

RESPONSIVE MEDIA QUERIES

Media types

Before media queries came along,
the only way we could deliver CSS
to different devices was with
media types.

The 10 media types

all	suitable for all devices
aural	for speech synthesizers
braille	for Braille tactile feedback devices
embossed	for paged Braille printers
handheld	for handheld devices
print	for print material
projection	for projected presentations
screen	for color computer screens
tty	for teletypes and terminals
tv	for television type devices

However, many of the 10 different media types are **not supported by the relevant devices.**

The only **effective media types**
are: “all”, “screen” and “print”

Also, media types give us **very limited control** over different types of media.

Media queries

What are they?

Media queries are **an extension** to the CSS2 media types.

Media queries give us **much more control** over how we deliver CSS to different devices.

A simple example?

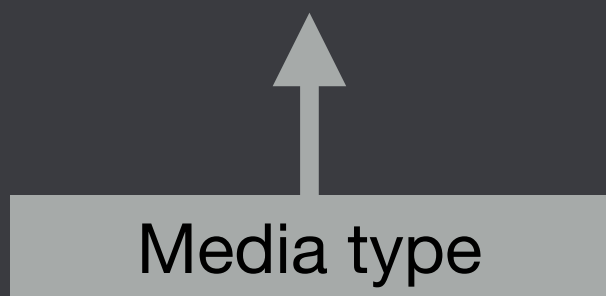
The following rules would only be applied by devices that support the media type of screen, and their viewport has a minimum width of 20em.

```
@media screen and (min-width: 20em)
{
    body { color: red; }
}
```

Media query syntax

Media queries can begin with a **basic media type**.

```
@media screen and (color)
{
}
```

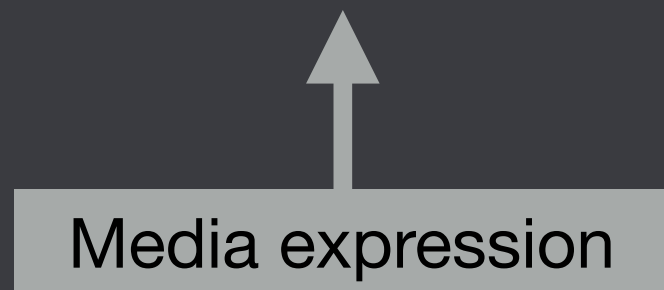


Media type

The diagram consists of a light gray rectangular box containing the text 'Media type'. A vertical arrow points upwards from the top center of this box to the word 'screen' in the CSS code block above it.

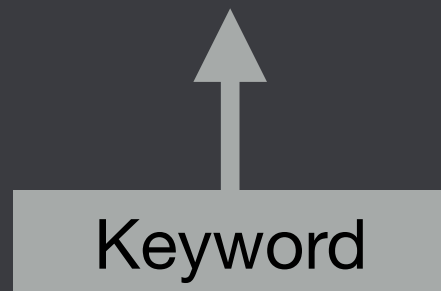
After the media type comes the
media expression.

```
@media screen and (color)
{
}
```



Inside the expression there is often a **keyword**, such as “and”.

```
@media screen and (color)
{
}
```



Media features are placed inside brackets.

```
@media screen and (color)
{
}
```



Media Feature
(inside brackets)

Media features can sometimes be written **with or without a unit value** - depending on the feature.

```
@media screen and (color)
{
}
```

Without unit value

```
@media screen and (color: 256)
{
}
```

With unit value

A media feature can be used without a media type or keyword. The media type is assumed to be **“all”**.

```
@media (color)
```

```
{  
}
```

```
@media [all and] (color)
```

```
{  
}
```

Some media features accept
“**min-**” or “**max-**” prefixes.

```
@media screen and (min-color: 1)
{
}
```

```
@media screen and (max-color: 256)
{
}
```

Multiple
expressions

You can use **multiple expressions** in a media query if you join them with the “and” keyword.

In the **following example**, the CSS rules will only be applied by screen devices, and only if their viewport width is greater than 20em and less than 40em.

```
@media screen and (min-width: 20em) and (max-  
width: 40em)  
{  
    body { color: red; }  
}
```


Comma separated

You can use **multiple, comma-separated** media queries. The comma acts like an “or” keyword.

In the **following example**, the CSS rules will be applied by screen devices with a viewport that has a min-width of 20em, or handheld devices with a viewport that has a max-width of 20em.

```
@media screen and (min-width: 20em), handheld  
and (max-width: 20em)  
{  
    body { color: red; }  
}
```

The “not” expression

You can use the **not keyword** in a media query if you want your CSS to be ignored by a specific device.

In the **following example**, the CSS rules will be applied by all devices except handheld devices that have a viewport with a max-width of 20em.

```
@media not handheld and (max-width: 20em)
{
    body { color: red; }
}
```


The “only”
expression

You can use the **only keyword** in a media query if you want your CSS to be used by a specific device.

In the **following example**, the CSS rules will only be applied by devices that support the screen media type and have a viewport with a min-width of 30em.

```
@media only screen and (min-width: 30em)
{
    body { color: red; }
}
```

Media query features

1. width

Value: <length>

Accepts min/max prefixes: yes

Describes the width of the viewport

```
@media (width: 10em) { }
```

```
@media (min-width: 10em) { }
```

```
@media (max-width: 10em) { }
```

2. height

Value: <length>

Accepts min/max prefixes: yes

Describes the height of the
viewport


```
@media (height: 10em) { }  
@media (min-height: 10em) { }  
@media (max-height: 10em) { }
```

3. device-width

Value: <length>

Accepts min/max prefixes: yes

Describes the rendering surface of the output device - or width of the screen

```
@media (device-width: 10em) { }  
@media (min-device-width: 10em) { }  
@media (max-device-width: 10em) { }
```

4. device-height

Value: <length>

Accepts min/max prefixes: yes

Describes the rendering surface of the output device - or height of the screen

```
@media (device-height: 10em) { }  
@media (min-device-height: 10em) { }  
@media (max-device-height: 10em) { }
```

5. orientation

Value: portrait | landscape

Accepts min/max prefixes: no

The 'orientation' is 'portrait' when the value of the 'height' media feature is greater than or equal to the value of the 'width' media feature.

```
@media (orientation: portrait) { }  
@media (orientation: landscape) { }
```

6. aspect-ratio

Value: <ratio>

Accepts min/max prefixes: yes

Defined as the ratio of the value of the 'width' media feature to the value of the 'height' media feature


```
@media (aspect-ratio: 1280/720) { }  
@media (min-aspect-ratio: 32/18) { }  
@media (max-aspect-ratio: 16/9) { }
```

7. device-aspect-ratio

Value: <ratio>

Accepts min/max prefixes: yes

Defined as the ratio of the value of the 'devicewidth' media feature to the value of the 'device-height' media feature

```
@media (device-aspect-ratio: 16/9) { }  
@media (device-min-aspect-ratio: 32/18) { }  
@media (device-max-aspect-ratio: 32/18) { }
```

8. color

Value: <integer>

Accepts min/max prefixes: yes

Describes the number of bits per color component of the output device

```
@media (color) { }  
@media (min-color: 1) { }  
@media (max-color: 256) { }
```

9. color-index

Value: <integer>

Accepts min/max prefixes: yes

Describes the number of entries in the color lookup table of the output device

```
@media (color-index) { }  
@media (min-color-index: 1) { }  
@media (max-color-index: 256) { }
```

10. monochrome

Value: <integer>

Accepts min/max prefixes: yes

Describes the number of bits per pixel in a monochrome frame buffer.


```
@media (monochrome) { }
```

```
@media (min-monochrome: 1) { }
```

```
@media (max-monochrome: 256) { }
```

11. resolution

Value: <resolution>

Units: dpi | dpcm

Accepts min/max prefixes: yes

Describes the resolution of the output device, i.e. the density of the pixels.

```
@media print and (resolution) { }  
@media print and (min-resolution: 300dpi) { }  
@media print and (max-resolution: 30dpcm) { }
```

12. scan

Value: progressive | interlace

Accepts min/max prefixes: no

Describes the scanning process of "tv" output devices.

```
@media tv and (scan: progressive) { }
```

```
@media tv and (scan: interlace) { }
```

13. grid

Value: <integer>

Accepts min/max prefixes: no

Used to query whether the output device is grid or bitmap

```
@media (grid) and (max-width: 15em) { }
```

Applying media queries

There are **four methods** can be used to apply CSS to web pages using media queries.

Method 1:

HTML `<link>`

Method 1:

A `<link>` element in the head of the HTML document can specify the target media of an external style sheet.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Title</title>
  <link rel="stylesheet" href="a.css"
    media="screen and (color)">
</head>
<body>
</body>
</html>
```

Strengths:

Devices only load the CSS that they need

Weaknesses:

Server requests for each individual CSS file

Method 2:
HTML @import

Method 2:

An @import in the head of your HTML document can specify the target media of an external style sheet.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Title</title>
  <style type="text/css"
    media="screen and (color)">
    @import "a.css";</style>
</head>
<body>
</body>
</html>
```


Weaknesses:

Server requests for each individual CSS file.

Outdated technique which can cause issues in some versions of Internet Explorer.

Method 3:
CSS *@import*

Method 3:

An `@import` in CSS document can specify the target media of an imported style sheet.

```
/* inside your CSS document */  
@import url("a.css") screen and (color);
```

Strengths:

Only one server request for all CSS.

Weaknesses:

Devices download all CSS - even if not relevant. Can cause queuing issues in older Internet Explorers.

Method 4:

CSS @media

Method 4:

An @media rule in the CSS document can specify the target media of subset of CSS rules.

```
/* inside your CSS document */  
@media screen and (color)  
{  
    body { color: blue; }  
}
```


Strengths:

Only one server request for all CSS.

Weaknesses:

Devices download all CSS - even if not relevant.

Browser support

Media Queries: resolution feature - REC

Allows a media query to be set based on the device pixels used per CSS unit. While the standard uses `min/max-resolution` for this, some browsers support the older non-standard `device-pixel-ratio` media query.

Global62.03% + 34.41% = 96.44%

unprefixed:62.03% + 14.34% = 76.37%

Current alignedUsage relativeShow all

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						

As you can see, media queries are supported by all modern browsers and devices **except Internet Explorer 6-8.**

If you need to support Internet Explorer 6-8, there are **three possible solutions**.

Respond.js

A simple JavaScript solution

<https://github.com/scottjehl/Respond>

CSS3-mediaqueries.js

An advanced JavaScript solution

<https://code.google.com/p/css3-mediaqueries-js/>

Give IE6/7/8 simpler CSS

Provide different style sheets to older Internet Explorer versions via Conditional Comments.


```
<!--[if lt IE 9]>  
    <link rel="stylesheet" href="b.css">  
<![endif]-->
```

Breakpoints

Breakpoints are the **points** (or specific viewport dimensions) where your RWD layout adapts or changes.

An example might be the point where a layout **changes from three columns to two columns** as the viewport transitions from a wide screen width to a narrower screen width.

Breakpoints are controlled using
CSS3 media queries.

Content-based breakpoints

In the past, many people used **common device widths** for breakpoints - such as 320, 480, 600, 768 and 1024.

This approach is **not maintainable** as there are new devices with different sized screens coming into the market all the time.

Break-points should be set according to the **needs of our content** (text, line-length, navigation etc) instead of devices.

Testing breakpoints

While there are some great tools out there for testing your layouts at various widths, be careful of **falling into two common traps.**

Device focussed testing

This is where you focus on testing at specific device widths - like for iPhone width only.

Viewport only testing

This is where you test only using a browser (moving the viewport in and out) and assume this is acceptable. Nothing beats testing in a range of real devices!

Three breakpoint approaches

There are **three overall approaches** to deliver CSS based on breakpoints. Each of these approaches have strengths and weaknesses.

XS
only

SM
only

MD
only

LG
only


```
// XS only
```

```
@media (max-width: @screen-xs-max)
```

```
// SM only
```

```
@media (min-width: @screen-sm-min) and (max-width: @screen-sm-max)
```

```
// MD only
```

```
@media (min-width: @screen-md-min) and (max-width: @screen-md-max)
```

```
// LG only
```

```
@media (min-width: @screen-lg-min)
```

XS & up →

SM & up →

MD & up →

LG

```
// SM
```

```
@media (min-width: @screen-sm-min)
```

```
// MD
```

```
@media (min-width: @screen-md-min)
```

```
// LG
```

```
@media (min-width: @screen-lg-min)
```

XS

← SM & down

← MD & down

← LG & down

```
// SM
```

```
@media (max-width: @screen-sm-max)
```

```
// MD
```

```
@media (max-width: @screen-md-max)
```

```
// LG
```

```
@media (max-width: @screen-lg-max)
```



Russ Weakley

Max Design

Site: maxdesign.com.au

Twitter: twitter.com/russmaxdesign

Slideshare: slideshare.net/maxdesign

Linkedin: linkedin.com/in/russweakley