RESPONSIVE IMAGES

Fluid
inline images

To make your images adapt to any size, you can use **a single, powerful declaration**. This declaration will allow images to be reduced in size, but never increase larger the images maximum size.

```css
img
{
    max-width: 100%;
}
```

We often want our images to sit properly within boxes, **without space under the image**. This can be achieved by converting the image from inline to block.

```
img
{
    max-width: 100%;
    vertical-align: middle;
}
```

A new declaration can be added to **remove border when inside an <a> element** in IE 8 and 9.

```
img
{
    max-width: 100%;
    vertical-align: middle;
    border: 0;
}
```

The **-ms-interpolation-mode property** improves image quality when scaled in IE7. It is on by default in IE8 and is not implemented in IE6.

The possible properties are:

bicubic
Complex interpolation algorithm to make higher quality large images by processing small ones.

nearest-neighbor

Simple interpolation algorithm to enlarge images. The image quality is lower than with the use of bicubic, but the process is faster this way.

```css
img
{
    max-width: 100%;
    vertical-align: middle;
    border: 0;
    -ms-interpolation-mode: bicubic;
}
```

The **image-rendering property** improves image quality when scaled in FF3.6 and up.

The possible properties are:

-moz-crisp-edges
Same as the optimizeSpeed

auto
Default. Same as the optimizeQuality

optimizeQuality
Bilinear interpolation algorithm

optimizeSpeed

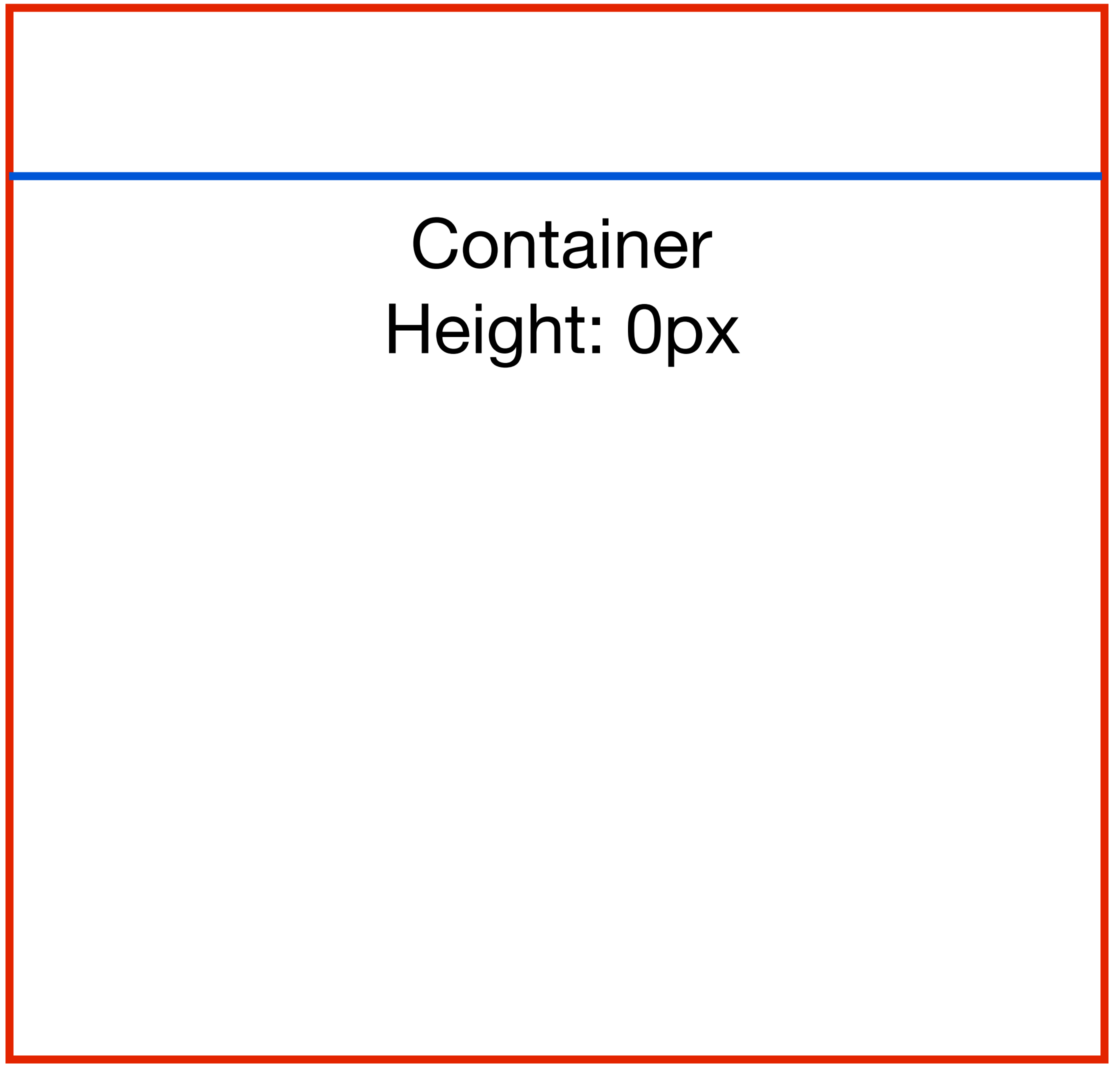
Nearest-neighbor interpolation algorithm

```css
img
{
    max-width: 100%;
    vertical-align: middle;
    border: 0;
    -ms-interpolation-mode: bicubic;
    image-rendering: optimizeQuality;
}
```

Responsive background images

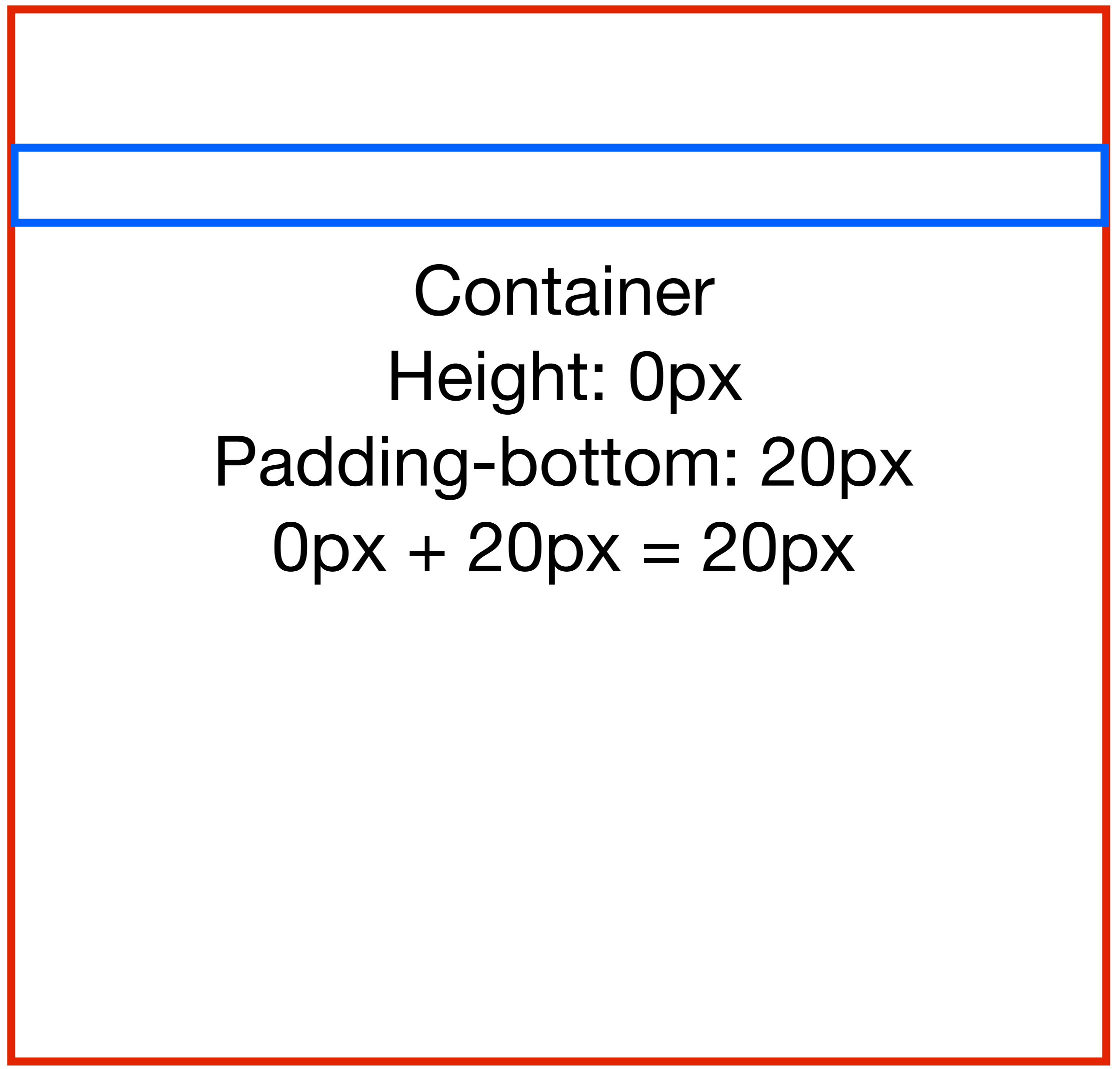
Before we look at responsive background images, we need to understand how containers work - **especially when they have nothing inside them**.

If we have a container with no content inside it, it will spread to the width of it’s parent container, or the viewport... However, it’s height will collapse - it will have **no height**.


Container
Height: 0px

If we add padding to the top or bottom of this container, this padding will be **added to the overall height**.

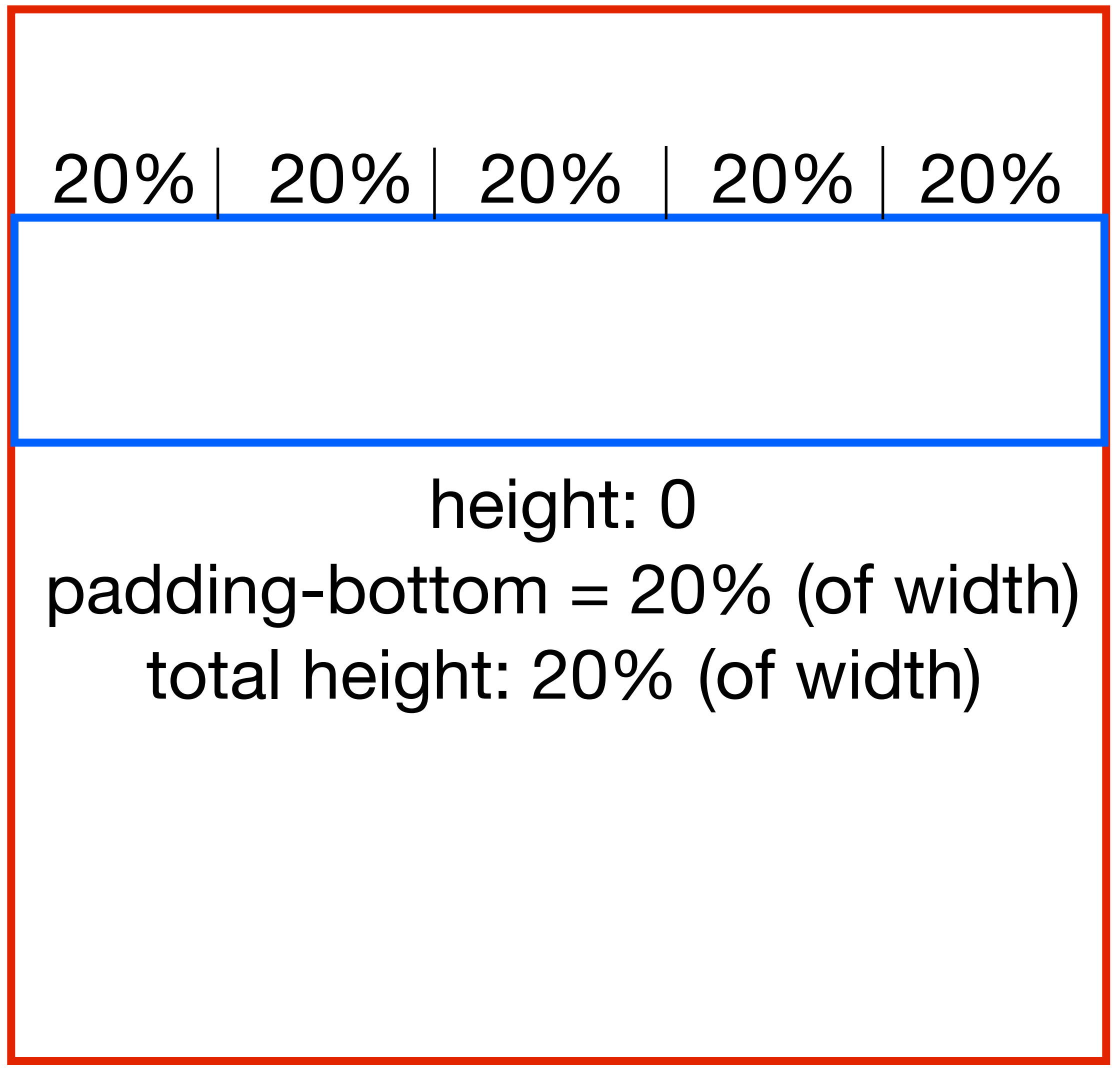
For example, if we have a container with nothing inside, it’s height will be “0”. If we add 20px of padding to the bottom of this container, the **overall height of the container will now be 20px**.


Container
Height: 0px
Padding-bottom: 20px
0px + 20px = 20px

We are going to use this method, but instead of using a length value for our padding- like 20px - **we are going to use a percentage value**.

But what if we add “padding-bottom: 20%”... what does the 20% mean?

This 20% actually means **“20% of the total width of the container”**.


20%
20%
20%
20%
20%
height: 0
padding-bottom = 20% (of width)
total height: 20% (of width)

for our fluid background image, we are going to **scales the height of the container** as the layout is reduced in width.

But how can we **force the height of the container to stay the same ratio as the width** of the container, even when we don’t know the actual width?

Step 1:

Calculate the ratio between height and width of the image and convert this into a percentage.

height ÷ width = ratio

460px ÷ 800px = 0.575 (57.5%)

Step 2:
Add add this percentage as padding to the container.

```css
.responsive-image
{
    width: 100%;
    padding-bottom: 57.5%;
    background-image:
        url(background-image.jpg);
    background-size: cover;
    background-position: center;
}
```

High resolution background images

There are **all sorts of devices coming out** with higher resolution screens. We have devices with device-pixel ratios of 1.5, 2 and even 3.

Here is a **test suite and detailed list** of devices and pixel density:

http://bjango.com/articles/min-device-pixel-ratio/

Resolution media queries

In order to target these different ratios we need a range of **resolution-based media queries**.

```
/* 1.25 dpr */
@media
    (-webkit-min-device-pixel-ratio: 1.25),
    (min-resolution: 120dpi)
    {
    }
```

```
/* 1.3 dpr */
@media
    (-webkit-min-device-pixel-ratio: 1.3),
    (min-resolution: 124.8dpi)
    {
    }
```

```
/* 1.5 dpr */
@media
    (-webkit-min-device-pixel-ratio: 1.5),
    (min-resolution: 144dpi)
    {
    }
```

```css
/* 2.0 dpr */
@media
     (-webkit-min-device-pixel-ratio: 2),
     (min-resolution: 192dpi)
     {
     }
```

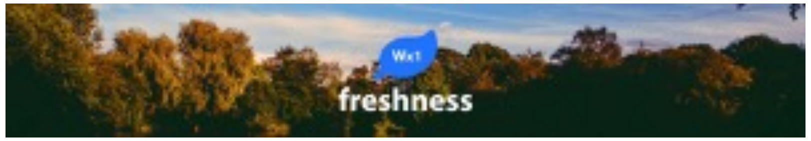
```
/* 3.0 dpr */
@media
    (-webkit-min-device-pixel-ratio: 3),
    (min-resolution: 350dpi)
    {
    }
```

The images

Our mission is to apply different background images based on **width and resolution**.

We want to deliver **smaller images to smaller screen devices**, but we also want to be able to deliver high res images to these small screen devices.

Nx1
freshness
Nx2
freshness
Nx3
freshness


Wx1
freshness

Wx2
freshness

Wx3
freshness

Responsive
inline images via
<picture>

<picture>

The **<picture> element** operates like the HTML5 <audio> and <video> elements.

The <picture> element doesn’t display your image, it just **tells the browser which image to display**.

```
<picture>
    <source media="(min-width: 1200px)"
srcset="filename-lg.jpg, filename-lg-hd.jpg
2x">
    <source media="(min-width: 992px)"
srcset="filename-md.jpg, filename-md-hd.jpg
2x">
    <source media="(min-width: 768px)"
srcset="filename-sm.jpg, filename-sm-hd.jpg
2x">
    <source srcset="filename-xs.jpg,
filename-xs-hd.jpg 2x">
    <img src="filename-xs.jpg" alt="">
</picture>
```

The **<source> element** is used to list all of the possible images files.

```html
<picture>
    <source srcset="filename-lg.jpg">
</picture>
```

The **srcset attribute** gives the browser a list of possible images, along with some (optional) “hints” about the screen resolution and screen size that correspond with each image source.

```html
<picture>
    <source srcset="filename-lg.jpg">
</picture>
```

The srcset attribute also allows you to provide a **comma-separated list of images** including their resolution.

```
<picture>
    <source srcset="filename-lg.jpg,
filename-lg-hd.jpg 2x">
</picture>
```

The **media attribute** is where you would put your @media query information. When the @media attribute evaluates to true, the browser then moves to the associated srcset.

```
<picture>
    <source media="(min-width: 1200px)"
srcset="filename-lg.jpg, filename-lg-hd.jpg
2x">
</picture>
```

The **sizes attribute** allows you to specify a set of intrinsic sizes for the images described in the srcset attribute.

```
<picture>
    <source media="(min-width: 1200px)"
sizes="100%" srcset="filename-lg.jpg,
filename-lg-hd.jpg 2x">
</picture>
```

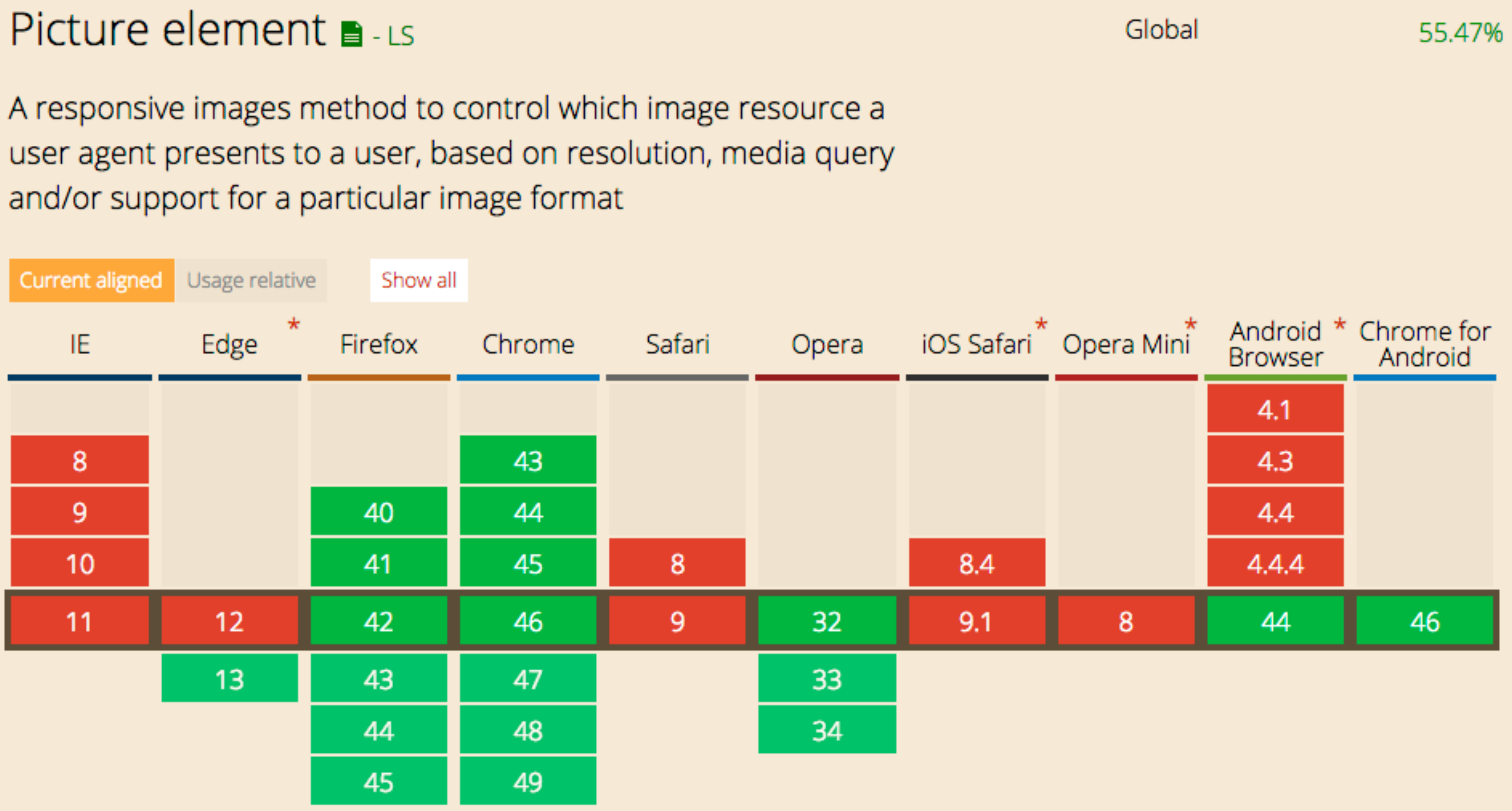
The <picture> element even allows you to include a **standard <img> element** for older browsers that do not support the <picture> element or the various <source> elements.

```
<picture>
    <source media="(min-width: 1200px)"
srcset="filename-lg.jpg, filename-lg-hd.jpg
2x">
    <img src="filename-xs.jpg" alt="">
</picture>
```

Here is a **full example**:

```
<picture>
    <source media="(min-width: 1200px)"
srcset="filename-lg.jpg, filename-lg-hd.jpg
2x">
    <source media="(min-width: 992px)"
srcset="filename-md.jpg, filename-md-hd.jpg
2x">
    <source media="(min-width: 768px)"
srcset="filename-sm.jpg, filename-sm-hd.jpg
2x">
    <source srcset="filename-xs.jpg,
filename-xs-hd.jpg 2x">
    <img src="filename-xs.jpg" alt="">
</picture>
```

Browser support



Picture element - LS

Global 55.47%

A responsive images method to control which image resource a user agent presents to a user, based on resolution, media query and/or support for a particular image format

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
								🟥 4.1	
🟥 8			🟩 43					🟥 4.3	
🟥 9		🟩 40	🟩 44					🟥 4.4	
🟥 10		🟩 41	🟩 45	🟥 8		🟥 8.4		🟥 4.4.4	
🟥 11	🟥 12	🟩 42	🟩 46	🟥 9	🟩 32	🟥 9.1	🟥 8	🟩 44	🟩 46
	🟩 13	🟩 43	🟩 47		🟩 33				
		🟩 44	🟩 48		🟩 34				
		🟩 45	🟩 49						

The <picture> element is only just becoming supported. However, There is a great **polyfill for the <picture> element**:

<http://scottjehl.github.io/picturefill/>

Another alternative is **imgix**, which uses suffixes to allow you to resize, re-res and even tint,darken, lighten your images.

https://www.imgix.com/

Russ Weakley

Max Design

Site: maxdesign.com.au

Twitter: twitter.com/russmaxdesign

Slideshare: slideshare.net/maxdesign

Linkedin: linkedin.com/in/russweakley