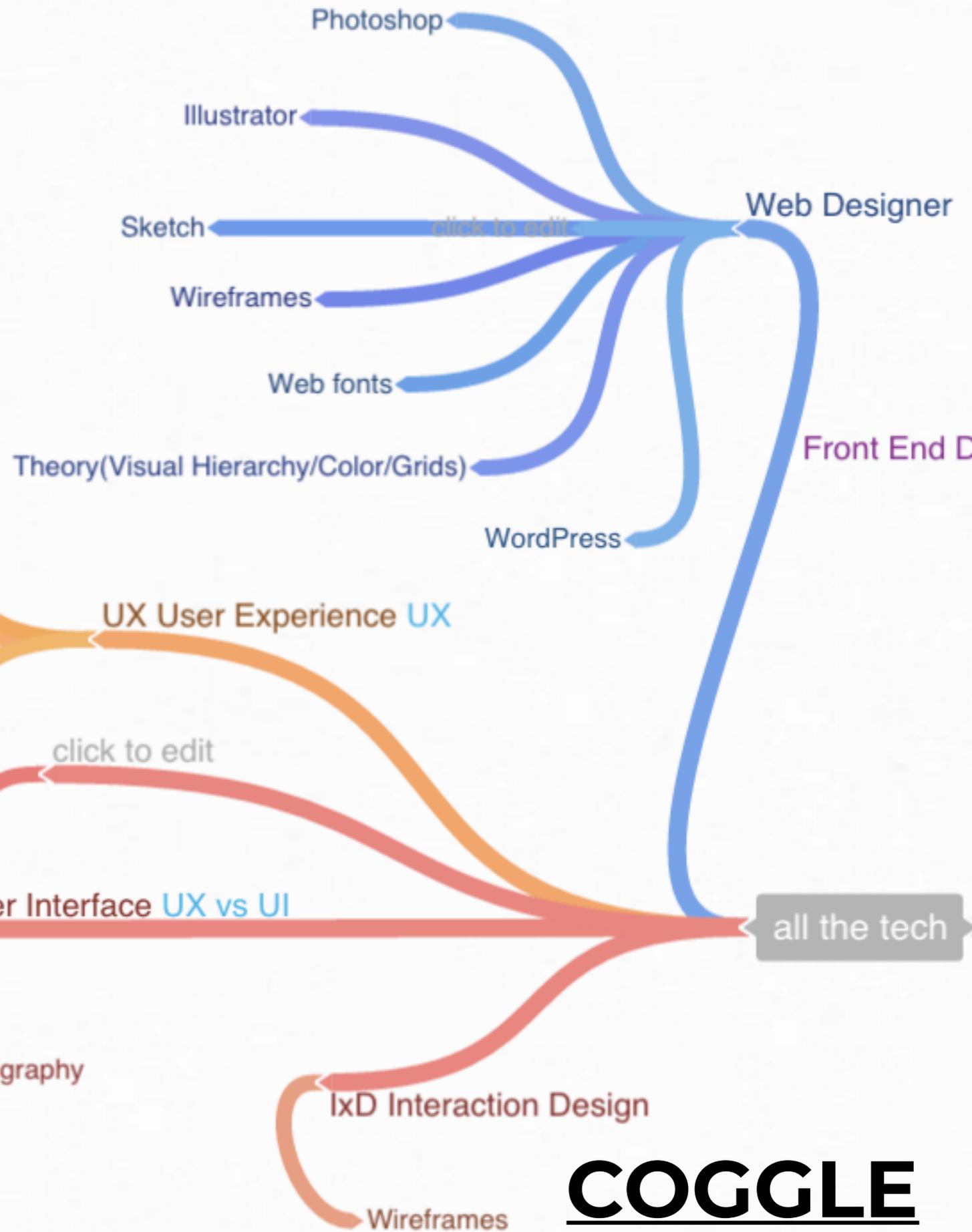


CAREER PATHS
HERE IS A WEB DESIGNER
MAKING \$6,500 A SITE WITH
AS MUCH CODING
SKILLS AS YOU ALL HAVE
LEARNED
ELLE AND COMPANY



COGGLE

Responsive Web Design

Instructor: Angela Wyman

Responsive Web Design



The history of design for the web began with print designers wanting to perfectly lay out a web page with all the elements in pixel perfect positions. But, the web is different than paper.

The traditional solution to this problem was to create a separate mobile site, which often meant a site designed specifically for the iPhone. BUT...

Responsive Web Design

An approach for building websites that work across **multiple screen resolutions; mobile devices, tablets, and desktop screens.**

The practice consists of a mix of fluid grids and layouts, flexible images in combination with CSS media queries.

Why bother with responsive?

We want our websites to be useable on all devices by responding to the user's behavior, screen size and screen orientation.

A collection of various electronic devices, including laptops, tablets, and smartphones, arranged on a wooden surface. The devices are mostly silver and black, with some blue and white ones. The text is overlaid on the image.

WHAT IS A DESIGNER TO DO...

**MAKE A SITE FOR
EVERY DEVICE?**

A LIST APART

2010 ETHAN MARCOTTE

This changed the way that many people think about designing web sites for multiple screen resolutions.

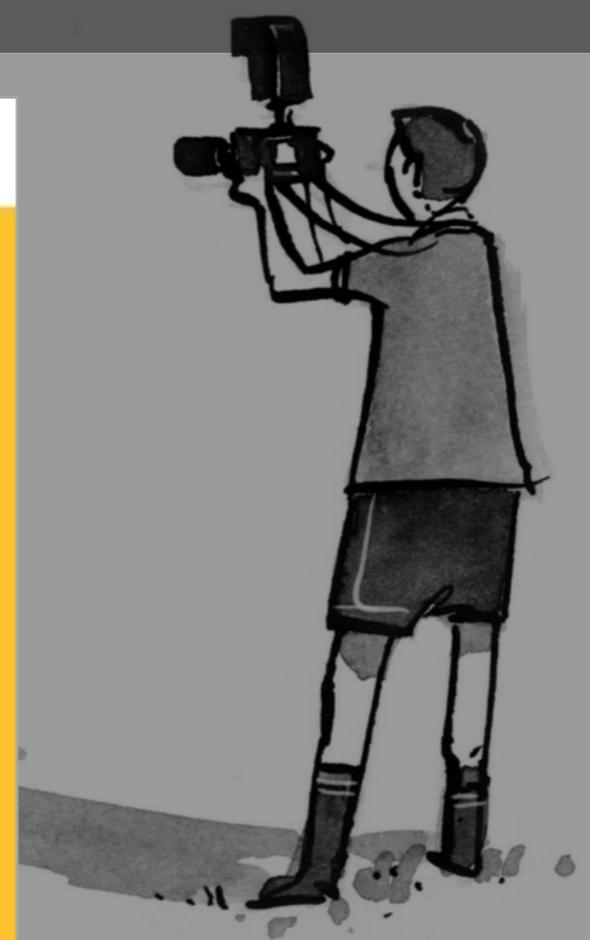
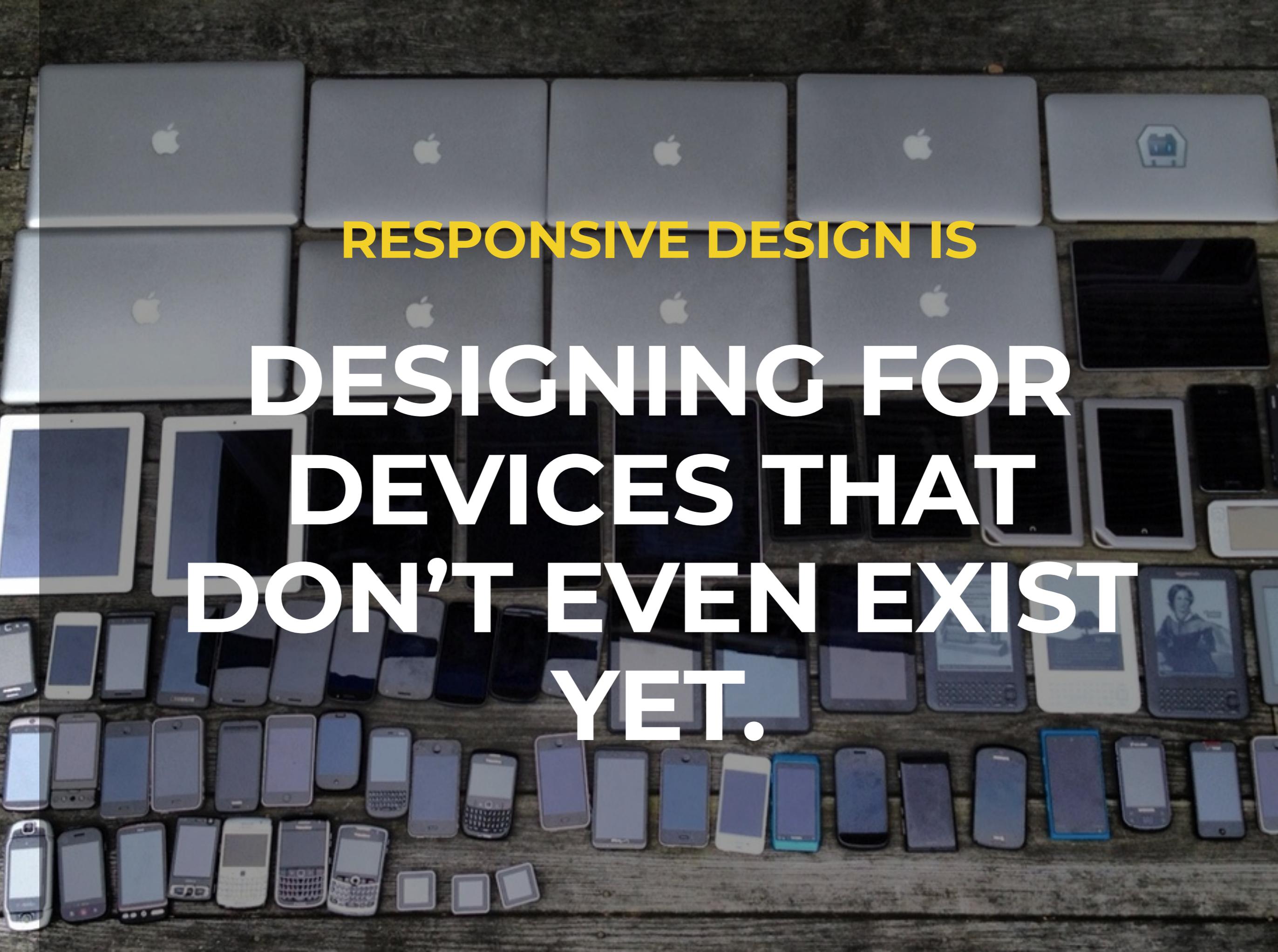


Illustration by Kevin Cornell

Responsive Web Design

by **Ethan Marcotte** · May 25, 2010

Published in [CSS](#), [Layout & Grids](#), [Mobile/Multidevice](#), [Responsive Design](#), [Interaction Design](#)

A collection of various electronic devices including laptops, tablets, and smartphones arranged on a wooden surface. The devices are mostly in shades of grey and blue, with some black and white. The text is overlaid on the center of the image.

RESPONSIVE DESIGN IS

**DESIGNING FOR
DEVICES THAT
DON'T EVEN EXIST
YET.**

This Is Responsive.

Patterns and resources for creating responsive web experiences.

<http://bradfrost.github.io/this-is-responsive/index.html>

Blackberry:
Curve 8530, Pearl Flip

Android:
Motorola Charm, Xperia X10

Symbian:
Nokia E63, others
320 x 240

Blackberry:
Torch,
Storm,
Bold
480 x 360

Apple:
iPhone, iPod

Android:
HTC Dream, Hero, others
320 x 480

AS IS THIS

Symbian:
Nokia N8, Nokia C6-01
360 x 640

Android:
Liquid A1, HTC Desire, Nexus One, i9000 Galaxy S

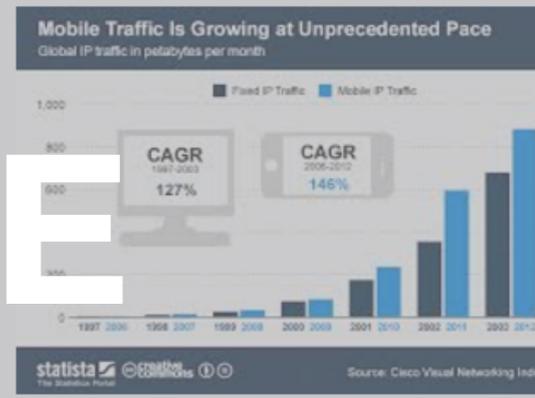
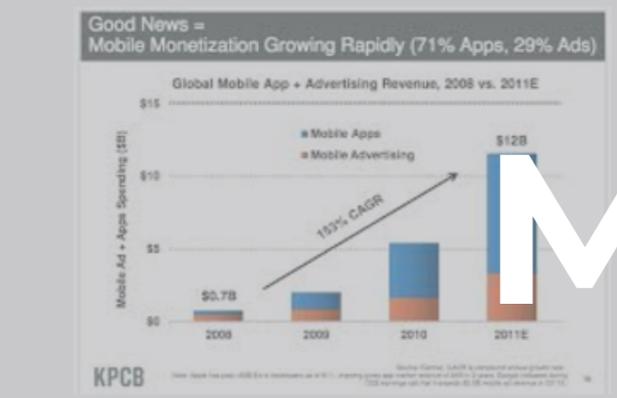
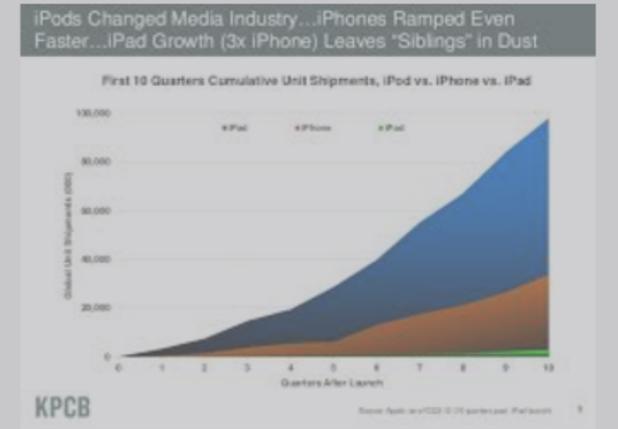
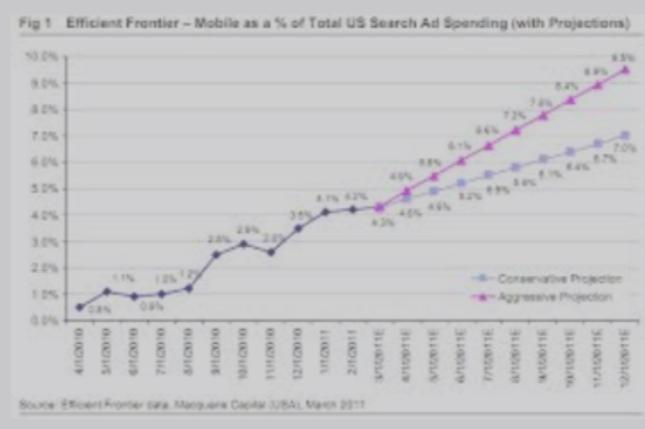
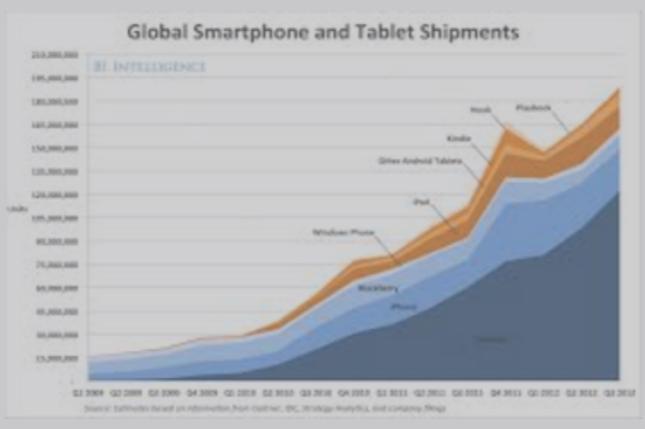
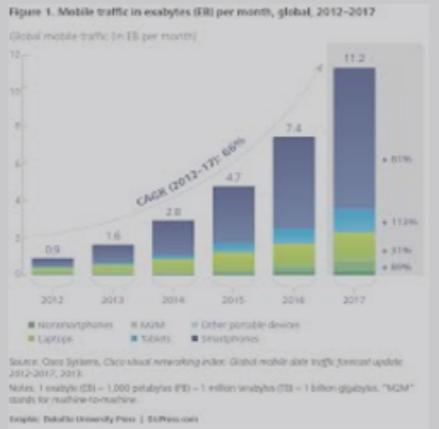
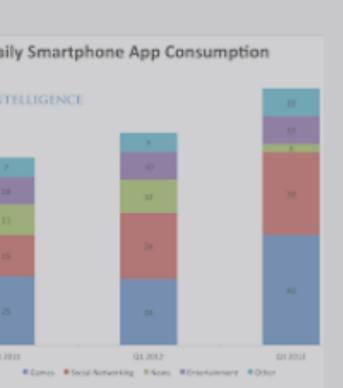
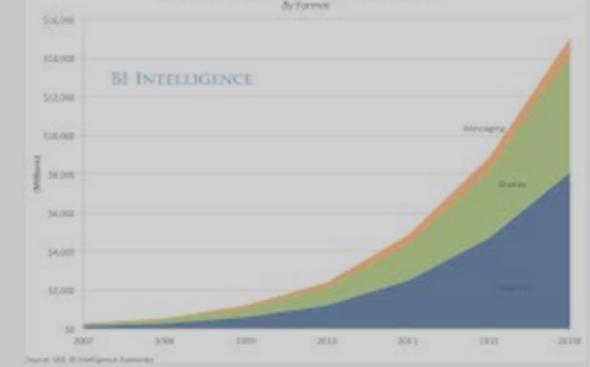
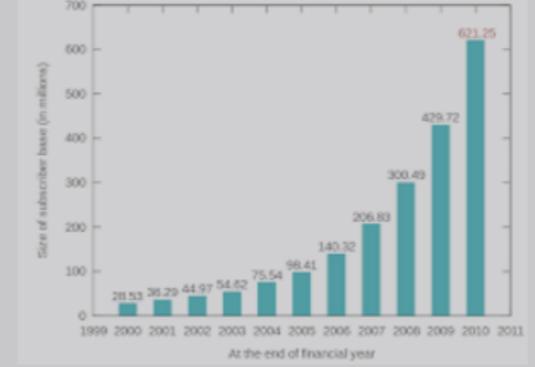
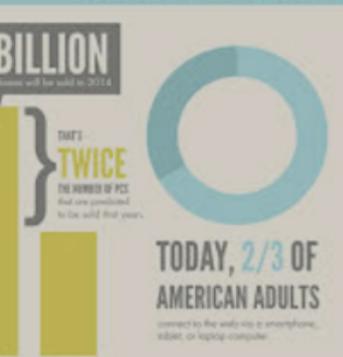
Windows:
Sharp S01SH, Venue Pro, Omnia 7, HTC 7 Pro
480 x 800

Apple:
iPhone 4

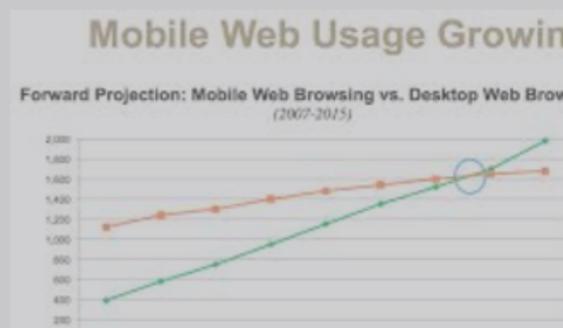
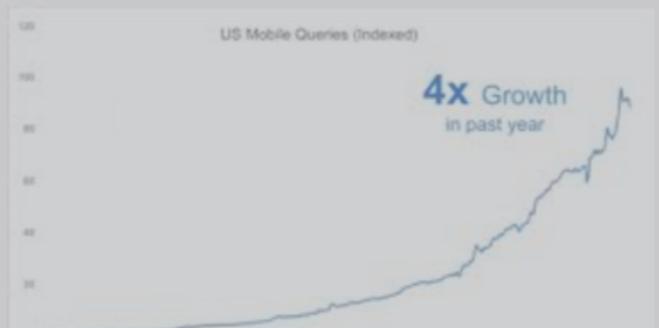
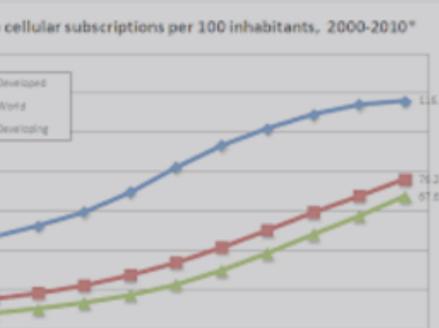
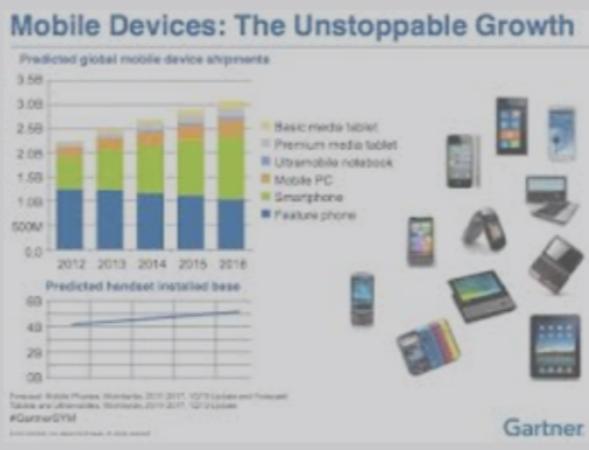
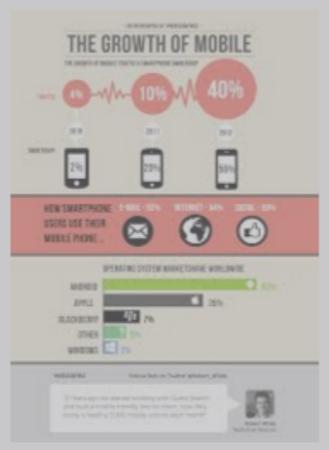
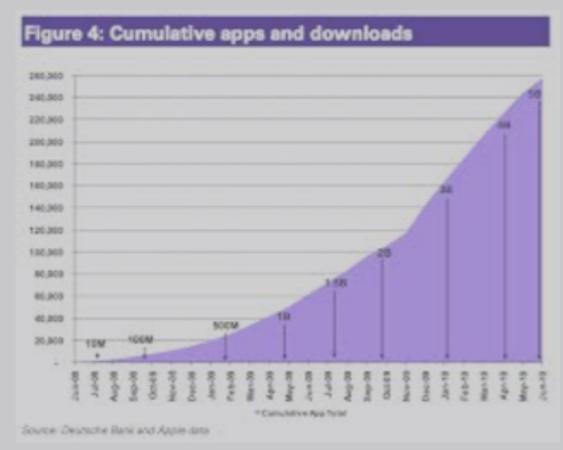
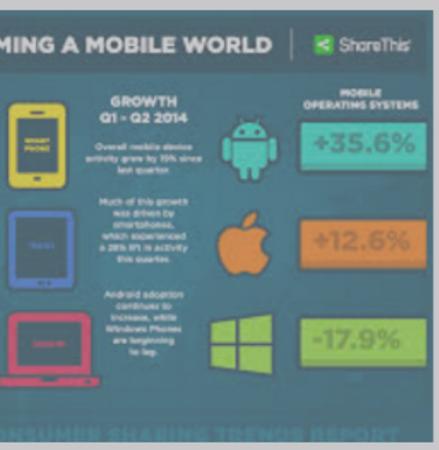
640 x 960
(320x480 with higher dpi)

Mobile First

Means start with mobile sizes and layer on styles optimized for larger screens only as needed. In other words, your mobile styles are the default and you won't have to override them later. — It's much simpler!

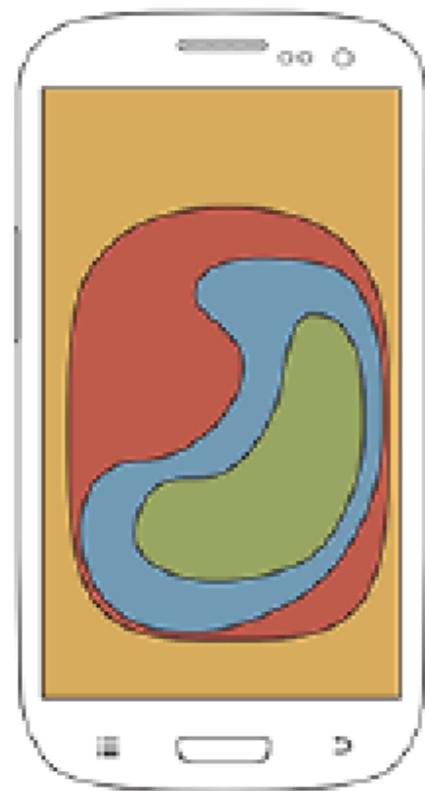


MOBILE IS HUGE



consider how people interact with mobile

multi-usage

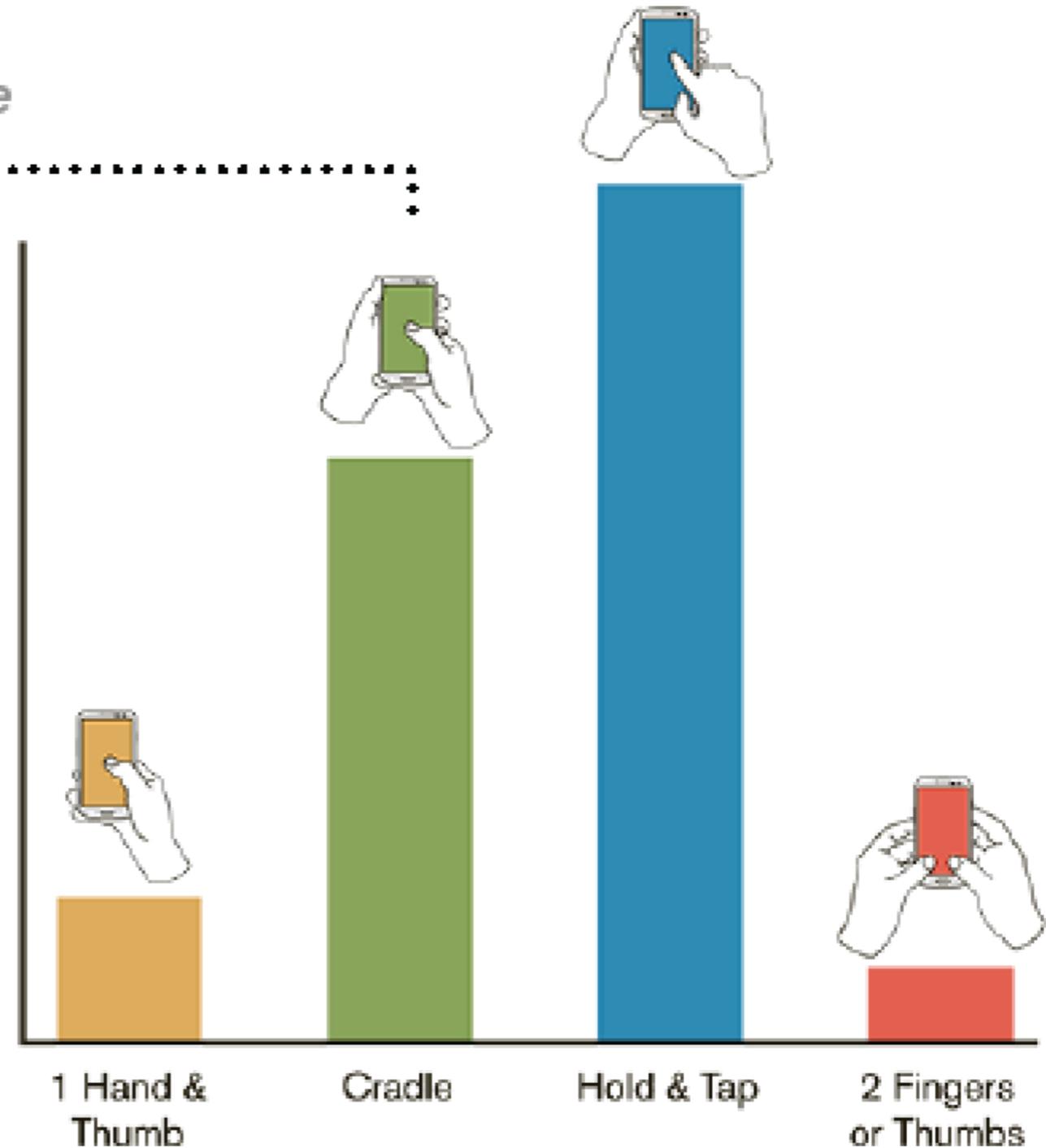


Hold & Tap,
Second-Order Cradle

Primary Tap Zone

First-Order Cradle

1 Hand & Thumb



WHAT SHOULD WE CONSIDER

- Buttons and touch targets are at least 44px by 44px
- Text has been removed from images
- The font size of body copy is at least 14px
- Image carousels are swipable
- Place content that is available on hover on the screen
- Elements which toggle open do not cover other elements
- Design both portrait and landscape orientations
- Forms use custom input types and are as short as possible
- Common touch actions fall within the easiest to reach areas
- Autocorrect and auto-capitalize are disabled on form fields
- Navigation is optimized for common mobile actions
- Calls to action are labelled with descriptive information
- Non-essential menu items have been moved to the footer
- The website loads in under three seconds on a 3G connection
- The context of each page is extremely clear

WHAT SHOULD WE CONSIDER

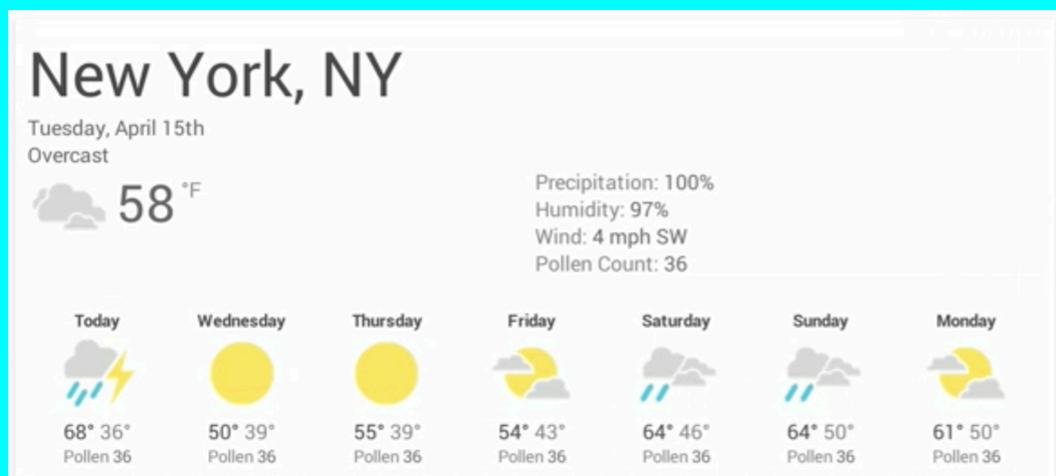
- ▶ **Average fingertip is between 1cm to 2cm wide, which roughly correlates to somewhere between 44px and 57px on a standard screen and 88px to 114px on a retina screen**

- ▶ **Mobile sites only have most crucial features**
- ▶ **Vertical navigation is common on more than 90% of mobile sites**
- ▶ **Hyperlinks are replaced by tabs and buttons**
- ▶ **Promotional images are often removed**
- ▶ **No breadcrumbs, progress indicators, persistent navigation or mega footers**
- ▶ **Frequent one-tap phone integration**
- ▶ **Localized and personalized search based on geo-location**

Responsive Web Design

Responsive web design, originally defined by [Ethan Marcotte in A List Apart](#), responds to the needs of the users and the devices they're using. The layout changes based on the size and capabilities of the device.

For example, on a phone users would see content shown in a single column view; a tablet might show the same content in two columns.

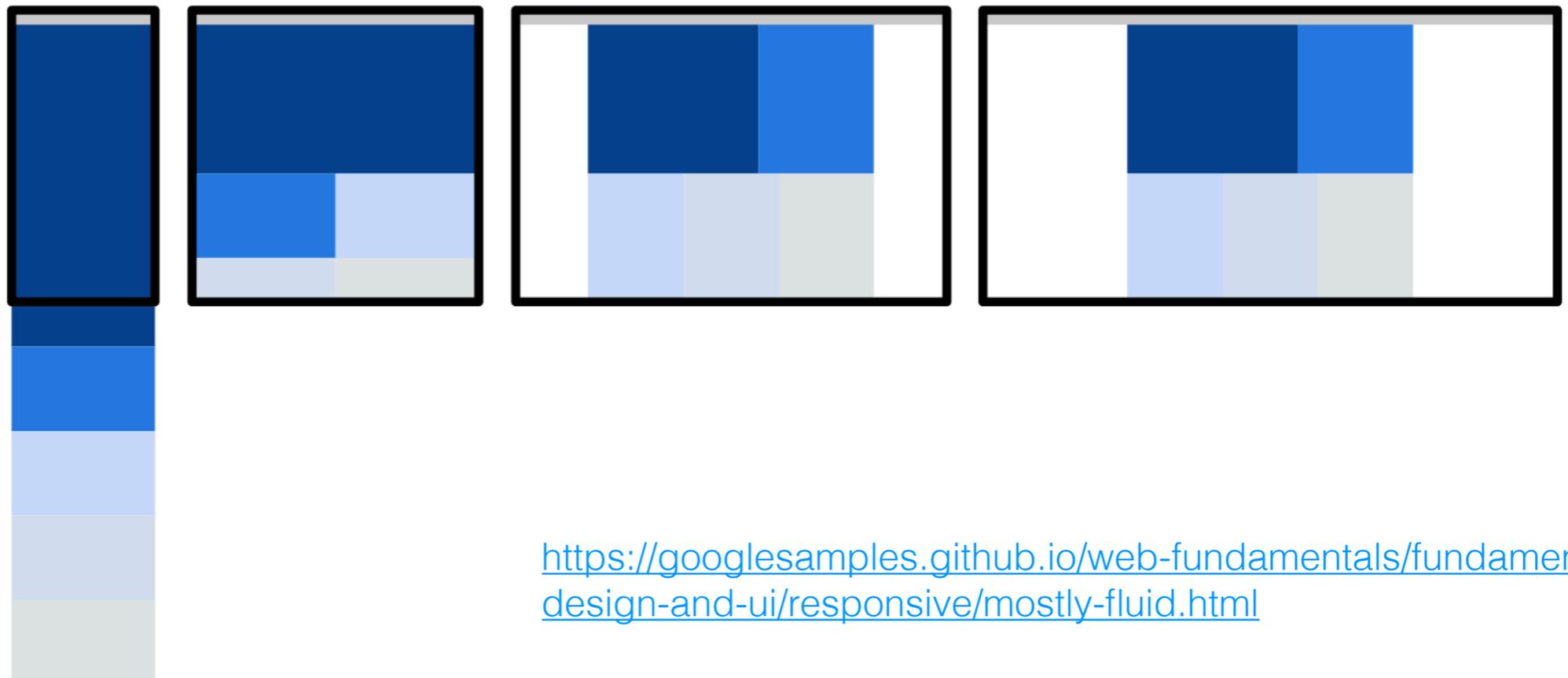


Mostly Fluid

Simply adjusts the margins on wider screens. Beyond 800px, the container div becomes fixed width and is centered on the screen.

On smaller screens, the fluid grid causes the main content to reflow, while columns are stacked vertically. One major advantage of this pattern is that it usually only requires one breakpoint between small screens and large screens.

Responsive Design Patterns



<https://googlesamples.github.io/web-fundamentals/fundamentals/design-and-ui/responsive/mostly-fluid.html>

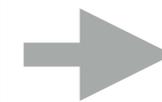
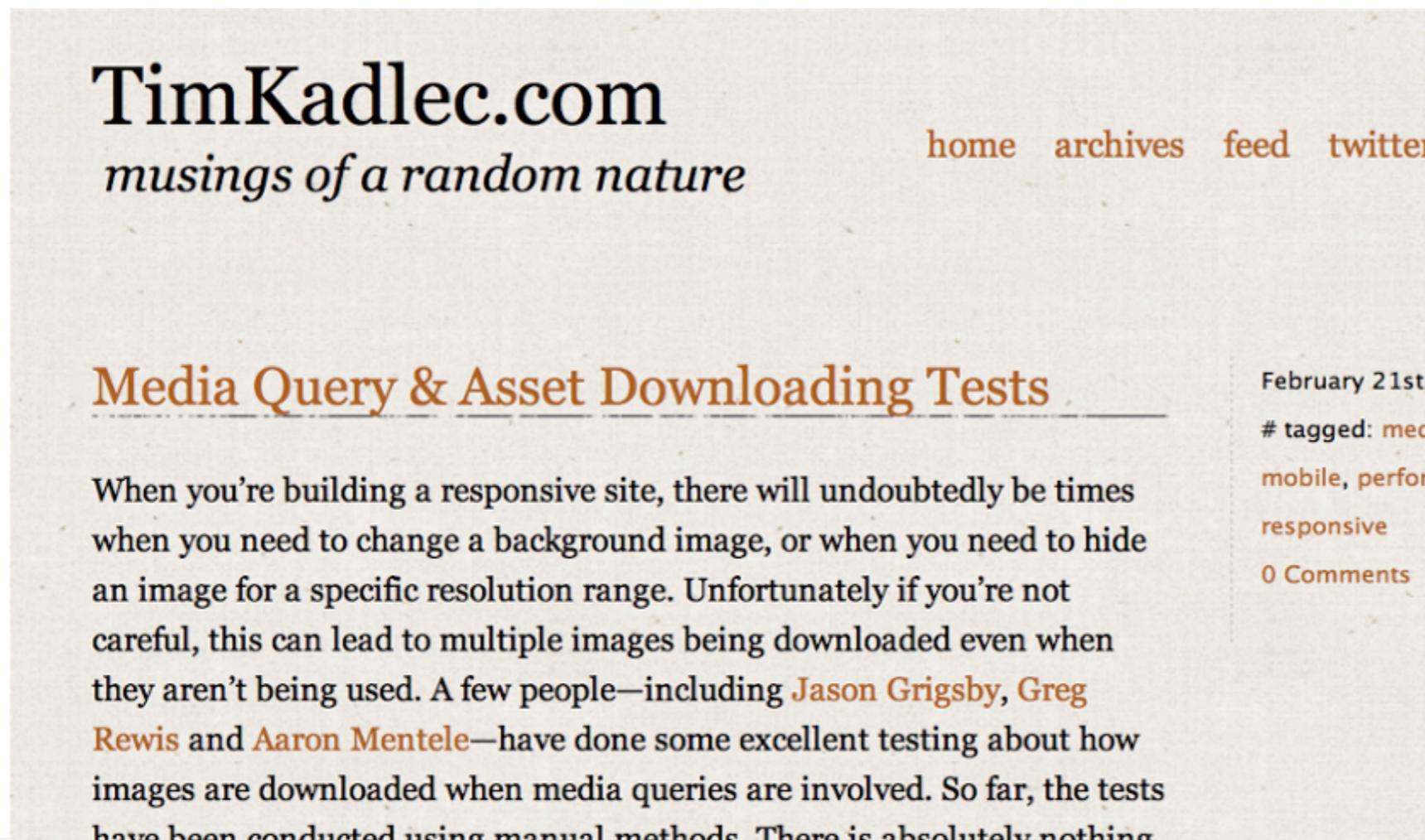
<http://simplebits.com/>

Top Nav

One of the easiest-to-implement solutions for navigation is to keep it at the top.

Problems: content-first, nav-second is preferred for mobile web experiences. Get the navigation out of user's way so they can focus on the core information on the page. What if you add more menu items? Links may be too close together for clicks.

Responsive Design Patterns

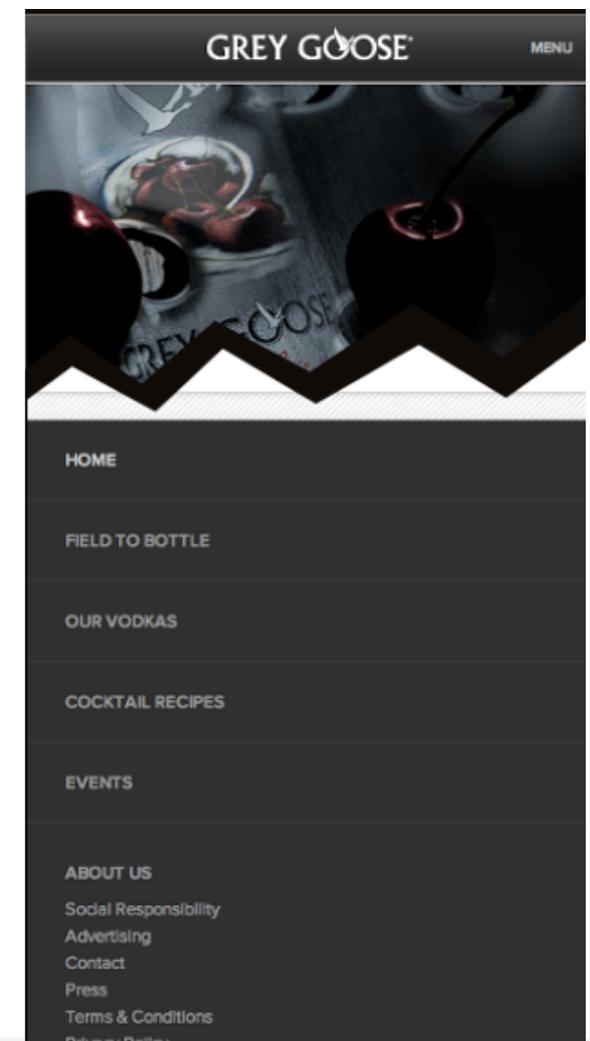
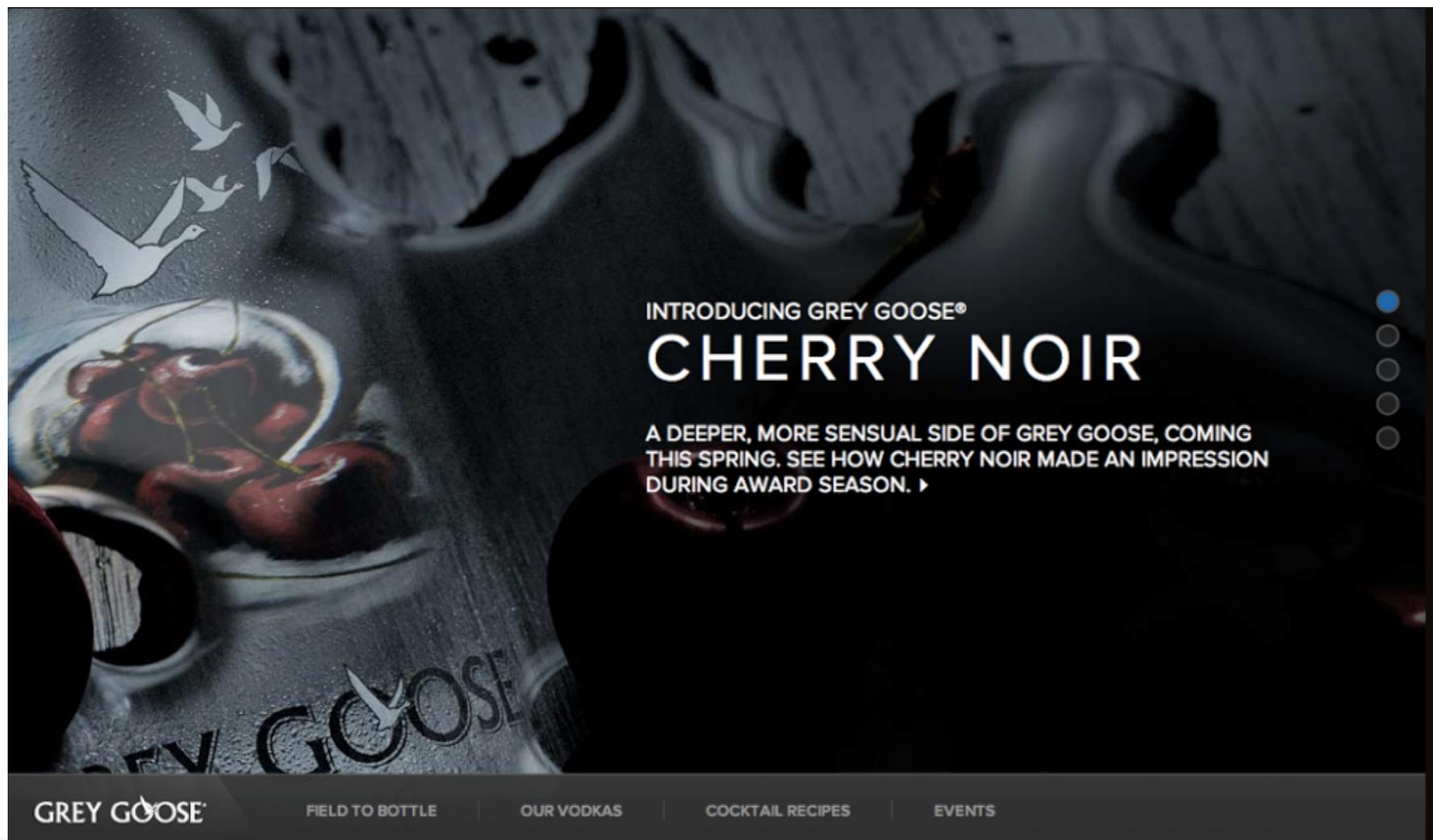


The Footer

Keep the nav list at the footer of the site, while the header contains a simple anchor link pointing to the footer nav. This approach clears up a lot of room for the core content while still providing quick access to the navigation.

Problems: jumping to the footer feels awkward.

Responsive Design Patterns

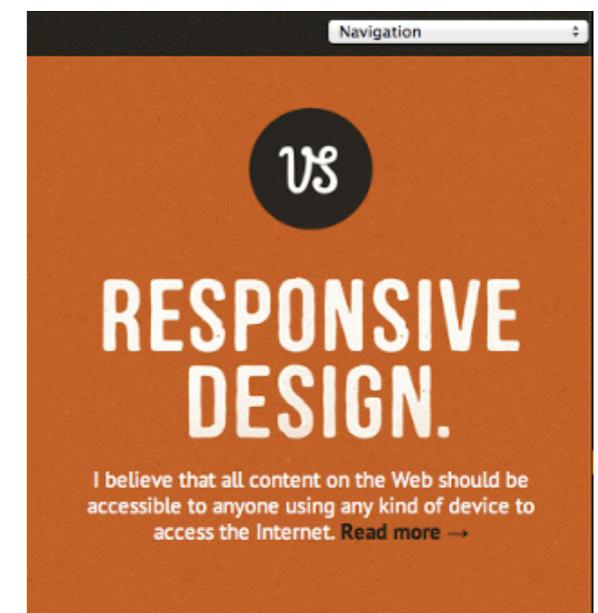
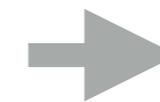


The Select Menu

You can transform a list of links into a **select menu** for small screens. This avoids the problems the top nav approach presents and is a clever way to save real estate.

Problem: nested lists handled by select menus can look weird. Child categories are usually handled by indenting with dashes

Responsive Design Patterns

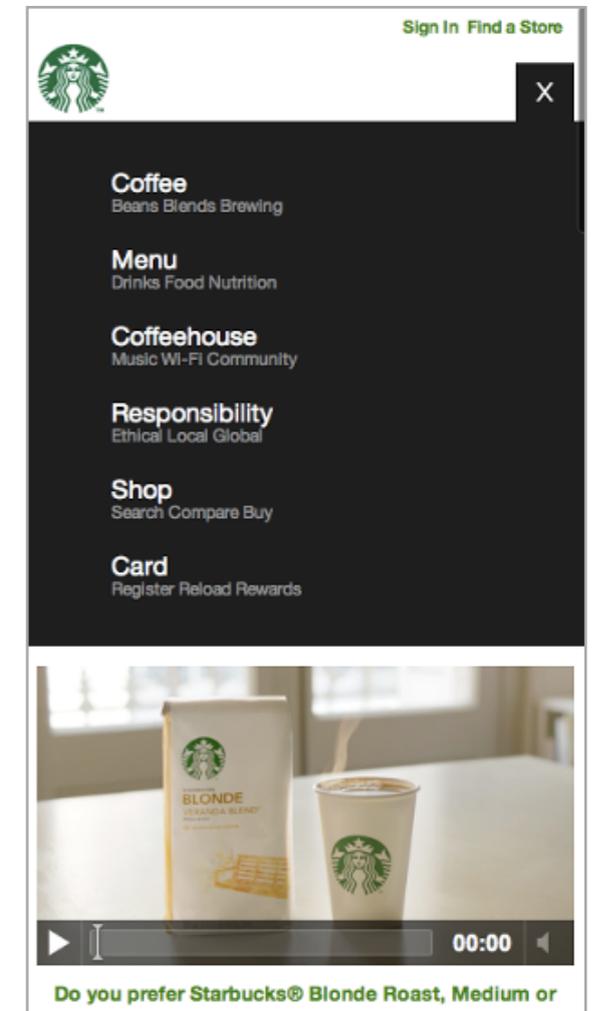
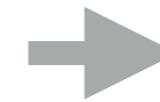
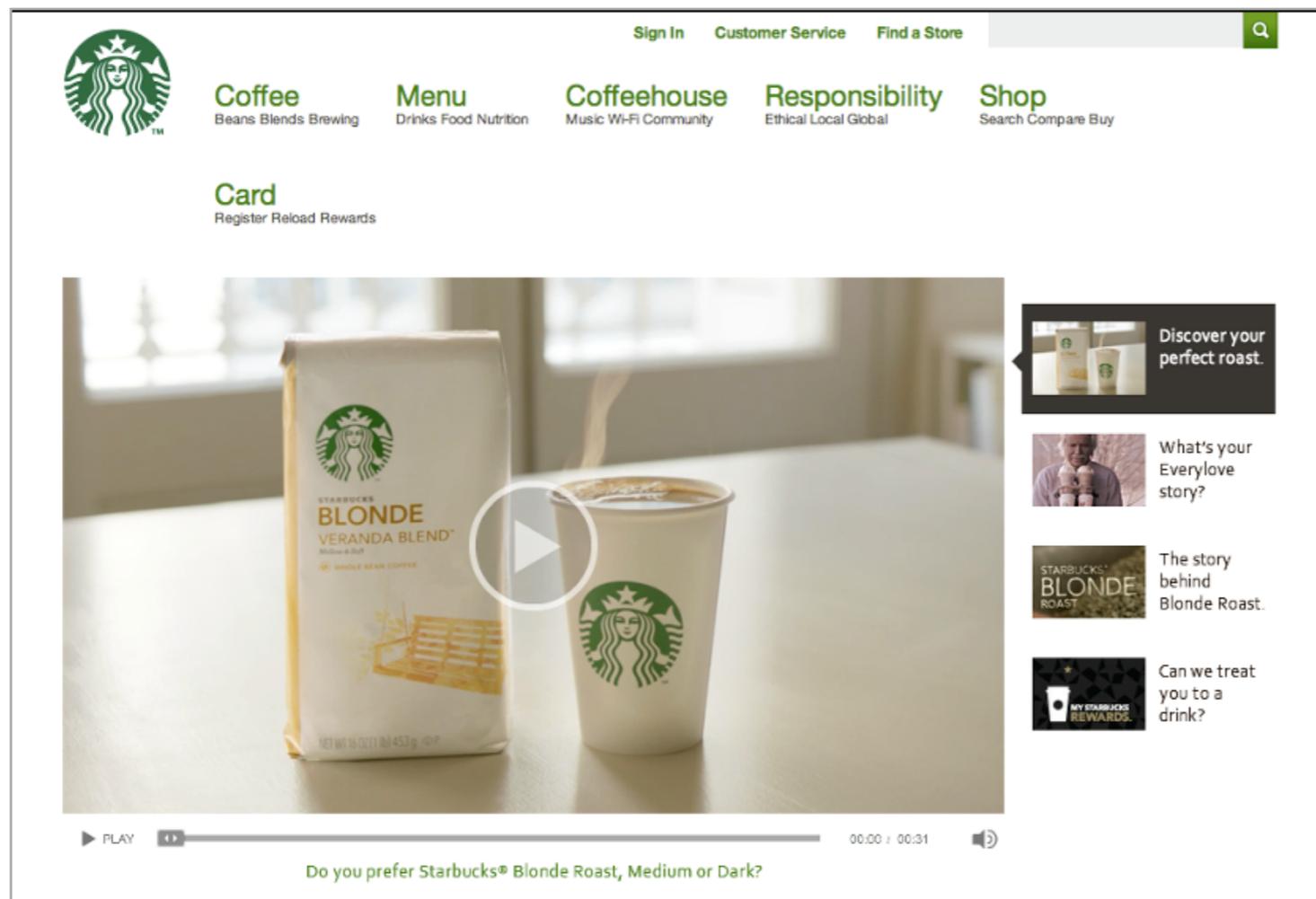


The Toggle

Similar to the footer anchor approach, but instead of jumping down to an anchor at the bottom of the page, the menu slides open right in the header. It's a good-looking approach and is relatively easy to implement.

Problem: this is dependent upon Javascript to trigger the toggle. Animations can also be janky.

Responsive Design Patterns

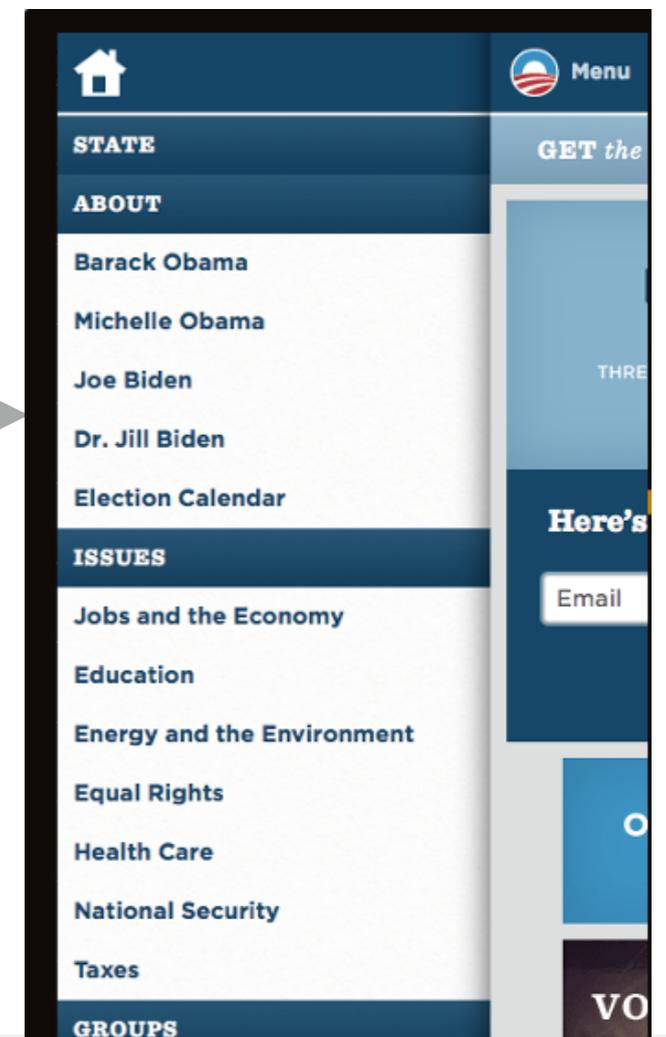
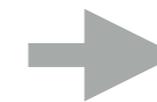


The Left Nav Flyout

The nav is accessed by a menu icon, which reveals a tray that slides in from the left and moves the main content over to the right.

Problem: this method is quite unique to mobile and doesn't necessarily scale up to large screens easily. Can be confusing.

Responsive Design Patterns

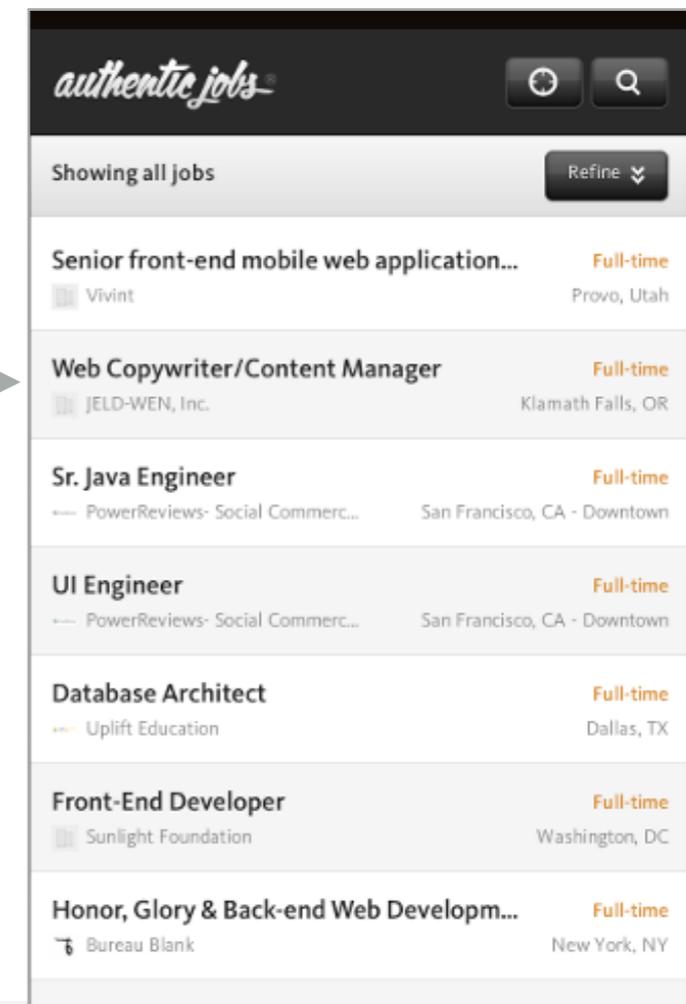
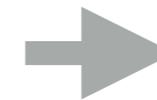
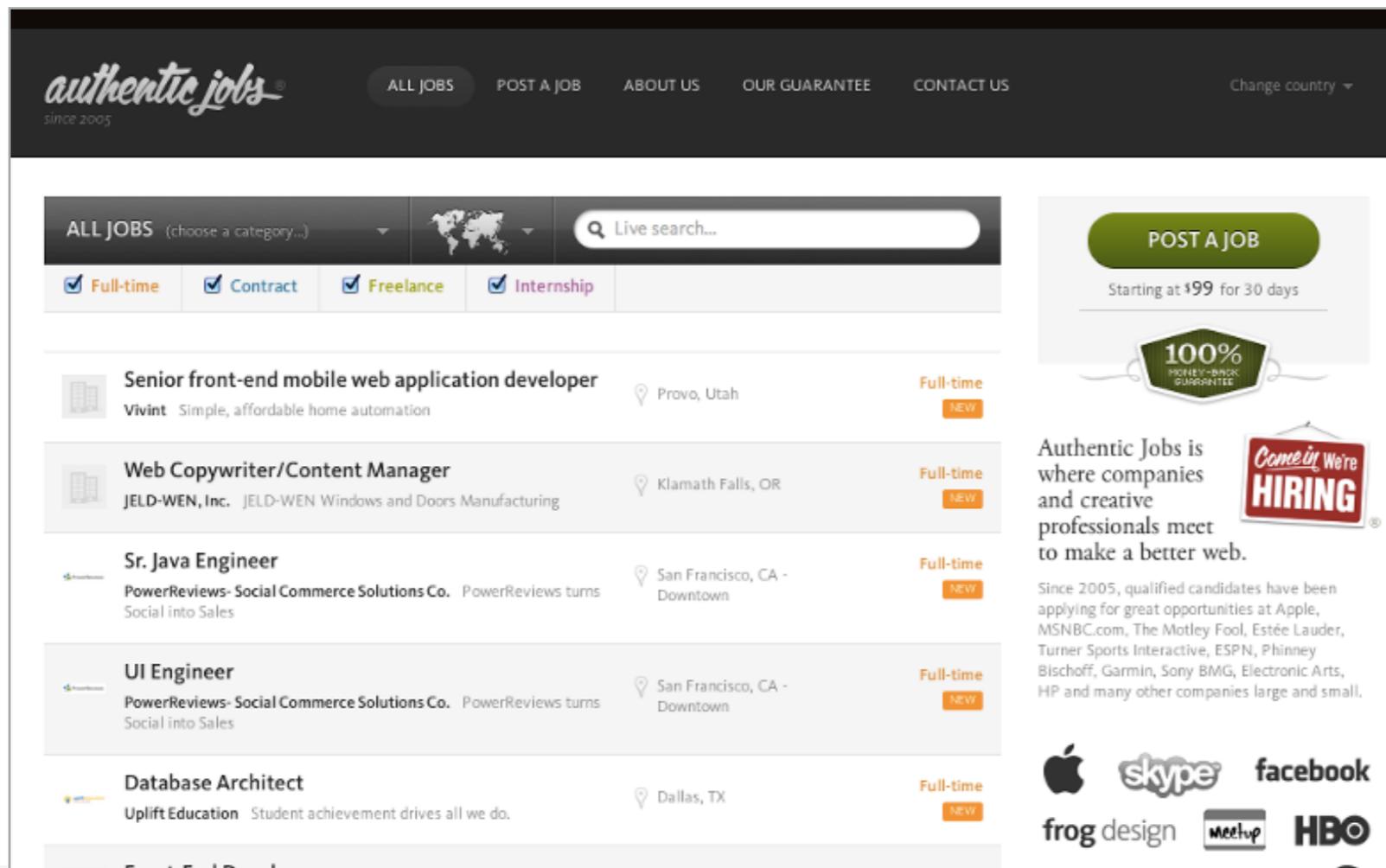


The Hide 'n Cry

Don't penalize users for visiting your site on smaller devices

Problem: **Removes content/functionality for mobile users**– Hiding links and content is not OK. If a responsive site is hiding content for mobile users it's no better. **Adds extra page weight**– Adding `display:none` for elements that are presumably unneeded for still gets download.

Responsive Design Patterns

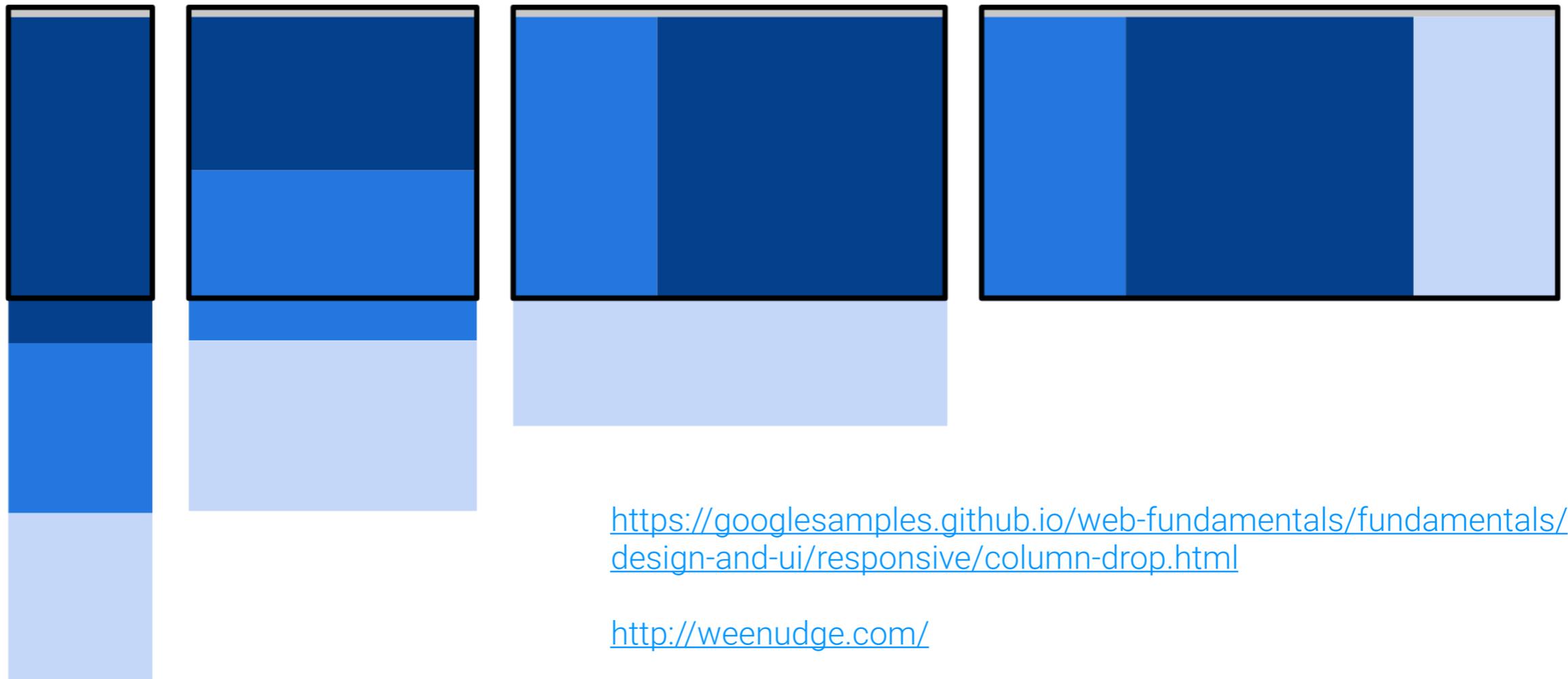


Column Drop

Column drop simply stacks the columns vertically as the window width becomes too narrow for the content.

Eventually this results in all of the columns being stacked vertically. Choosing breakpoints for this layout pattern is dependent on the content and changes for each design.

Responsive Design Patterns



<https://googlesamples.github.io/web-fundamentals/fundamentals/design-and-ui/responsive/column-drop.html>

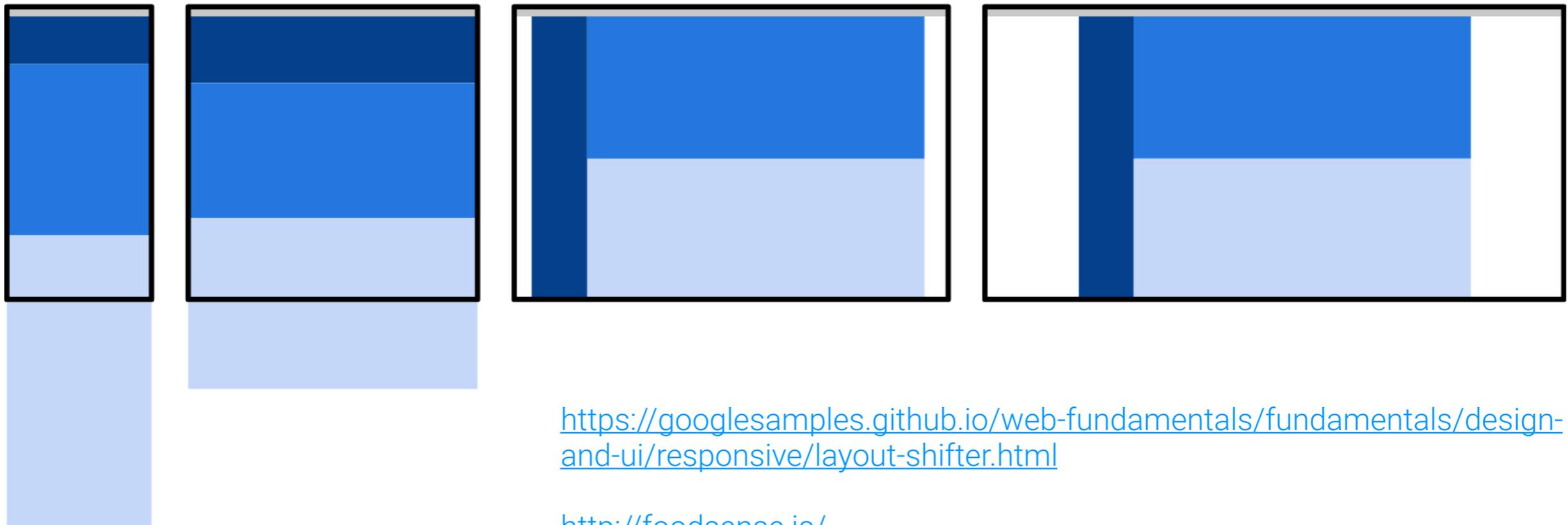
<http://weenudge.com/>

Layout Shifter

This is the most responsive pattern, with multiple breakpoints across several screen widths.

Key to this layout is the way content moves about, instead of reflowing and dropping below other columns. Due to the significant differences between each major breakpoint, it is more complex to maintain and likely involves changes within elements, not just overall content layout.

Responsive Design Patterns



<https://googlesamples.github.io/web-fundamentals/fundamentals/design-and-ui/responsive/layout-shifter.html>

<http://foodsense.is/>

<https://alistapart.com/d/responsive-web-design/ex/ex-site-FINAL.html>

Tiny Tweaks

Tiny tweaks simply makes small changes to the layout, such as adjusting font size, resizing images, or moving content around in very minor ways.

It works well on single column layouts such as one page linear websites and text-heavy articles.

Responsive Design Patterns



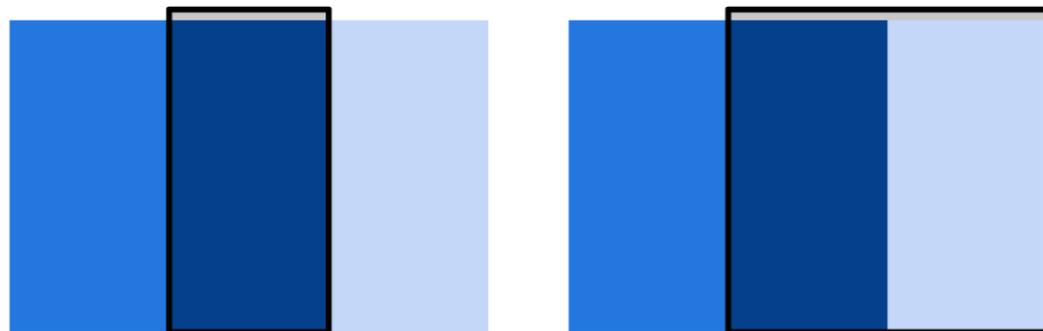
<https://googlesamples.github.io/web-fundamentals/fundamentals/design-and-ui/responsive/tiny-tweaks.html>

<http://futurefriendlyweb.com/>

Off Canvas

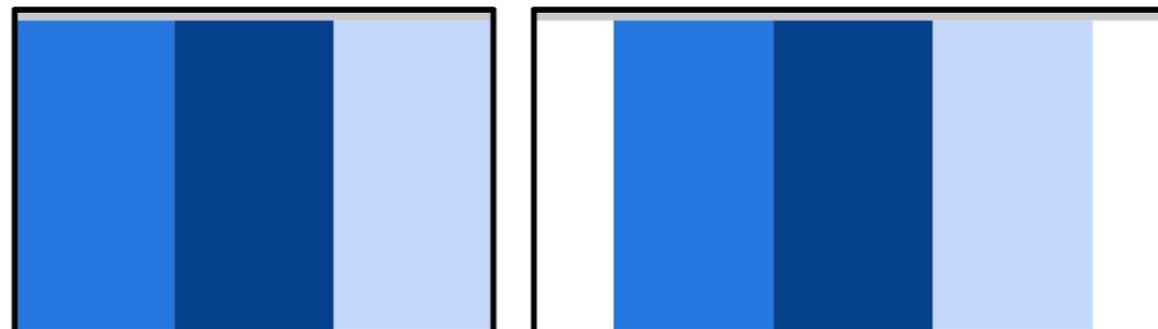
Rather than stacking content vertically, the off canvas pattern places less frequently used content—perhaps navigation or app menus—off screen, only showing it when the screen size is large enough, and on smaller screens, content is only a click away.

Responsive Design Patterns



<https://googlesamples.github.io/web-fundamentals/fundamentals/design-and-ui/responsive/off-canvas.html>

<https://www.google.com/nexus/>



1.

definition: affordance

J. J. Gibson, a visual psychologist coined the word “affordance” in the late 1970s. Gibson envisioned affordances as *the relationships between an environment and an actor.*



1.

definition: *perceived* affordance



Norman argued for **perceived affordances**, essentially translating a term that may have never left the realm of psychology. Perceived affordances describe the relationships that actors *perceive* within an environment. This is an important distinction. Perceived affordances would soon become a tool that designers of (everyday) products would use to *invite* users to use them.

1.

emotional connection to mobile

- portable and lightweight
- constant access



63% of people felt lost if their smartphone was not in easy reach. They described their mobile devices as “‘alive’ ... an extension of their own body and personality ”

1.

physicality

- bandwidth and connectivity (reduced data)
- smaller dimensions
- touch screens and gestures
- viewing conditions -in different lights like outdoors, in theatres, etc
- distractions - used while doing something else or in unquiet environments



Interaction Design



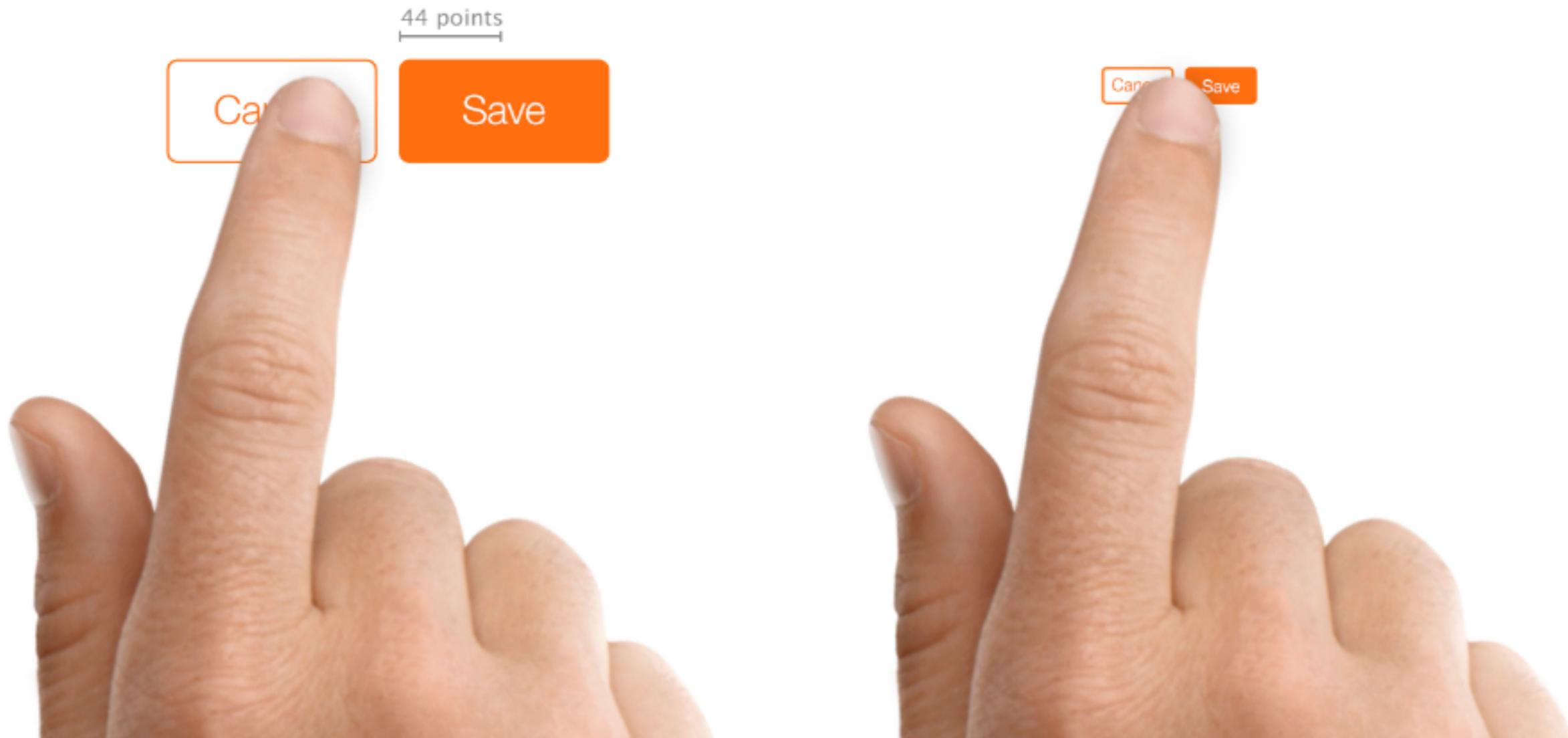
Tip 1

we look at the impact these differences have when developing the structure and functionality

2.

ergonomics

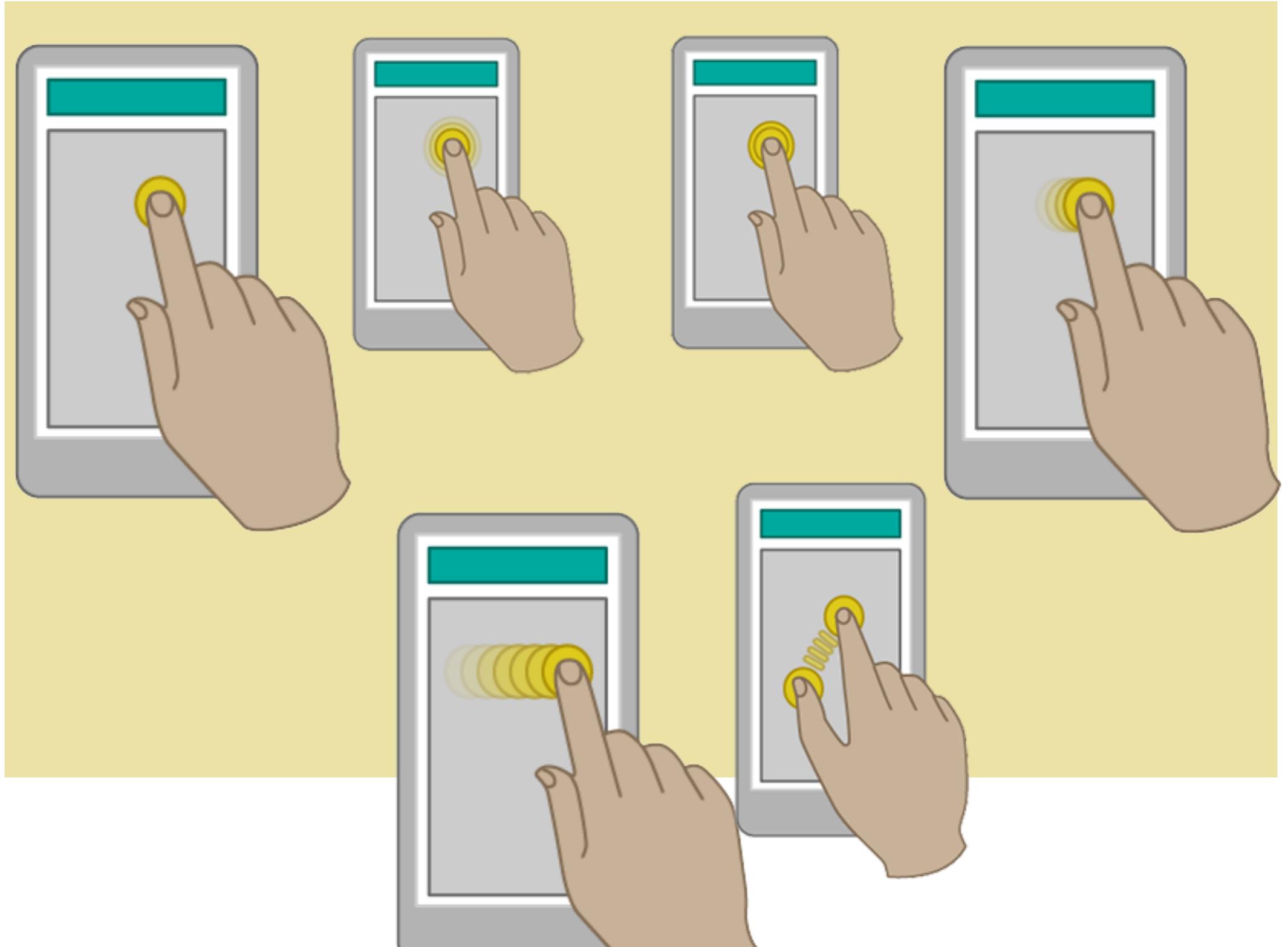
Tap Targets are places (links, buttons, forms, ads, etc.) that a user interacts to do something



For mobile-friendly designs, make sure you use at least 48px

2.

gestures



2.

touch gesture guide

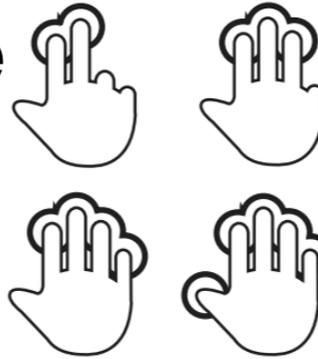
Tap



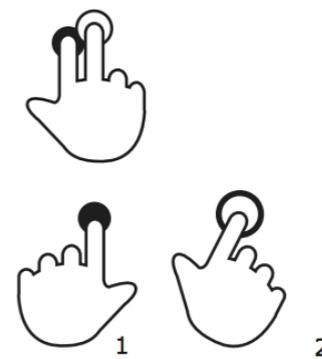
Press



Multi-finger tap



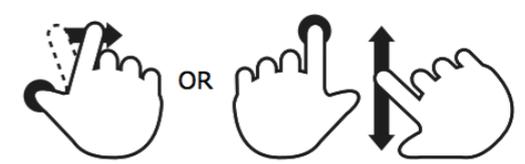
Press and tap



Double tap



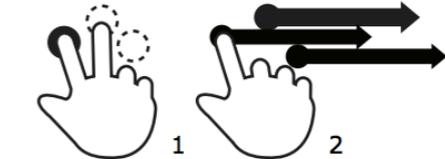
Press and drag



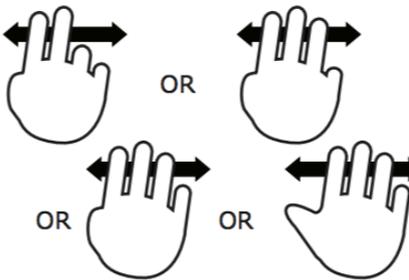
Drag



Press and tap, then drag



Multi-finger drag



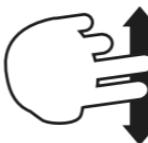
Pinch



Squeeze



Two-finger drag



Spread



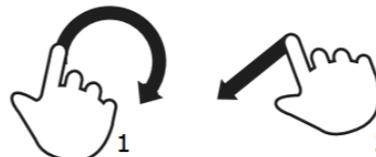
Flick



Splay

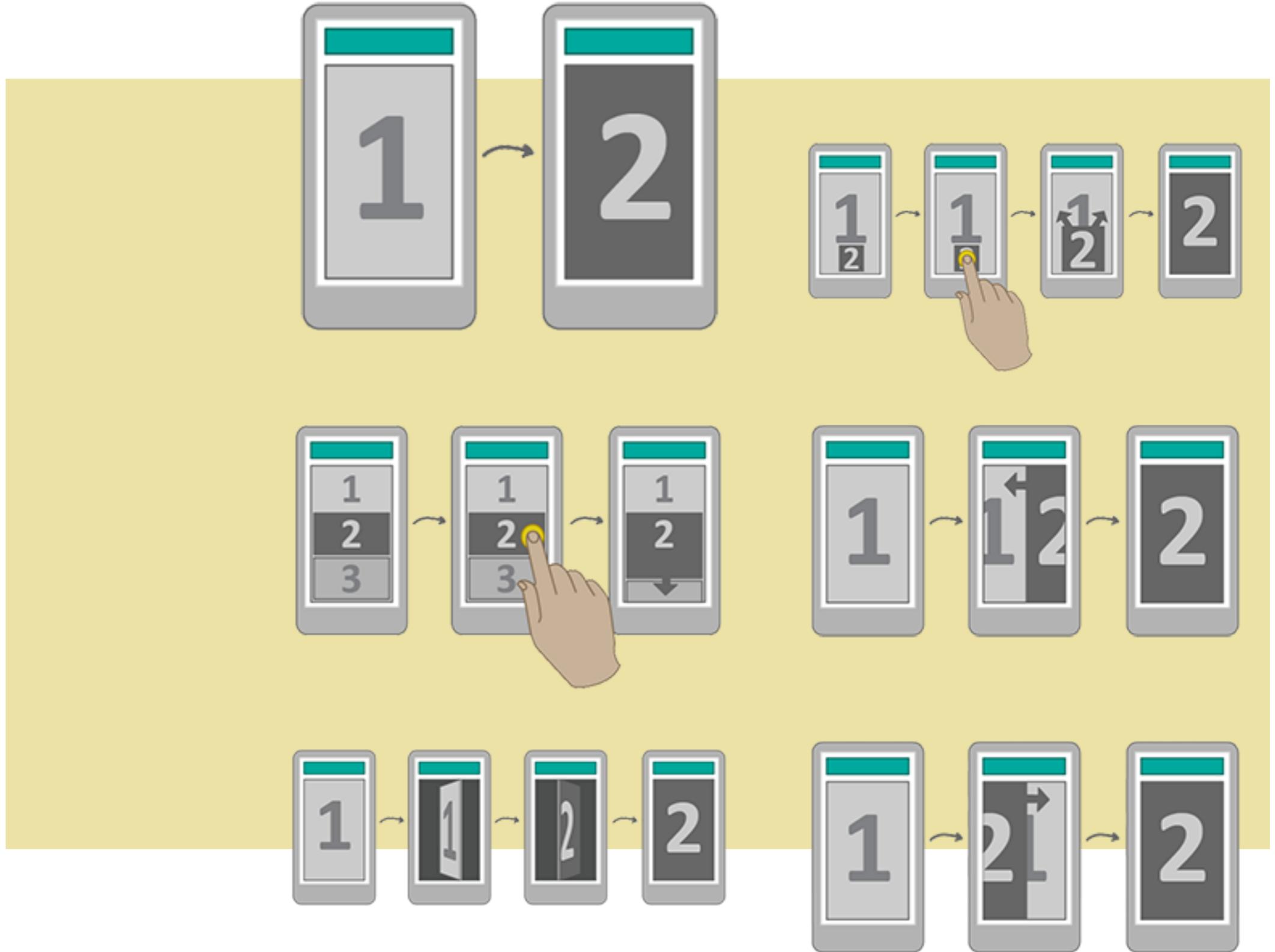


Lasso and cross



2.

transitions



2.

common patterns

- **expanding menu**
- **side menu**
- **tabbed menu**
- **hub and spoke menu**

2.

selecting content

- **press to move forward**
- **overflow**
- **flipover peel back**

2.

signing in

- **auto sign in**
- **remember details**
- **pin codes**

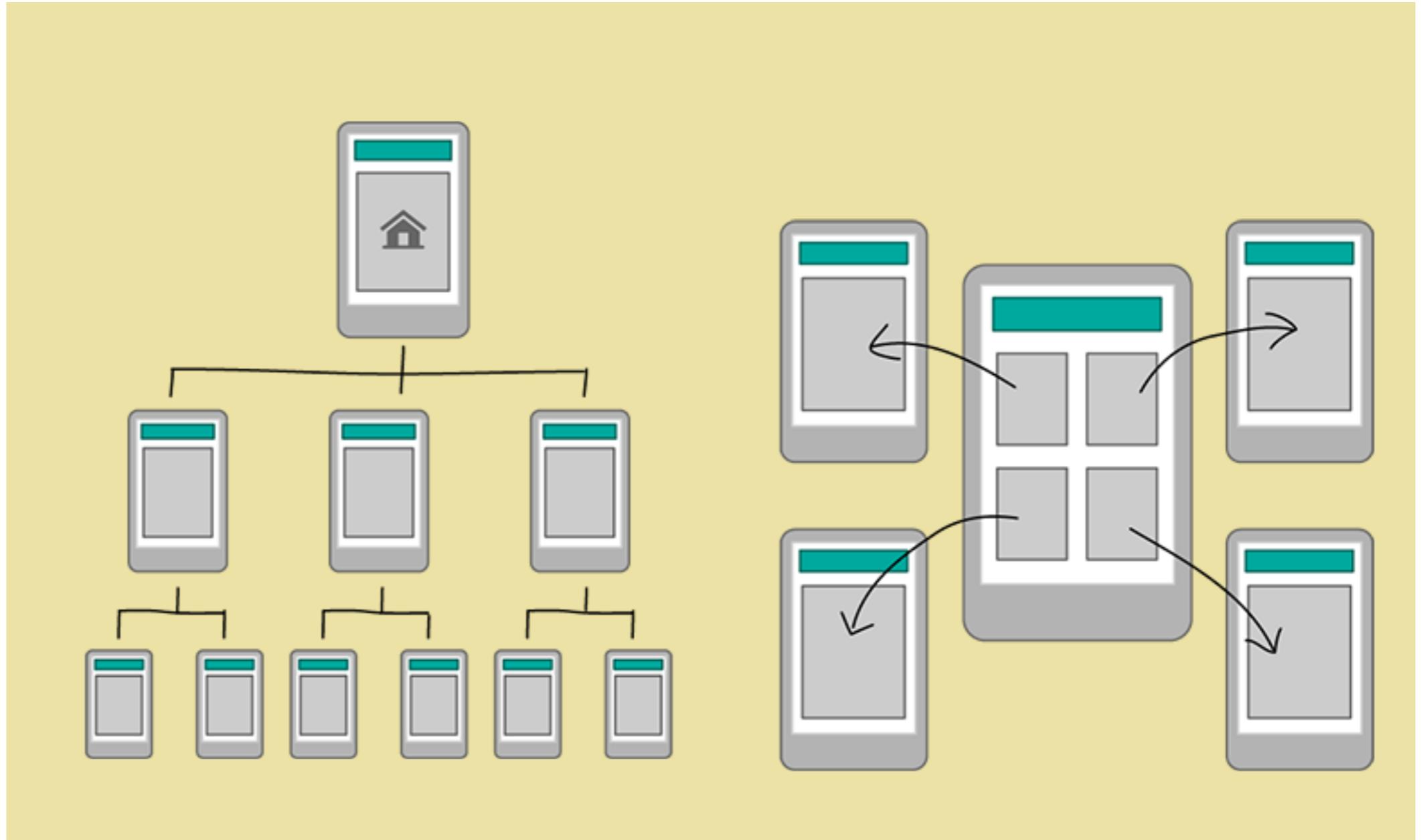
2.

using forms

- **save user details**
- **provide the right keyboard**
- **progress bars**

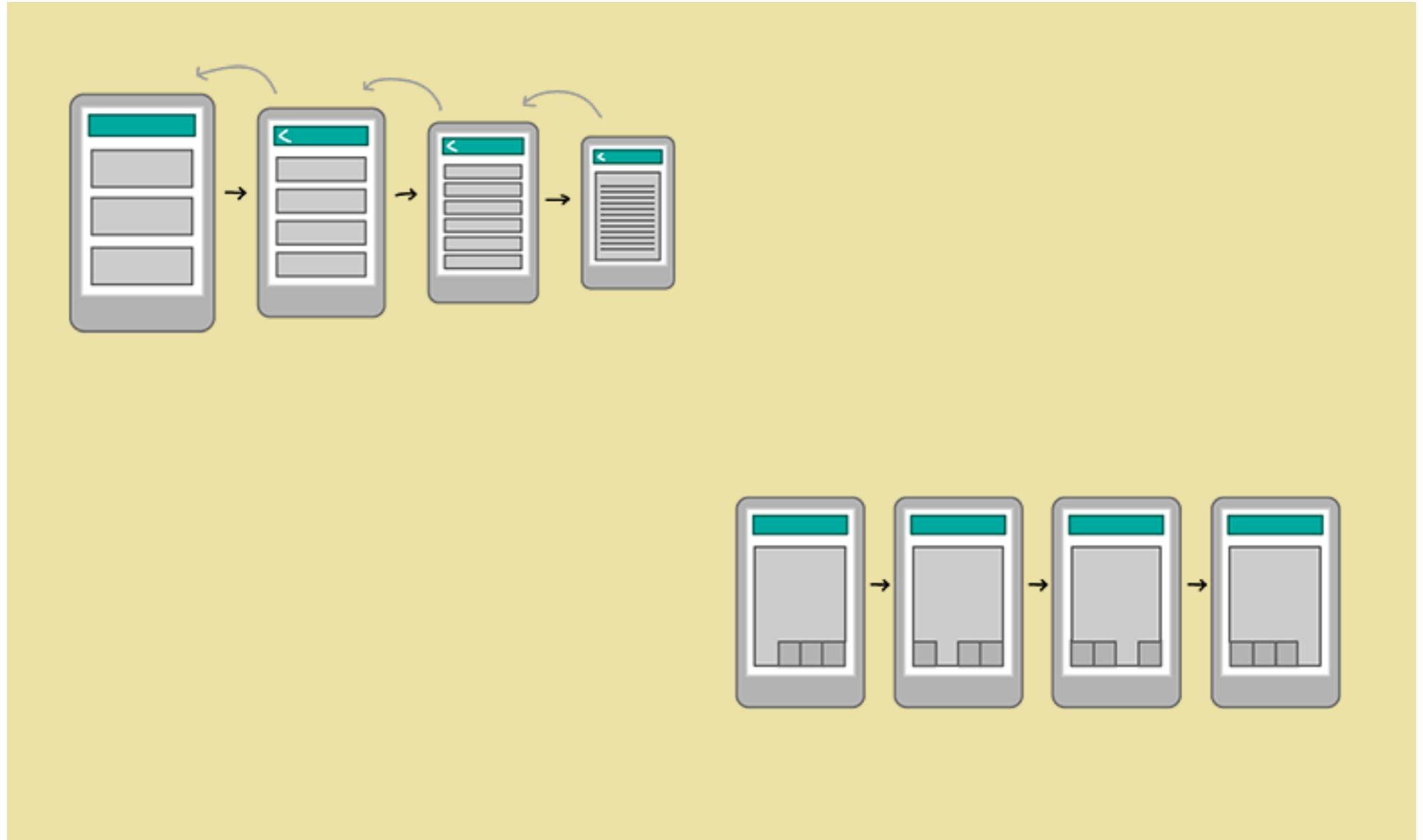
3.

information architecture



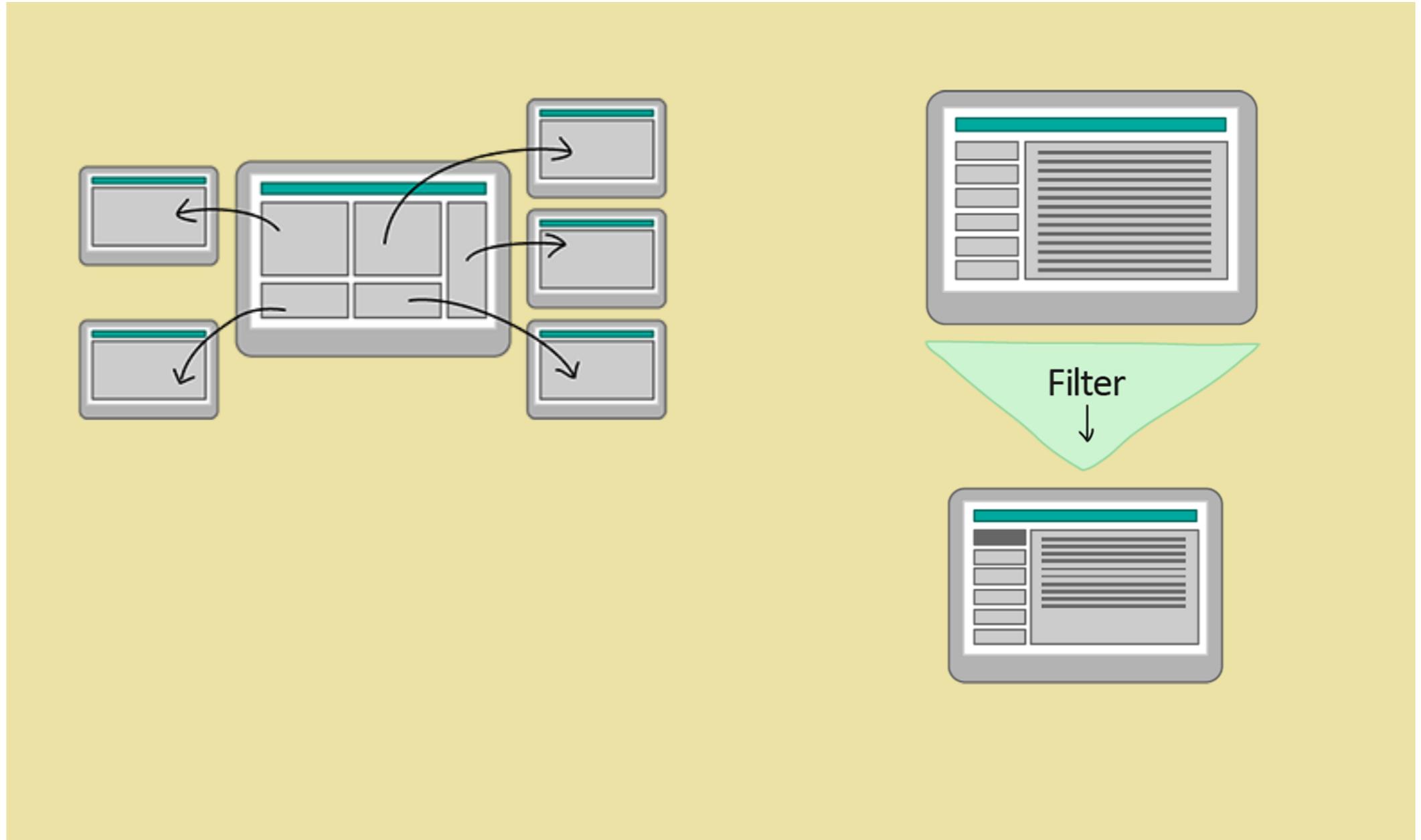
3.

information architecture



3.

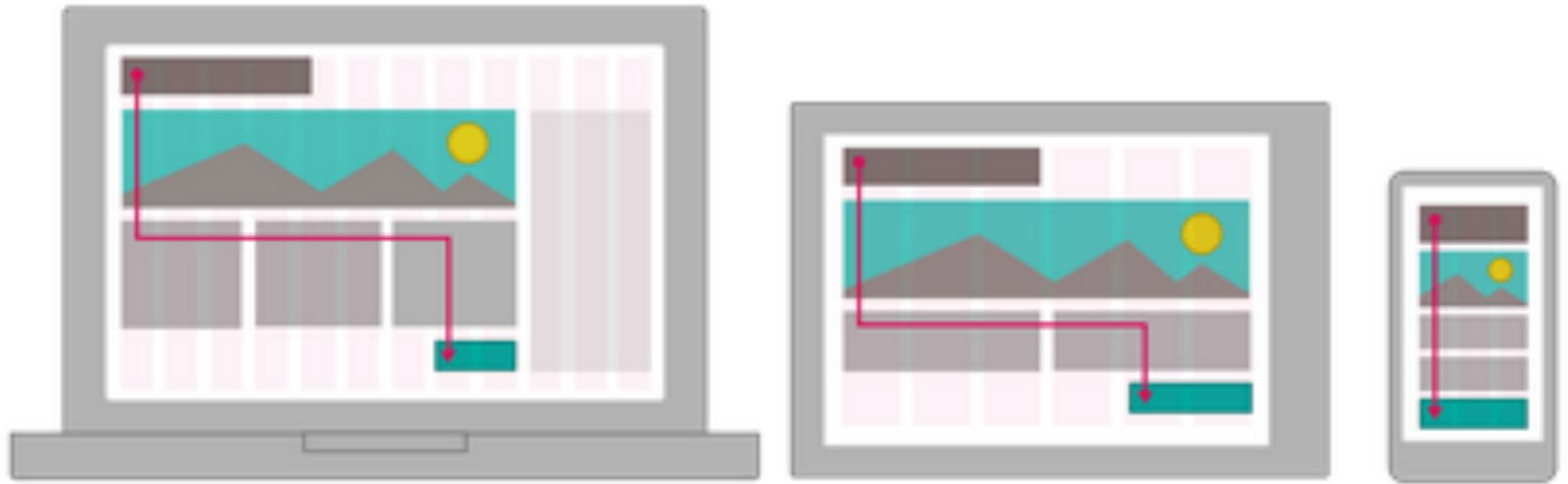
information architecture



3.

usable layouts

limited space



Tips

1.

2.

3.

user friendly interactions

target areas



1.

2.

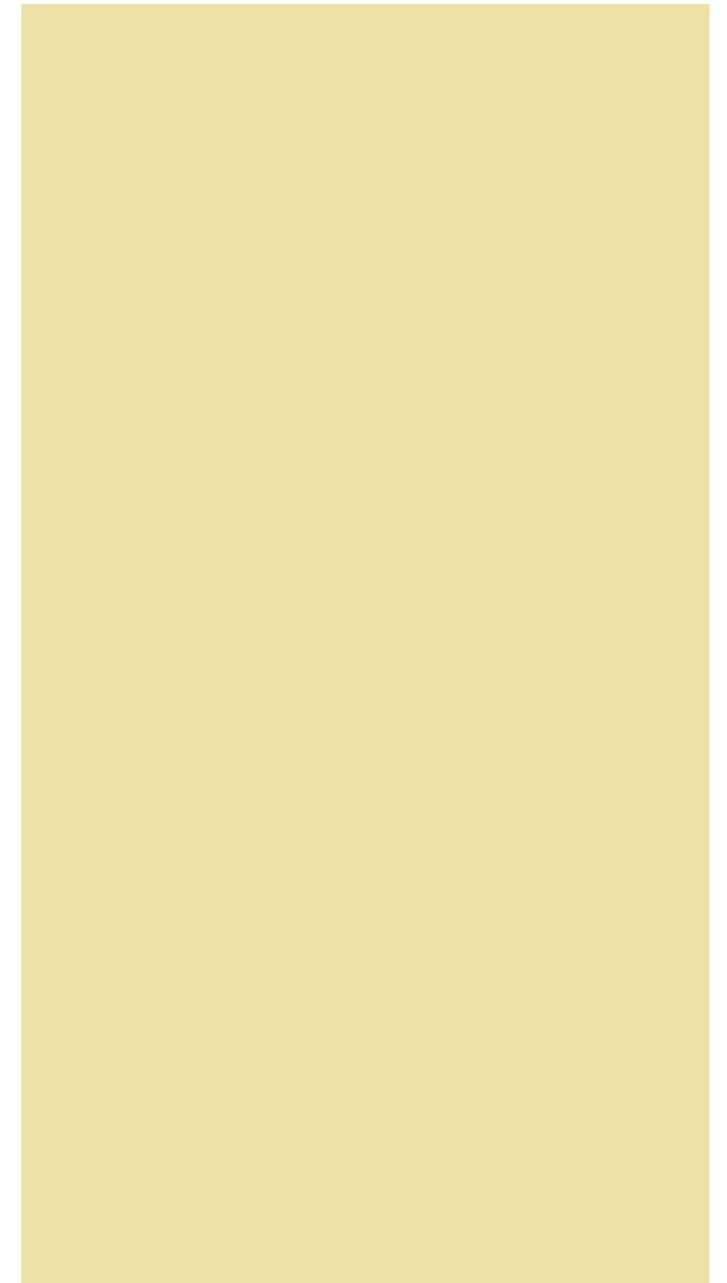


3.

designing for communication

vertical rhythm

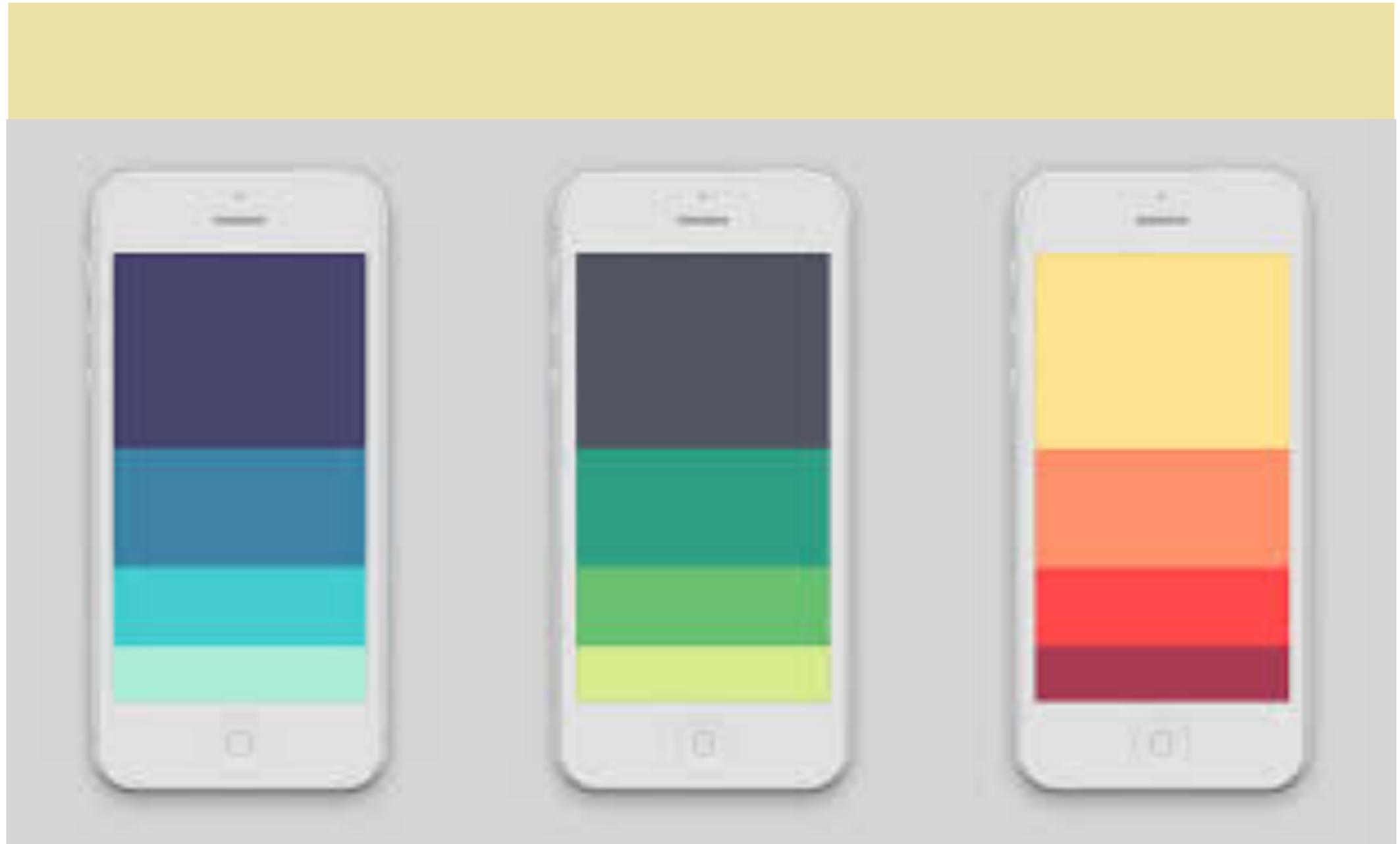
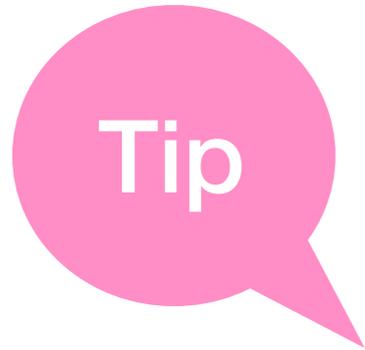
Tip



3.

designing for communication

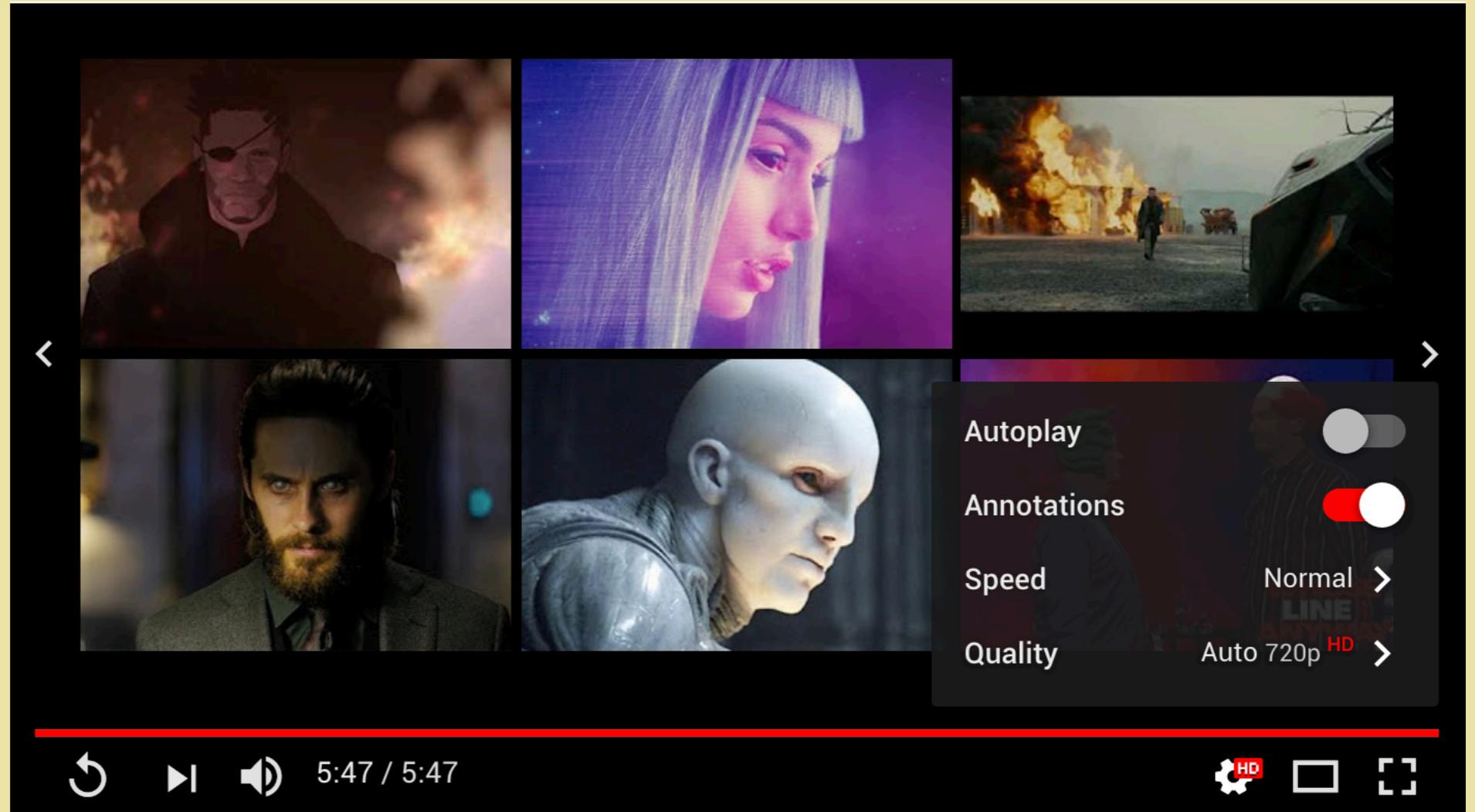
using color



3.

designing for communication

using color as a visual language



3.

designing for communication

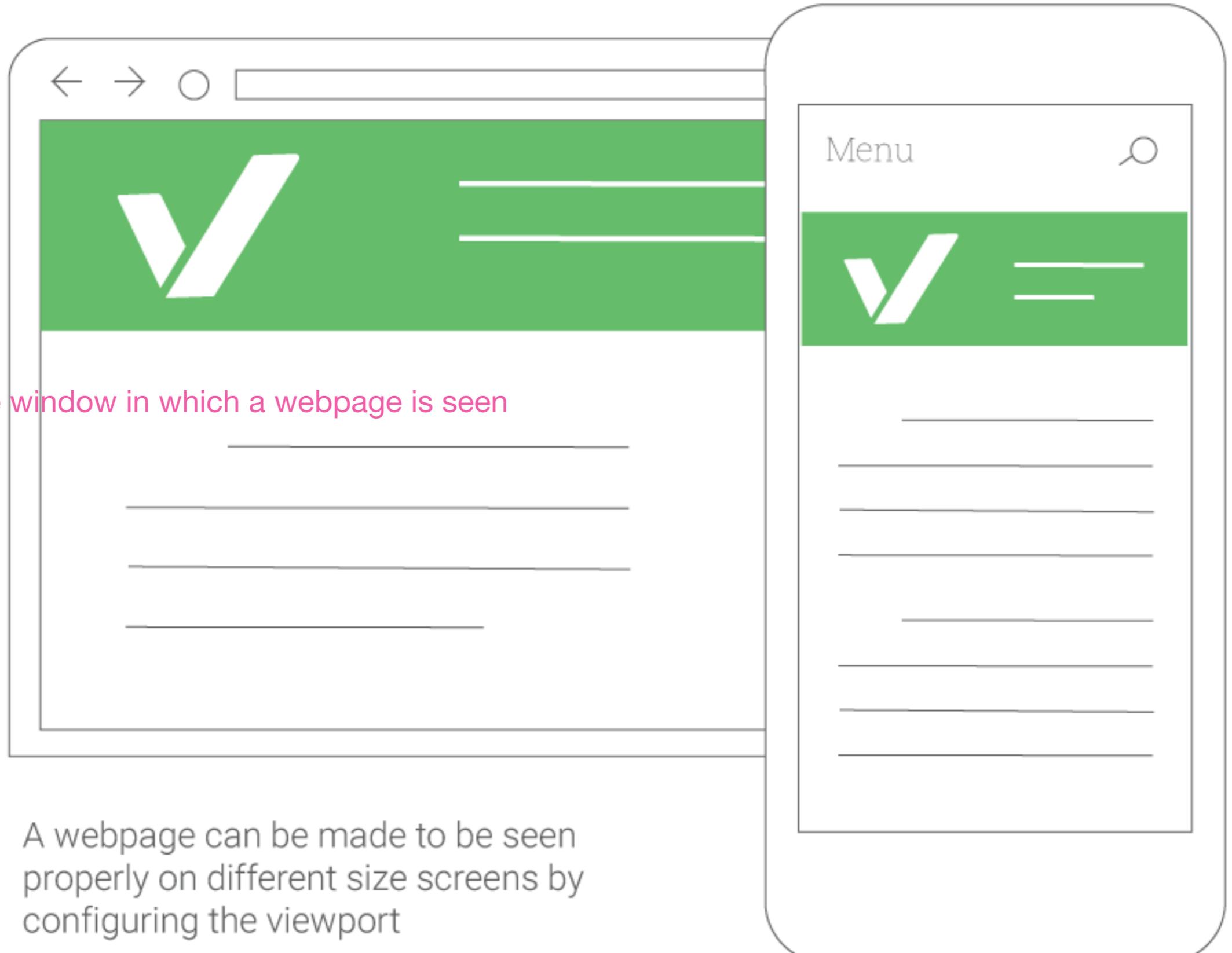
using metaphor



What is a mobile viewport?

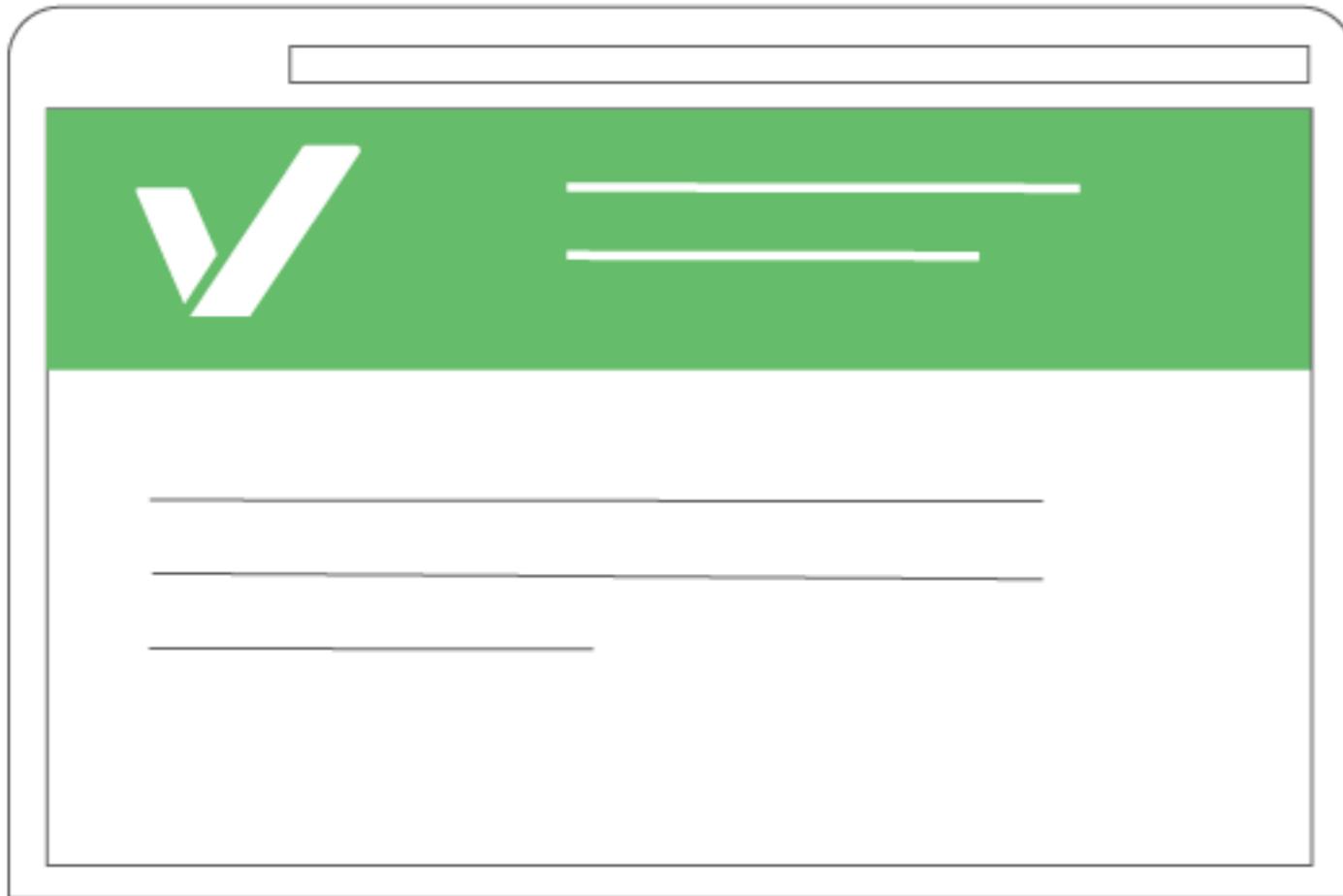


A viewport is the window in which a webpage is seen



A webpage can be made to be seen properly on different size screens by configuring the viewport

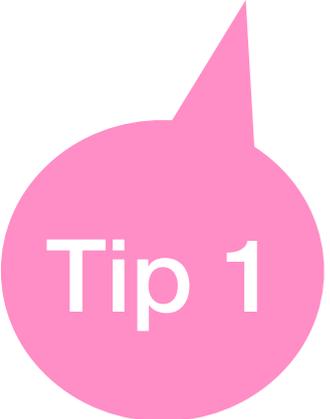
Without a viewport a site will look like this:



When a user goes to this page on mobile they have to zoom in just to make the text large enough to read.

How to configure the viewport?

Add this to the head of the html document.

A pink circular callout with a white border and a small tail pointing towards the tip text.

Tip 1

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

The viewport metatag says

- `width=device-width`
- `initial-scale=1`

What this means:

- **Width:** Make the width of the page the same width as whatever screen it is being shown on.
 - **Initial scale:** If the page is shown in landscape then make the page as wide as it can be within that screen.
-

Responsive Web Design

relies on three main concepts

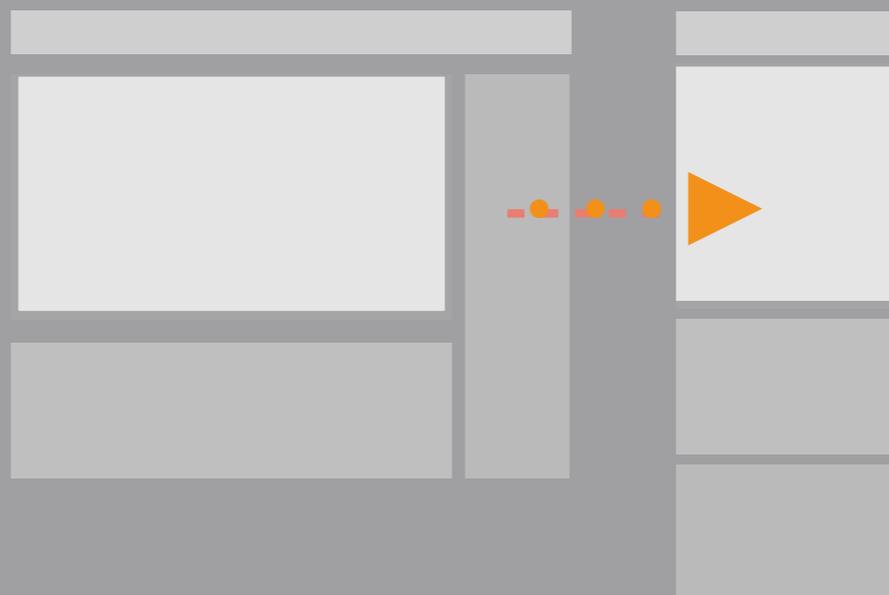
#1

FLUID GRIDS



#2

MEDIA QUERIES



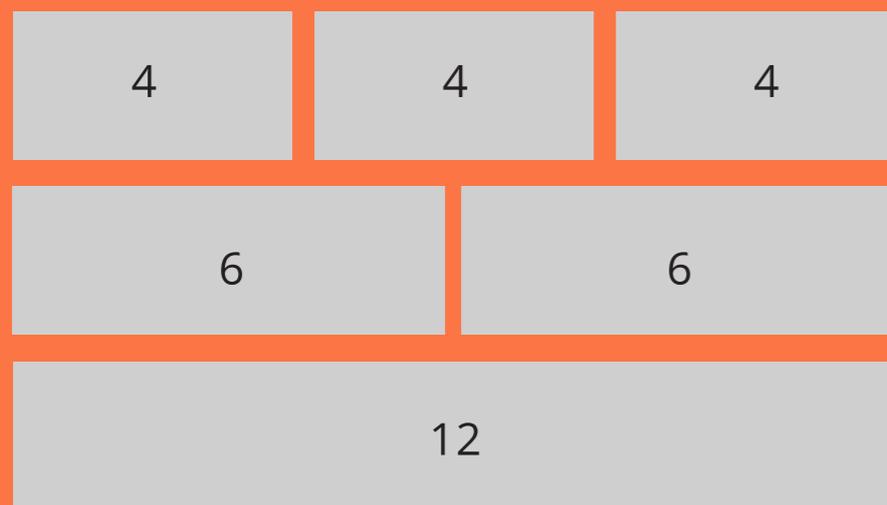
#3

FLEXIBLE MEDIA



FLUID GRIDS

MODULAR

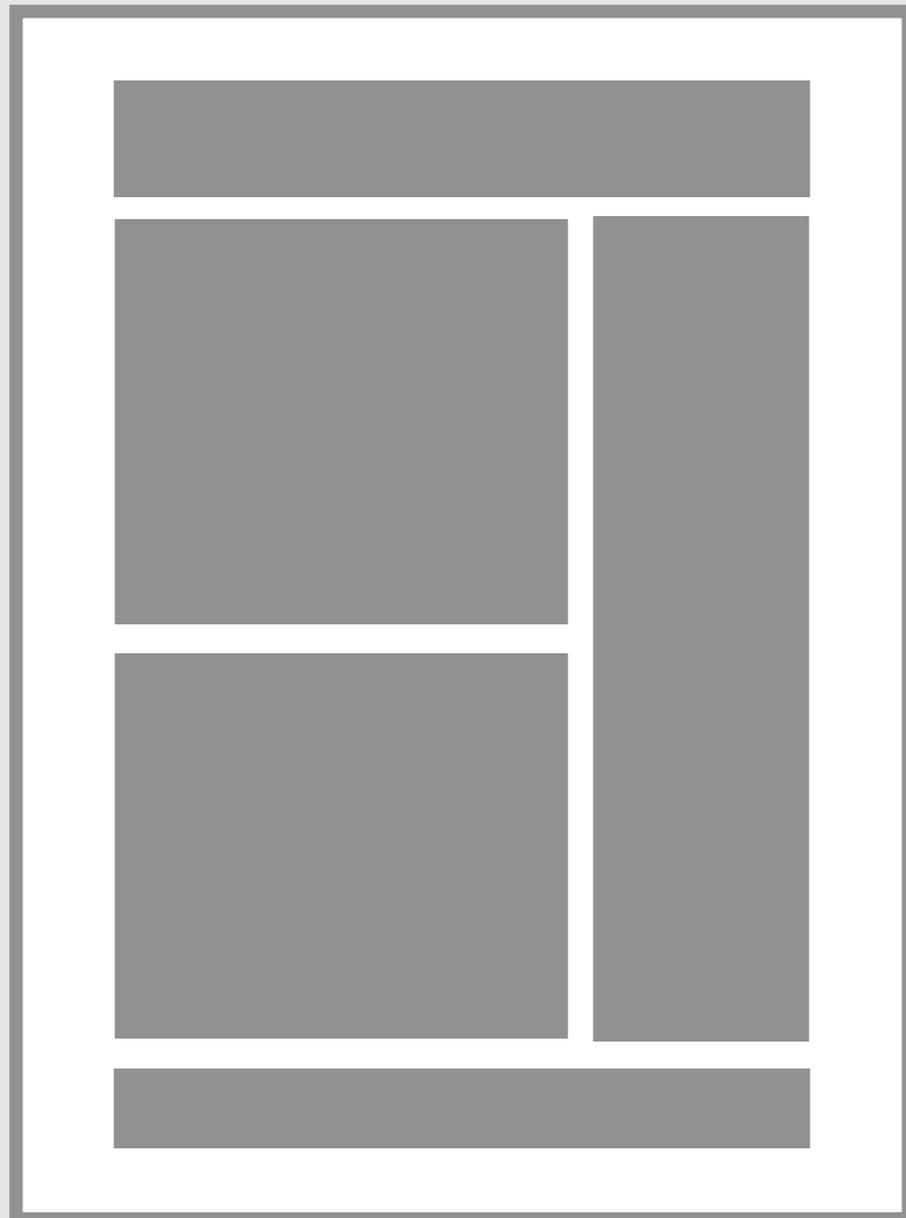


Building a site using modular grids that can flow and adapt to various screen sizes by using percentages, not pixels (as well as a little help from some media queries).

FIXED VS. FLUID

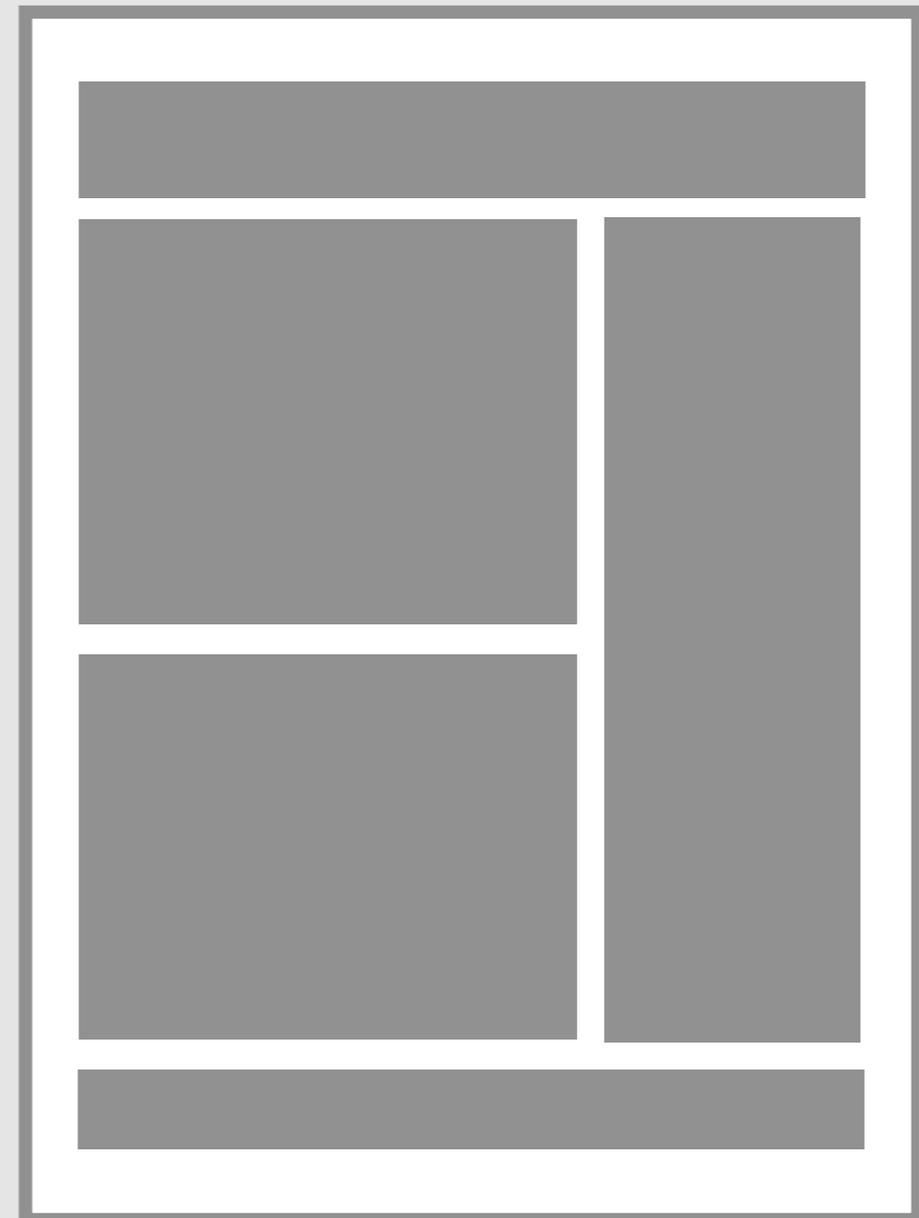
FIXED GRID

← 960px →



FLUID GRID

← 100% →



FLUID GRIDS SCALE & FLEX

And still allow you to align your content.

Logo

Navigation

Article

Sidebar

Footer

Log

Navigation

Article

Sidebar

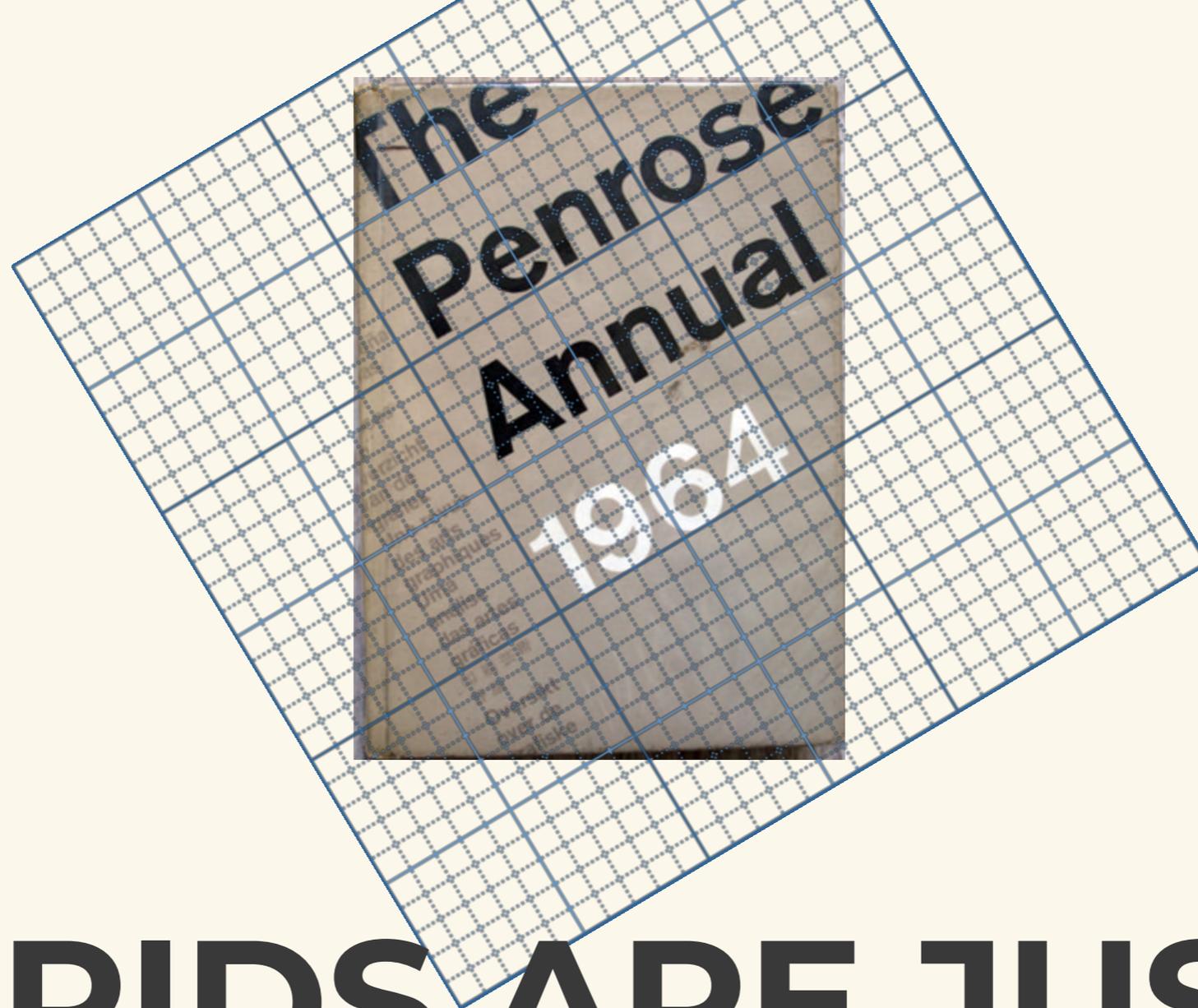
Footer

CHROME GRID PLUGIN

A very simple grid system overlay plug-in. Allows you to see the underlying grid used to layout elements for the website's page.

You can manage :
adjust columns, width and gutters.

The image shows the Grrrid Chrome extension interface. At the top, there's a puzzle piece icon with a red 'X' and the name 'Grrrid' offered by 'david marsalone'. It has a 4.5-star rating from 14 reviews and is categorized as 'Developer Tools' with 1,990 users. Below this are tabs for 'OVERVIEW', 'REVIEWS', 'SUPPORT', and 'RELATED'. The main content area displays a screenshot of a web browser showing the Bootstrap components page. The Grrrid overlay is active, showing a grid of vertical red lines. A settings panel is open on the right, allowing users to adjust: 'Common grids' (a dropdown menu), 'Number of columns' (set to 12), 'Column width' (set to 70), 'Gutter width' (set to 30), 'Align' (set to Center), 'Offset' (set to 0), and 'Grid opacity' (set to 20). The browser page shows a navigation bar, a sidebar with a 'Thumbnails' menu item highlighted, and a main content area with 'More examples' and 'Alerts' sections. A warning alert is visible at the bottom of the page: 'Warning! Best check yo self, you're not looking too good.'

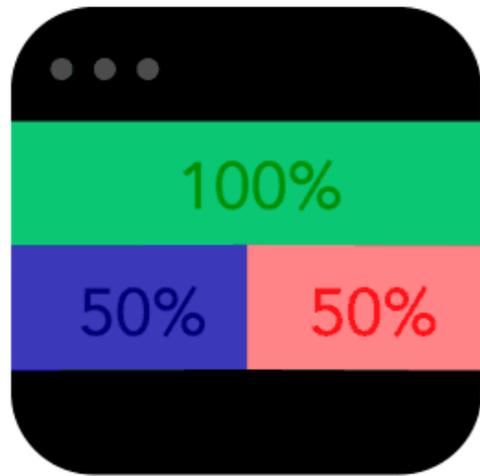


GRIDS ARE JUST GUIDES TO HELP ALIGN

They help you to align elements and provide structure.

They're not perfect.

Relative Units



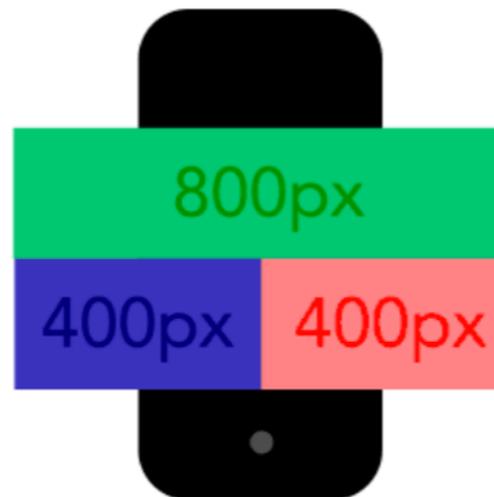
Static Units



Relative Units



Static Units



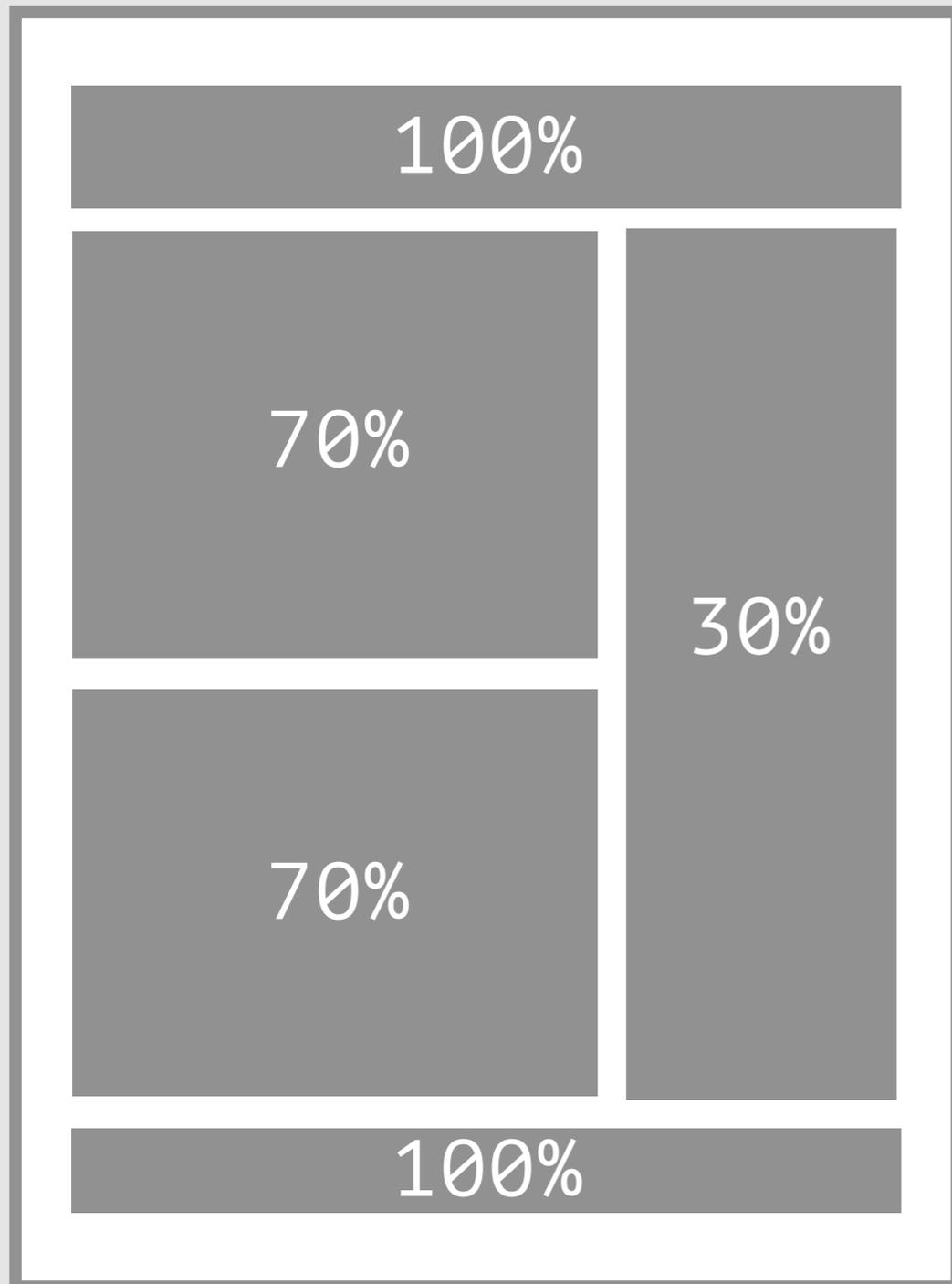
USE %

Relative units are written in percentages.

Pixel density can vary so we need units that are flexible or fluid. This method keeps the layout proportionate to the viewport.

FLUID LAYOUT

FLUID GRID

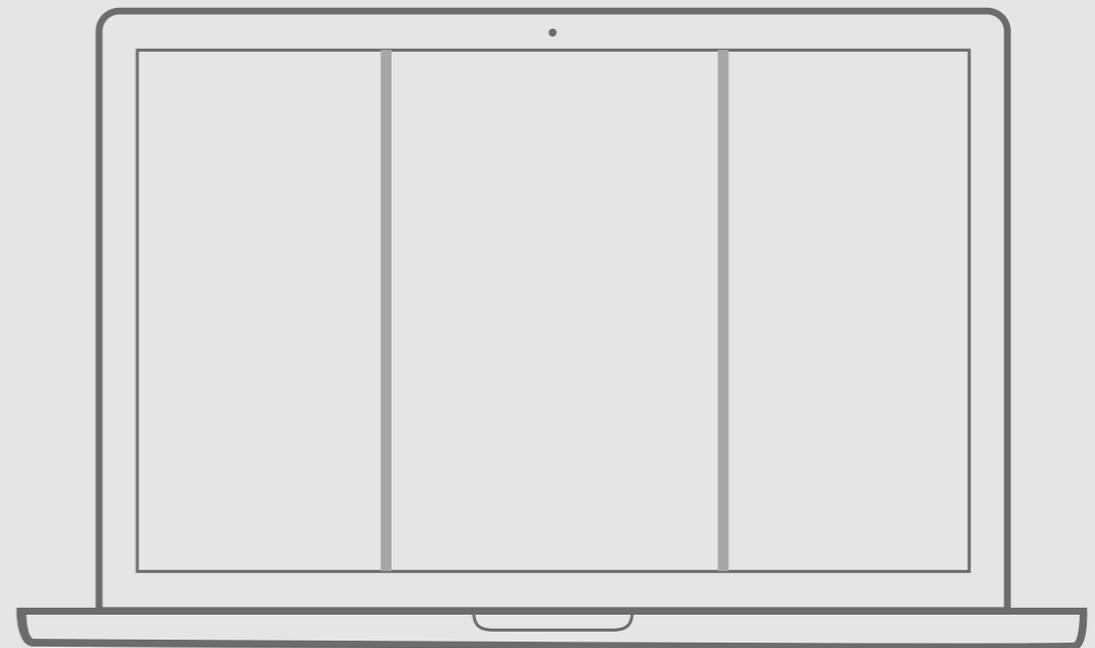
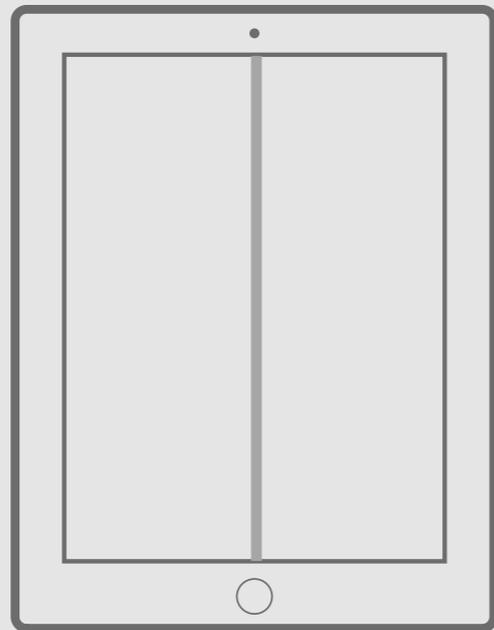


```
#header {  
    width: 100%;  
}  
#side-bar {  
    float: right;  
    width: 30%;  
}  
#content {  
    float: left;  
    width: 70%;  
}
```

BREAKPOINTS

Are the points at which your site's content will respond to a device size to provide the user with the best possible layout to consume the information.

Both Responsive & Adaptive use breakpoints :)



BREAKPOINTS



<http://mediaqueri.es/popular/>

Think Vitamin



Hi there, Welcome to the New Think Vitamin. Keep up to date by signing up to our [Facebook](#) and subscribing to our [RSS Feed](#).

THE WEB PRACTITIONER'S BLOG

Home Membership About Online Conferences Podcast Archive Write For Us

Communicating Freelance Development

By [Nick Pallant](#)
20 December 2010 | Category: [Business](#)

Freelance development- chances are you've done some at one time or another. Be it as your day to day job or a few hours on the side, most developers at some point in their career have dipped their fingers into this particular honey pot.

Many a time though, although the honey is undeniably sweet, you can pull your finger out and find that you've been stung. Freelancing is akin to traveling over rocky roads and rivers- it can get tough sometimes, but once you're on that peak with that oh so beautiful view, you know that long hard journey was so worth it.

Topics: Business, CSS3, HTML5, JavaScript, PHP, Ruby on Rails, UX, Podcast

Think Vitamin Membership Give the gift of web knowledge this Christmas!

The perfect Christmas gift for web geeks!

Twitter: [Tweet from @jkrums](#)
Powered by [@thinkvitamin](#) | 21 December 2010 | Category: [Articles](#)

Wow, that is incredible! Google Demo Slides: Epic Dice Animation <http://post.jp/AMV/>

Around the Web: Clone Delicious, Forrst

Hi there, Welcome to the New Think Vitamin. Keep up to date by signing up to our [Facebook](#) and subscribing to our [RSS Feed](#).

THE WEB PRACTITIONER'S BLOG

Home Membership About Online Conferences Podcast Archive Write For Us

Communicating Freelance Development

By [Nick Pallant](#)
20 December 2010 | Category: [Business](#)

Freelance development- chances are you've done some at one time or another. Be it as your day to day job or a few hours on the side, most developers at some point in their career have dipped their fingers into this particular honey pot.

Many a time though, although the honey is undeniably sweet, you can pull your finger out and find that you've been stung. Freelancing is akin to traveling over rocky roads and rivers- it can get tough sometimes, but once you're on that peak with that oh so beautiful view, you know that long hard journey was so worth it.

Topics: Business, CSS3, HTML5, JavaScript, PHP, Ruby on Rails, UX, Podcast

Think Vitamin Membership Give the gift of web knowledge this Christmas!

The perfect Christmas gift for web geeks!

Twitter: [Tweet from @jkrums](#)
Powered by [@thinkvitamin](#) | 21 December 2010 | Category: [Articles](#)

Wow, that is incredible! Google Demo Slides: Epic Dice Animation <http://post.jp/AMV/>

Around the Web: Clone Delicious, Forrst V3

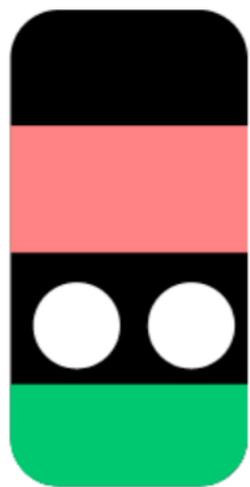
With Breakpoints



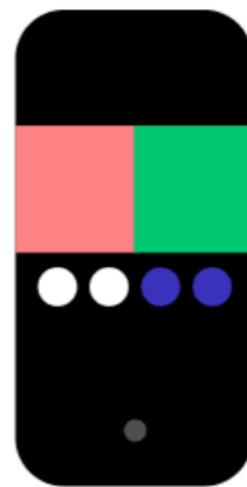
Without Breakpoints



With Breakpoints



Without Breakpoints



BREAK-POINTS

allow the layout to change at predefined points, ie having 3 columns switch to two on mobile. Most CSS properties can be modified for different screen sizes.

#3

FLEXIBLE MEDIA

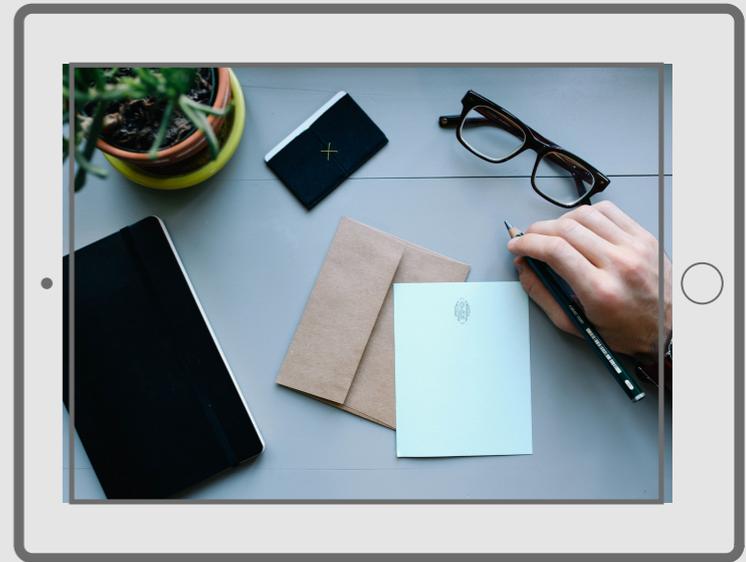


Ensuring images are large enough to appear on the biggest of screens and are coded to scale proportionately on smaller devices.

#3

FLEXIBLE

Can scale across different device sizes while maintaining its resolution and proportion.



SIMPLE LINE OF CODE

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

MAX-WIDTH 100%

This works on most devices but you may have to do a couple of hacks to have these work on IE6 and below.

If the image is nested inside a parent container that's smaller than their pixel value, then the image should shrink. So for example, if an image with a width of 800px is placed inside a container that's only 600px wide, the image will also shrink to be 600px wide. The height will be calculated automatically and will maintain the original aspect ratio.

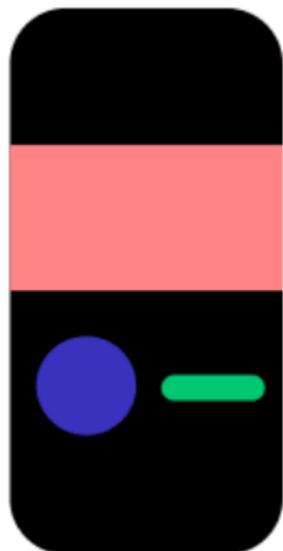
Max width



No max width



Max width



No max width

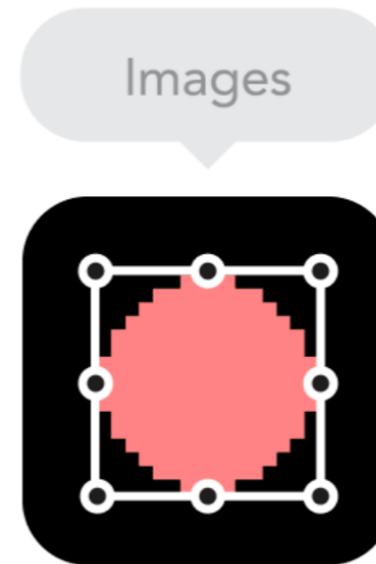
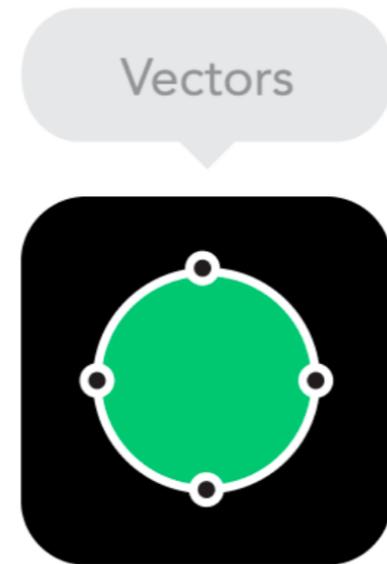


```
/* Small screens (default) */  
html { font-size: 100%; }  
  
/* Medium screens (640px) */  
@media (min-width: 40rem) {  
  html { font-size: 112%; }  
}  
  
/* Large screens (1024px) */  
@media (min-width: 64rem) {  
  html { font-size: 120%; }  
}
```

MAX AND MIN WIDTH

MEDIA QUERIES

Introduce layout-specific rules only when you need them. Use min-width to layer complexity on your layout as the viewport widens.



Bitmap images vs Vectors

Does your icon have lot of details and some fancy effects applied?

If not, consider using a vector image.

For vectors the best choice would be a SVG or an icon font.

If you answered yes, use a bitmap. For bitmaps use a jpg, png or a gif.

Each has some benefits and some drawbacks.

WHAT WE'VE LEARNED



There are 3 elements that make up Responsive Design: Media Queries, Fluid Grids, and Flexible Media.



Media Queries are magical lines of CSS styles that can tell what size device you're viewing on and provide the styles accordingly.



Fluid Grids adapt to different screen-sizes by using percentages instead of pixels.



Flexible Media are images and videos that can maintain resolution and proportion on all screen sizes.

VIEWPORT TAG

Place in the `<head>` of your HTML.

This enables use of media queries for cross-device layouts.

VIEWPORT

(HTML DOC)

```
<!DOCTYPE html>
  <head>
    <meta charset="utf-8">
    <title>My First Fluid Layout</title>
    <meta name="description" content="">
    <meta name="keywords" content="">
    <meta name="viewport" content="width=device-
width,initial-scale=1.0, user-scalable=yes"/>
  </head>
```

@MEDIA SCREEN

(MEDIA QUERY IN YOUR CSS DOC)

```
@media screen and (min-width: 400px) {  
  header {background: red;}  
}
```

@MEDIA SCREEN

(MEDIA QUERY IN YOUR CSS DOC)

```
/* ...mobile styles here...  
*/ @media screen and (min-width: 600px) {  
/* ...tablet styles here... */ }
```

```
@media screen and (min-width: 900px) {  
/* ...desktop styles here... */ }
```

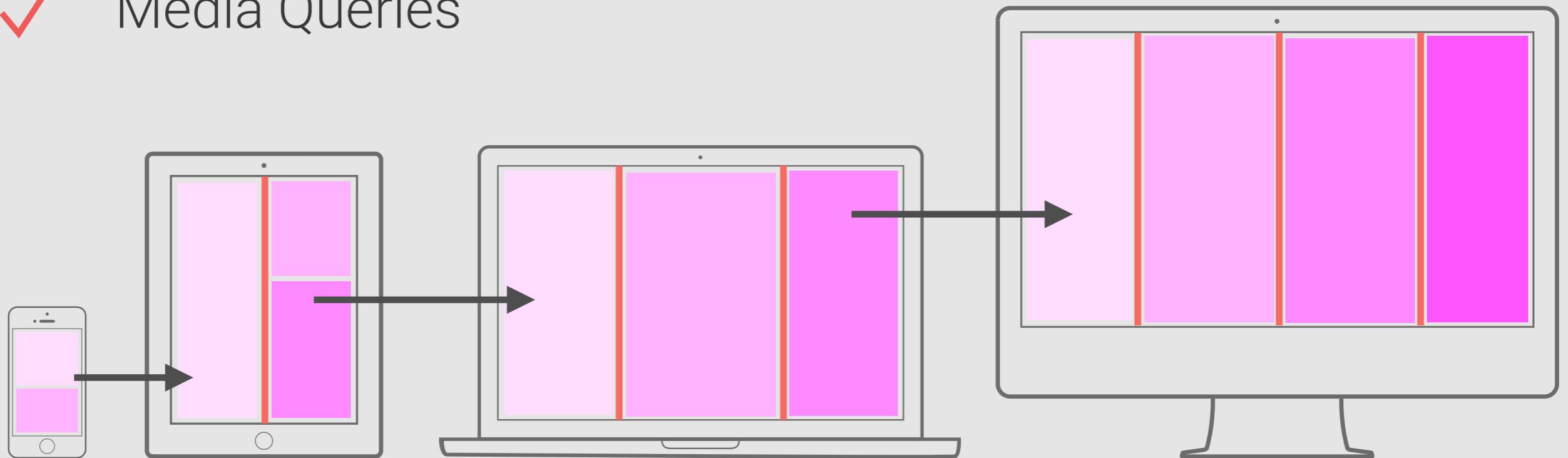
This is just a small example. In practice, you might have many more media queries at sizes that are appropriate for your content.

I recommend using your site's content as a guide for creating these breakpoints. But, in case you need it, Screensiz.es has the best listing I've seen of device resolutions

OTHER SUGGESTED PRACTICES

RESPONSIVE DESIGN

- ✓ Fluid Grids
- ✓ Flexible Media
- ✓ Media Queries



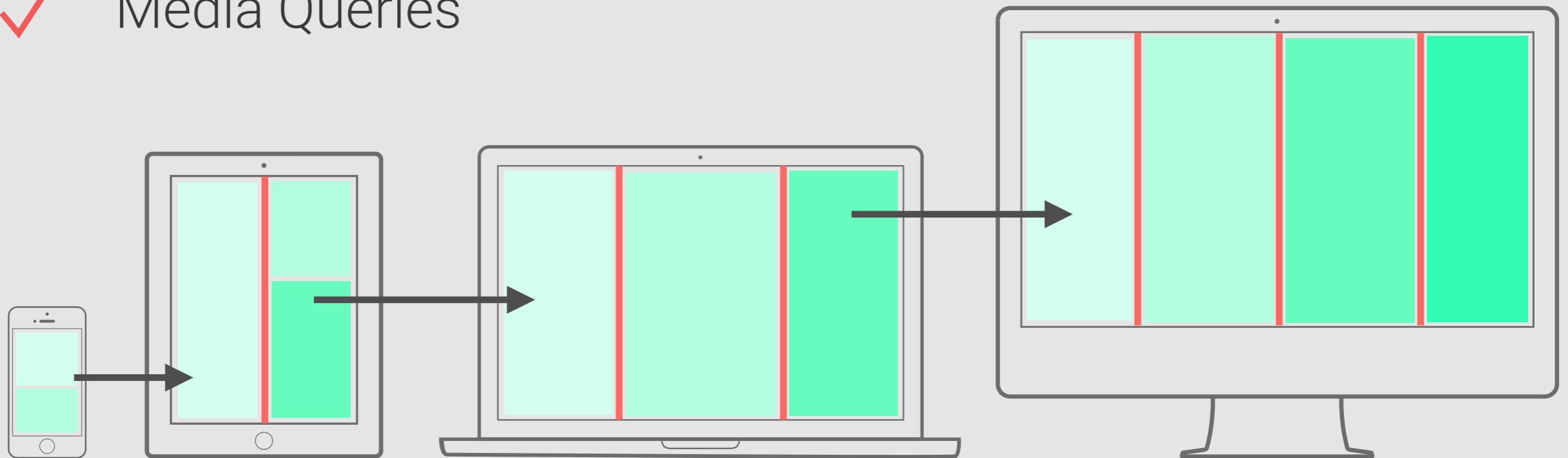
ADAPTIVE DESIGN

Still has breakpoints for different device sizes, but it's *not* fluid.

~~Fluid Grids~~

~~Flexible Media~~

✓ Media Queries



CROSS-BROWSER TESTING

Test your responsive site on different browsers.
With online testing tools you can check them all at once!

The screenshot shows the BrowserStack website interface. At the top, there is a navigation bar with the BrowserStack logo and links for LIVE, AUTOMATE, SCREENSHOTS, and RESPONSIVE. On the right side of the navigation bar, there are links for Pricing, Docs, Help, and Sign in, along with a FREE TRIAL button. Below the navigation bar, a blue banner contains the text "Rapidly test your website for cross browser compatibility across 700+ browsers." and a search input field with a History dropdown. Below the search field, there is a button for "Start local testing" and a link for "or Use binaries". A notification box indicates "15 browsers pre-selected". The main content area displays a grid of devices categorized by platform: iOS, Android, and Windows. Each device is represented by its logo and a list of models. Some models are highlighted with a blue background, indicating they are pre-selected for testing.

Platform	Device	Version
iOS	iPad Mini 2	v8
	iPhone 6	v8
	iPhone 6 Plus	v8
	iPhone 5S	v7
	iPhone 5	v6
	iPhone 4S	v6
	iPad 3	v5.1
	iPad 2	v5
Android	Galaxy S5	Samsung
	Galaxy S4	Samsung
	Galaxy S3	Samsung
	Galaxy S2	Samsung
	Galaxy Tab 2	Samsung
	Galaxy S5 Mini	Samsung
	Galaxy Note 3	Samsung
	Galaxy Note 2	Samsung
	Galaxy Note 10.1	Samsung
	Galaxy Note	Samsung
	Kindle Fire 2	Amazon
	Kindle Fire HD 8.9	Amazon
	Kindle Fire HDX	Amazon
	Razr	Motorola
	Droid Razr	Motorola
Razr Maxx HD	Motorola	
Nexus 7	Google	
Nexus 5	Google	
Nexus 4	Google	
Nexus	Google	
One M8	HTC	
One X	HTC	
Wildfire	HTC	
Xperia Tipo	Sony	
Windows	Windows 8.1	
	11 Desktop	
	30	Chrome
	29	Chrome
	28	Chrome
	27	Chrome
	26	Chrome
	25	Chrome
	24	Chrome
	23	Chrome
36	Chrome	
35	Chrome	
34	Chrome	
33	Chrome	
32	Chrome	
31	Chrome	
30	Chrome	
29	Chrome	
28	Chrome	
27	Chrome	
26	Chrome	
25	Chrome	
24	Chrome	
12.16	Internet Explorer	
12.15	Internet Explorer	
12.14	Internet Explorer	
12.10	Internet Explorer	
12	Internet Explorer	
5.1*	Internet Explorer	

VALIDATE YOUR CODE

Check your code to check for errors.

The screenshot shows a web browser window with the URL `validator.w3.org`. The page features a blue header with the W3C logo and the text "Markup Validation Service" and "Check the markup (HTML, XHTML, ...) of Web documents". Below the header, there are three tabs: "Validate by URI", "Validate by File Upload", and "Validate by Direct Input". The "Validate by URI" tab is active, showing a form with the label "Validate by URI" and "Validate a document online:". The form includes an "Address:" label and a text input field. Below the input field is a link for "More Options". At the bottom of the form is a "Check" button. Below the form, there is a paragraph of text explaining the validator's purpose and providing links to other tools. At the bottom of the page, there is a logo for "W3C VALIDATOR Suite", a link to "Try now the W3C Validator Suite™", and a "Donate" link. A small "I ♥ VALIDATOR" logo is in the bottom left, and a "5083" counter is in the bottom right.

W3C[®] Markup Validation Service
Check the markup (HTML, XHTML, ...) of Web documents

Validate by URI | Validate by File Upload | Validate by Direct Input

Validate by URI

Validate a document online:

Address:

► More Options

Check

This validator checks the [markup validity](#) of Web documents in HTML, XHTML, SMIL, MathML, etc. If you wish to validate specific content such as [RSS/Atom feeds](#) or [CSS stylesheets](#), [MobileOK content](#), or to [find broken links](#), there are [other validators and tools](#) available. As an alternative you can also try our [non-DTD-based validator](#).

W3C VALIDATOR Suite

Try now the [W3C Validator Suite™](#) premium service that checks your entire website and evaluates its conformance with W3C open standards to quickly identify those portions of your website that need your attention.

The W3C validators rely on community support for hosting and development. [Donate](#) and help us build better tools for a better web.

I ♥ VALIDATOR

5083

RESOURCES

Grid Generator

Use the Calculator to Build a Responsive Web Site Your Way

Decide the number of columns you want in a row & set the margin you want to use.

Some sweet maths will do the heavy lifting for you.

<http://www.responsivegridsystem.com/calculator/>



9 Basic Principles of Responsive Web Design

<http://blog.froont.com/9-basic-principles-of-responsive-web-design/>



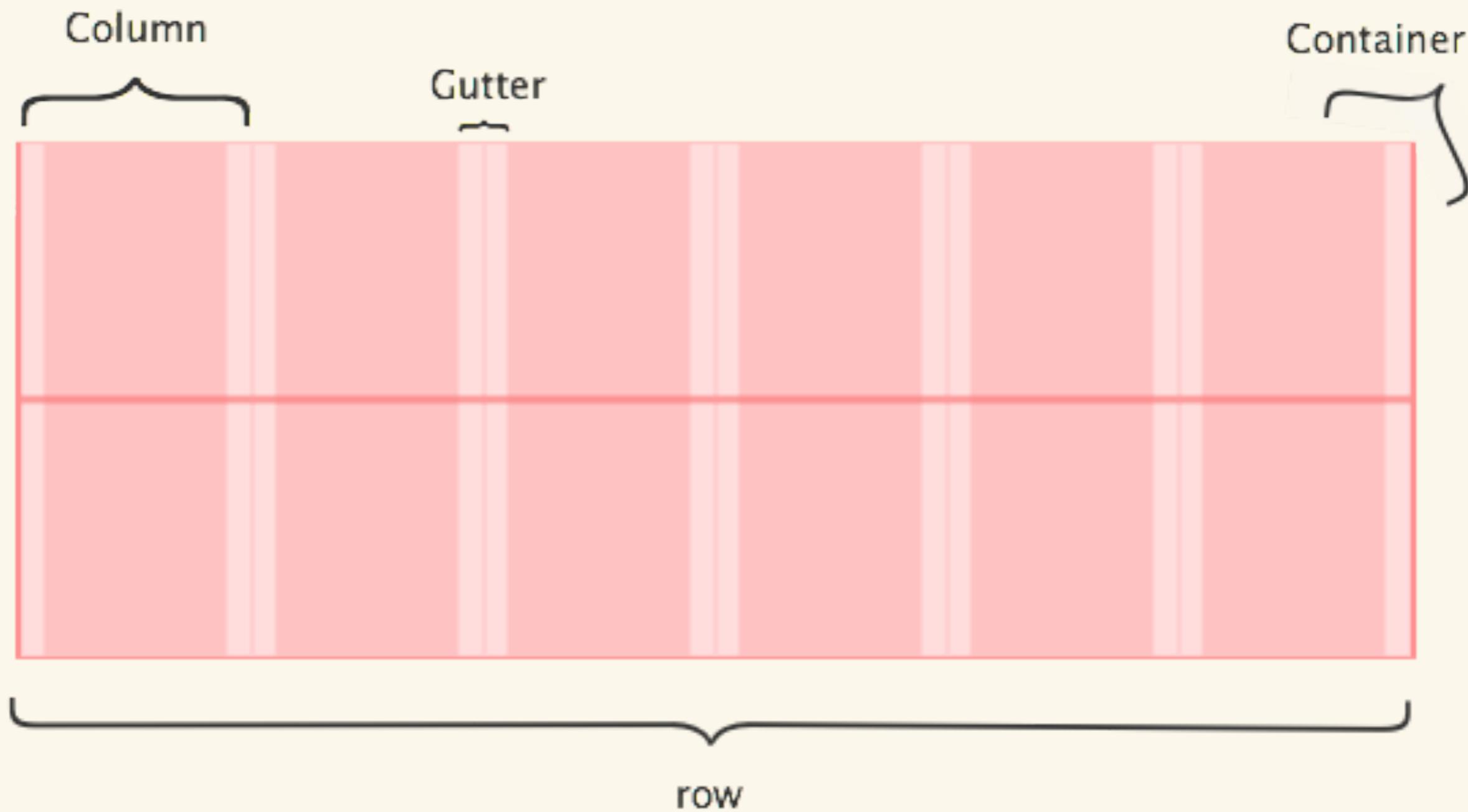
Wild Experiments in the Latest Coding

<http://beta.theexpressiveweb.com/#!/css3-media-queries>

frameworks

- A FLUID GRID LAYOUT
- RESPONSIVE DESIGN
- CUSTOM FORM ELEMENTS
- TYPOGRAPHY
- JAVASCRIPT INTERACTION
- CROSS BROWSER COMPATABILITY
- IT IS FAST
- IT IS CUSTOMIZABLE

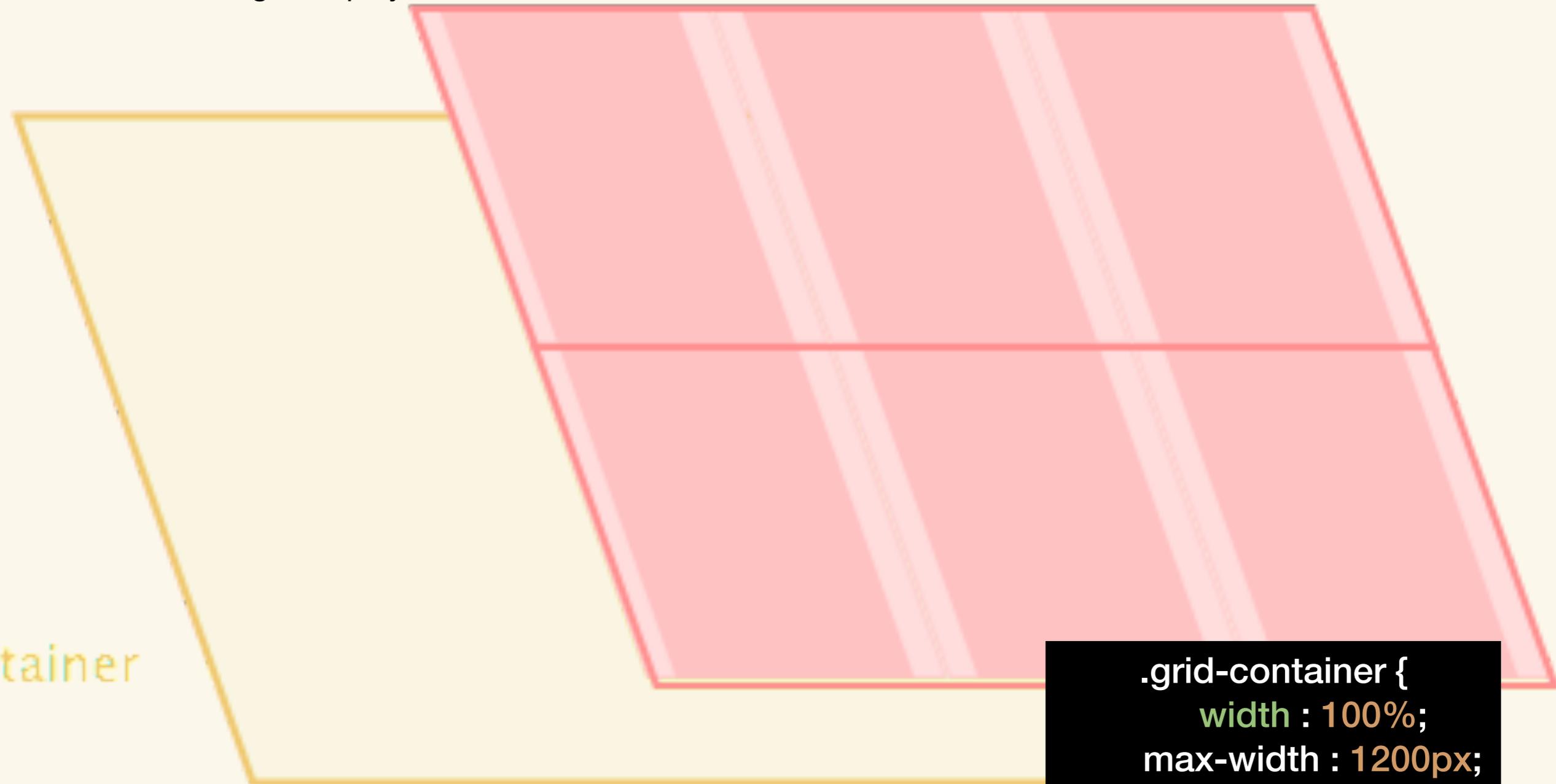
ELEMENTS OF A CSS GRID



CONTAINER The purpose of the container is to set a width

The width of the container is generally 100%, but you might want to set a max-width for larger displays.

Container



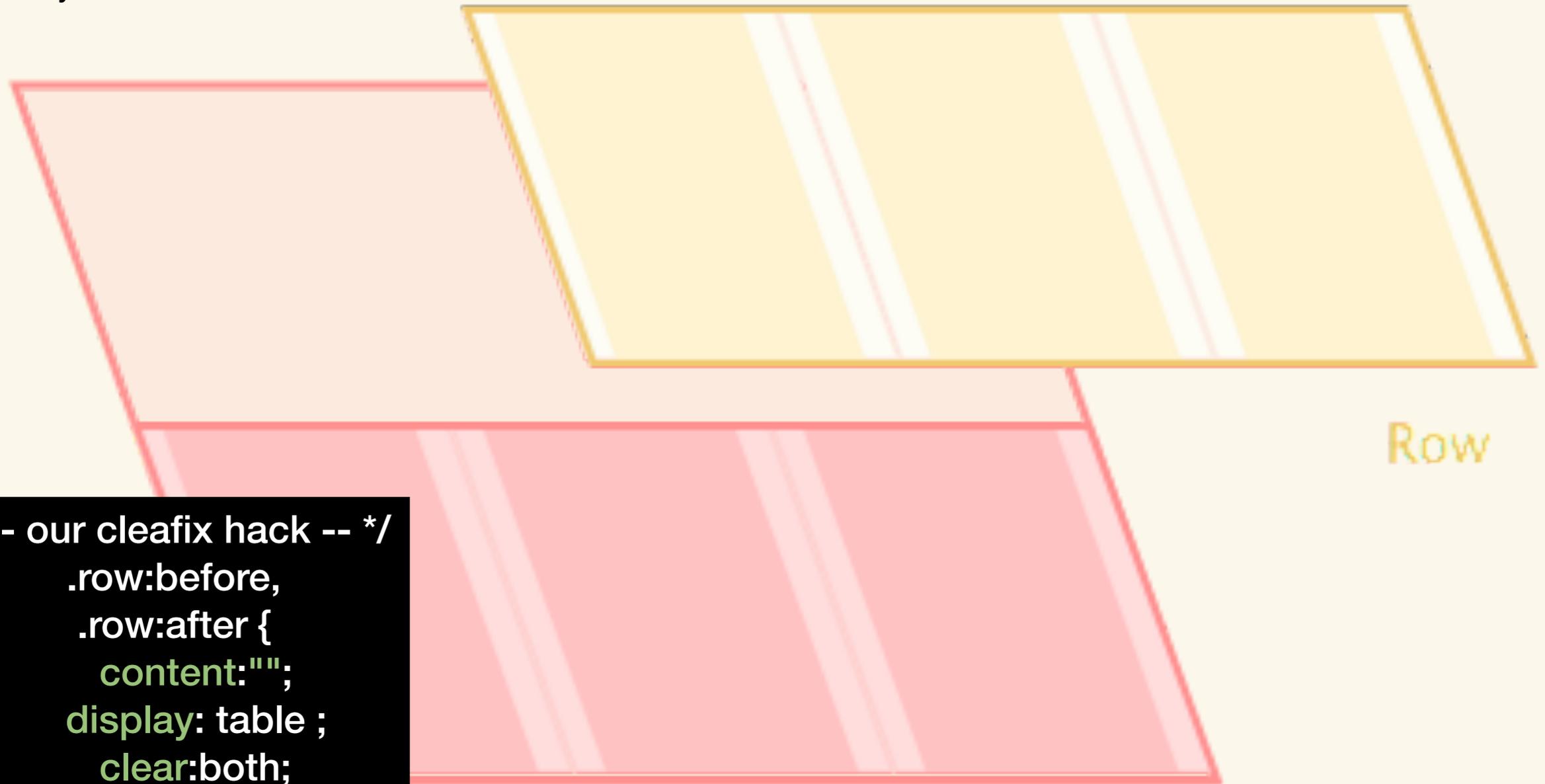
```
.grid-container {  
  width : 100%;  
  max-width : 1200px;  
}
```

BOX SIZING BORDER-BOX

```
html {  
  box-sizing: border-box;  
}  
  
*,  
*:before,  
*:after {  
  box-sizing: inherit;  
}
```

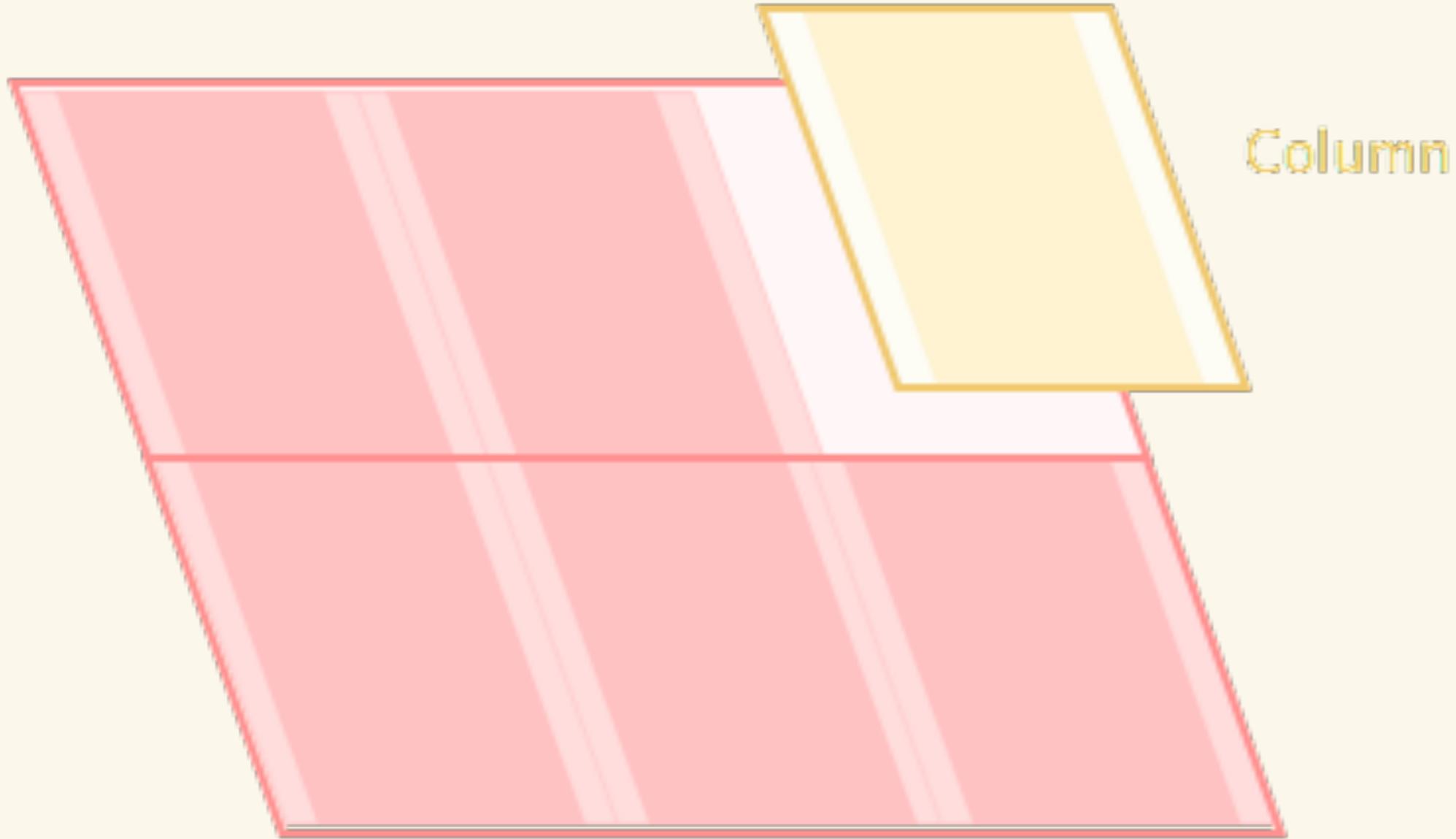
ROW

The purpose of the row element is to keep the columns inside it from overflowing onto other rows. To achieve this, we'll use the clearfix hack to make sure everything inside the row stays inside the row.



```
/*-- our cleafix hack -- */  
.row:before,  
.row:after {  
  content:"";  
  display: table ;  
  clear:both;  
}
```

THE COLUMN



Floats, inline-blocks, display-table, display-flex. These are all different ways of positioning columns in CSS. From my personal experience, the least error prone and most widely used of these methods is the 'float' method. If our columns are empty however, our floated columns will stack on top of each other. To prevent this, we'll give our columns a minimum height of 1px as well as float them.

```
[class*='col-'] {  
  float: left;  
  min-height: 1px;  
}
```

COLUMN WIDTHS

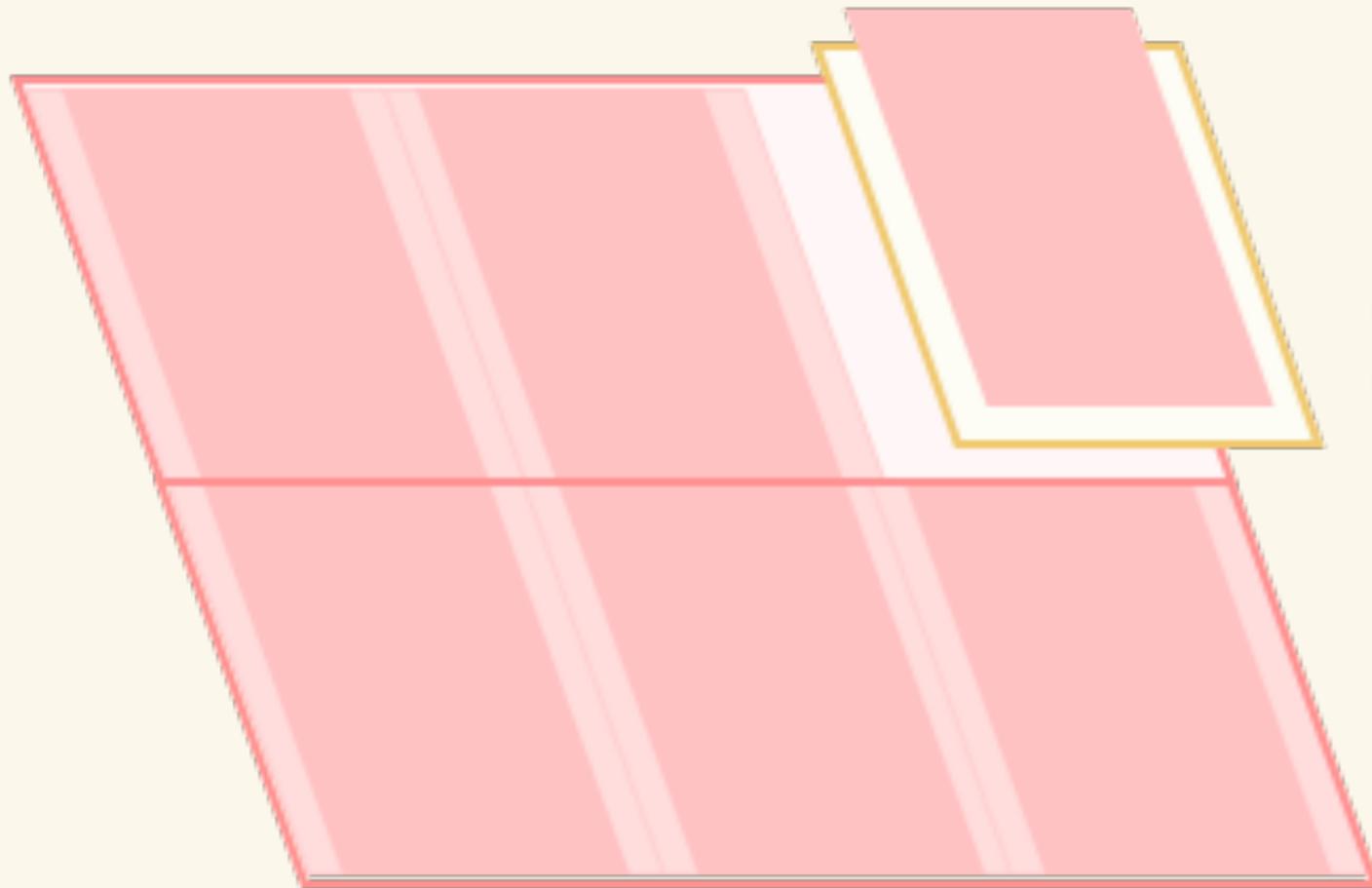
```
[class*='col-'] {  
  float: left;  
  min-height: 1px;  
  width: 16.66%;  
}
```

To find the width of one column, all we have to do is divide the total number of columns by the width of the container. In our case, the width of the container is 100%, and we want 6 columns, so $100/6 = 16.66$, so our base column width is 16.66%.

If we want a section that's 2 columns wide, we have to create an 2-column-wide column.

```
.col-1{  
  width: 16.66%;  
}  
.col-2{  
  width: 33.33%;  
}  
.col-3{  
  width: 50%;  
}  
.col-4{  
  width: 66.66%;  
}  
.col-5{  
  width: 83.33%;  
}  
.col-6{  
  width: 100%;  
}
```

COLUMN GUTTERS



Column
Gutters

Before the 'border-box' box-sizing model, giving percentage-width elements a static padding was a real pain. Luckily, using the 'border-box' model, we can create gutters with ease.

```
/*-- setting border box on all
elements inside the grid --*/
.grid-container *{
  box-sizing: border-box;
}

[class*='col-'] {
  float: left;
  min-height: 1px;
  width: 16.66%;
  /*-- our gutter --*/
  padding: 12px;
}
```

HTML

```
<div class="row">
<div class="col-1"><p>col-1</p></div>
<div class="col-1"><p>col-1</p></div>
<div class="col-1"><p>col-1</p></div>
<div class="col-1"><p>col-1</p></div>
<div class="col-1"><p>col-1</p></div>
<div class="col-1"><p>col-1</p></div>
</div>
<div class="row">
<div class="col-2"><p>col-2</p></div>
<div class="col-2"><p>col-2</p></div>
<div class="col-2"><p>col-2</p></div>
</div>
<div class="row">
<div class="col-3"><p>col-3</p></div>
<div class="col-3"><p>col-3</p></div>
</div>
</div>
```

CSS

```
grid-container{
  width: 100%;
  max-width: 1200px;
}

/*-- our cleafix hack -- */
.row:before,
.row:after {
  content: "";
  display: table ;
  clear: both;
}

[class*='col-'] {
  float: left;
  min-height: 1px;
  width: 16.66%;
  /*-- our gutter -- */
  padding: 12px;
  background-color: #FFDCDC;
}

.col-1{ width: 16.66%; }
.col-2{ width: 33.33%; }
.col-3{ width: 50%; }
.col-4{ width: 66.66%; }
.col-5{ width: 83.33%; }
.col-6{ width: 100%; }

.outline, .outline *{
  outline: 1px solid #F6A1A1;
}
```

MAKE THE GRID RESPONSIVE

```
<div class="grid-container outline">
  <div class="row">
    <div class="col-1"><p>col-1</p></div>
    <div class="col-1"><p>col-1</p></div>
    <div class="col-1"><p>col-1</p></div>
    <div class="col-1"><p>col-1</p></div>
    <div class="col-1"><p>col-1</p></div>
    <div class="col-1"><p>col-1</p></div>
  </div>
  <div class="row">
    <div class="col-2"><p>col-2</p></div>
    <div class="col-2"><p>col-2</p></div>
    <div class="col-2"><p>col-2</p></div>
  </div>
  <div class="row">
    <div class="col-3"><p>col-3</p></div>
    <div class="col-3"><p>col-3</p></div>
  </div>
</div>
<hr/>
```

Adjusting our grid for mobile layouts is pretty easy. All we have to do is adjust the widths of the columns.

Let's double the widths of the columns for screens under 800px.

```
@media all and (max-width:800px){
```

```
  .col-1{ width: 33.33%; }
```

```
  .col-2{ width: 50%; }
```

```
  .col-3{ width: 83.33%; }
```

```
  .col-4{ width: 100%; }
```

```
  .col-5{ width: 100%; }
```

```
  .col-6{ width: 100%; }
```

```
  .row .col-2:last-of-type{  
    width: 100%;  
  }
```

```
  .row .col-5 ~ .col-1{  
    width: 100%;  
  }  
}
```

The only thing to watch out for is a few exceptions where the last column in the row hangs off the end. Such as in the case of the .col-2 columns and the .col-1 beside the .col-5 column. To counter this, we'll make the last .col-2 and .col-1 in the row 100% wide.

For screens that are even smaller than 800px, we'll make all the columns except the very smallest 100%.

```
@media all and (max-width:650px){  
  .col-1{ width: 50%; }  
  .col-2{ width: 100%; }  
  .col-3{ width: 100%; }  
  .col-4{ width: 100%; }  
  .col-5{ width: 100%; }  
  .col-6{ width: 100%; }  
}
```

```
<div class="grid-container outline">  
  <div class="row">  
    <div class="col-1"><p>col-1</p></div>  
    <div class="col-1"><p>col-1</p></div>  
    <div class="col-1"><p>col-1</p></div>  
    <div class="col-1"><p>col-1</p></div>  
    <div class="col-1"><p>col-1</p></div>  
    <div class="col-1"><p>col-1</p></div>  
  </div>  
  <div class="row">  
    <div class="col-2"><p>col-2</p></div>  
    <div class="col-2"><p>col-2</p></div>  
    <div class="col-2"><p>col-2</p></div>  
  </div>  
  <div class="row">  
    <div class="col-3"><p>col-3</p></div>  
    <div class="col-3"><p>col-3</p></div>  
  </div>  
  <div class="row">  
    <div class="col-4"><p>col-4</p></div>  
    <div class="col-2"><p>col-2</p></div>  
  </div>  
  <div class="row">  
    <div class="col-5"><p>col-5</p></div>  
    <div class="col-1"><p>col-1</p></div>  
  </div>  
  <div class="row">  
    <div class="col-6"><p>col-6</p></div>  
  </div>  
</div>
```

MIN-WIDTH

A mobile-first approach to styling means that styles are applied first to mobile devices. Advanced styles and other overrides for larger screens are then added into the stylesheet via media queries. This approach uses `min-width` media queries.

```
// This applies from 0px to 600px
body {
  background: red;
}

// This applies from 600px onwards
@media (min-width: 600px) {
  body {
    background: green;
  }
}
```

MAX-WIDTH

On the flipside, a desktop-first approach to styling means that styles are applied first to desktop devices. Advanced styles and overrides for smaller screens are then added into the stylesheet via media queries.

This approach uses max-width media queries.

```
// This applies from 600px onwards
body {
  background: green;
}

// This applies from 0px to 600px
@media (max-width: 600px) {
  body {
    background: red;
  }
}
```

references

HTML GRID SYSTEM

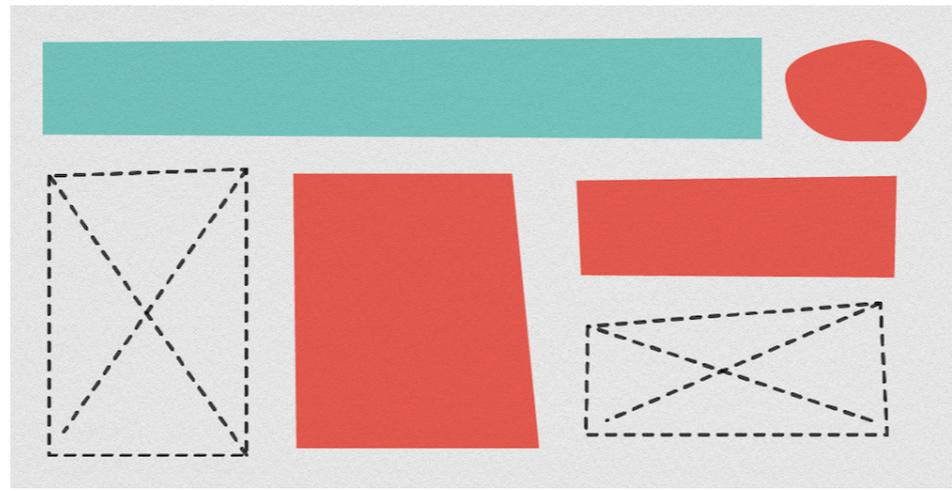
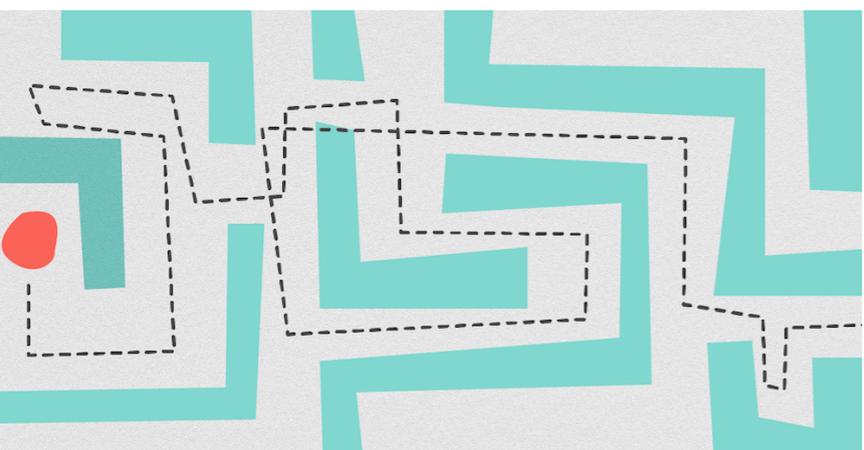
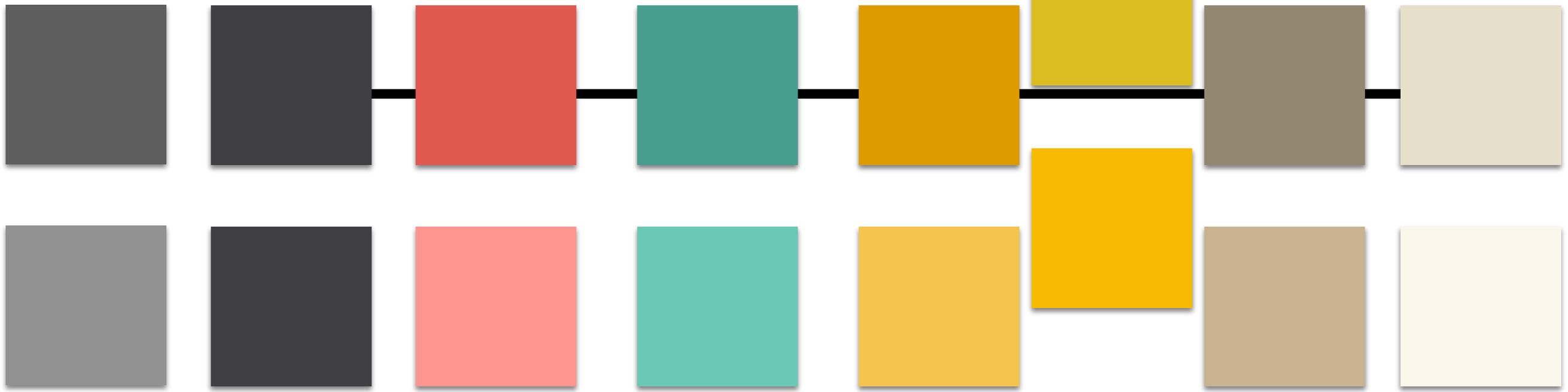
<http://j4n.co/blog/Creating-your-own-css-grid-system>

<https://zellwk.com/blog/from-html-grids-to-css-grids/>

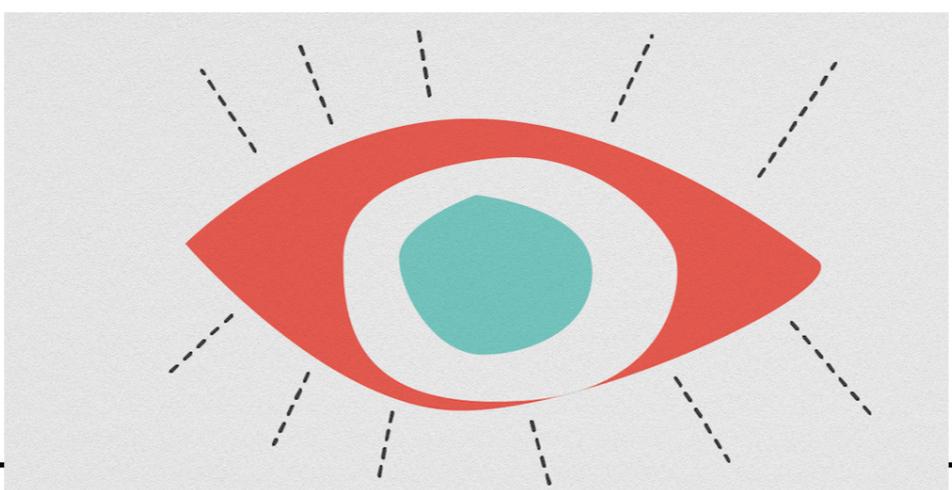
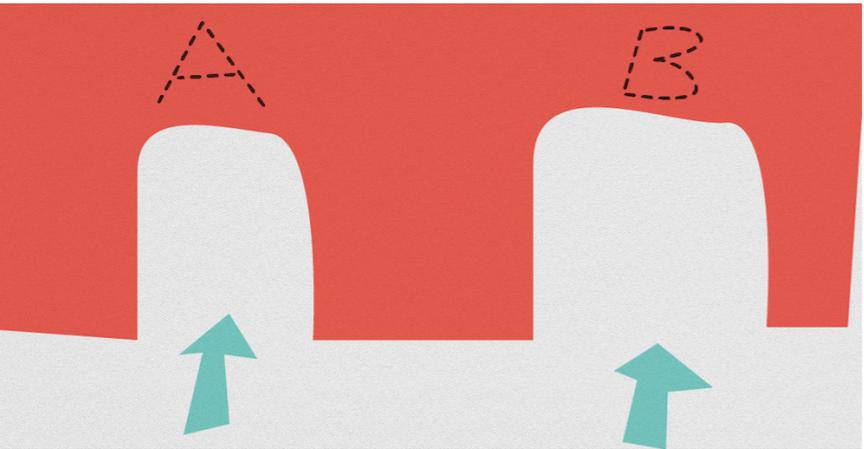
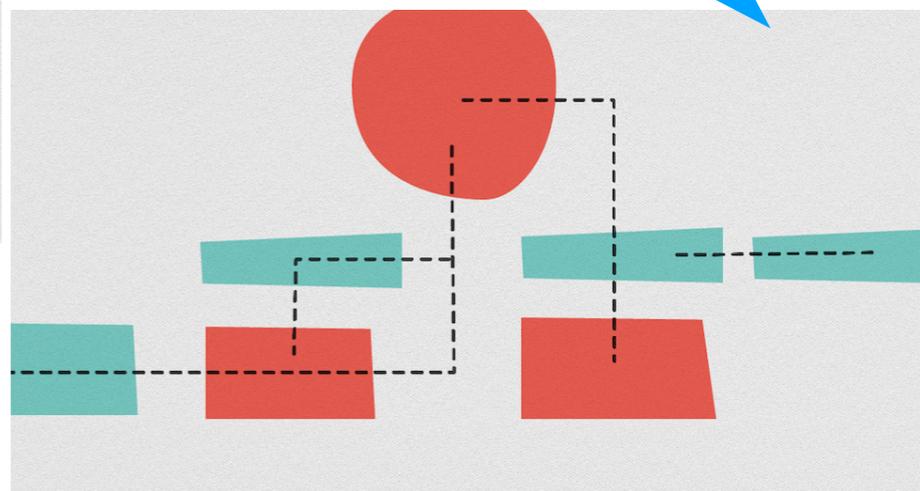
CSS GRID SYSTEM

<https://zellwk.com/blog/responsive-grid-system/>

<https://www.digitalocean.com/community/tutorials/an-introduction-to-jquery>



Tip 1



Rule of Thumb Across Most Browsers:

1EM = 16PX = 100%

Is the default

vertical space between lines of type

LEADING

leading

print

line-height

css code

leading

Design Elements

POINT | LINE | SHAPE | TONE |
TEXTURE | FORM | COLOR |

color terms

hue | saturation | value | contrast

Instructor: Angela Wyman

typography
rule
of three
typefaces

Instructor: Angela Wyman

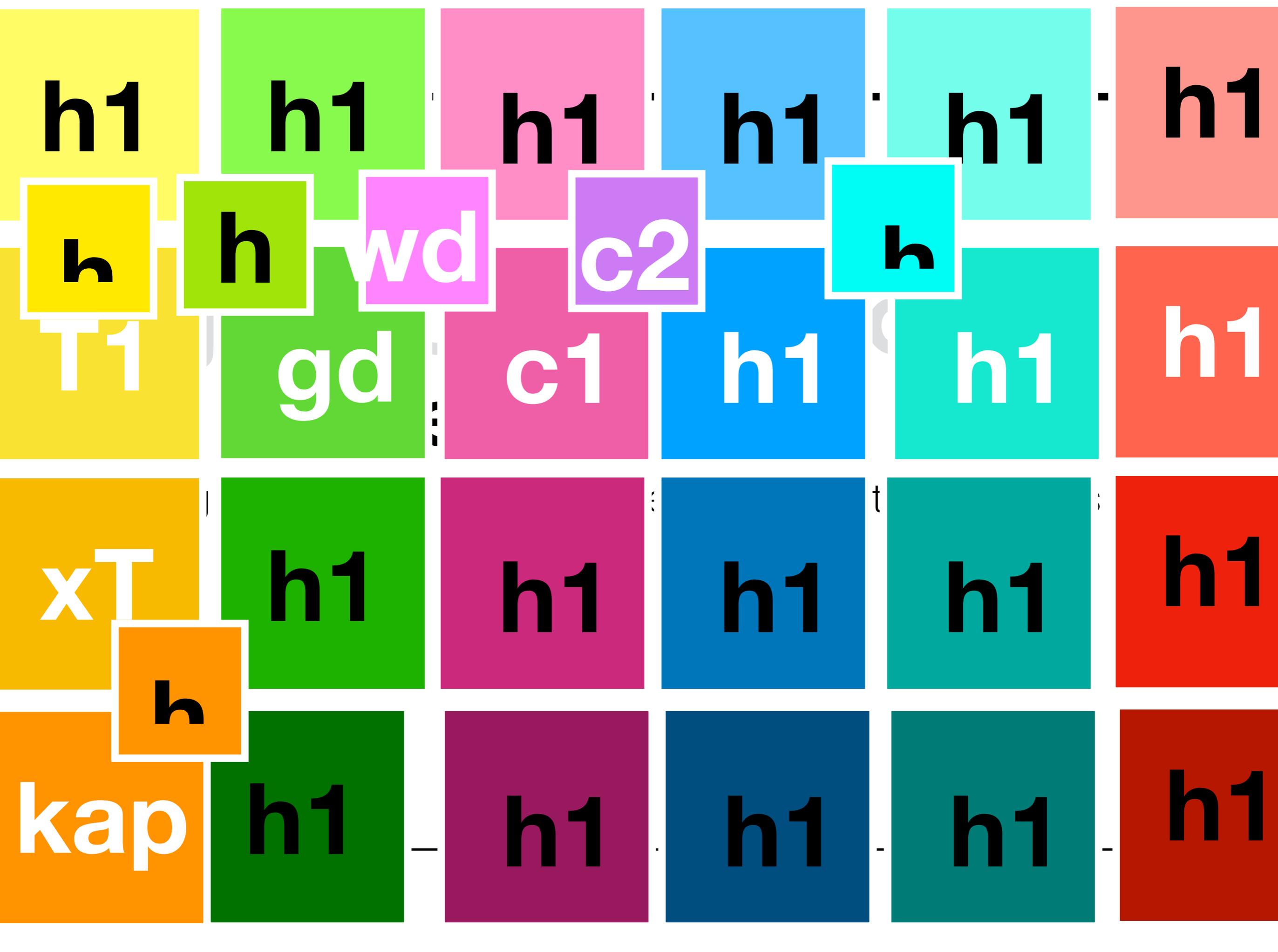
typography

terms

the hyphen, en
dash & em dash

quiz

Instructor: Angela Wyman



h1

h1

h1

h1

h1

h1

h

h

wd

c2

h

T1

gd

c1

h1

h1

h1

xT

h1

h1

h1

h1

h1

h

kap

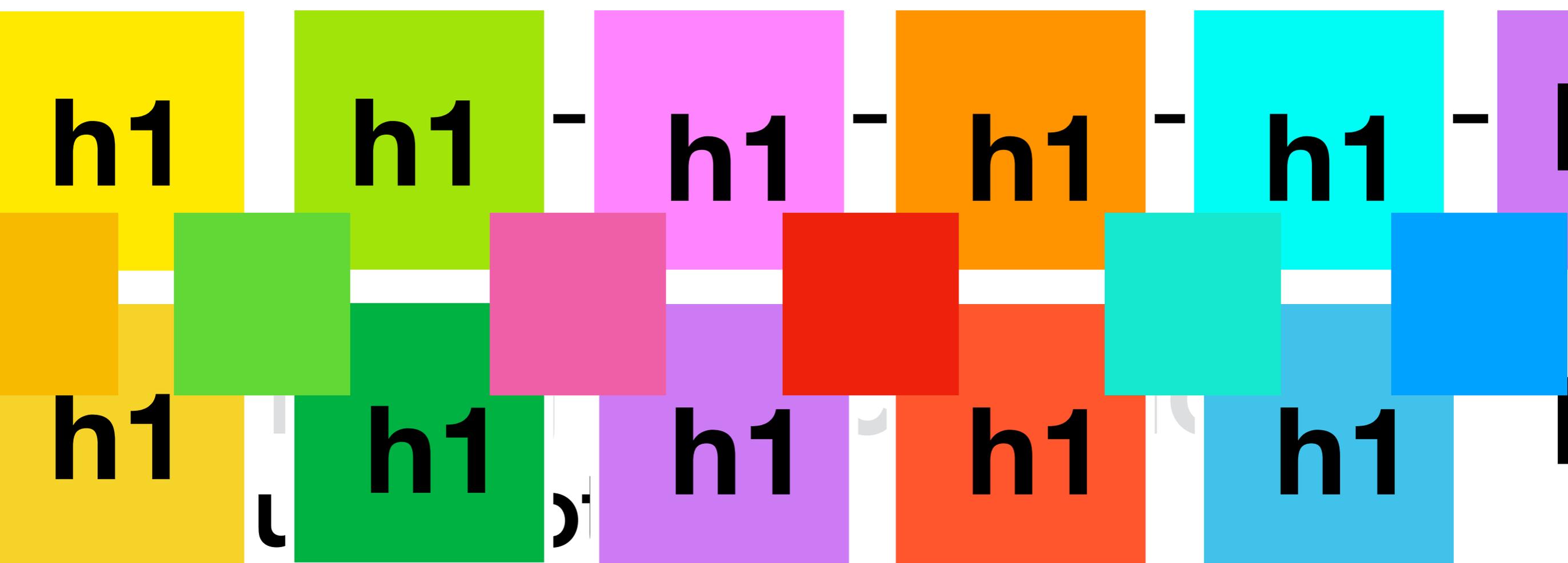
h1

h1

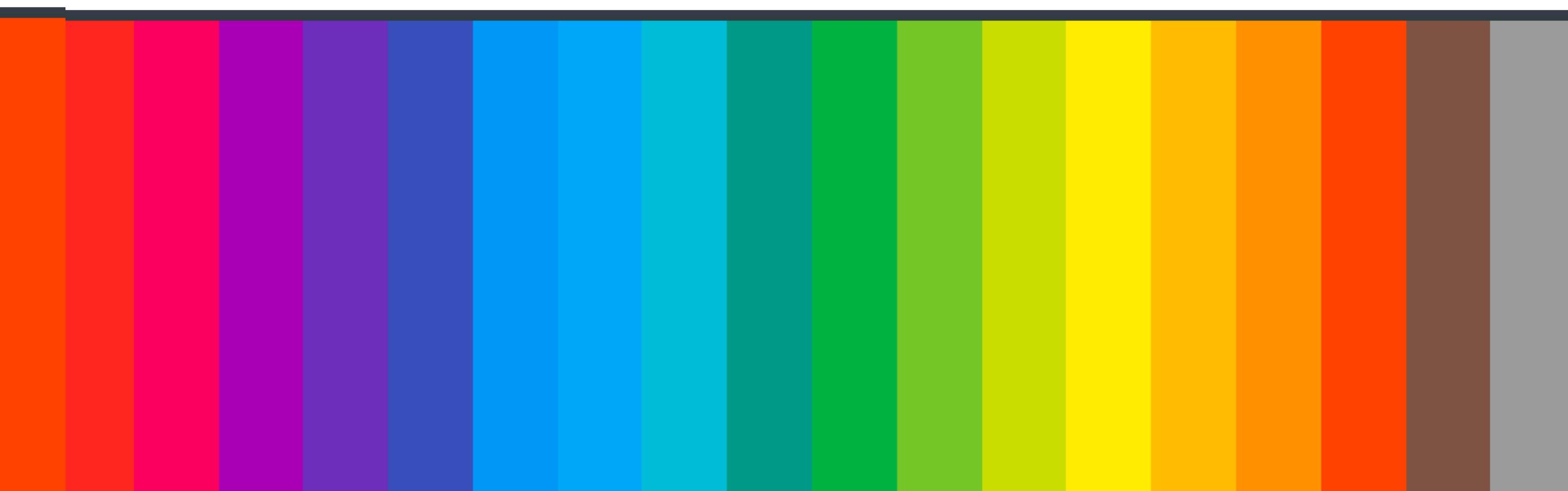
h1

h1

h1



With hanging punctuation the flow of text on the left hand side is





SUB HEADING

SUB HEADING
aside text

SUB HEADING
Aside text

SUB HEADING
Paragraph text
• Bullet
• Text

no-
nos



Intro to C2

Digital Asset
& File
Supply
Images

