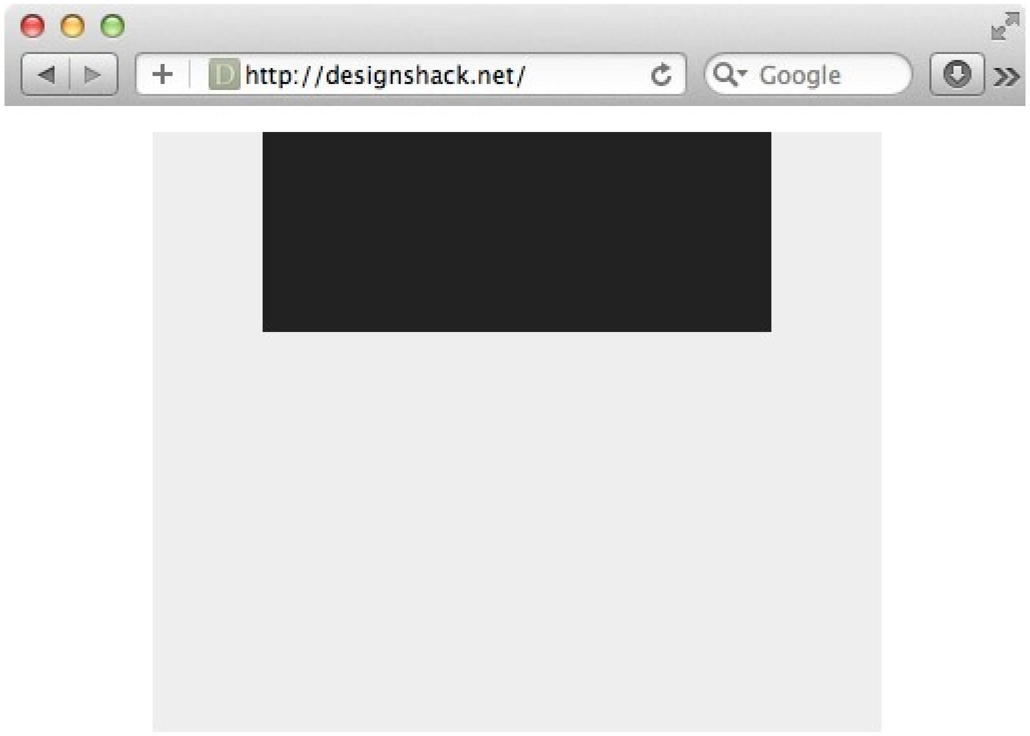
Using this logic, we apply a left margin of negative fifty pixels along with a left value of 50% and our div is once aga
in perfectly centered on the x-axis.

```css
.container {
  height: 300px;
  width: 70%;
  background: #eee;
  margin: 10px auto;
  position: relative;
}

.box {
  height: 100px;
  width: 100px;
  background: #222;
  position: absolute;

  /*Centering Method 2*/
  margin: 0px 0 0 -50px;
  left: 50%;
}
```



It’s important to note that this would also work if our child element had a fluid width. We use the same steps as bef ore and come up with something like the following:

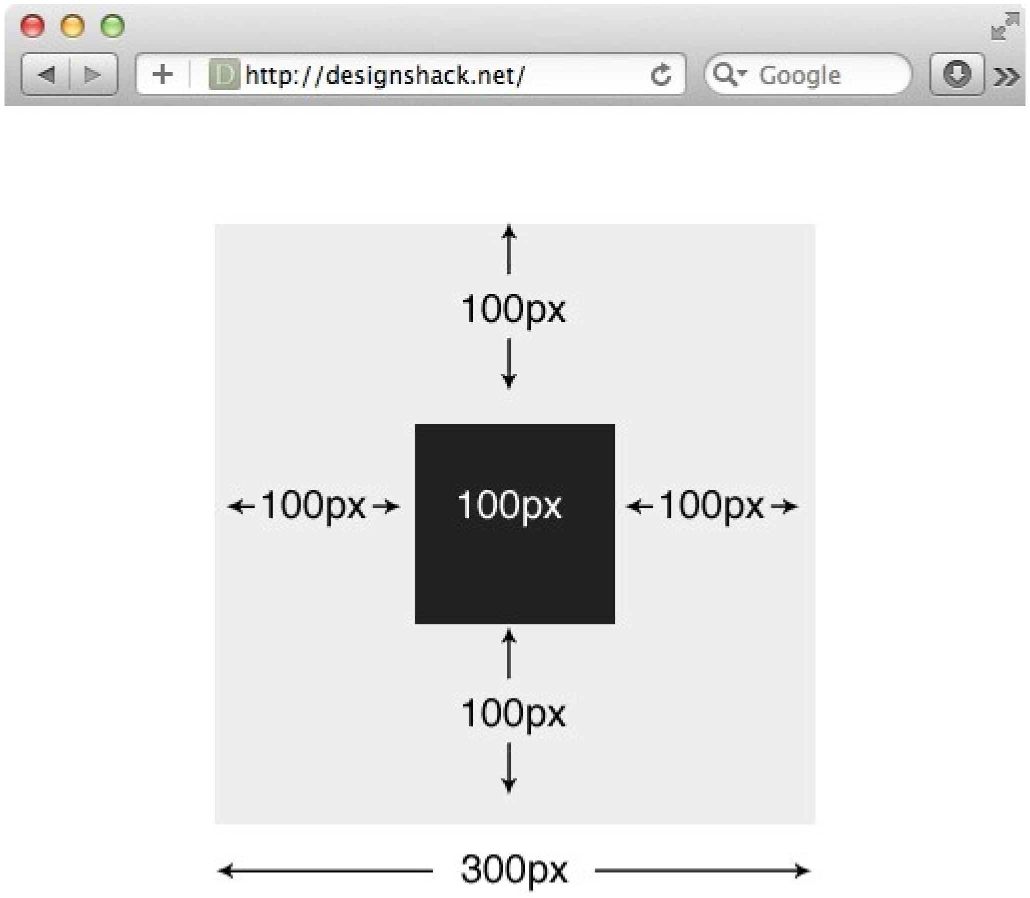
```
.container {
  height: 300px;
  width: 70%;
  background: #eee;
  margin: 10px auto;
  position: relative;
}

.box {
  height: 100px;
  width: 70%;
  background: #222;
  position: absolute;

  /*Centering Method 2*/
  margin: 0px 0 0 -35%; /* Half of 70% /*
  left: 50%;
}
```



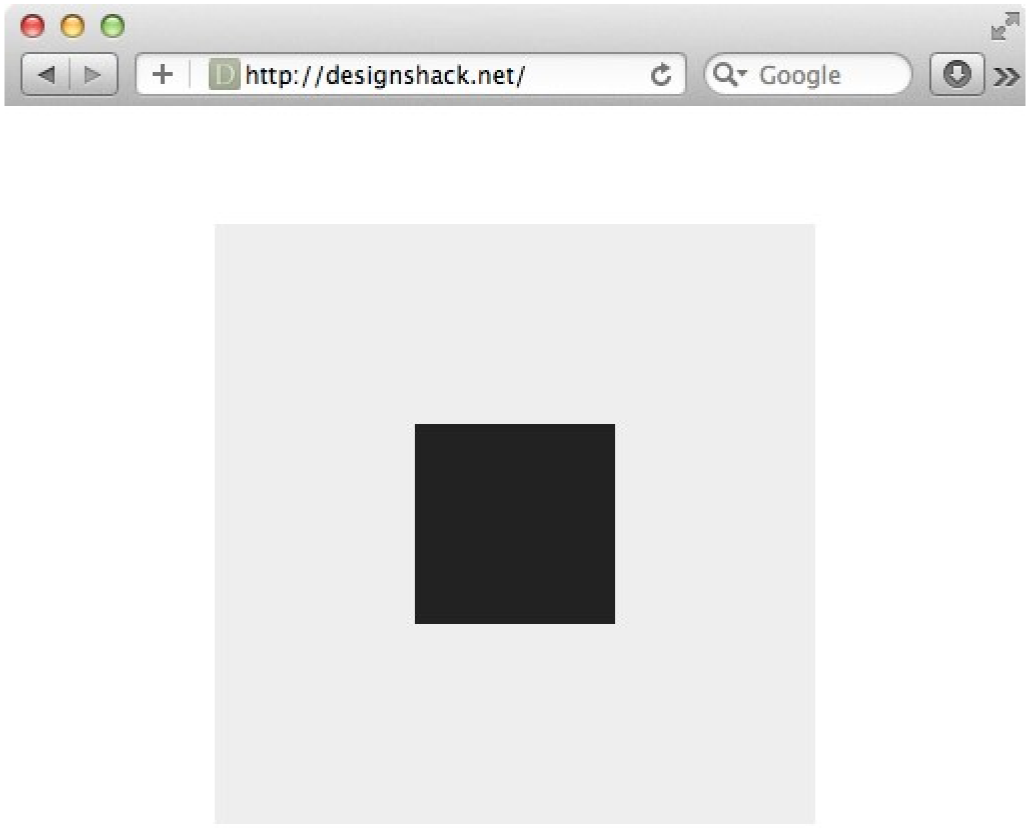
Dead Center an Element

Now that we have a few simple and complicated centering methods in our tool belt, it’s time to tackle the puzzle of perfectly centering an element both horizontally and vertically.
Fortunately, to pull this off, we can use the same method that we just learned, we just have to account for height. This time around we’re also going to center both the parent and the child both vertically and horizontally. Here’s the code to pull it off:

```
.container {
  height: 300px;
  width: 300px;
  background: #eee;
  position: absolute;

  margin: -150px 0 0 -150px;
  left: 50%;
  top: 50%;
}

.box {
  height: 100px;
  width: 100px;
  background: #222;
  position: absolute;

  /*Centering Method 2*/
  margin: -50px 0 0 -50px;
  left: 50%;
  top: 50%;
}
```

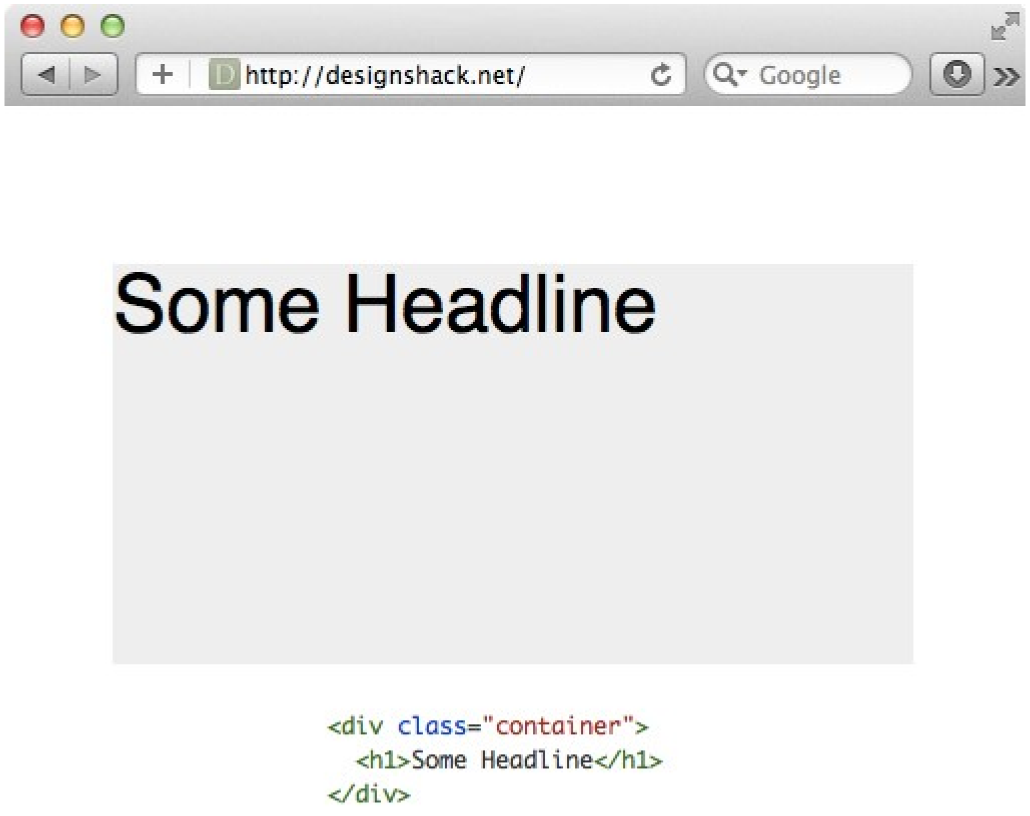


There are a few things that you need to notice here. First, this time **both** the parent and the child are absolutely po sitioned. From here, I used our negative margins trick with the left property on the container div, then did the same for the box div.
The result is that our content is completely centered and will stay that way as the browser changes size in any direc tion (even vertically!). Click on the image below to tinker with a live demo.



Centering Text

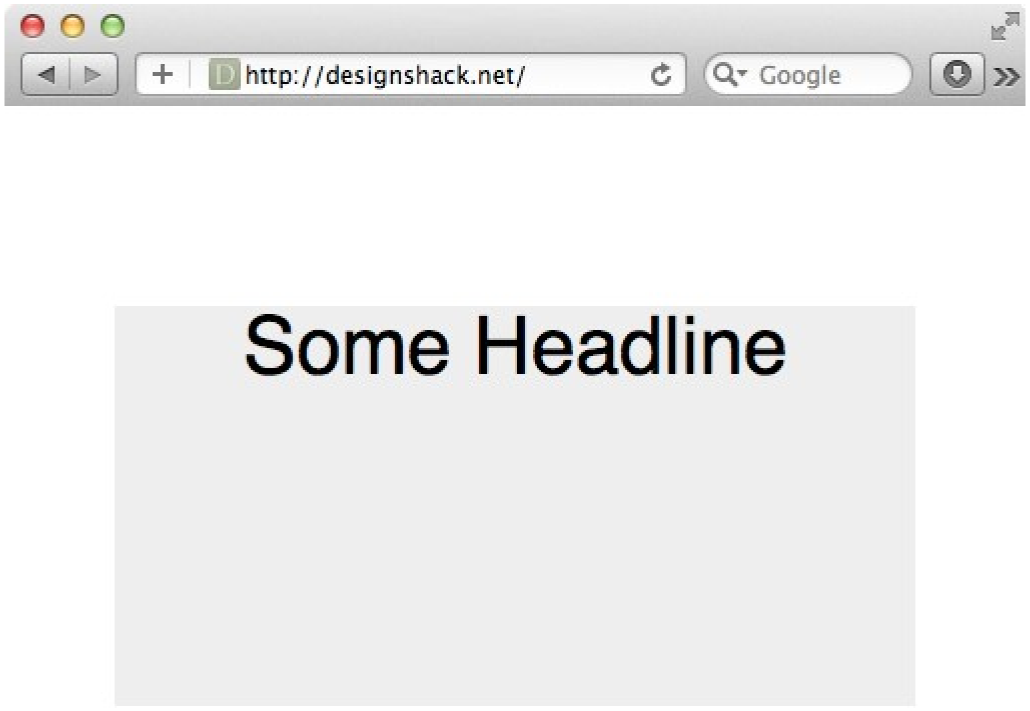
For my next trick, I’ll teach you something cool about centering text. We’ll start with a simple h1 element inside of a container div.



```
<div class="container">
  <h1>Some Headline</h1>
</div>
```

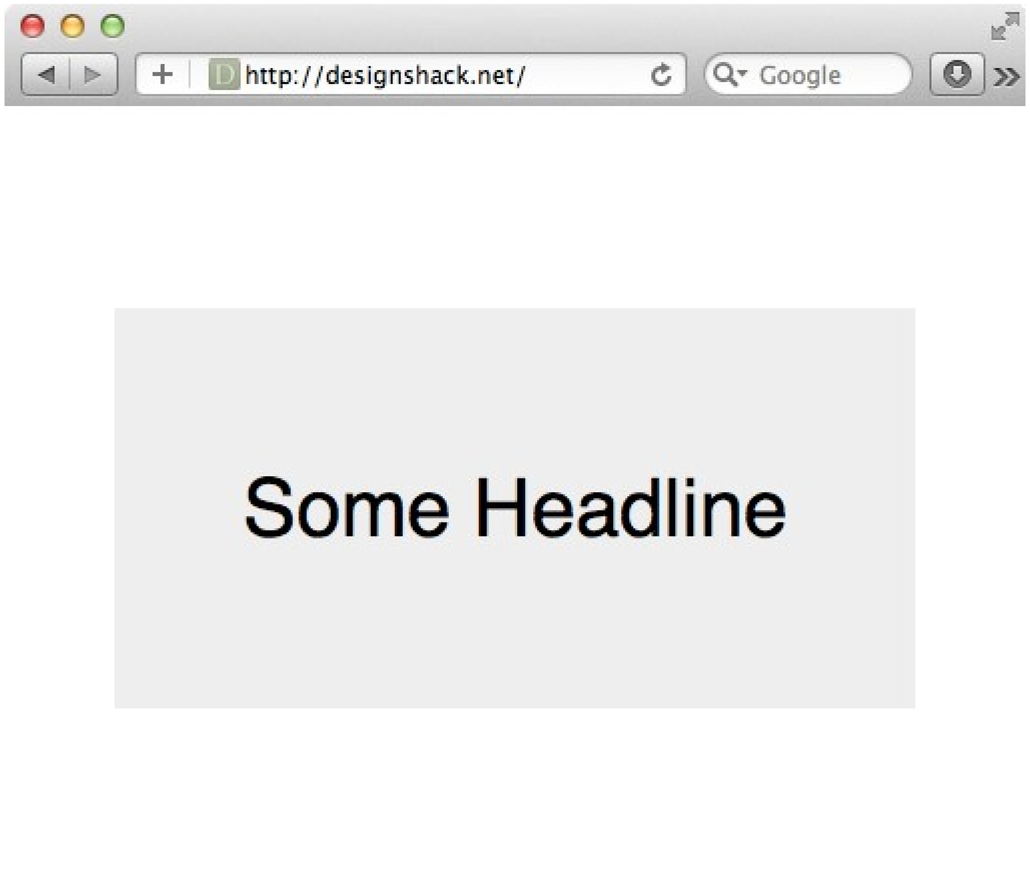
Now, I’m sure that you already know how to center this text horizontally in the space. It’s typically one of the first things you learn in CSS. Just set the *text-align* property to center.

```
.container {
  height: 400px;
  width: 400px;
  background: #eee;
  margin: 50px auto;
}

h1 {
  font: 40px/1 Helvetica, sans-serif;
  text-align: center;
}
```



Easy right? But now let's say we want to center this line of text vertically as well. If this were a paragraph, we would probably take the methods above into account, but since it's only a single line, we can use a nifty trick.
All we have to do is set the *line-height* property to the height of the container. I accomplished this below using the shorthand font syntax.

```
.container {
  height: 200px; /*Set line-height to this value*/
  width: 400px;
  background: #eee;
  margin: 150px auto;
}

h1 {
  font: 40px/200px Helvetica, sans-serif;
  text-align: center;
}
```



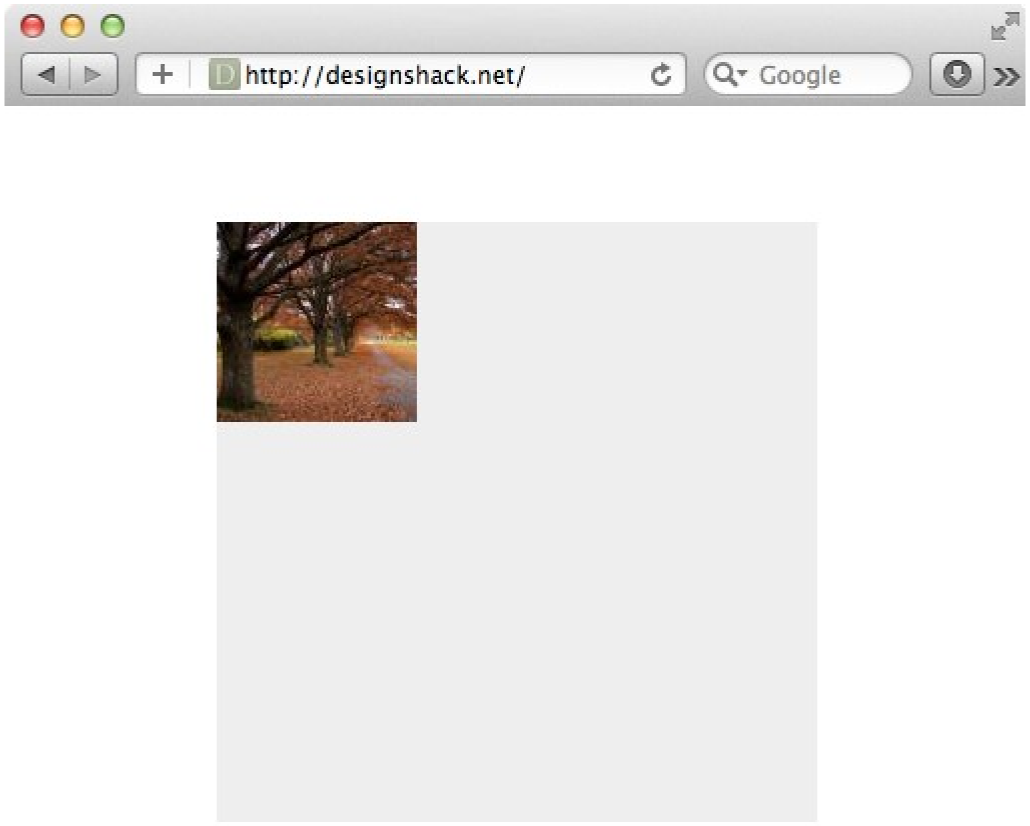
Warning: This trick only works with a single line of text, and is a bit hacky so it may not be appropriate for all situations.

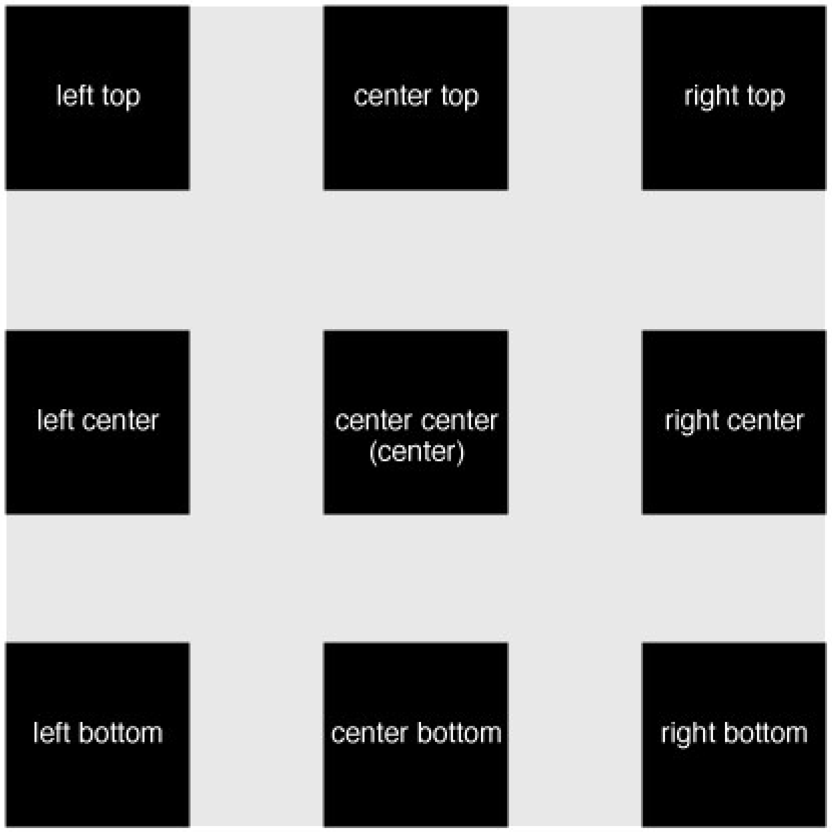
Centering a Background Image

The last thing that we're going to learn to center is a CSS background image. To get started with this, we'll create another container div, but this time we'll keep in empty and toss in an image using CSS.

```
.container {
  height: 300px;
  width: 300px;
  margin: 150px auto;
  background: #eee url(http://lorempixum.com/100/100/nature/4) no-repeat;
}
```

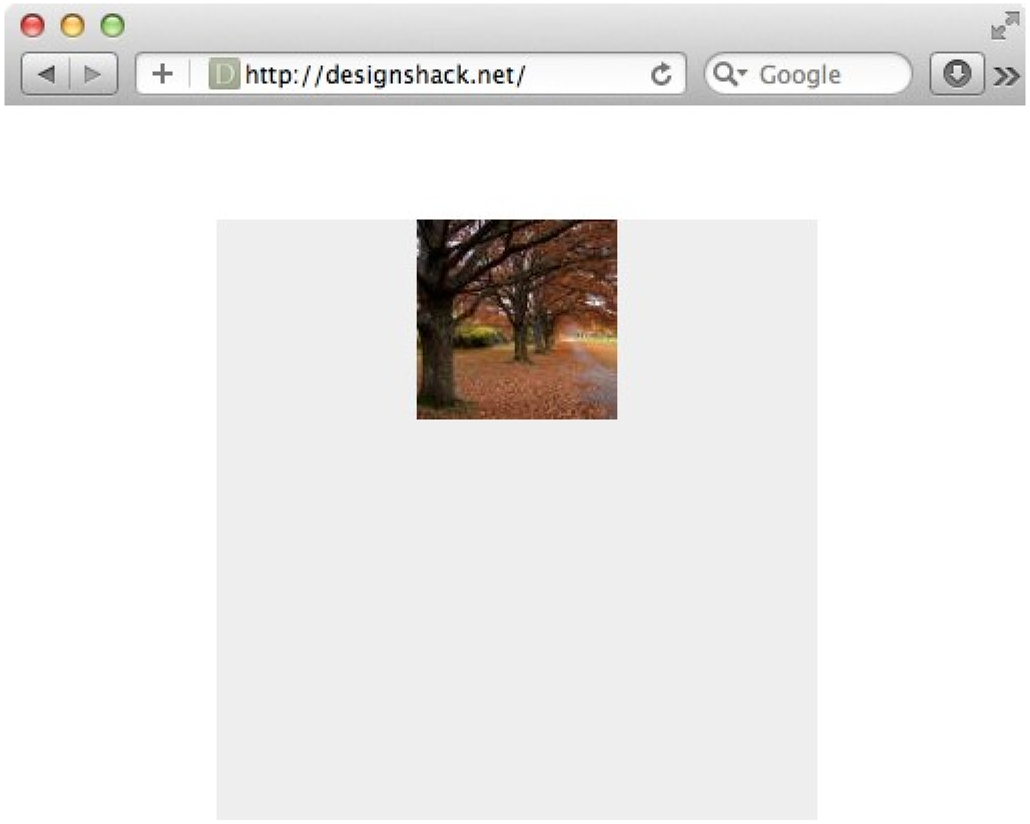
As you can see, the default place for an image to appear if no repeat is set is the top left. It turns out though that y ou can move it to one of nine different preassigned slots. These are shown below:





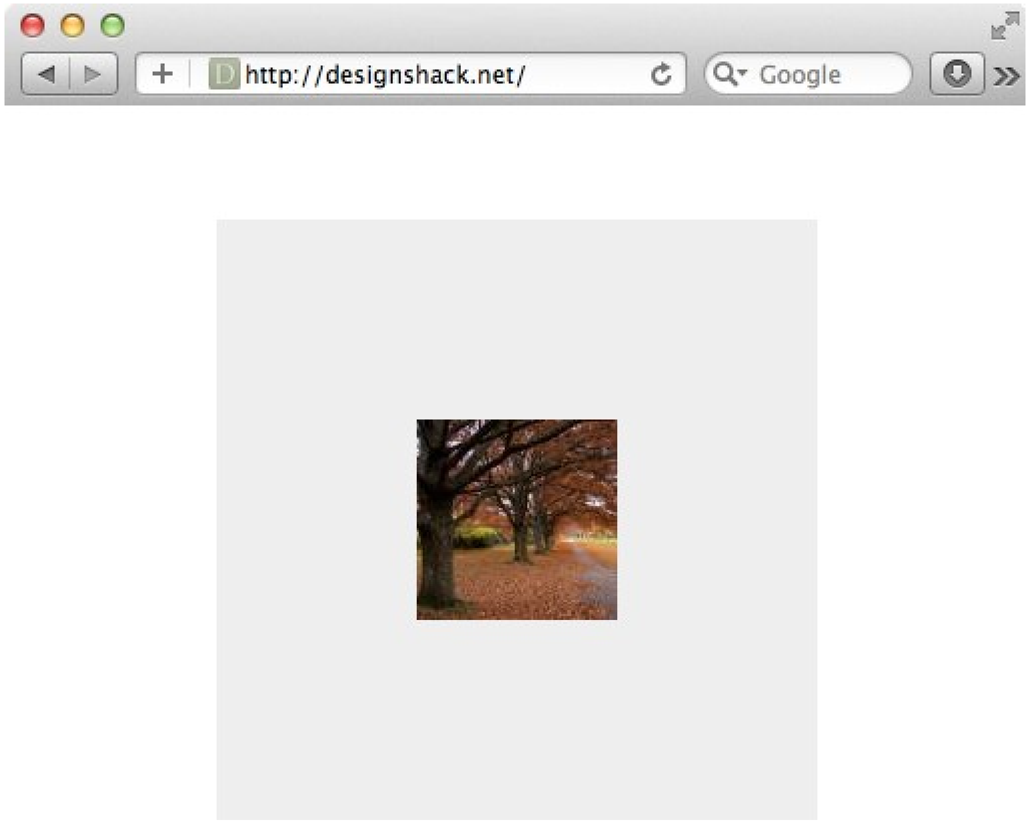
We accomplish this movement through the use of the *background-position* property. Simply call this property and se t any of the values listed above.

```css
.container {
  height: 300px;
  width: 300px;
  margin: 150px auto;
  background: #eee url(http://lorempixum.com/100/100/nature/4) no-repeat;
  background-position: top center;
}
```



As an alternative, we can use the shorthand syntax for this. Simply toss in one of the values at the end of your stre am of values on the *background* property.

```
.container {
  height: 300px;
  width: 300px;
  margin: 150px auto;
  background: #eee url(http://lorempixum.com/100/100/nature/4) no-repeat center;
}
```



Conclusion

There you have it! You should now be completely confident in your ability to tackle almost any centering situation wi th CSS, from dead centering a div to vertically centering a line of text within its container and beyond.
Reach out to us on Facebook or Twitter and let us know if you found this information to be helpful and if you have a ny questions.

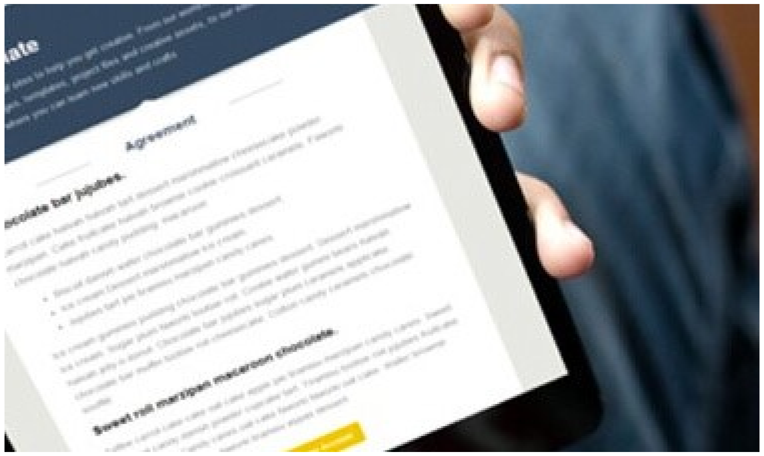
Join the Discussion

More

ARTICLES

10 Tips for Designing Icons That Don’t Suck

Almost every designer is thinking about app design these days. One of the smallest features of every app is



How To Center Anything With CSS

Recently, we took a dive into the very core concepts behind CSS layout and explored the differences between...

1

https://codemyviews.com/blog/how-to-center-anything-with-css

1

http://blog.csdn.net/freshlover/article/details/11579669

1

1. 绝对定位居中Absolute Centering技术
    1. 一容器内Within Container
    2. 二视区内Within Viewport
    3. 三边栏 Offsets
    4. 四响应式自适应Responsive
    5. 五 溢出情况Overflow
    6. 六重绘Resizing
    7. 七图片Images

Ⅰ.绝对定位居中(Absolute Centering)技术

我们经常用margin:0 auto来实现水平居中，而一直认为margin:auto不能实现垂直居中......实际上，实现垂直居中仅需要声明元素高度和下面的CSS:

```css
.Absolute-Center {
  margin: auto;
  position: absolute;
  top: 0; left: 0; bottom: 0; right: 0;
}
```

我不是这种实现方法的第一人，可能这只是非常常见的一种小技术，我斗胆将其命名为**绝对居中(Absolute Centering)**，虽然如此，但是大多数讨论垂直居中的文章却从来不提这种方法，直到我最近浏览《How to Center Anything WithCSS》这篇文章的评论时候才发现这种用法。在评论列表中Simon和Priit都提及了此方法。

如果你有任何扩展的功能或建议，可以在此跟帖：

CodePen

SmashingMagazine

Twitter @shshaw

优点：

1.支持跨浏览器，包括IE8-IE10.

2.无需其他特殊标记，CSS代码量少

3.支持百分比%属性值和min-/max-属性

4.只用这一个类可实现任何内容块居中

5.不论是否设置padding都可居中（在不使用box-sizing属性的前提下）

6.内容块可以被重绘。

7.完美支持图片居中。

缺点：

1.必须声明高度（查看可变高度Variable Height）。

2.建议设置overflow:auto来防止内容越界溢出。（查看溢出Overflow）。

3.在Windows Phone设备上不起作用。

浏览器兼容性：

Chrome,Firefox, Safari, Mobile Safari, IE8-10.

绝对定位方法在最新版的Chrome,Firefox, Safari, Mobile Safari, IE8-10.上均测试通过。

对比表格：

绝对居中法并不是唯一的实现方法，实现垂直居中还有些其他的方法，并各有各的优势。采用哪种技术取决于你的浏览器是否支持和你使用的语言标记。这个对照表有助于你根据自己的需求做出正确的选择。

Technique	Browser Support	Responsive	Overflow	resize:both	Variable Height	Major Caveats
Absolute Centering	Modern & IE8+	Yes	Scroll, can overflow container	Yes	Yes*	Variable Height not perfect cross-browser
Negative Margins	All	No	Scroll	Resizes but doesn't stay centered	No	Not responsive, margins must be calculated manually
Transform	Modern & IE	Yes	Scroll, can overfl	Yes	Yes	Blurry rendering

Technique	Browser Support	Responsive	Overflow	resize:both	Variable Height	Major Caveats
s	9+		ow container			
Table-Cell	Modern & IE 8+	Yes	Expands container	No	Yes	Extra markup
Inline-Block	Modern, IE8+ & IE7*	Yes	Expands container	No	Yes	Requires container, hacky styles
Flexbox	Modern & IE 10+	Yes	Scroll, can overflow container	Yes	Yes	Requires container, vendor prefixes

解释：
通过以上描述，绝对居中（AbsoluteCentering）的工作机理可以阐述如下：
1、在普通内容流（normal content flow）中，margin:auto的效果等同于margin-top:0;margin-bottom:0。
W3C中写道If 'margin-top', or'margin-bottom' are 'auto', their used value is 0.
2、position:absolute使绝对定位块跳出了内容流，内容流中的其余部分渲染时绝对定位部分不进行渲染。
Developer.mozilla.org:...an element that is positioned absolutely is taken out of the flow and thustakes up no space
3、为块区域设置top: 0; left: 0; bottom: 0; right: 0;将给浏览器重新分配一个边界框，此时该块block将填充其父元素的所有可用空间，父元素一般为body或者声明为position:relative;的容器。
Developer.mozilla.org:For absolutely positioned elements, the top, right, bottom, and left propertiesspecify offsets from the edge of the element's containing block (what theelement is positioned relative to).
4、 给内容块设置一个高度height或宽度width，能够防止内容块占据所有的可用空间，促使浏览器根据新的边界框重新计算margin:auto
Developer.mozilla.org: The margin of the[absolutely positioned] element is then positioned inside these offsets.
5、由于内容块被绝对定位，脱离了正常的内容流，浏览器会给margin-top,margin-bottom相同的值，使元素块在先前定义的边界内居中。
W3.org: If none of the three [top, bottom,height] are 'auto': If both 'margin-top' and 'margin-bottom' are 'auto', solvethe equation under the extra constraint that the two margins get equal values.AKA: center the block vertically

这么看来， margin:auto似乎生来就是为绝对居中(Absolute Centering)设计的，所以绝对居中(Absolute Centering)应该都兼容符合标准的现代浏览器。

简而言之（TL;DR）：绝对定位元素不在普通内容流中渲染，因此margin:auto可以使内容在通过top: 0; left: 0; bottom: 0;right: 0;设置的边界内垂直居中。

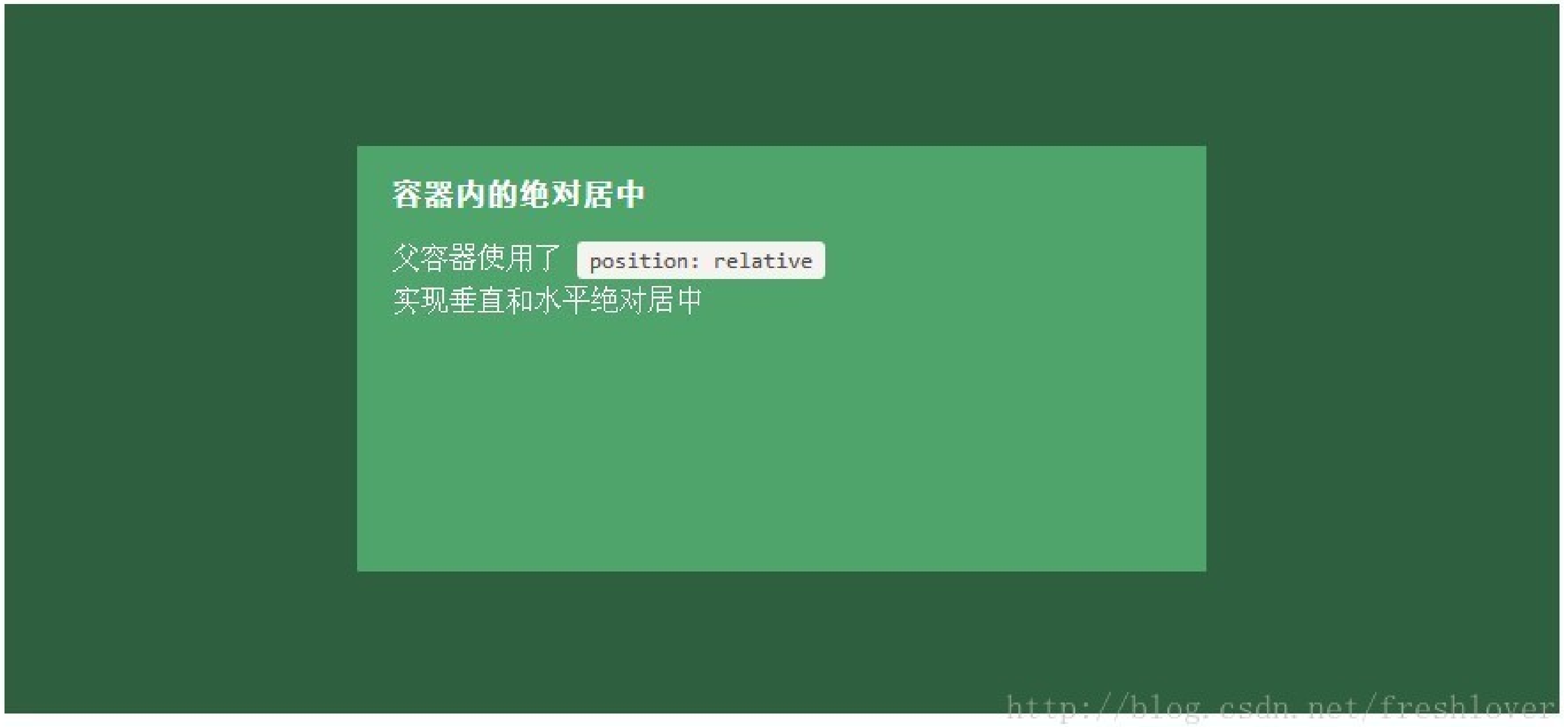
居中方式：

一、容器内（Within Container）

内容块的父容器设置为position:relative，使用上述绝对居中方式，可以使内容居中显示于父容器。

```css
.Center-Container {
  position: relative;
}

.Absolute-Center {
  width: 50%;
  height: 50%;
  overflow: auto;
  margin: auto;
  position: absolute;
  top: 0; left: 0; bottom: 0; right: 0;
}
```

一、容器container内的居中


容器内的绝对居中

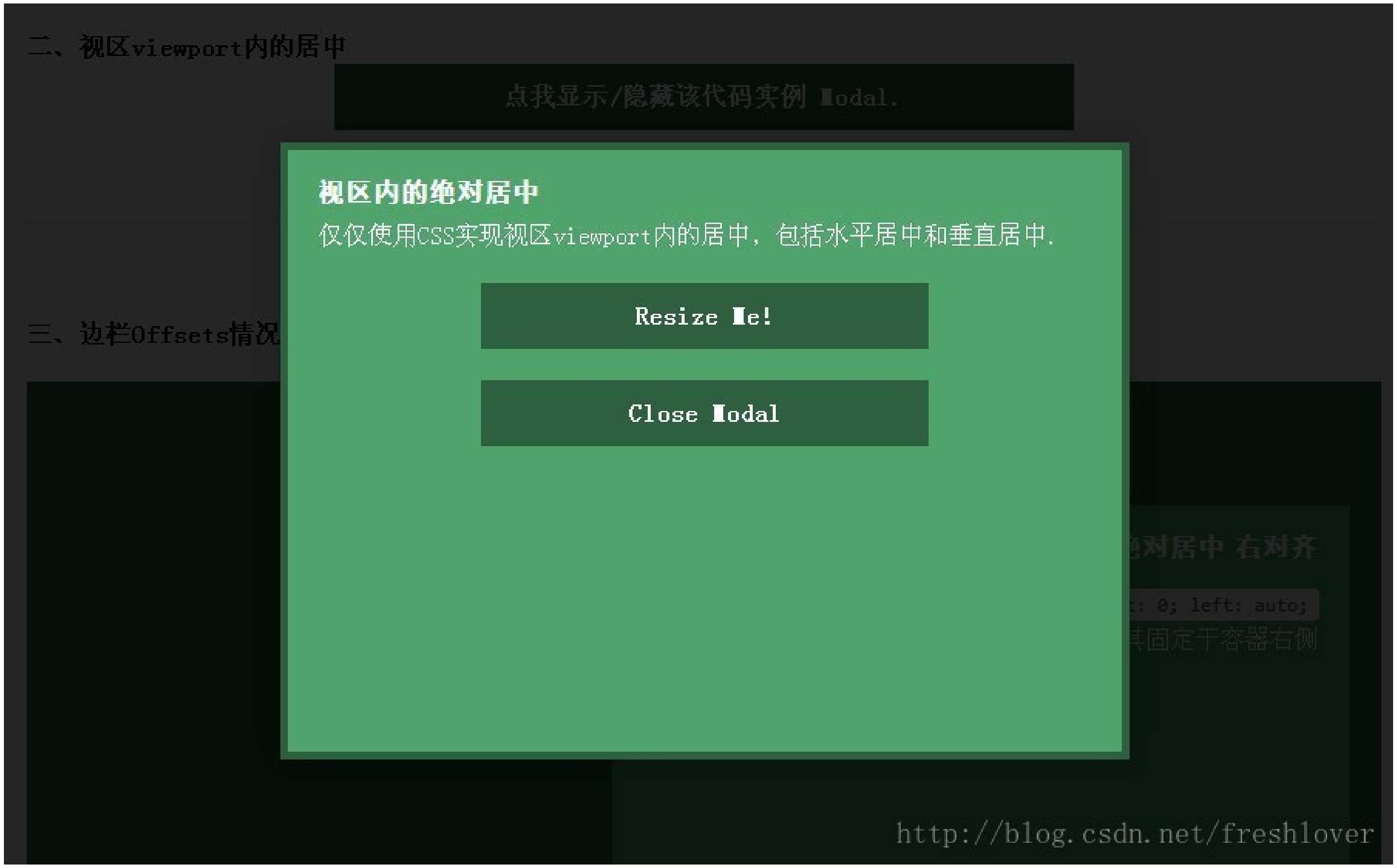
父容器使用了 `position: relative`
实现垂直和水平绝对居中

以下其余的demo默认上面的CSS样式已引用包括进去，在此基础上只提供额外的类供用户追加以实现不同的功能。

二、视区内（Within Viewport）

想让内容块一直停留在可视区域内？将内容块设置为position:fixed;并设置一个较大的z-index层叠属性值。

```css
.Absolute-Center.is-Fixed {
  position: fixed;
  z-index: 999;
}
```



注意：对MobileSafari，若内容块不是放在设置为position:relative;的父容器中，内容块将垂直居中于整个文档，而不是可视区域内垂直居中。

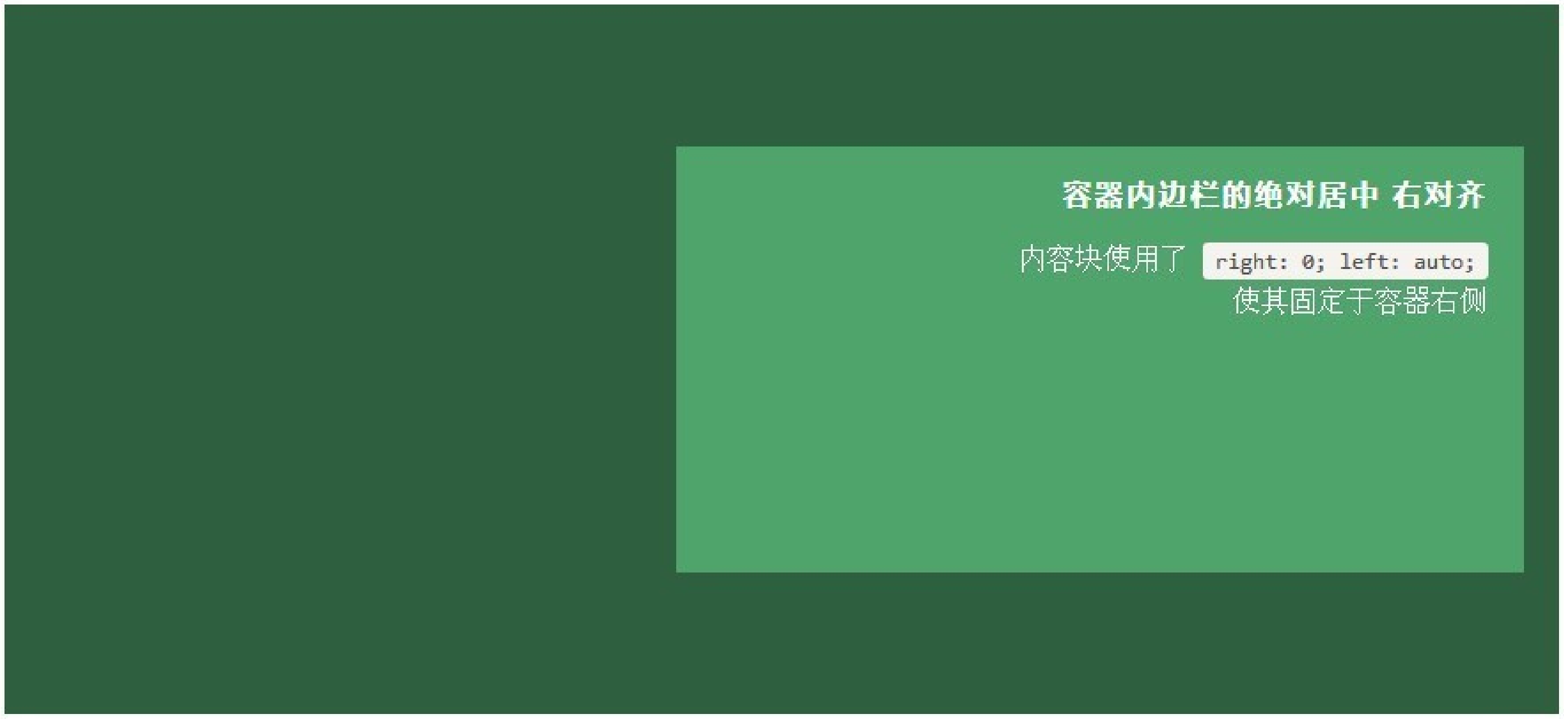
三、边栏 (Offsets)

如果你要设置一个固顶的头或增加其他的边栏，只需要在内容块的样式中加入像这样的CSS样式代码：top:70px;bottom:auto;由于已经声明了margin:auto;，该内容块将会垂直居中于你通过top,left,bottom和right属性定义的边界框内。

你可以将内容块固定与屏幕的左侧或右侧，并且保持内容块垂直居中。使用right:0;left:auto;固定于屏幕右侧，使用left:0;right:auto;固定与屏幕左侧。

```
.Absolute-Center.is-Right {
  left: auto; right: 20px;
  text-align: right;
}

.Absolute-Center.is-Left {
  right: auto; left: 20px;
  text-align: left;
}
```

三、边栏Offsets情况的居中

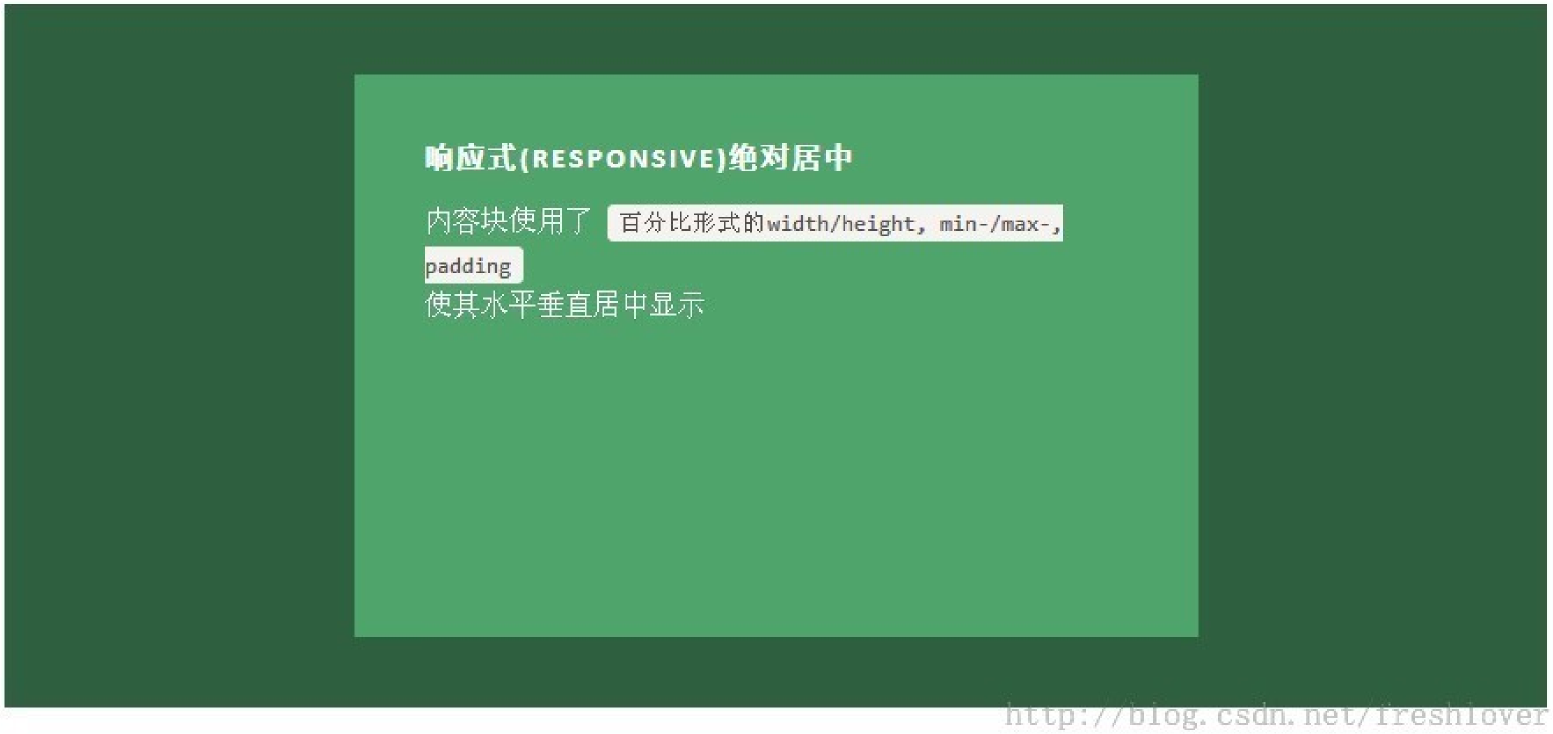

容器内边栏的绝对居中 右对齐
内容块使用了 right: 0; left: auto; 使其固定于容器右侧

四、响应式/自适应(Responsive)

绝对居中最大的优势应该就是对百分比形式的宽高支持的非常完美。甚至min-width/max-width 和min-height/max-height这些

属性在自适应盒子内的表现也和预期很一致。

四、响应式/自适应居中


响应式(RESPONSIVE)绝对居中
内容块使用了 百分比形式的width/height, min-/max-, padding
使其水平垂直居中显示
http://blog.csdn.net/freshlover

```
.Absolute-Center.is-Responsive {
  width: 60%;
  height: 60%;
  min-width: 200px;
  max-width: 400px;
  padding: 40px;
}
```

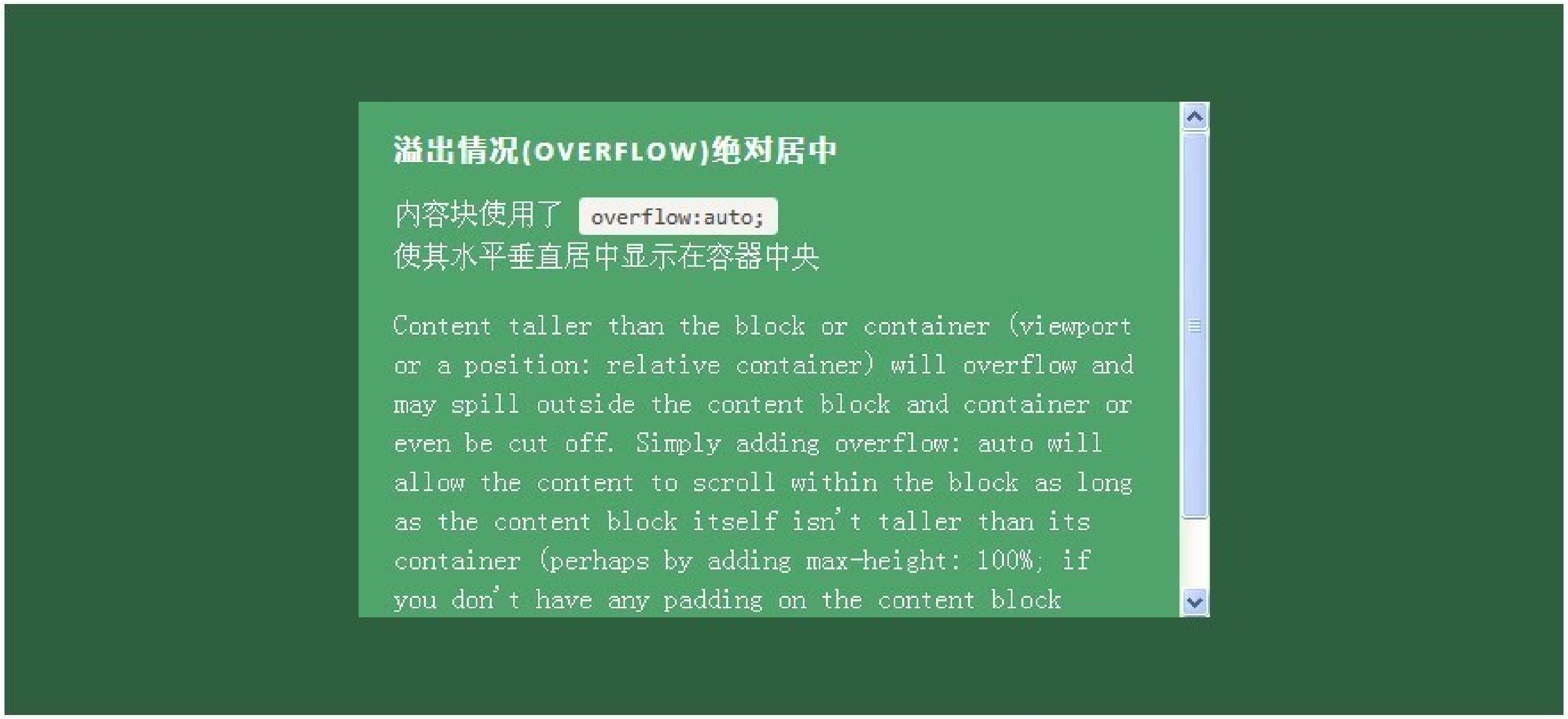
给内容块元素加上padding也不影响内容块元素的绝对居中实现。

五、 溢出情况(Overflow)

内容高度大于块元素或容器（视区viewport或设为position:relative的父容器）会溢出，这时内容可能会显示到块与容器的外面，或者被截断出现显示不全（分别对应内容块overflow属性设置为visible和hidden的表现）。
加上overflow: auto会在内容高度超过容器高度的情况下给内容块显示滚动条而不越界。

```
.Absolute-Center.is-Overflow {
  overflow: auto;
}
```

五、溢出居中



内容块使用了 `overflow:auto;`
使其水平垂直居中显示在容器中央

Content taller than the block or container (viewport or a position: relative container) will overflow and may spill outside the content block and container or even be cut off. Simply adding overflow: auto will allow the content to scroll within the block as long as the content block itself isn't taller than its container (perhaps by adding max-height: 100%; if you don't have any padding on the content block

如果内容块自身不设置任何padding的话，可以设置max-height: 100%;来保证内容高度不超越容器高度。

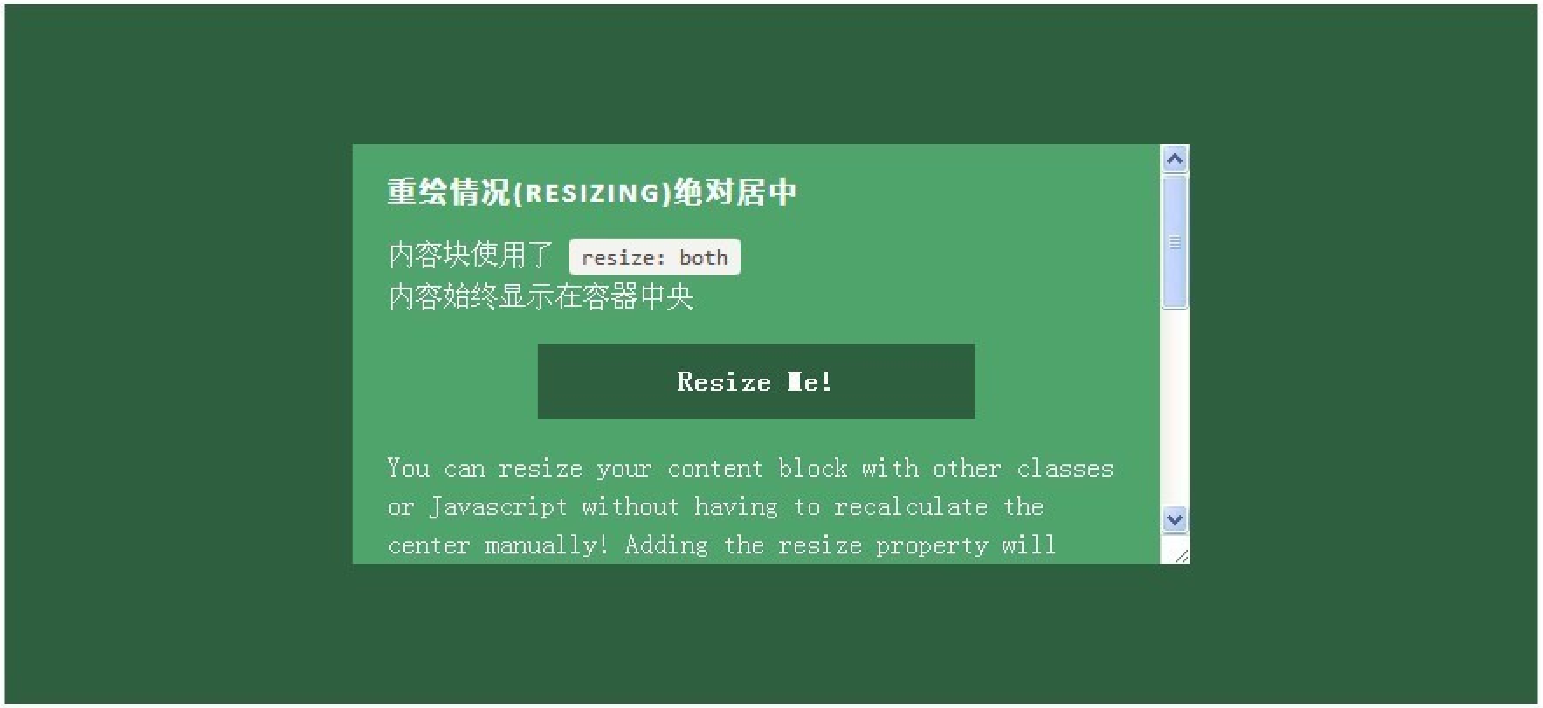
六、重绘(Resizing)

你可以使用其他class类或javascript代码来重绘内容块同时保证居中，无须手动重新计算中心尺寸。当然，你也可以添加resize属性来让用户拖拽实现内容块的重绘。

绝对居中（Absolute Centering）可以保证内容块始终居中，无论内容块是否重绘。可以通过设置min-/max-来根据自己需要限制内容块的大小，并防止内容溢出窗口/容器。

```
.Absolute-Center.is-Resizable {
  min-width: 20%;
  max-width: 80%;
  min-height: 20%;
  max-height: 80%;
  resize: both;
  overflow: auto;
}
```

六、重绘居中



内容块使用了 `resize: both`
内容始终显示在容器中央

You can resize your content block with other classes or Javascript without having to recalculate the center manually! Adding the resize property will

如果不使用resize:both属性，可以使用CSS3动画属性transition来实现重绘的窗口之间平滑的过渡。一定要设置overflow:auto;以

防重绘的内容块尺寸小于内容的实际尺寸这种情况出现。

绝对居中（AbsoluteCentering）是唯一支持resize:both属性实现垂直居中的技术。

注意：

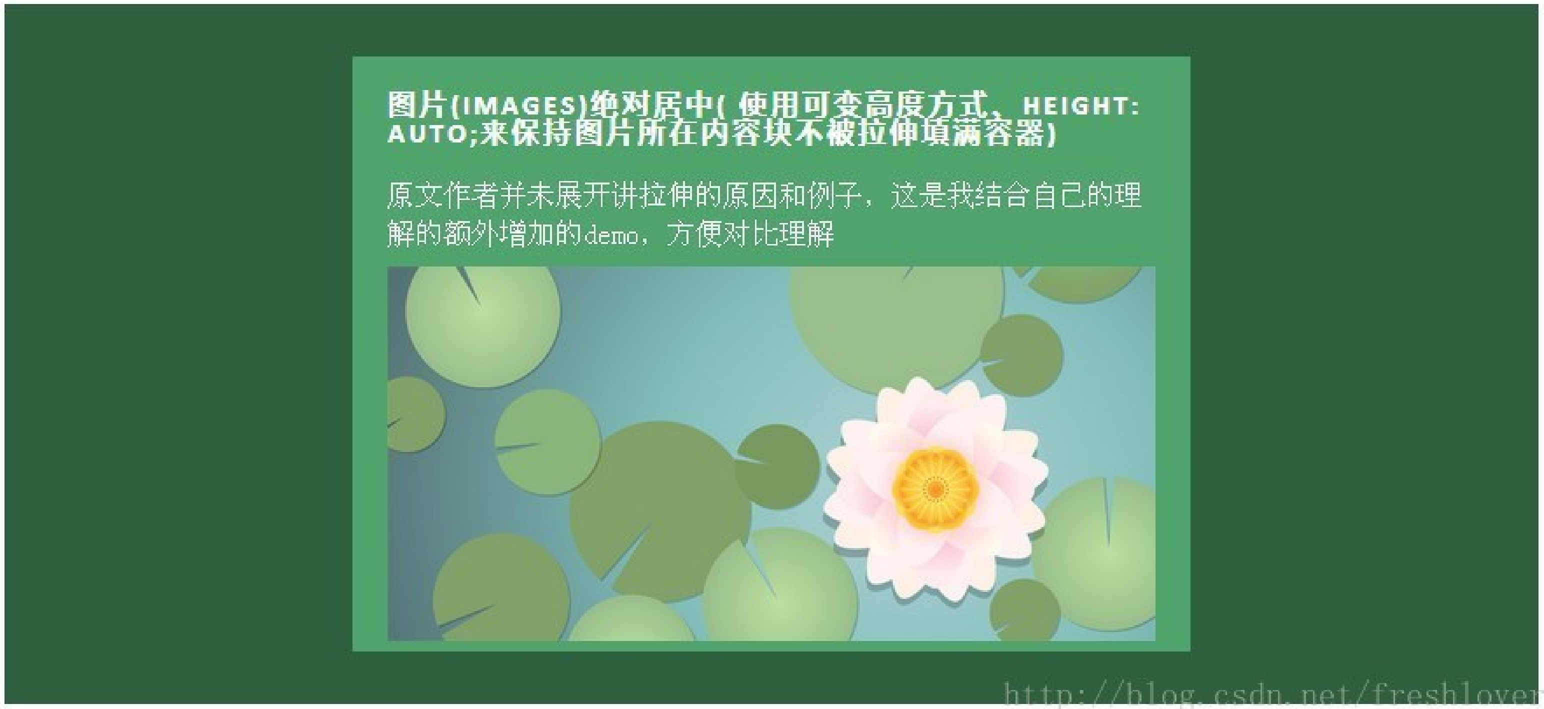
1. 要设置max-width/max-height属性来弥补内容块padding，否则可能溢出。
2. 手机浏览器和IE8-IE10浏览器不支持resize属性，所以如果对你来说，这部分用户体验很必要，务必保证对resizing你的用户有可行的退路。
3. 联合使用resize 和 transition属性会在用户重绘时，产生一个transition动画延迟时间。

七、图片(Images)

绝对居中（**AbsoluteCentering**）也适用于图片。对图片自身应用class类或CSS样式，并给图片添加height:auto样式，图片会自适应居中显示，如果外层容器可以resize则随着容器的重绘，图片也相应重绘，始终保持居中。

需要注意的是height:auto虽然对图片居中有用，但如果是在图片外层的内容块上应用了height:auto则会产生一些问题：规则的内容块会被拉伸填充整个容器。这时，我们可以使用可变高度(Variable Height)方式解决这个问题。问题的原因可能是渲染图片时要计算图片高度，这就如同你自己定义了图片高度一样，浏览器得到了图片高度就不会像其他情况一样去解析margin:auto垂直居中了。所以我们最好对图片自身应用这些样式而不是父元素。

七、图片居中（第一种用法，需要结合可变高度居中方式(即内容块添加is-Variable类)修复才能实现）



图片(IMAGES)绝对居中(使用可变高度方式，HEIGHT: AUTO;来保持图片所在内容块不被拉伸填满容器)

原文作者并未展开讲拉伸的原因和例子，这是我结合自己的理解的额外增加的demo，方便对比理解

HTML:

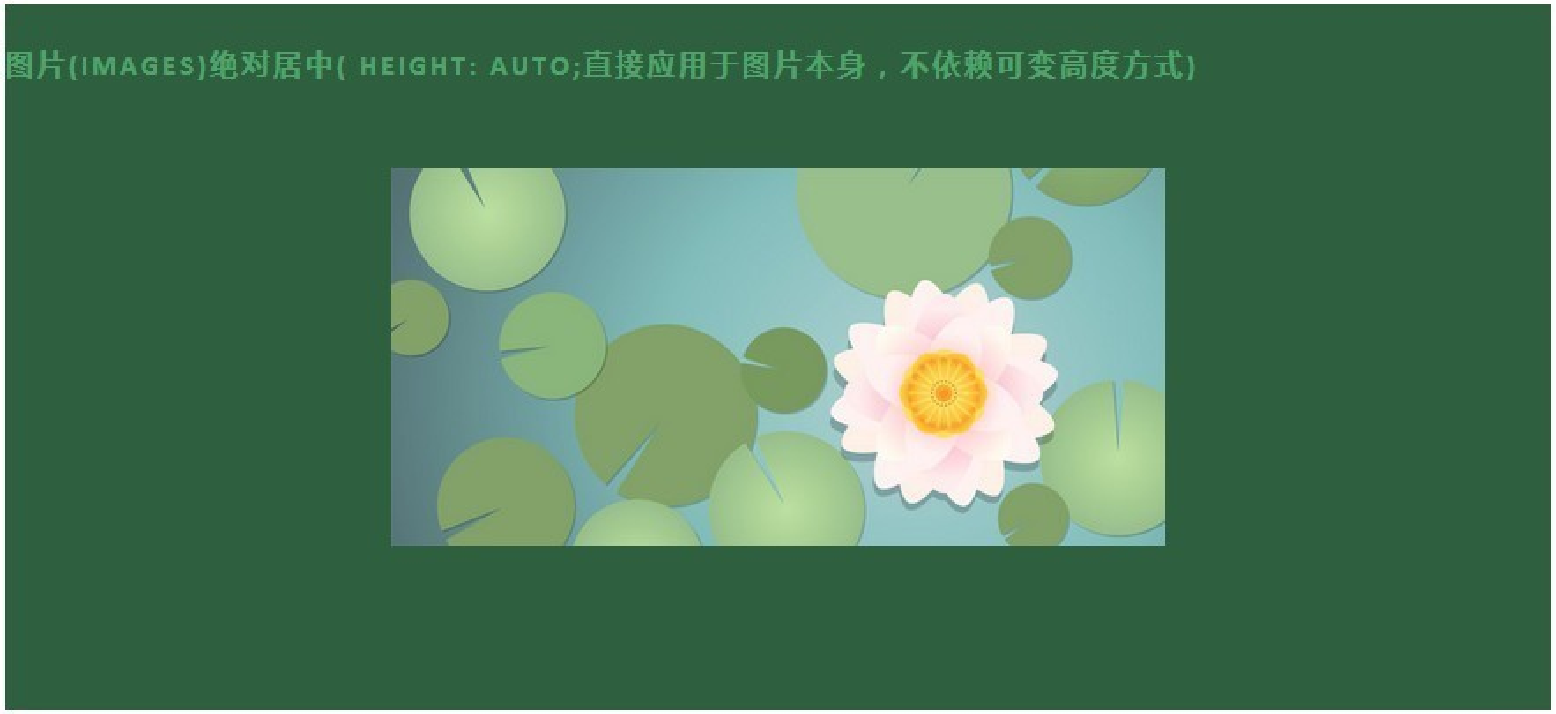
```
<img src="http://placekitten.com/g/500/200" class="Absolute-Center is-Image" alt="" />
```

CSS:

```
.Absolute-Center.is-Image {
  height: auto;
}

.Absolute-Center.is-Image img {
  width: 100%;
  height: auto;
}
```

最好是对图片自身应用此方法，效果如下图：

七、图片居中(第二种用法，即原文所列demo。测试通过)


图片(IMAGES)绝对居中(HEIGHT: AUTO;直接应用于图片本身，不依赖可变高度方式)

八、可变高度（Variable Height）

这种情况下实现绝对居中（AbsoluteCentering）必须要声明一个高度，不管你是基于百分比的高度还是通过max-height控制的高度，还有，别忘了设置合适的overflow属性。对自适应/响应式情景，这种方法很不错。

与声明高度效果相同的另一种方法是设置display:table;这样无论实际内容有多高，内容块都会保持居中。这种方法在一些浏览器（如IE/FireFox）上会有问题，我的搭档Kalley

在ELL Creative（访问ellcreative.com ）上写了一个基于Modernizr插件的检测函数，用来检测浏览器是否支持这种居中方法，进一步增强用户体验。

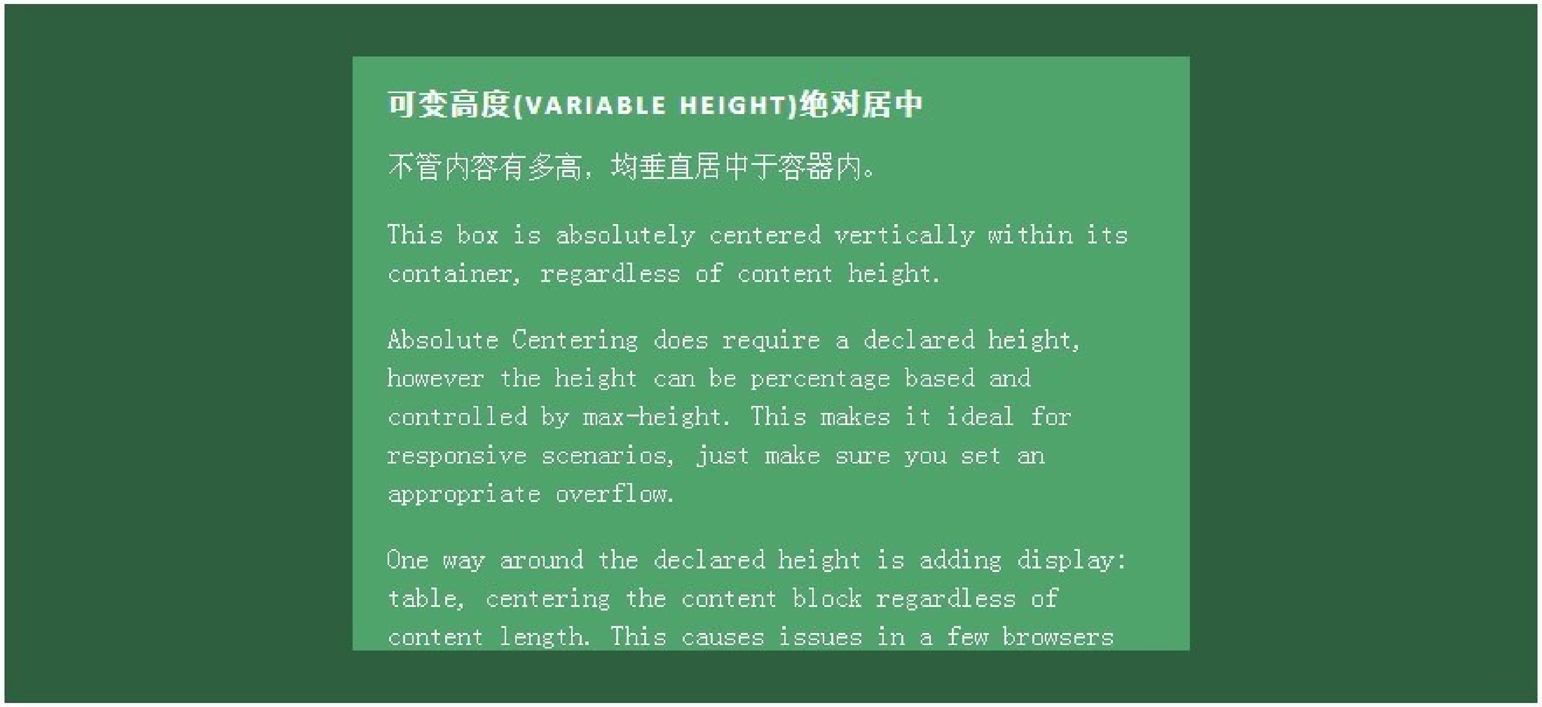
Javascript :

```
/* Modernizr Test for Variable Height Content */
Modernizr.testStyles('#modernizr { display: table; height: 50px; width: 50px; margin: auto; position: absolute; top
: 0; left: 0; bottom: 0; right: 0; }', function(elem, rule) {
  Modernizr.addTest('absolutecentercontent', Math.round(window.innerHeight / 2 - 25) === elem.offsetTop);
});
```

CSS :

```
.absolutecentercontent .Absolute-Center.is-Variable {
  display: table;
  height: auto;
}
```

八、可变高度



可变高度(VARIABLE HEIGHT)绝对居中

不管内容有多高，均垂直居中于容器内。

This box is absolutely centered vertically within its container, regardless of content height.

Absolute Centering does require a declared height, however the height can be percentage based and controlled by max-height. This makes it ideal for responsive scenarios, just make sure you set an appropriate overflow.

One way around the declared height is adding display: table, centering the content block regardless of content length. This causes issues in a few browsers

缺点：

浏览器兼容性不太好，若Modernizr不能满足你的需求，你需要寻找其他方法解决。

1. 与上述重绘(Resizing)情况的方法不兼容
2. Firefox/IE8:使用display:table会使内容块垂直居上，不过水平还是居中的。
3. IE9/10: 使用display:table会使内容块显示在容器左上角。
4. Mobile Safari:内容块垂直居中；若使用百分比宽度，水平方向居中会稍微偏离中心位置。

Ⅱ.其他居中实现技术

绝对居中（Absolute Centering）是一种非常不错的技术，除此之外还有一些方法可以满足更多的具体需求，最常见的推荐：NegativeMargins, Transforms,Table-Cell, Inline-Block方式和新出现的Flexbox.方式。这些方法许多文章都有深入讲解，这里只做简

单阐述。

九、负外边距(Negative Margins)

这或许是当前最流行的使用方法。如果块元素尺寸已知，可以通过以下方式让内容块居中于容器显示：

外边距margin取负数，大小为width/height（不使用box-sizing: border-box时包括padding，）的一半，再加上top: 50%; left : 50%;。即：

```css
.is-Negative {
        width: 300px;
        height: 200px;
        padding: 20px;
        position: absolute;
        top: 50%; left: 50%;
        margin-left: -170px; /* (width + padding)/2 */
        margin-top: -120px; /* (height + padding)/2 */
}
```

九、其他居中方式：margin负间距

MARGIN负间距(NEGATIVE MARGINS)绝对居中
通过设置Margin负间距，使其水平垂直居中于容器内。

测试表明，这是唯一在IE6-IE7上也表现良好的方法。

优点：

1.　良好的跨浏览器特性，兼容IE6-IE7。

2.　代码量少。

缺点：

1.　不能自适应。不支持百分比尺寸和min-/max-属性设置。

2.　内容可能溢出容器。

3.　边距大小与padding,和是否定义box-sizing: border-box有关，计算需要根据不同情况。

十、变形（Transforms）

这是最简单的方法，不近能实现绝对居中同样的效果，也支持联合可变高度方式使用。内容块定义transform: translate(-50%,-50

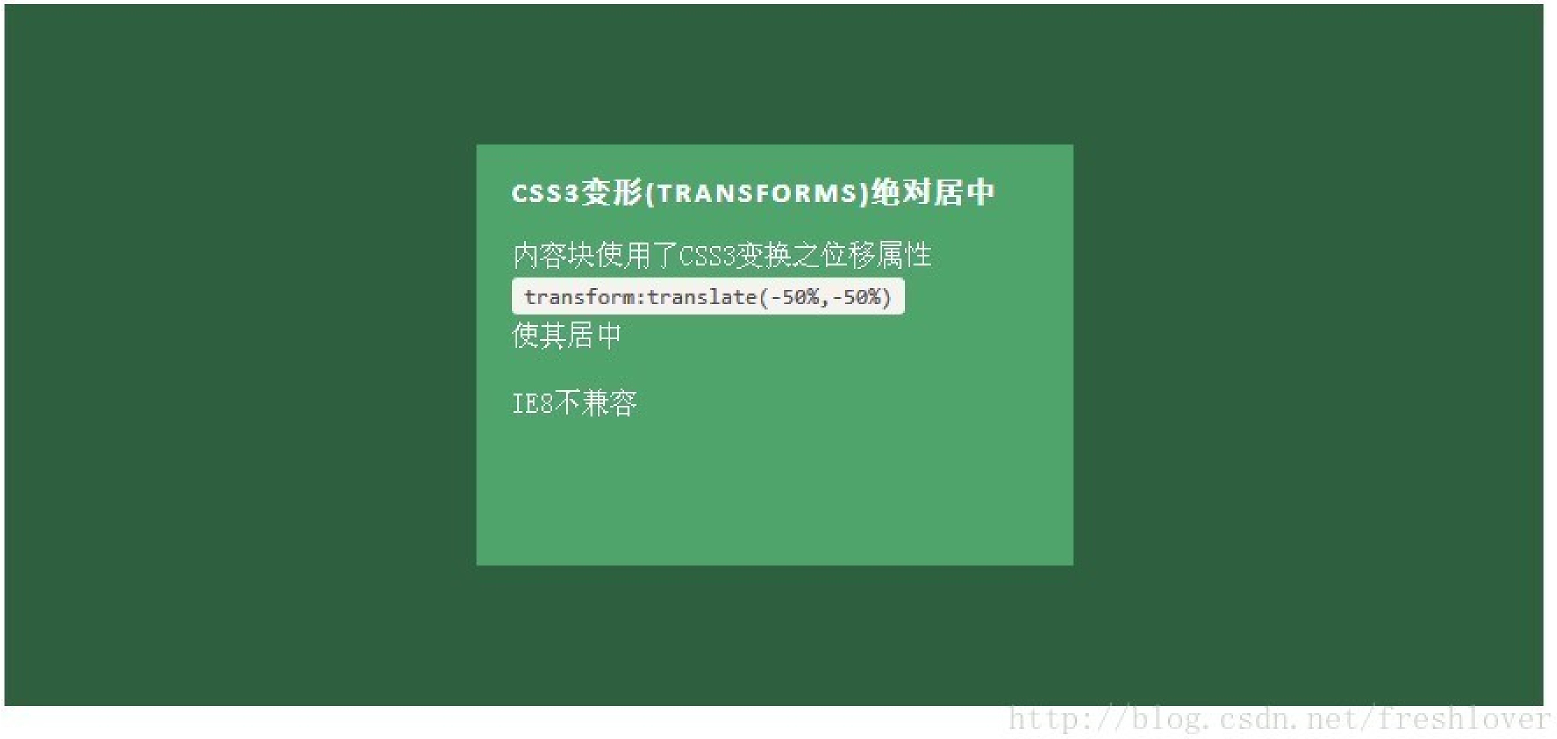
%)必须带上浏览器厂商的前缀，还要加上

top: 50%; left: 50%;

代码类：

```
.is-Transformed {
  width: 50%;
  margin: auto;
  position: absolute;
  top: 50%; left: 50%;
  -webkit-transform: translate(-50%,-50%);
      -ms-transform: translate(-50%,-50%);
          transform: translate(-50%,-50%);
}
```

十、其他居中方式：CSS3变形Transforms


CSS3变形(TRANSFORMS)绝对居中

内容块使用了CSS3变换之位移属性

```
transform:translate(-50%,-50%)
```

使其居中

IE8不兼容

优点：

1. 内容可变高度
2. 代码量少

缺点：

1. IE8不支持
2. 属性需要写浏览器厂商前缀
3. 可能干扰其他transform效果
4. 某些情形下会出现文本或元素边界渲染模糊的现象

进一步了解transform实现居中的知识可以参考CSS-Tricks的文章《Centering PercentageWidth/Height Elements》

十一、表格单元格（Table-Cell）

总的说来这可能是最好的居中实现方法，因为内容块高度会随着实际内容的高度变化，浏览器对此的兼容性也好。最大的缺点是需要

大量额外的标记，需要三层元素让最内层的元素居中。

HTML：

```
<div class="Center-Container is-Table">
  <div class="Table-Cell">
    <div class="Center-Block">
    <!-- CONTENT -->
    </div>
  </div>
</div>
```

CSS：

```
.Center-Container.is-Table { display: table; }
.is-Table .Table-Cell {
  display: table-cell;
  vertical-align: middle;
}
.is-Table .Center-Block {
  width: 50%;
  margin: 0 auto;
}
```

十一、其他居中方式：Table-Cell实现居中

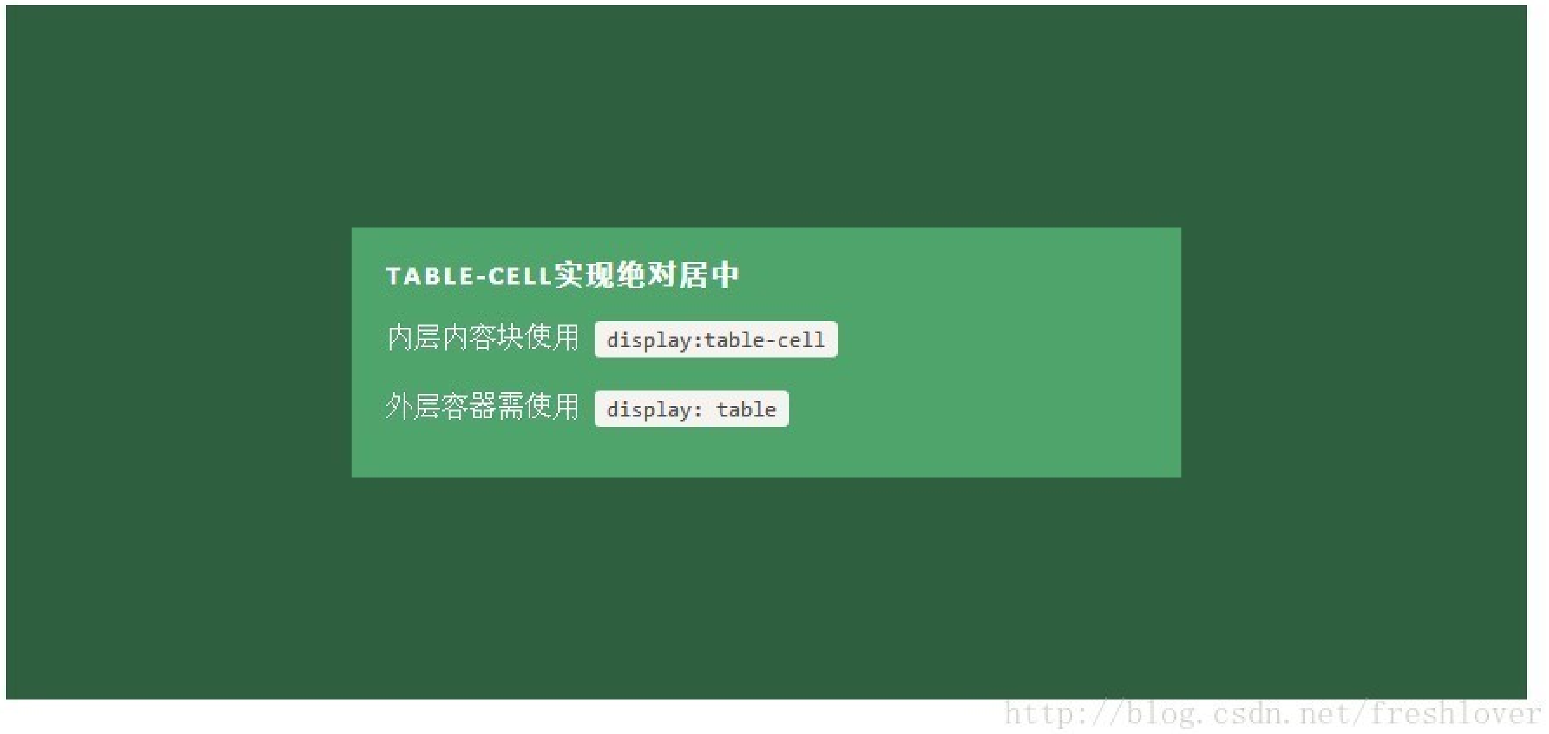


TABLE-CELL实现绝对居中

内层内容块使用 `display:table-cell`

外层容器需使用 `display: table`

优点：

1. 高度可变
2. 内容溢出会将父元素撑开。
3. 跨浏览器兼容性好。

缺点：

需要额外html标记

了解更多表格单元格实现居中的知识，请参考Roger Johansson发表在456bereastreet的文章《Flexibleheight vertical centering with CSS, beyond IE7》

十二、行内块元素（Inline-Block）

很受欢迎的一种居中实现方式，基本思想是使用display: inline-block, vertical-align: middle和一个伪元素让内容块处于容器中央

。这个概念的解释可以参考CSS-Tricks上的文章《Centering in the Unknown》

我这个例子也有一些其他地方见不到的小技巧，有助于解决一些小问题。

如果内容块宽度大于容器宽度，比如放了一个很长的文本，但内容块宽度设置最大不能超过容器的100%减去0.25em，否则使用伪元素:after内容块会被挤到容器顶部，使用:before内容块会向下偏移100%。

如果你的内容块需要占据尽可能多的水平空间，可以使用max-width: 99%;（针对较大的容器）或max-width: calc(100% -0.25em)（取决于支持的浏览器和容器宽度）。

HTML：

```html
<div class="Center-Container is-Inline">
  <div class="Center-Block">
    <!-- CONTENT -->
  </div>
</div>
```

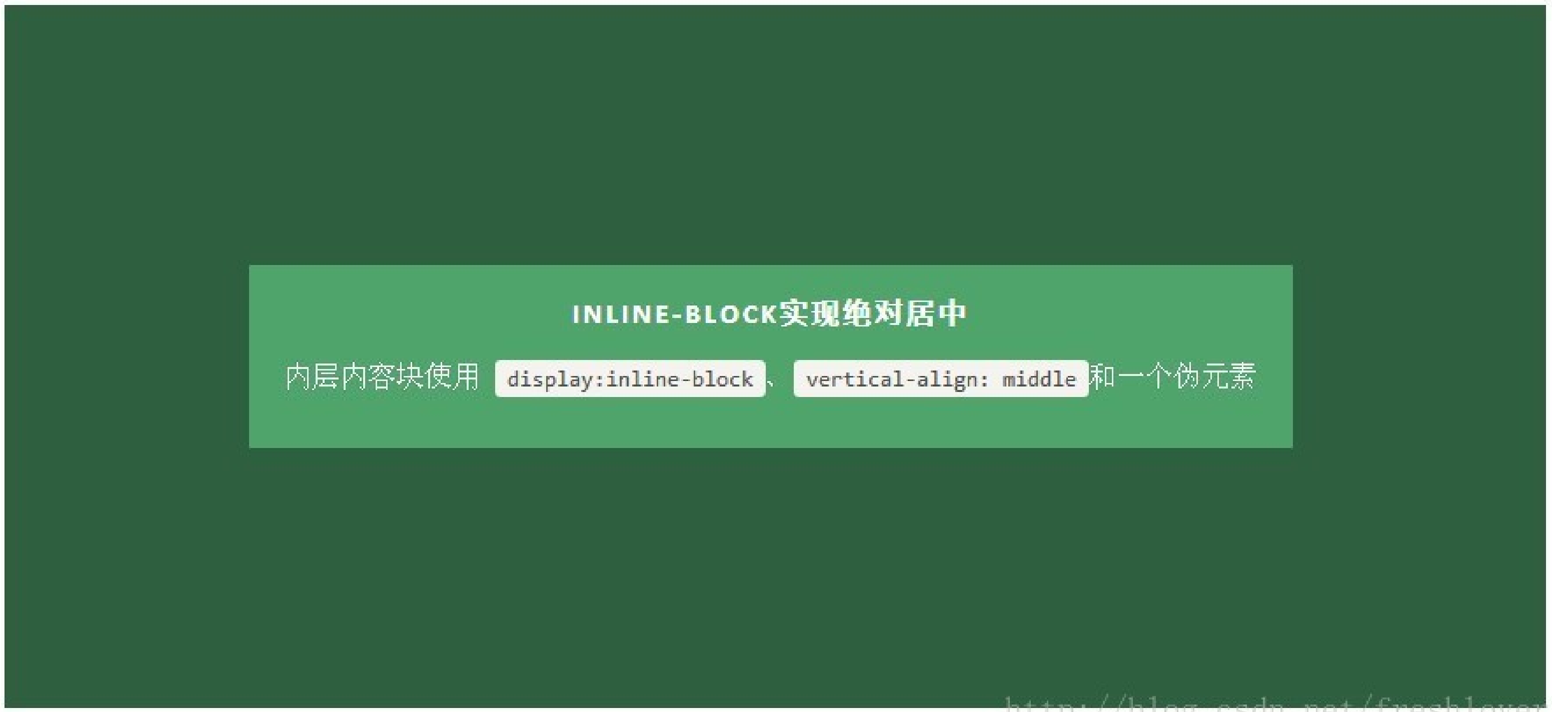
CSS：

```css
.Center-Container.is-Inline {
  text-align: center;
  overflow: auto;
}

.Center-Container.is-Inline:after,
.is-Inline .Center-Block {
  display: inline-block;
  vertical-align: middle;
}

.Center-Container.is-Inline:after {
  content: '';
  height: 100%;
  margin-left: -0.25em; /* To offset spacing. May vary by font */
}

```

```
.is-Inline .Center-Block {
  max-width: 99%; /* Prevents issues with long content causes the content block to be pushed to the top */
  /* max-width: calc(100% - 0.25em) /* Only for IE9+ */
}
```

这种方法的优劣和单元格Table-Cell方式差不多，起初我把这种方式忽略掉了，因为这确实是一种hack方法。不过，无论如何，这是很流行的一种用法，浏览器支持的也很好。

十二、其他居中方式：Inline-Block实现居中


INLINE-BLOCK实现绝对居中
内层内容块使用 display:inline-block 、 vertical-align: middle 和一个伪元素

优点：

1. 高度可变
2. 内容溢出会将父元素撑开。
3. 支持跨浏览器，也适应于IE7。

缺点：

1.需要一个容器

2.水平居中依赖于margin-left: -0.25em;该尺寸对于不同的字体/字号需要调整。

3.内容块宽度不能超过容器的100% - 0.25em。

更多相关知识参考ChrisCoyier的文章《Centeringin the Unknown》

十三、Flexbox

这是CSS布局未来的趋势。Flexbox是CSS3新增属性，设计初衷是为了解决像垂直居中这样的常见布局问题。相关的文章如《Centering Elements with Flexbox》

记住Flexbox不只是用于居中，也可以分栏或者解决一些令人抓狂的布局问题。

十三、其他居中方式：Flexbox实现居中

优点：

1.内容块的宽高任意，优雅的溢出。

2.可用于更复杂高级的布局技术中。

缺点：

1.　　IE8/IE9不支持。

2.　　Body需要特定的容器和CSS样式。

3.　　运行于现代浏览器上的代码需要浏览器厂商前缀。

4.　　表现上可能会有一些问题

有关Flexbox Centering的文章可以参考David Storey的文章《Designing CSS Layouts WithFlexbox Is As Easy As Pie》

建议：

每种技术都有其优劣之处。你选择哪一种技术取决于支持的浏览器和你的编码。使用上面的对照表有助于你做出决定。

作为一种简单的替代方案，绝对居中(Absolute Centering)技术表现良好。曾经你使用负边距（Negative Margins）的地方，现在可以用绝对居中(Absolute Centering)替代了。你不再需要处理讨厌的边距计算和额外的标记，而且还能让内容块自适应大小居中。

如果你的站点需要可变高度的内容，可以试试单元格(Table-Cell)和行内块元素(Inline-Block)这两种方法。如果你处在流血的边缘，试试Flexbox，体验一下这一高级布局技术的好处吧。

1

1

1

1

1

1

1

1

xgqfrms

标签: css, CSS3实现, 垂直居中, 水平居中, 的技巧, How To Center Anything With CSS, Front End, Code My Views

好文要顶 关注我 收藏该文

xgqfrms
关注 - 1
粉丝 - 2

1 推荐　0 反对

(请您对文章做出评价)

« 上一篇：Cordova 入门和基础：Cordova,html5, hybrid,android,

» 下一篇：如何成为一个让人喜欢的好室友/陌生人！

posted @ 2016-04-17 19:36 xgqfrms 阅读(12) 评论(3) 编辑 收藏

评论列表

#1楼[楼主] 2016-04-29 13:06 xgqfrms　修改 删除

!important

支持(0) 反对(0)

#2楼[楼主] 2016-07-07 07:56 xgqfrms　修改 删除

https://codemyviews.com/blog/the-lowdown-on-absolute-vs-relative-positioning

The Lowdown On Absolute vs. Relative Positioning

支持(0) 反对(0)

#3楼[楼主] 2016-07-07 07:57 xgqfrms 修改 删除

lowdown生词本

英 ['ləʊdaʊn] 美 ['loˌdaʊn]

n. <俚>**内幕，真相，实情**

网 络

内幕； 真相； 实情

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

发表评论

修改成功

昵称： xgqfrms

评论内容：

提交评论 不改了 退出登录

修改成功

[Ctrl+Enter快捷键提交]

最新IT新闻:
· 有理想情怀的极客，运气不会差
· 贾跃亭致北美员工内部信曝光 乐视或要在美国卖手机
· 微软前合伙人转投阿里巴巴 任阿里云首席科学家
· 武汉暴雨致联想工厂被大水包围 ZUK Z2手机线上供应受影响
· 高通诉讼魅族案新进展？魅族总裁称将对新机MX 6进行涨价
» 更多新闻...

最新知识库文章:
· 抽象：程序员必备的能力
· 编程同写作，写代码只是在码字

· 遇见程序员男友
· 设计师的视觉设计五项修炼
· 我听到过的最精彩的一个软件纠错故事
» 更多知识库文章...