Responsive Web Design

# Viewport

The area of the screen that the browser can render content to.

The viewport width is not the same as the pixel density of the screen it is rendering to.

## Hardware Pixels

The actual width of the screen in pixels

## Device Independent Pixels

The browser reports the screen width in DIPS. Which is

Unit of measurement that relate pixels to a real distance.

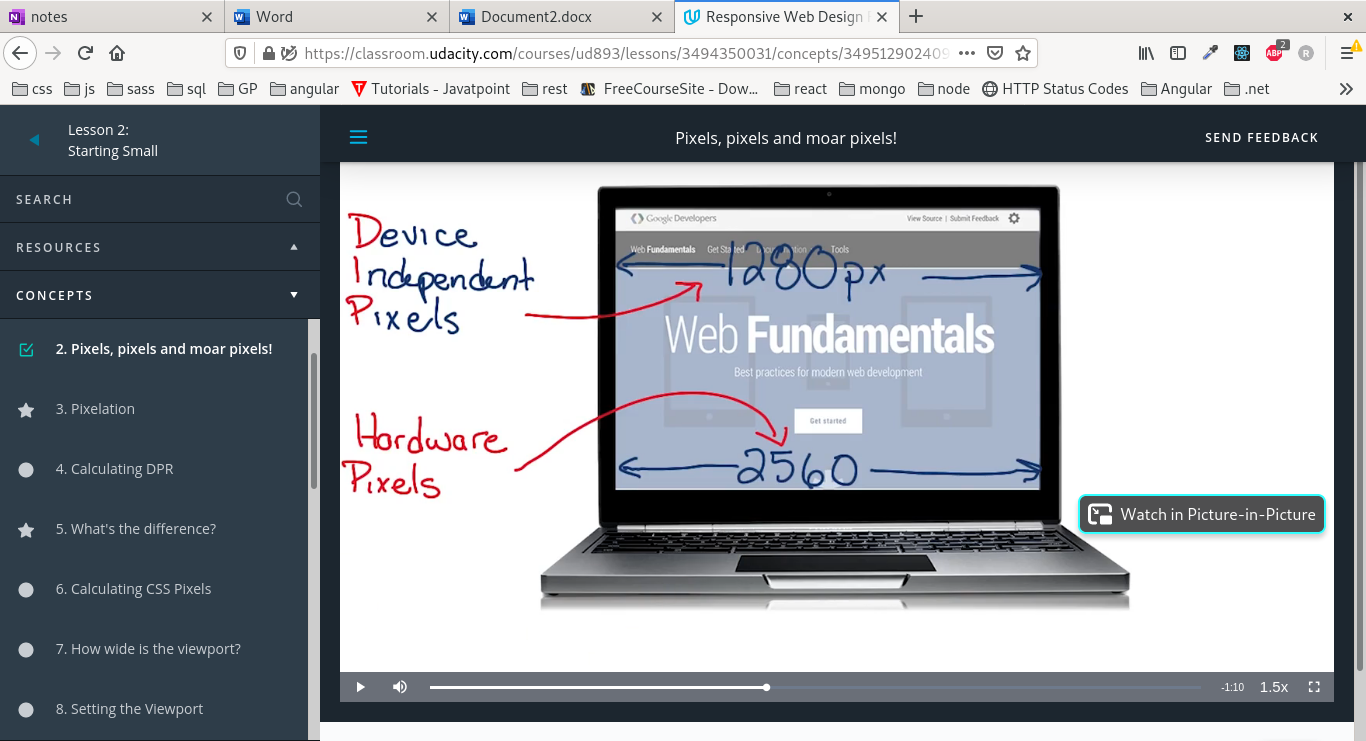
It is regardless of the pixel density of the display it is rendering the content to.

## Device Pixel ratio

The DIPS get scaled to actual Hardware Pixels.

DPR = HW Pixels / DIPS

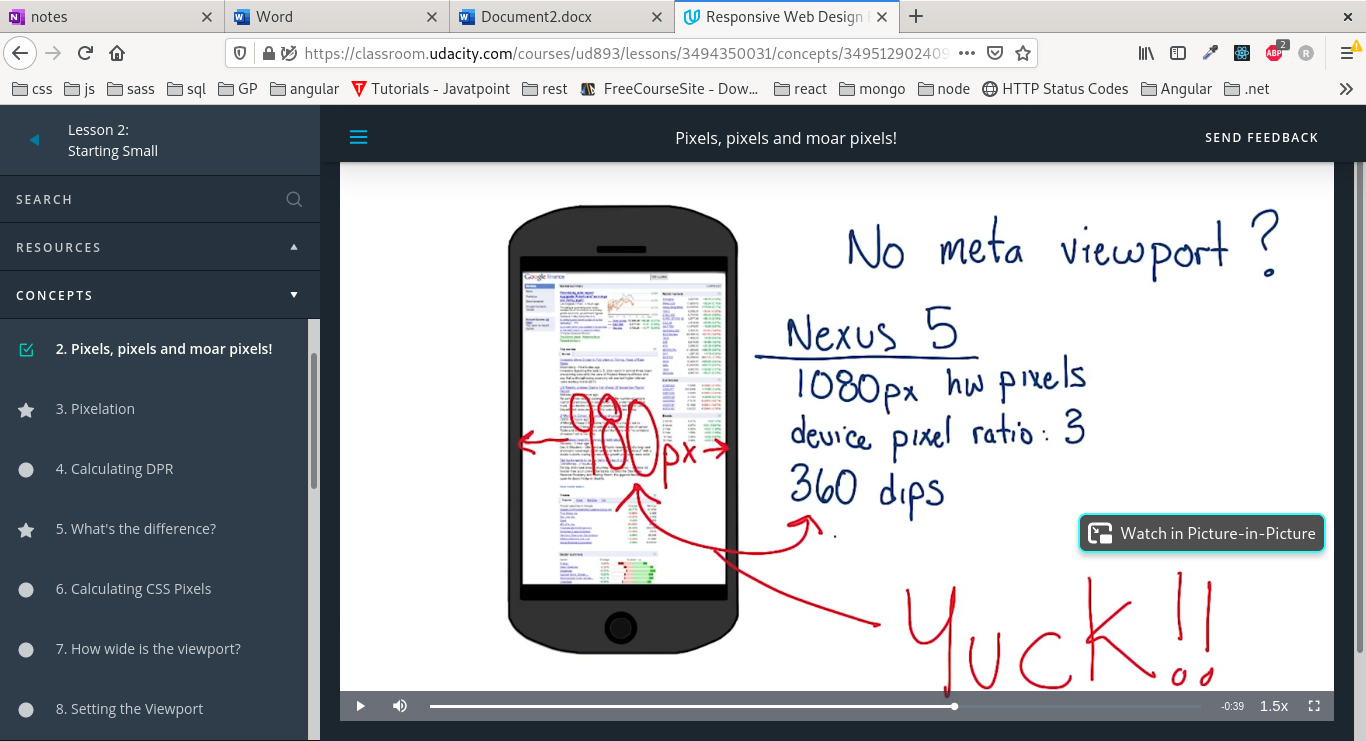
So the device pixel ratio in the following example is 2560/1280 = **2**



## Setting the Viewport in meta tag

If we don’t tell the browser that our website is designed to work on a small screen, it assumes that it weren't.

And it renders the content to the small screen as if it were a large screen, get scaled to half or more.



1080/3 = 360 DIPS

So a content that was designed to run on 980 DIPS is running on 360 DIPS < 0.5.

### Font Boosting

The browser also tries to enlarge some of the content of the page by increasing the font size, so some elements will have large and others will have small font sizes

### Setting the viewport

<meta name="viewport" content="width=device-width,initial-scale=1">

Width=device-width: tells the browser to match the device width in DIPS

Initial-scale=1: tells the browser that the ratio between DIPS and CSS pixels is 1

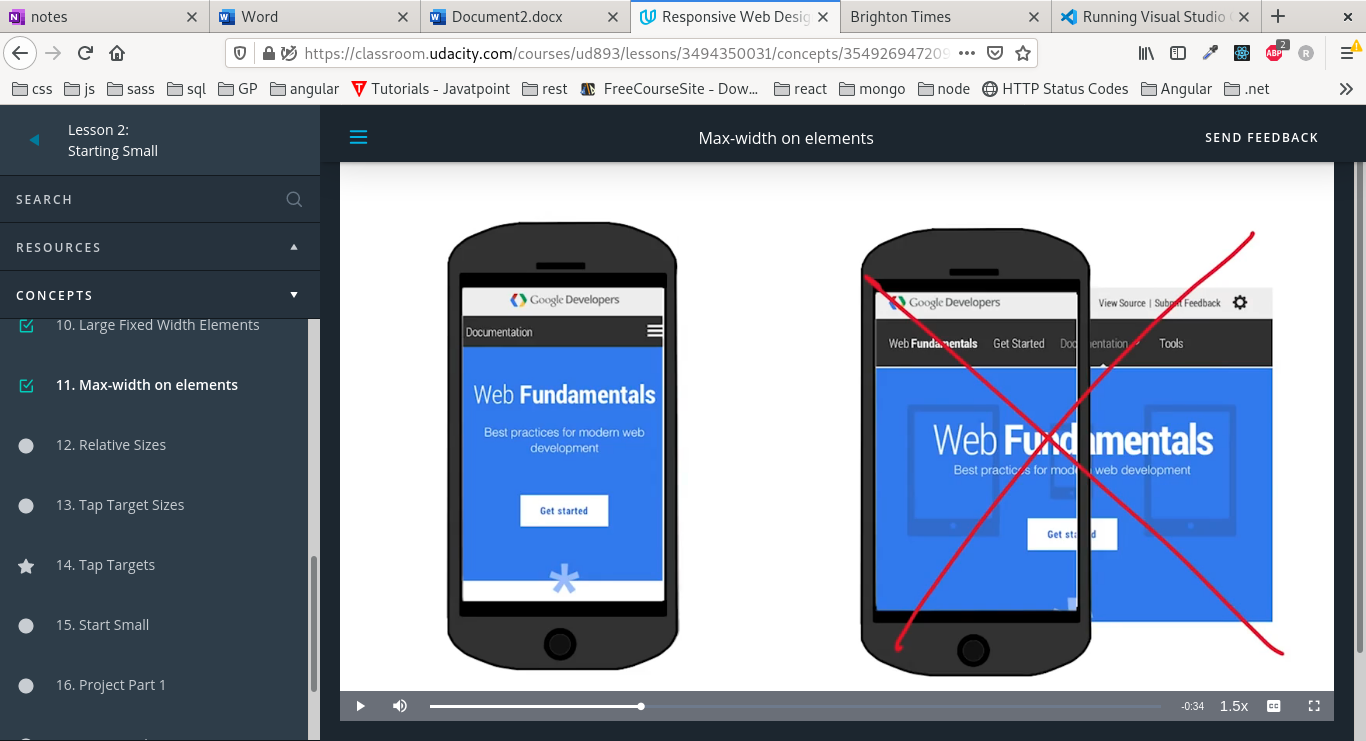
# CSS

## Over-flowing the viewport

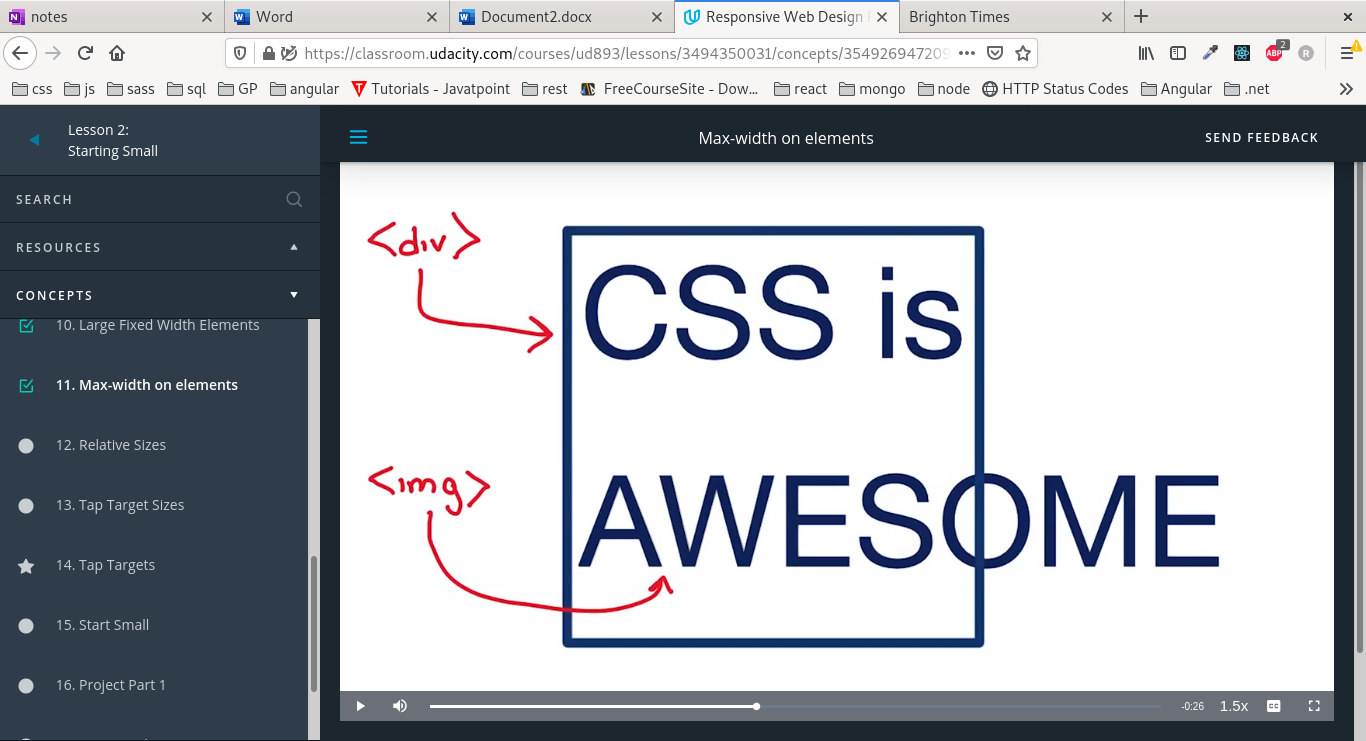
Fixed CSS and Absolute position rely on the width of the viewport, which is different from one device to another.

We shouldn’t rely on it to render images and content. Instead, we should use relative position and percentages in width.

This will prevent elements from over-flowing the viewport.

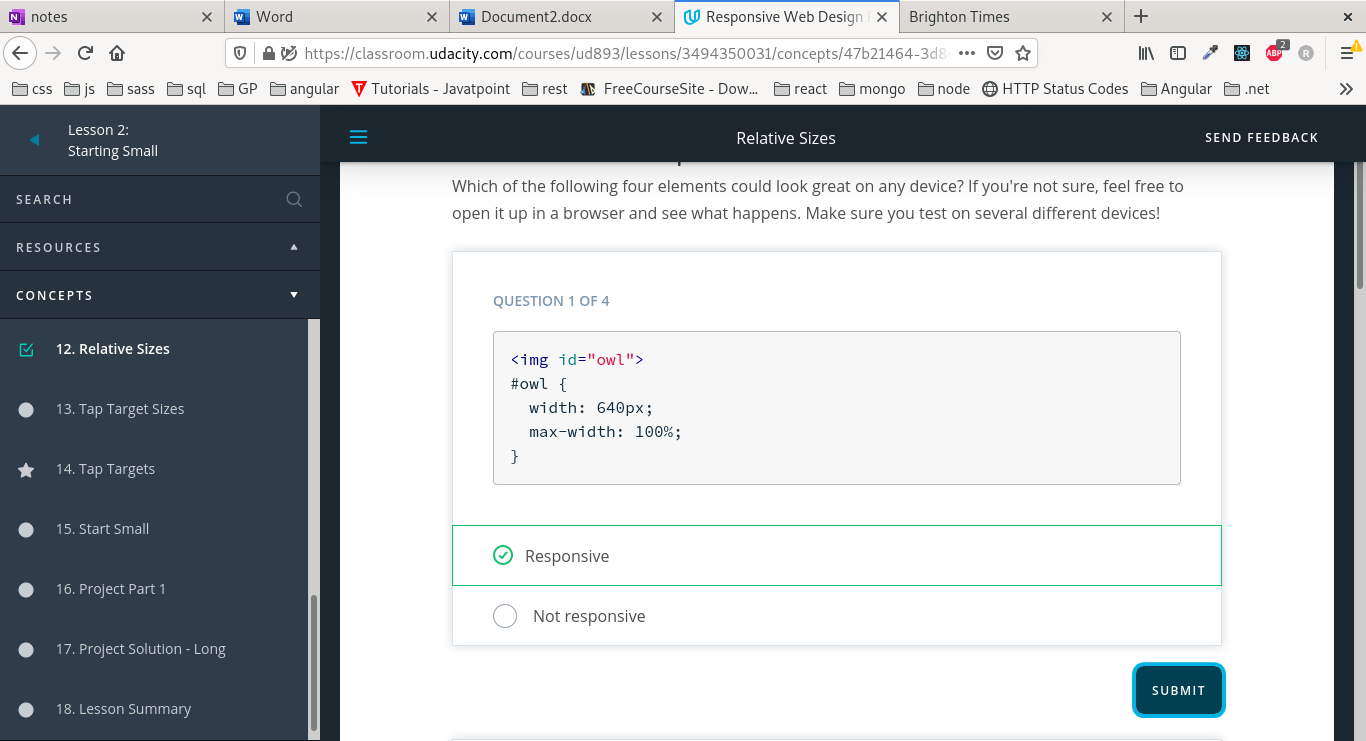


## Over-flowing the container



Sometimes images and other content overflow its container so we should set their max-width to 100% of the parent width.

Max-width overrides width so if we set the css as the following it will be responsive.



# Tap Targets

A finger is 10 mm width which is half an inch, which translates to 40px.

Its ok to have buttons smaller than that but we need at least 40px of space between any two tap targets.

We should ensure that any buttons or anchors have min-height and min-width of 40 px.

# Responsive design concepts

1. Start small: design to smaller screen then see if we need to have design for larger screens.
2. Prioritize content: what is the most important content the user needs to see on the screen.
3. Code from small to large: write code for smaller screen then to larger screen.
4. Think about Performance from smaller to larger screens.

# Media Queries

## Types

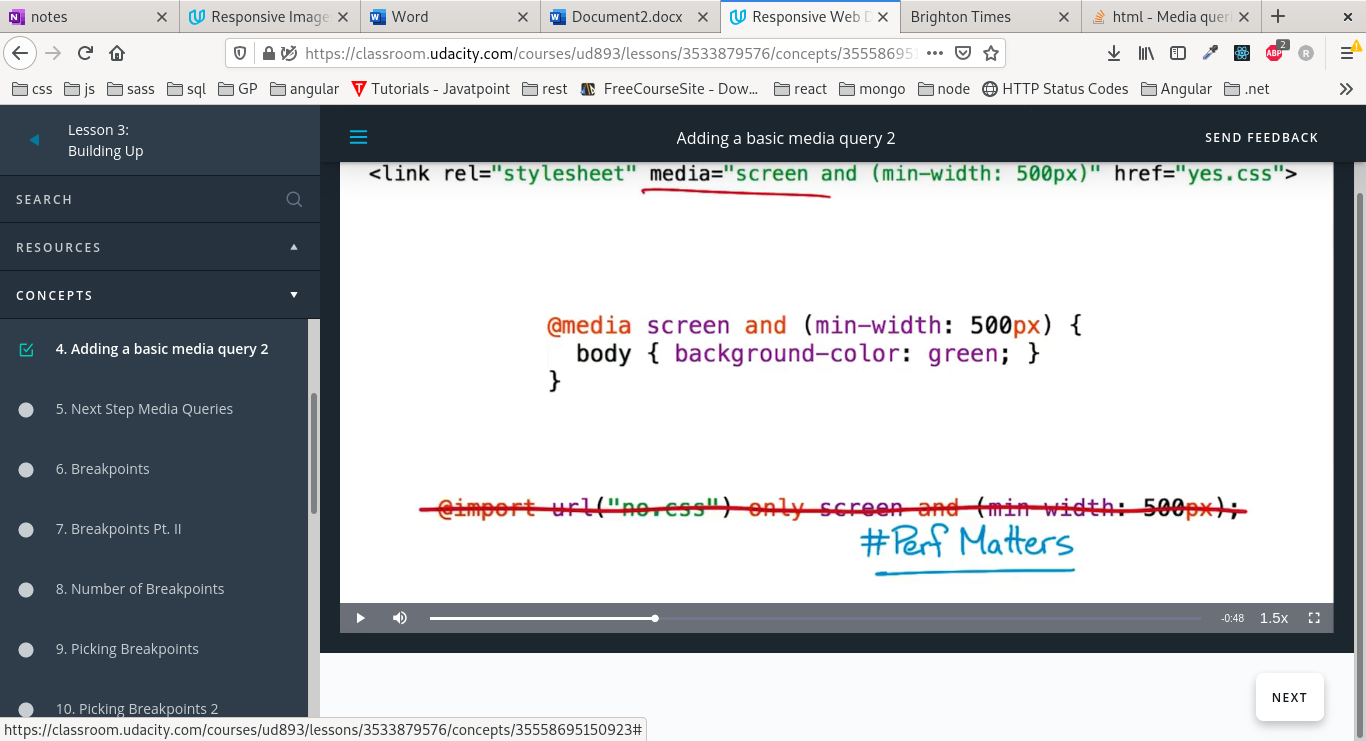
Screen and print are the most important types

## Ways of using media queries

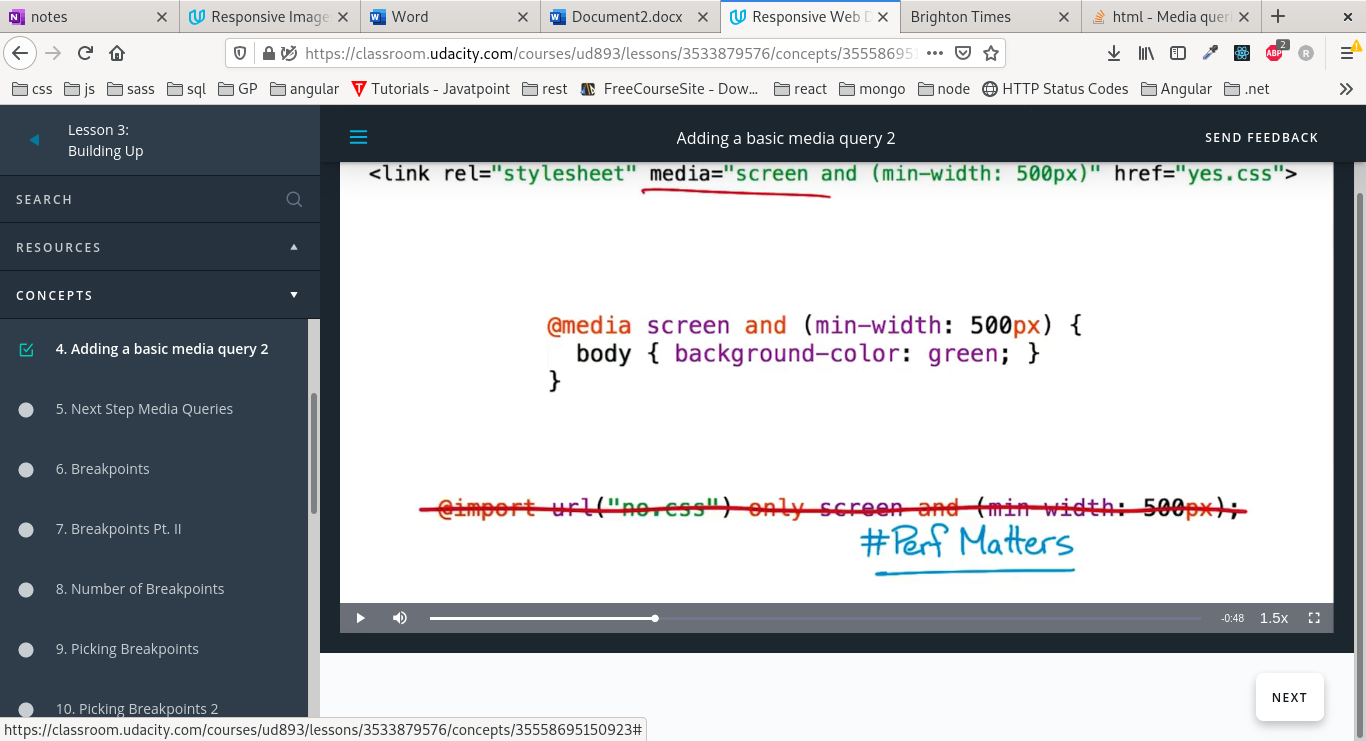
1. Add different style sheets per screen size.

<link rel="stylesheet" type="text/css" href="small-screen.css" media="screen and (max-width:600px)">

1. @media tags:



1. @import css file: expensive to make http requests and not recommended for performance.



## Types of queries:

* Min-width or max or width
* Min-aspect-ratio or max or width
* Min-height or max or width
* Min-resolution or max or width
* Color or min or max
* Min-device-width or max: strongly discouraged because it is based on the HW width of the device and not the browser window.

## Break points

The width of the screen where we want to make changes.

### Minor breakpoints

A break point that makes a small change to the screen but not a large layout change.

### Picking breakpoints

We shouldn’t choose breakpoints based on the size of the screen rather than the content of the page.

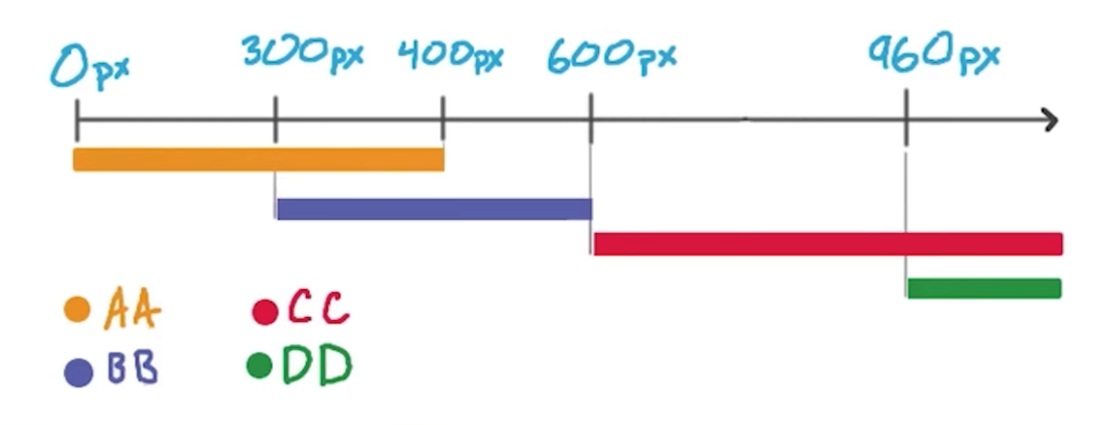
We should see what is the content we want to display and how it should be different if the screen is larger or smaller

Starting small then enlarging the browser window, how the content looks when the screen gets bigger should tell me it needs a break point.

The content could have too much space when the screen gets wider.

## Order of queries

The order of media queries like css determines how overlapping styles will be handled



In case of overlap between AA and BB, BB will be applied because its order comes after AA in css



# Building Layouts

## Grids

Fluid grids columns wrap when the width of the browser window decreases

Example is bootstrap

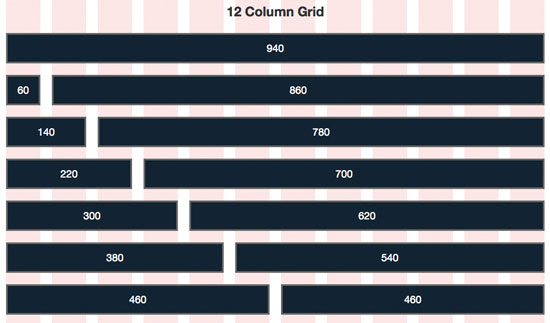
### 960 grid layout system

It provides cross-browser-tested and optimized preset column widths for you to set your content into.

The 960 Grid System is simply a way to lay out websites using a grid that is 960 pixels wide.

The reason it’s 960 pixels wide is because the number 960 makes for a lot of clean divisions utilizing whole numbers when factoring in column widths and margins. And it fits nicely on the majority of screens.

The 960 GS comes in two primary variants: a 12-column grid and a 16-column grid



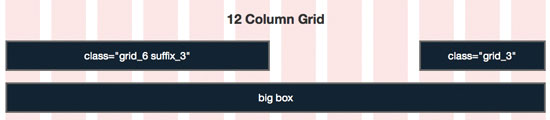
Pushing and pulling:



We can push or pull columns which moves them a column distance to right or left

Prefix and suffix:

To create spaces between columns



There are many implementations of the grid which adds some utility classes to make it better

<https://960.gs/>

<https://www.webfx.com/blog/web-design/the-960-grid-system-made-easy/>

## Flexbox

Flex box ignores the width property set in children and sets their width based on the browser width to be on the same line

# Responsive Tables

## Problem

Table width is wider than the browser window



## Solution 1: Hidden columns

Hides columns based on their importance as the screen gets smaller

We can also use abbreviated data instead of hiding columns, in case of smaller screen

Give certain columns display:none at a certain viewport width

## Solution 2: No more tables (transpose)

We invert the table to show only the rows as table header and columns as rows

<https://codepen.io/JohnMav/pen/BoGJNy>

<https://css-tricks.com/responsive-data-table-roundup/>

## Solution 3: contained table(scrollable)

We set a fixed width on the container of the table to make it scrollable horizontally

# Fonts

The ideal measure of number of characters per line is 40-90 cpl, 65 is the best,

We can set the max-width in characters

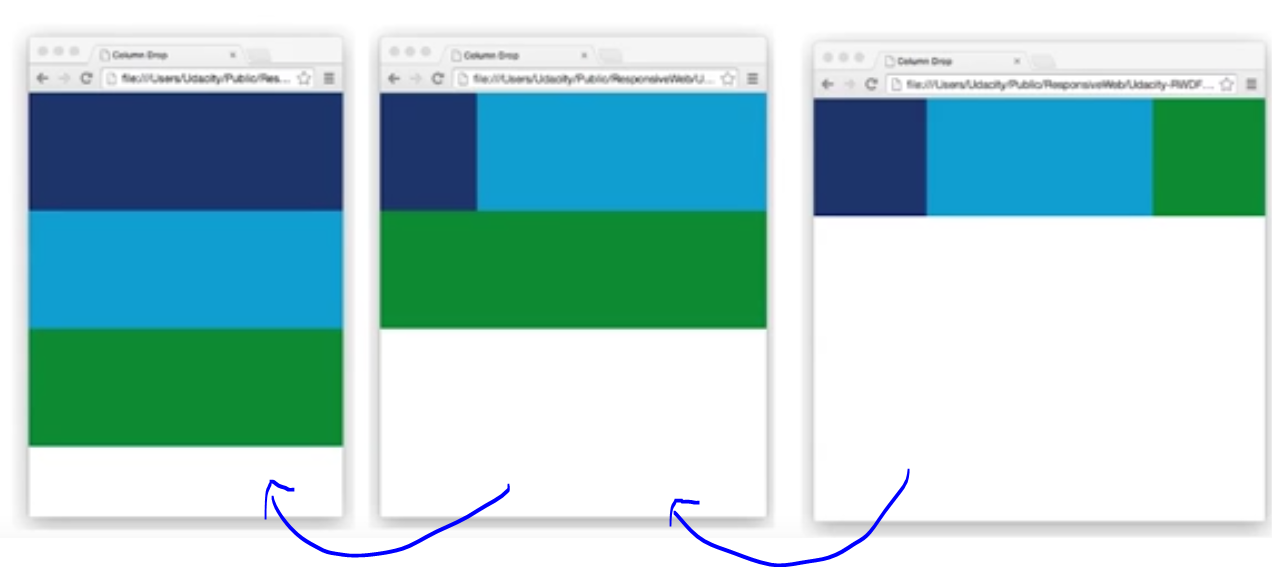
We need fonts to be big enough to read on different devices at least 16px

# Responsive Patterns

## Column drop

As the viewport gets smaller, the columns are stacked on top of each other.

The column drop layout stacks one column at a time as the viewport is reduced.



Every time the view port is reduced by a break point it wraps one column and makes its width 100%

## Mostly fluid

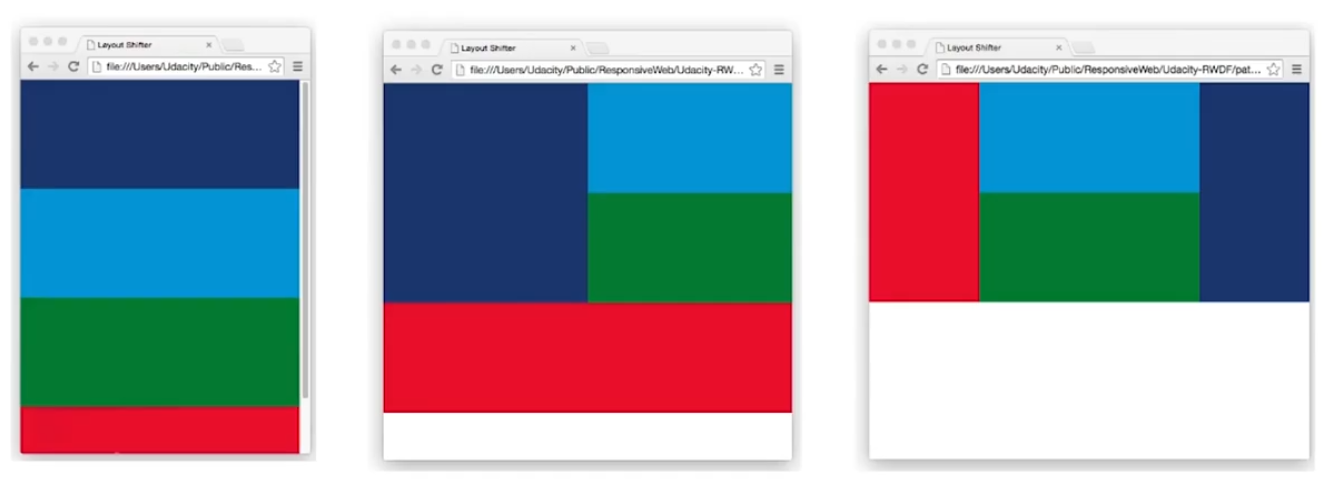
Similar to column drop but it wraps like a grid. It maintains grid structure at all break points.



1. First it removes margins
2. It wraps columns but tries to maintain some columns next to each other as a grid

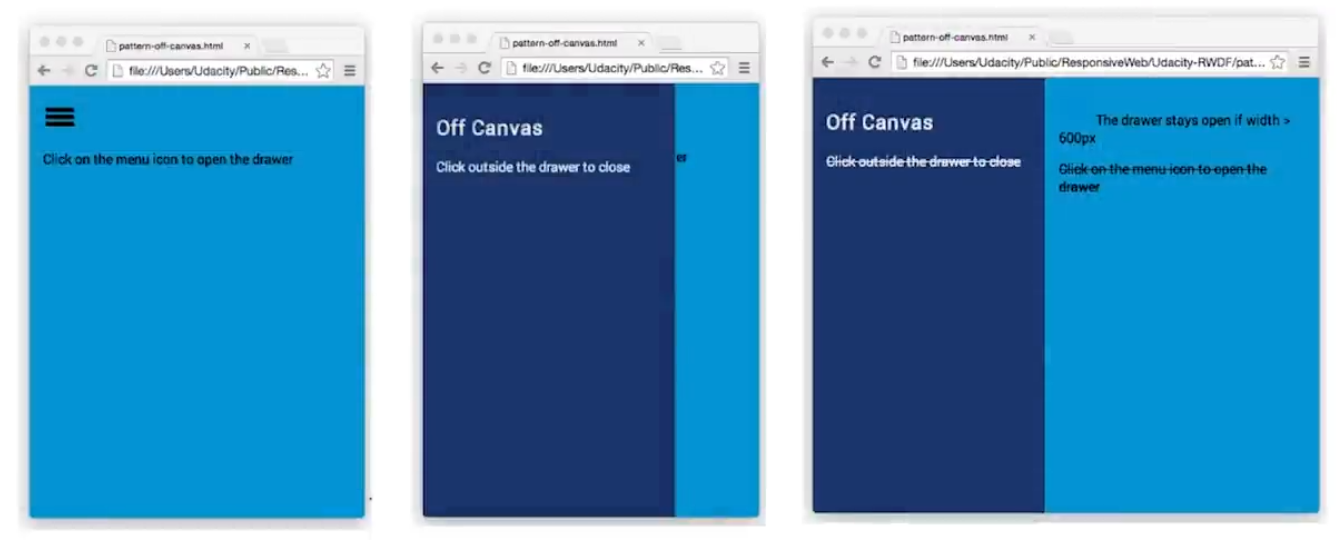
## Layout shifter

The layout changes at different breakpoints, and the order of columns change



## Off canvas

Content that is hidden in smaller screen and comes from off-canvas, but is shown by default in larger screen. Ex: side menu



## More patterns

<https://responsivedesign.is/patterns/>

Responsive Images

# Image size

## Decreasing the image size while maintaining how it looks

1. Decreasing the size in pixels
2. Maintaining the size but using compression

## Image size

Equals: the number of pixels \* number of bits per pixel

The ideal case is to keep images as small as possible(number of pixels) and compressed as possible(number of bits per pixel)

How to reduce size while maintain quality?

## Max-width:100%

Allows images to grow or shrink according to viewport or window size but it will stop growing when it reaches it max width

For example the image width is 420px and the viewport is 200, then the image width will be 200px

If the viewport is 500, then the image width will be 420px

# Image optimization

We can use tools like page speed insights to see how images in our website are optimized

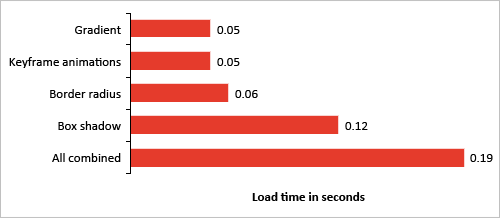
<https://developers.google.com/speed/pagespeed/insights/>

# improving performance

The best performance is to reduce image size by compression and optimize images as much as possible

1. Use text instead of text images: will be better for scaling
2. Use css styling:
   1. Animations
   2. Shadows
   3. Gradient

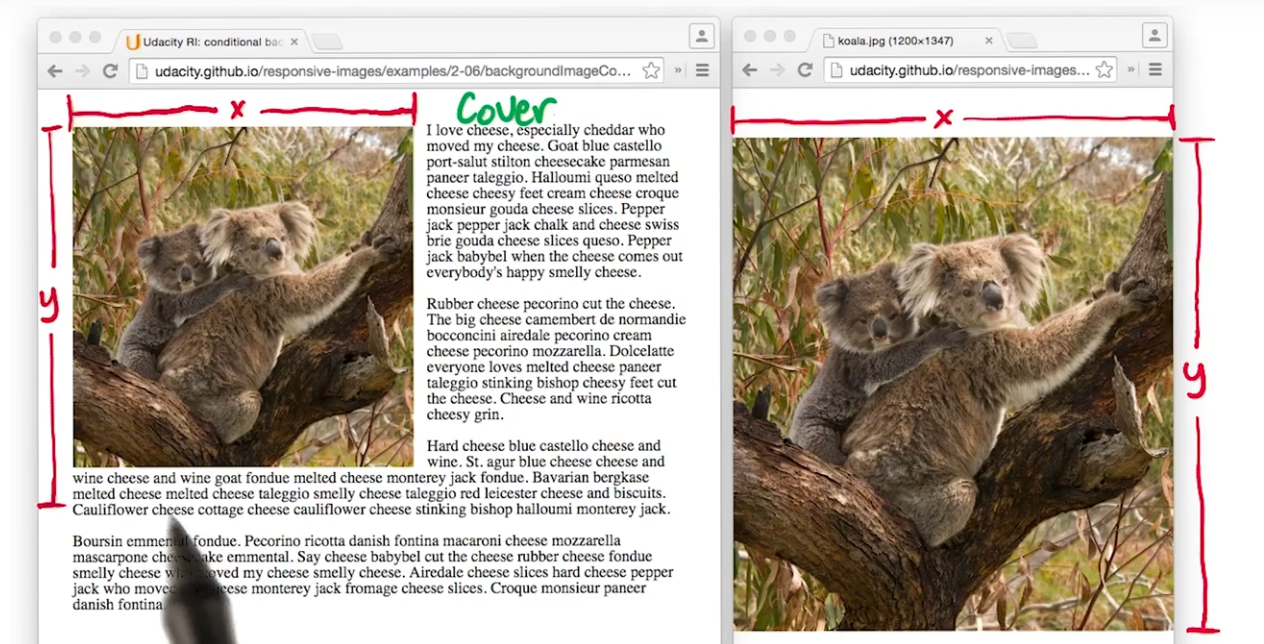
The styling also has effect on performance



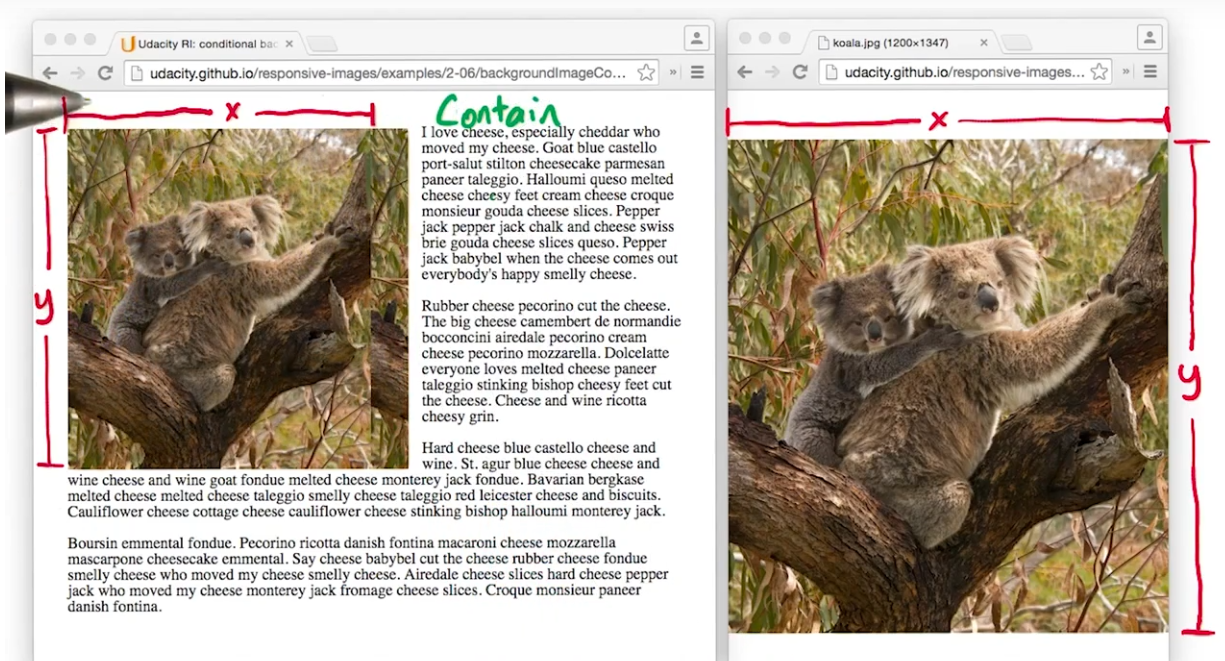
<https://www.smashingmagazine.com/2013/04/build-fast-loading-mobile-website/>

1. Use background image:
   1. Cover and contain
      1. Cover: the image’s smallest dimension must be contained in the container

If the larger dimension of the images overflows the container, it is cut off



* + 1. Contain: the image’s largest dimension is contained in the container, and if there is remaining space it is repeated the image



* 1. Gradients
  2. Conditional display of images using media queries
  3. Image-set

<https://developer.mozilla.org/en-US/docs/Web/CSS/image/image-set()>

* 1. Use symbols instead of svg images:
     1. They are scalable as fonts
     2. Need to add <meta http-equiv=”Content-type” content=”text/html;charset=utf-8”
     3. A list of all symbols: <https://unicode-table.com/en/>
     4. Use icon fonts:

<https://fonts.google.com/icons>

<https://smcllns.github.io/css-social-buttons/>

<https://fontawesome.com/>

<https://css-tricks.com/examples/IconFont/>

* 1. Use inline svg and inline image: Instead of adding a source file, embed the image as base-64 string using data uri

This will reduce the number of requests we have to make

However if we use the image on multiple pages, we should make it a source file so that the browser can cache it

* 1. Svg optimization

<http://petercollingridge.appspot.com/svg-optimiser>

* 1. Examples of svg manipulation

<https://www.creativebloq.com/design/examples-svg-7112785>

<https://codepen.io/chrisgannon/pen/MwMpBQ>