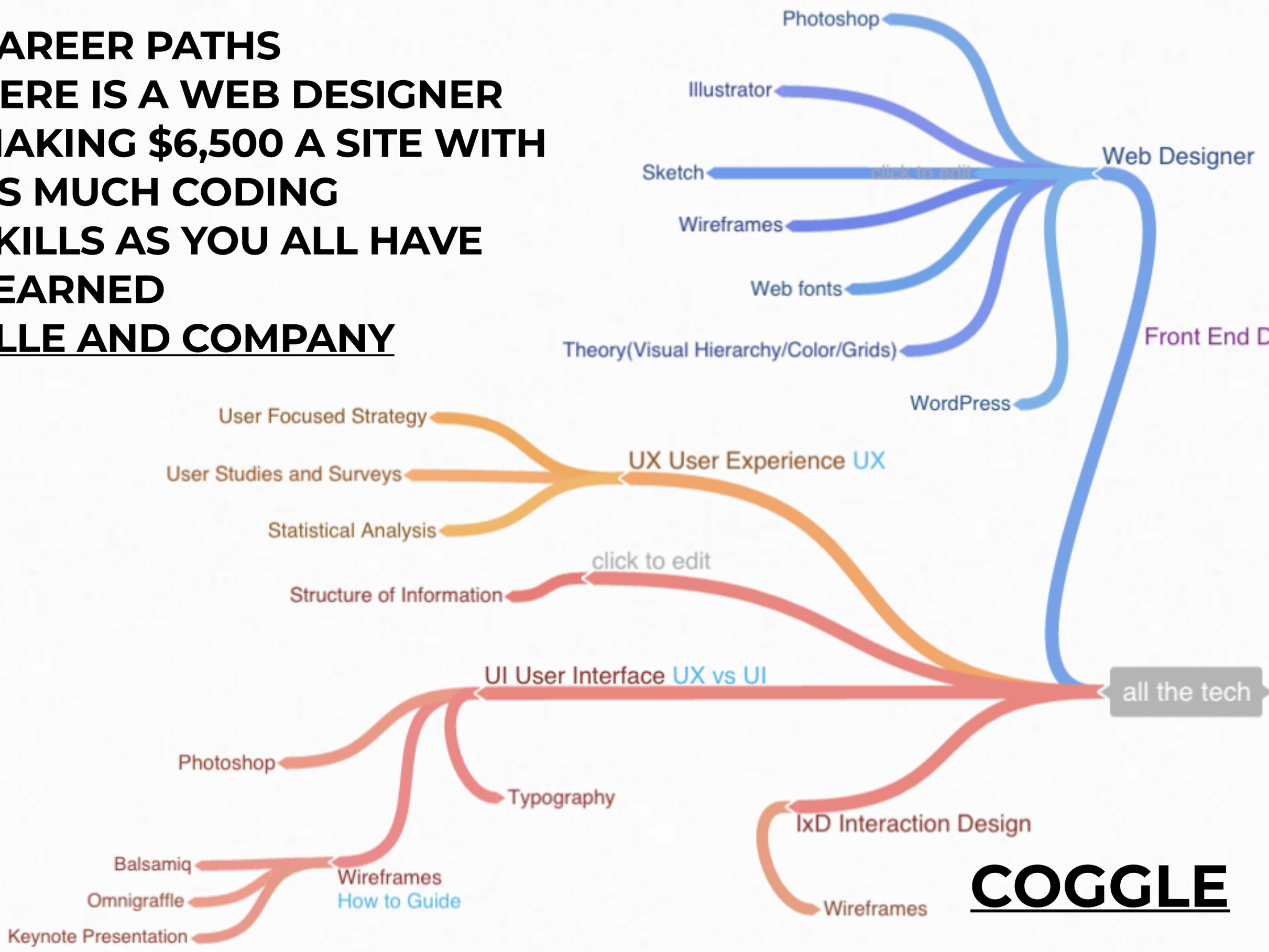
CAREER PATHS
HERE IS A WEB DESIGNER MAKING $6,500 A SITE WITH AS MUCH CODING SKILLS AS YOU ALL HAVE LEARNED
ELLE AND COMPANY



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 will 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.

WHAT IS A DESIGNER TO DO...

MAKE A SITE FOR EVERY DEVICE?

listapart.com/article/responsive-web-design

COLUMNS BLOG EVENTS TOPICS WRITE FOR US

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

Issue № 306

Responsive Web Design

by **Ethan Marcotte** · May 25, 2010

Published in CSS, Layout & Grids, Mobile/Multidevice, Responsive Design, Interaction Design

"The control which designers know in the print medium, and often desire in the web medium, is simply a function of the limitation of the printed page. We should embrace the fact that the web doesn't have the same constraints, and design for this flexibility. But first, we must 'accept the ebb and flow of things.'"

—John Allsopp, "A Dao of Web Design"

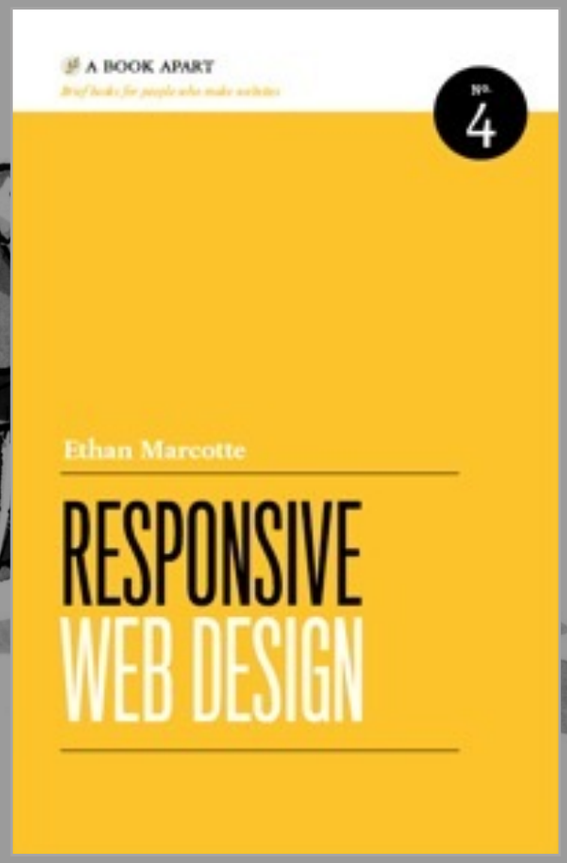


Illustration by Kevin Cornell

Responsive Web Design

by **Ethan Marcotte** · May 25, 2010

Published in CSS, Layout & Grids, Mobile/Multidevice, Responsive Design, Interaction Design

RESPONSIVE DESIGN IS

DESIGNING FOR DEVICES THAT DON'T EVEN EXIST YET.

This Is Responsive.

Patterns and resources for creating responsive web experiences.

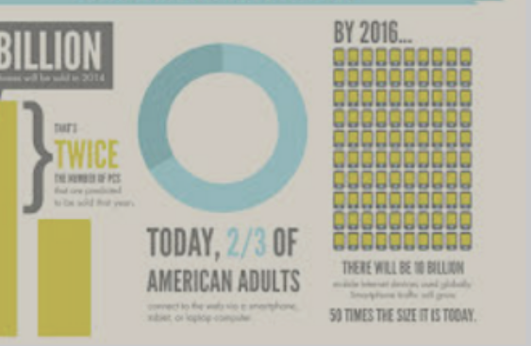
http://bradfrost.github.io/this-is-responsive/index.html

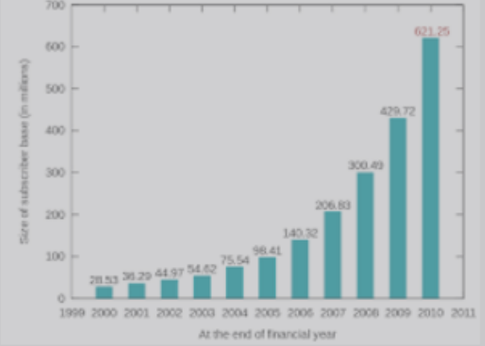

Blackberry:
Curve 8530, Pearl Flip
Android:
Motorola Charm, Xperia X10
Symbian:
Nokia E63, others
320 x 240
Blackberry:
Torch,
Storm,
Bold
480 x 360
Apple:
iPhone, iPod
Android:
HTC Dream, Hero, others
320 x 480
AS IS THIS
Symbian:
Nokia N8, Nokia C6-01
360 x 640
Android:
Liquid A1, HTC Desire, Nexus One, i9000 Galaxy S
Windows:
Sharp S01SH, Venue Pro, Omnia 7, HTC 7 Pro
480 x 800
Apple:
iPhone 4
640 x 960
(320×480 with higher dpi)
Graphics: http://webdev-il.blogspot.com

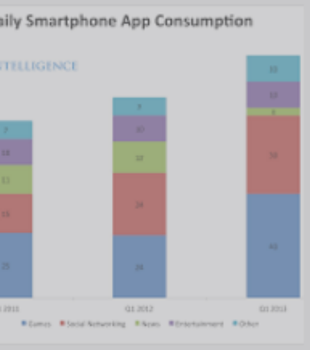
Mobile First

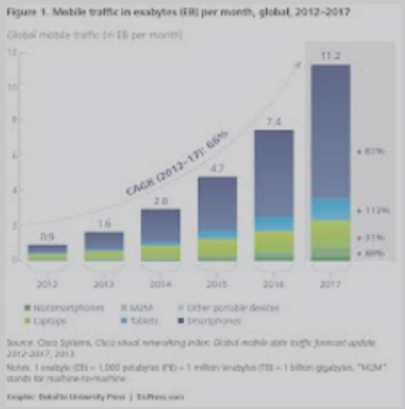
Means start with mobile sizes and layer on styles optimized for larger screens only as needed. In other words, your mobile styles are the default and you won’t have to override them later. — It’s much simpler!

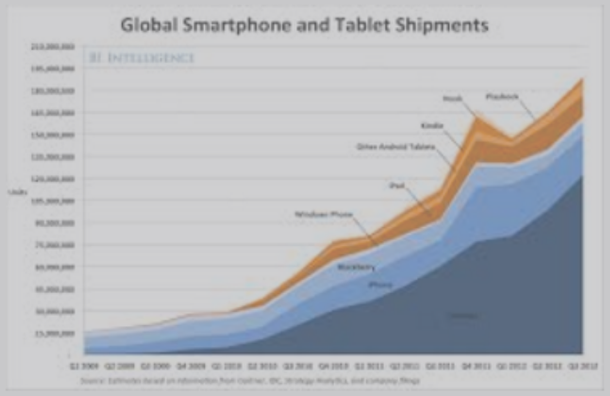
MOBILE IS HUGE

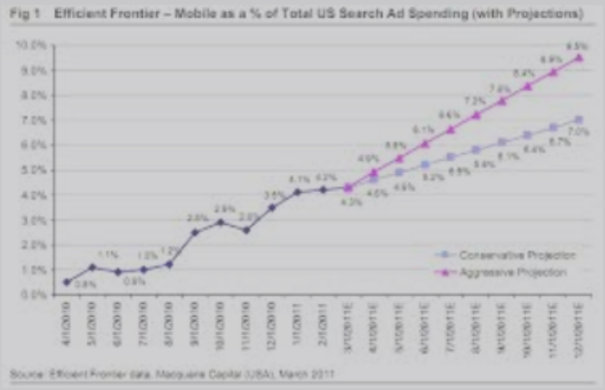


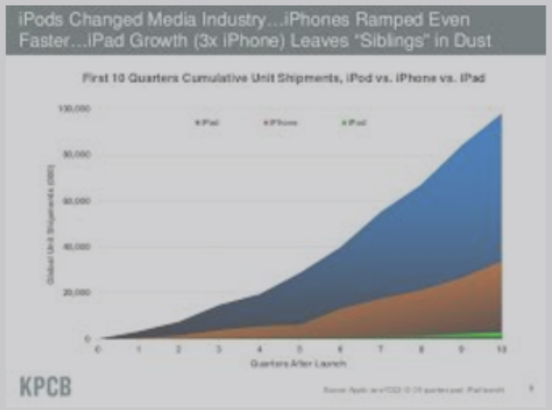


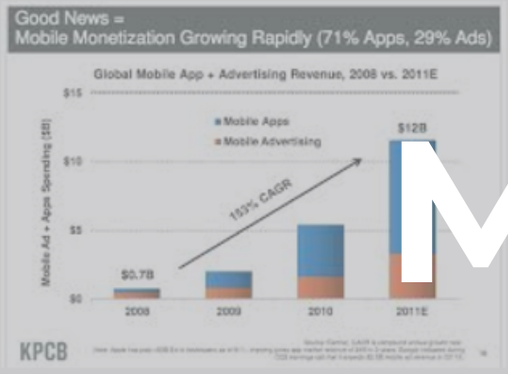


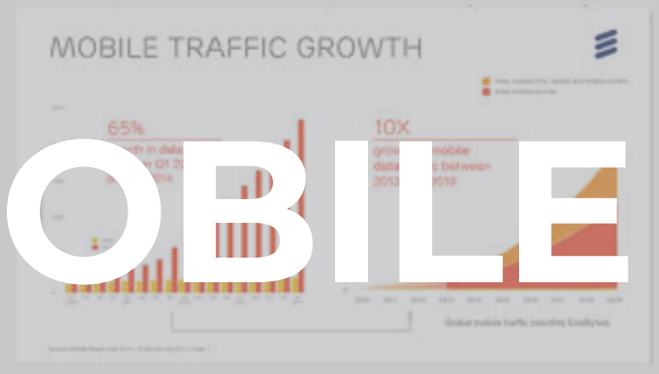


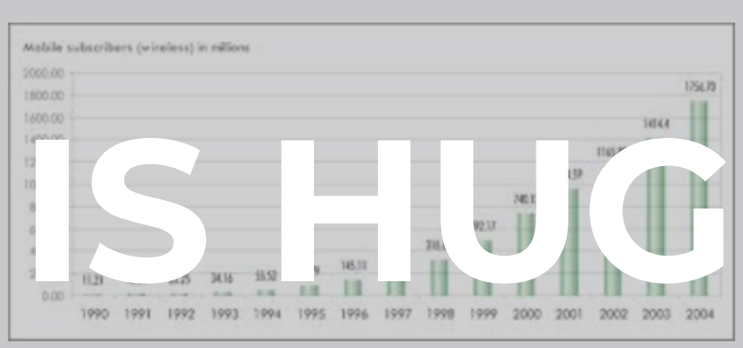


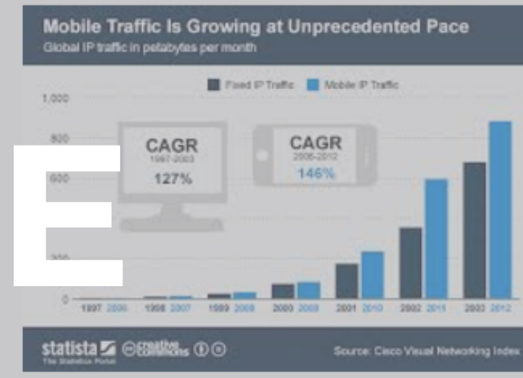


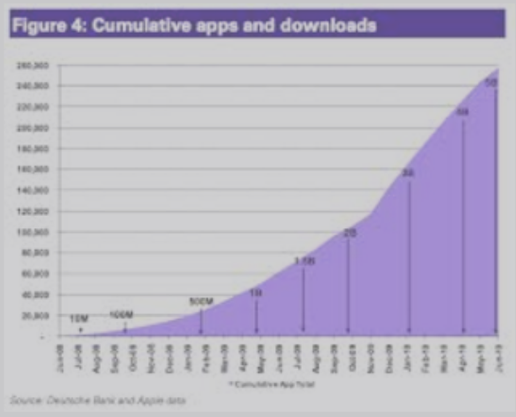


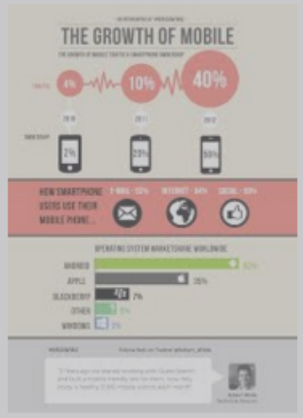


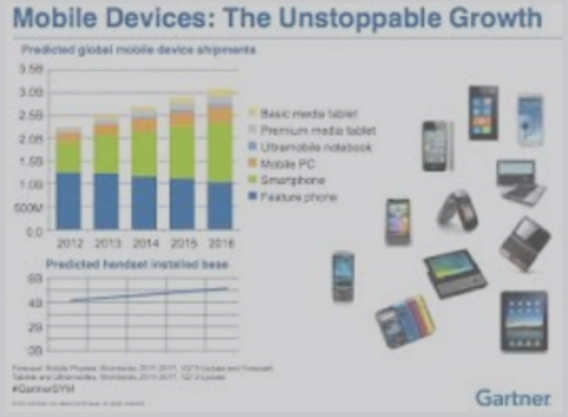


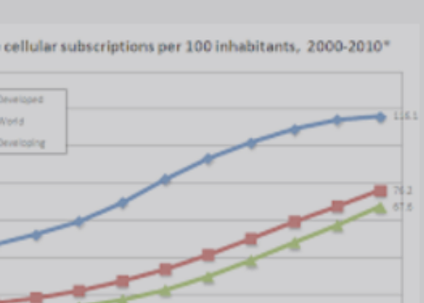


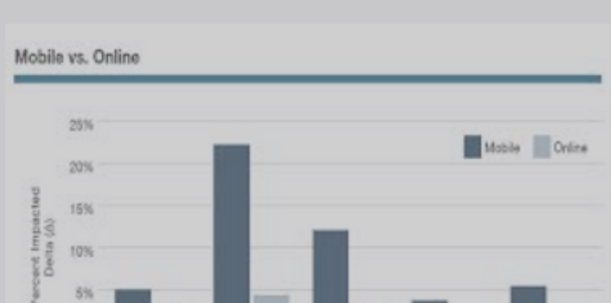


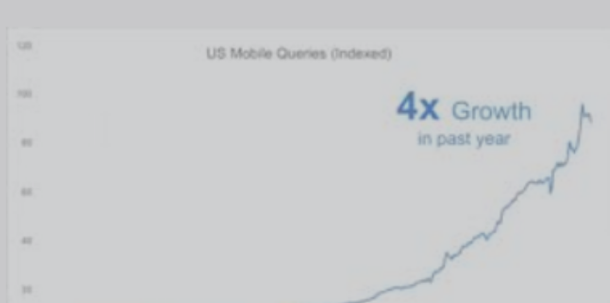


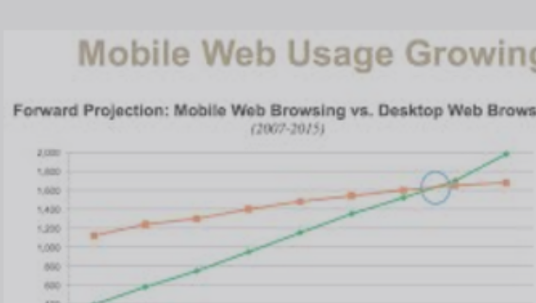




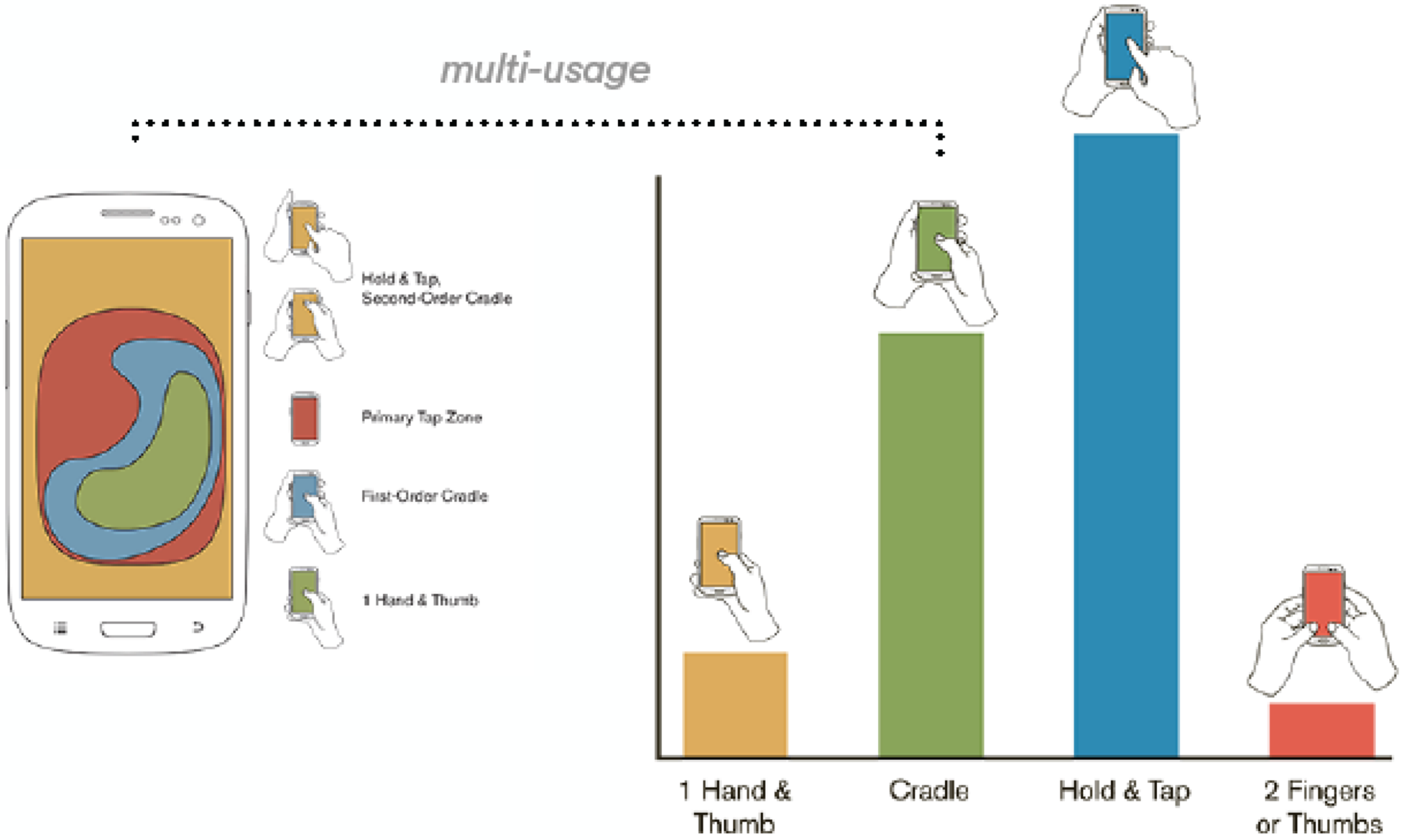








consider how people interact with mobile



WHAT SHOULD WE CONSIDER

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

Mobile UX Checklist

WHAT SHOULD WE CONSIDER

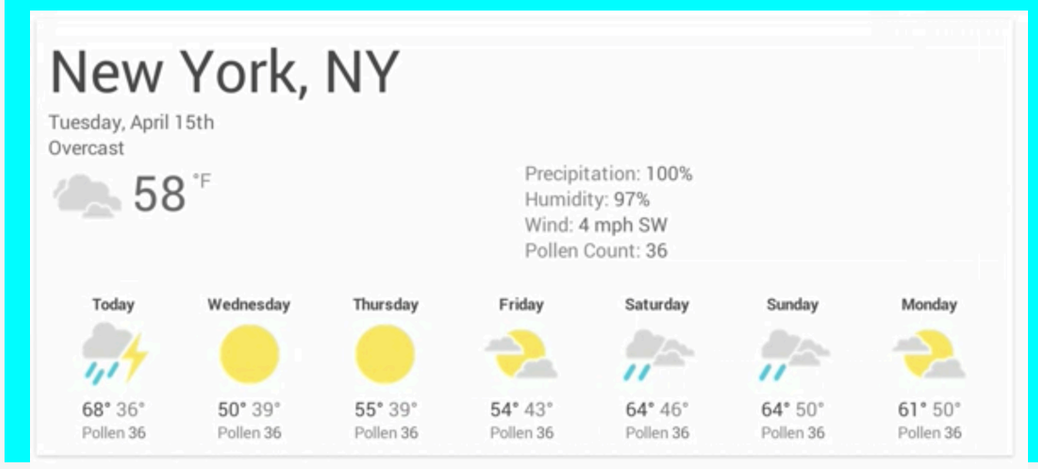
- Average fingertip is between 1cm to 2cm wide, which roughly correlates to somewhere between 44px and 57px on a standard screen and 88px to 114px on a retina screen

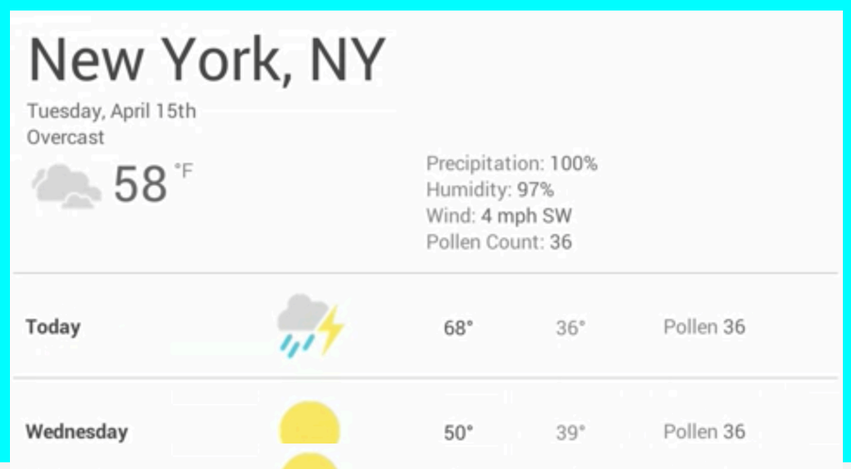
- Mobile sites only have most crucial features
- Vertical navigation is common on more than 90% of mobile sites
- Hyperlinks are replaced by tabs and buttons
- Promotional images are often removed
- No breadcrumbs, progress indicators, persistent navigation or mega footers
- Frequent one-tap phone integration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

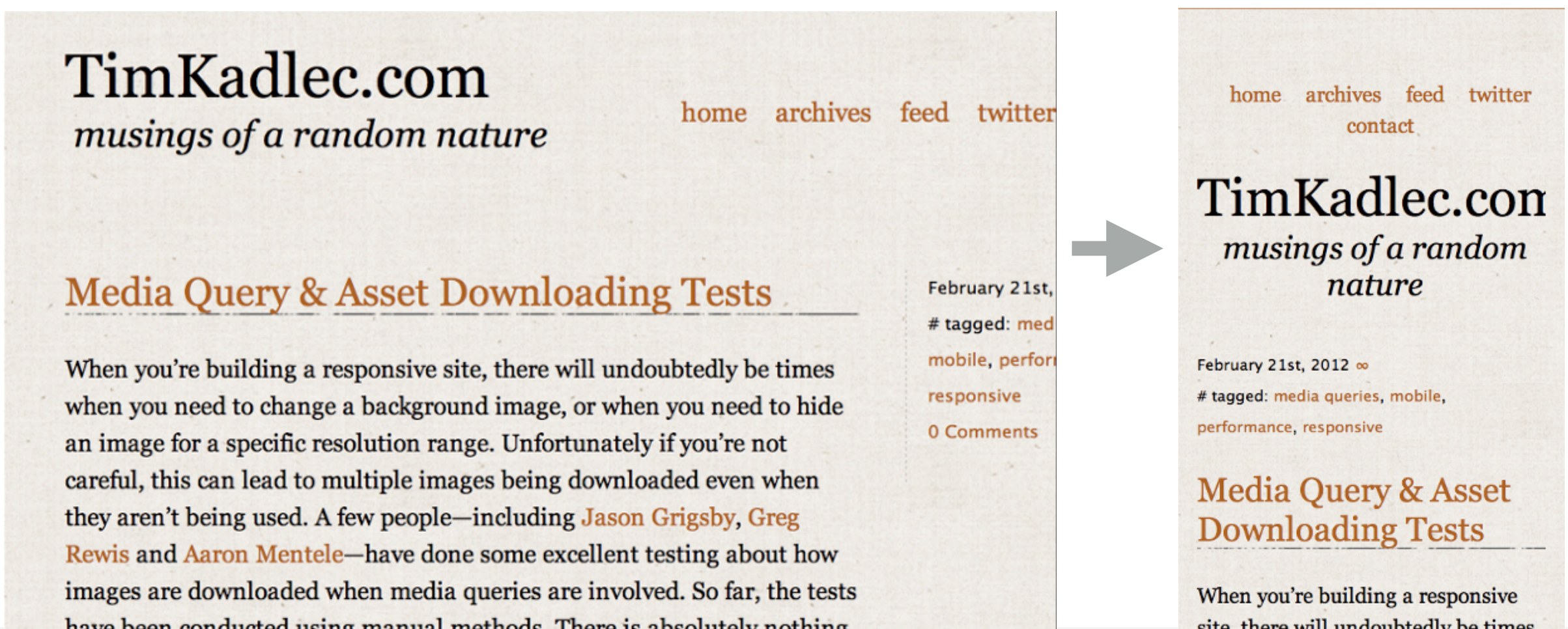
http://simplebits.com/

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

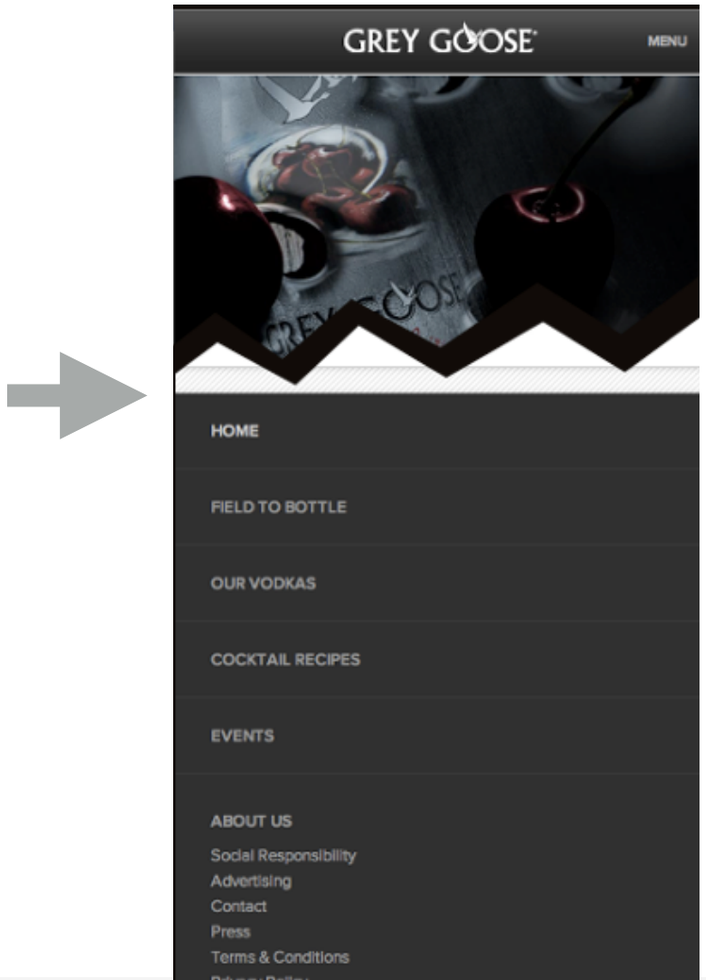


The Footer

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



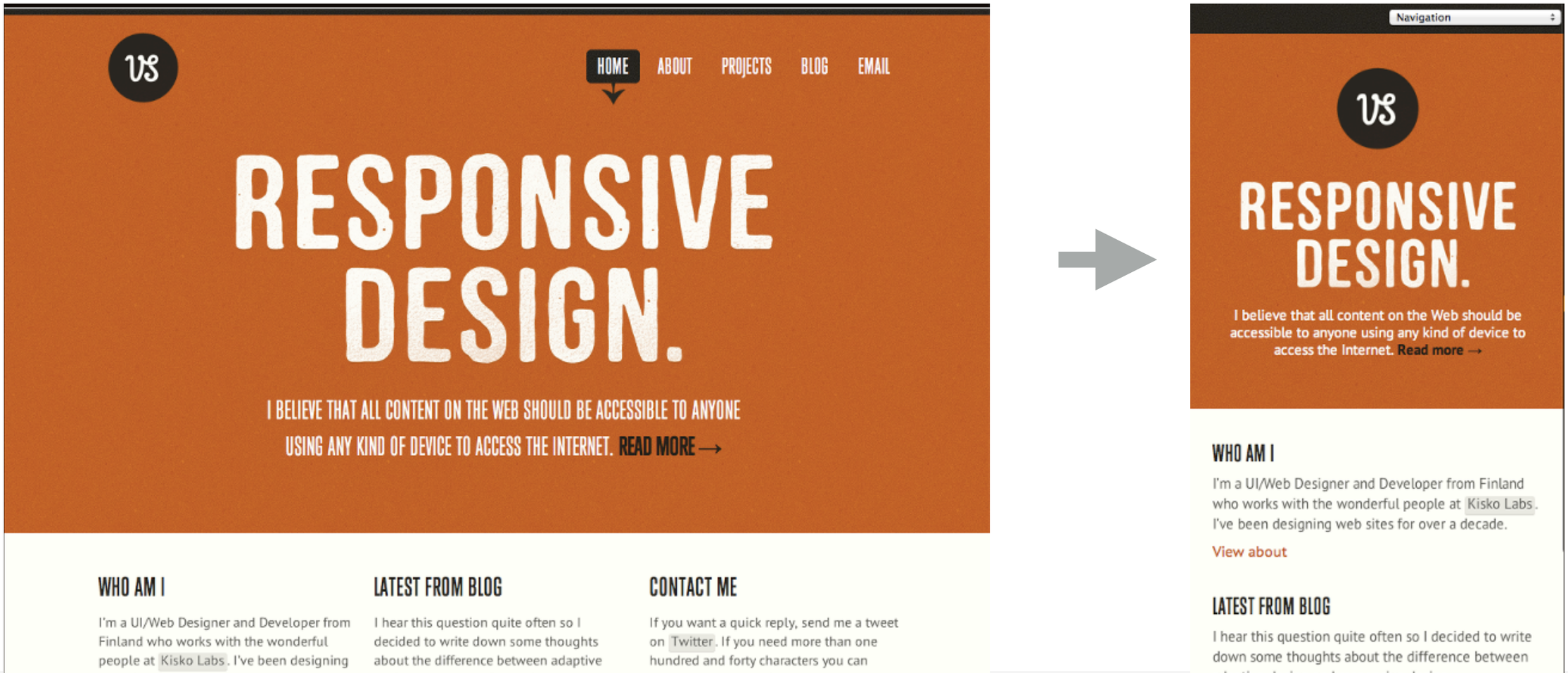
http://tinynav.com/
https://css-tricks.com/convert-menu-to-dropdown/

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

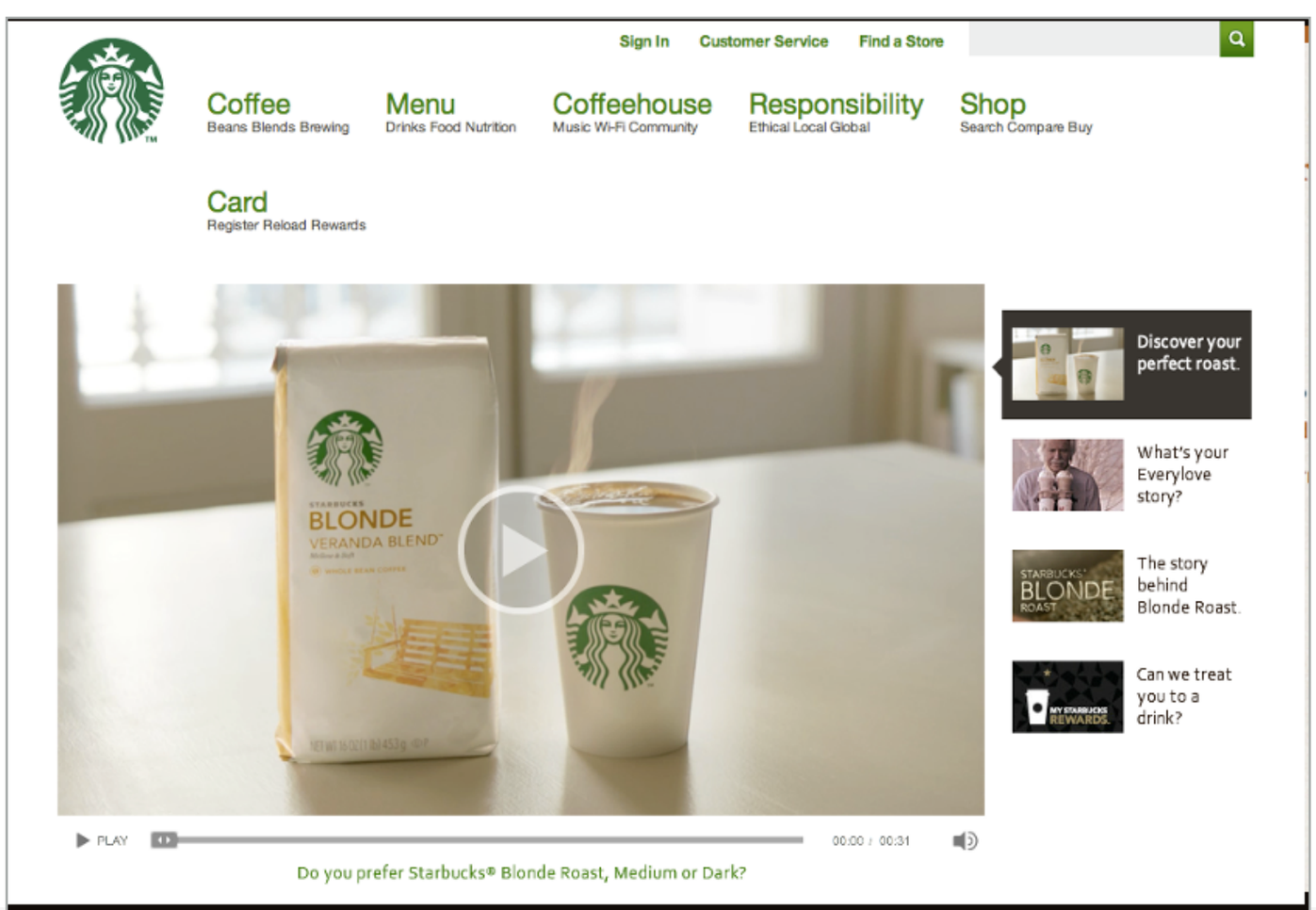
Responsive Design Patterns

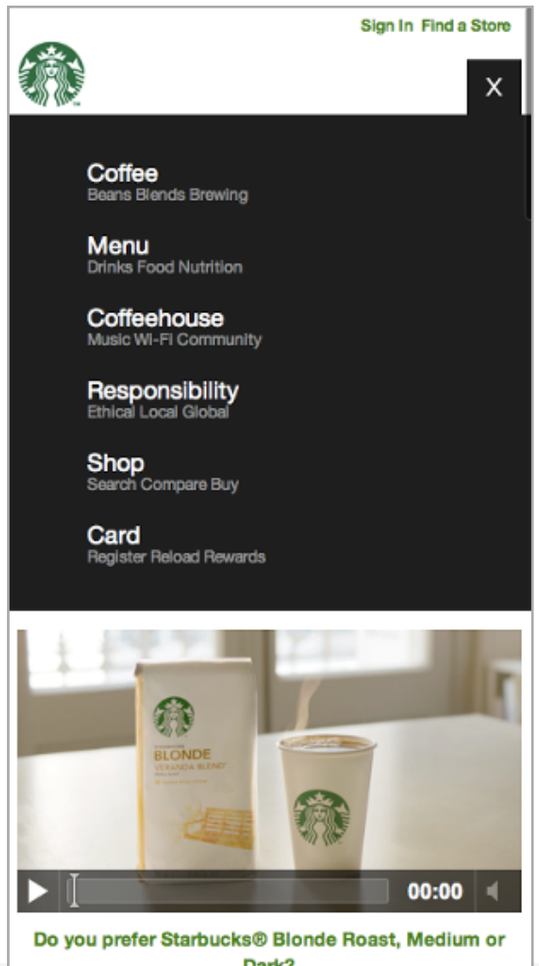


The Toggle

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.



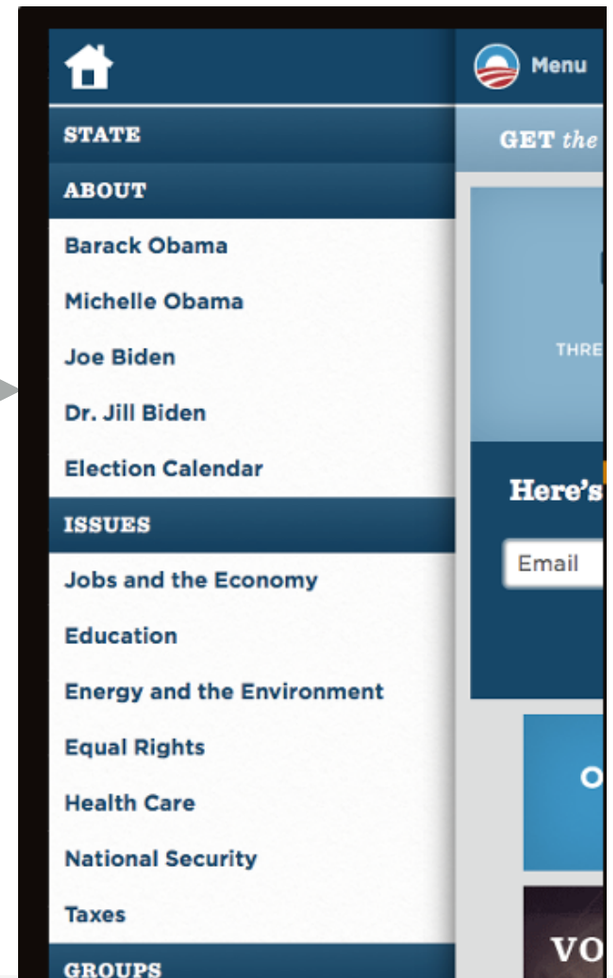


The Left Nav Flyout

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.

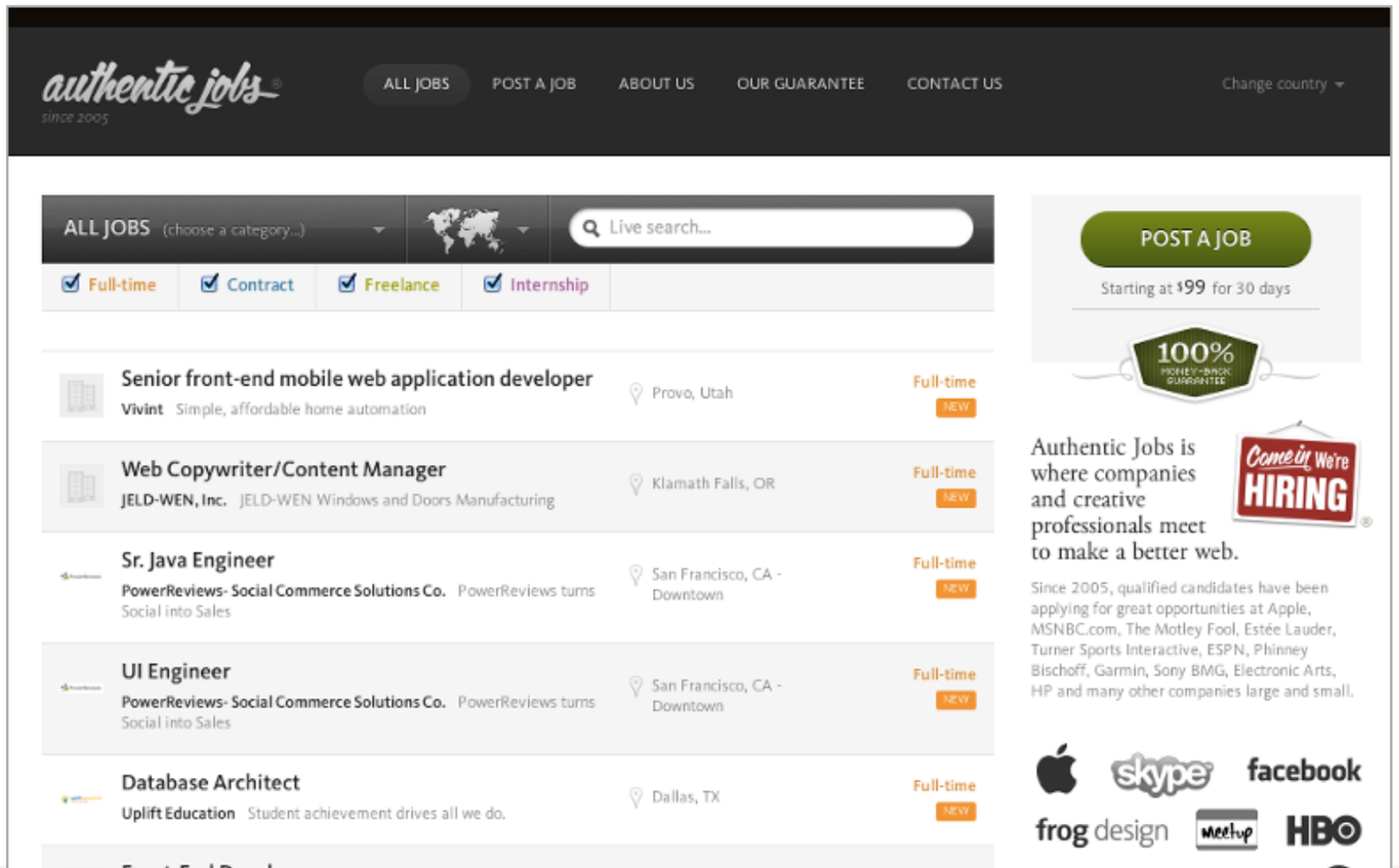
Responsive Design Patterns

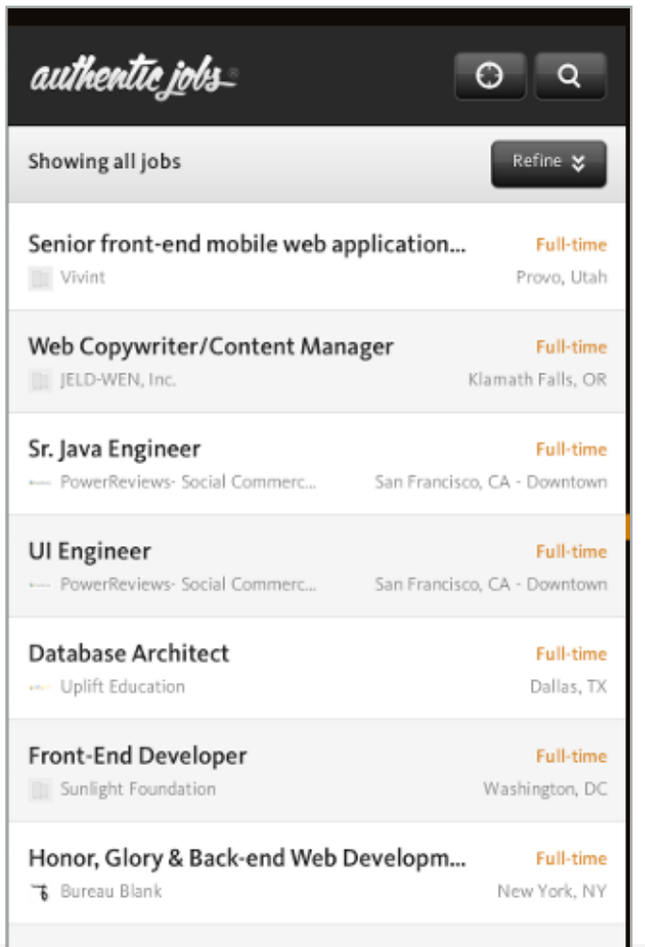


The Hide ’n Cry

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.



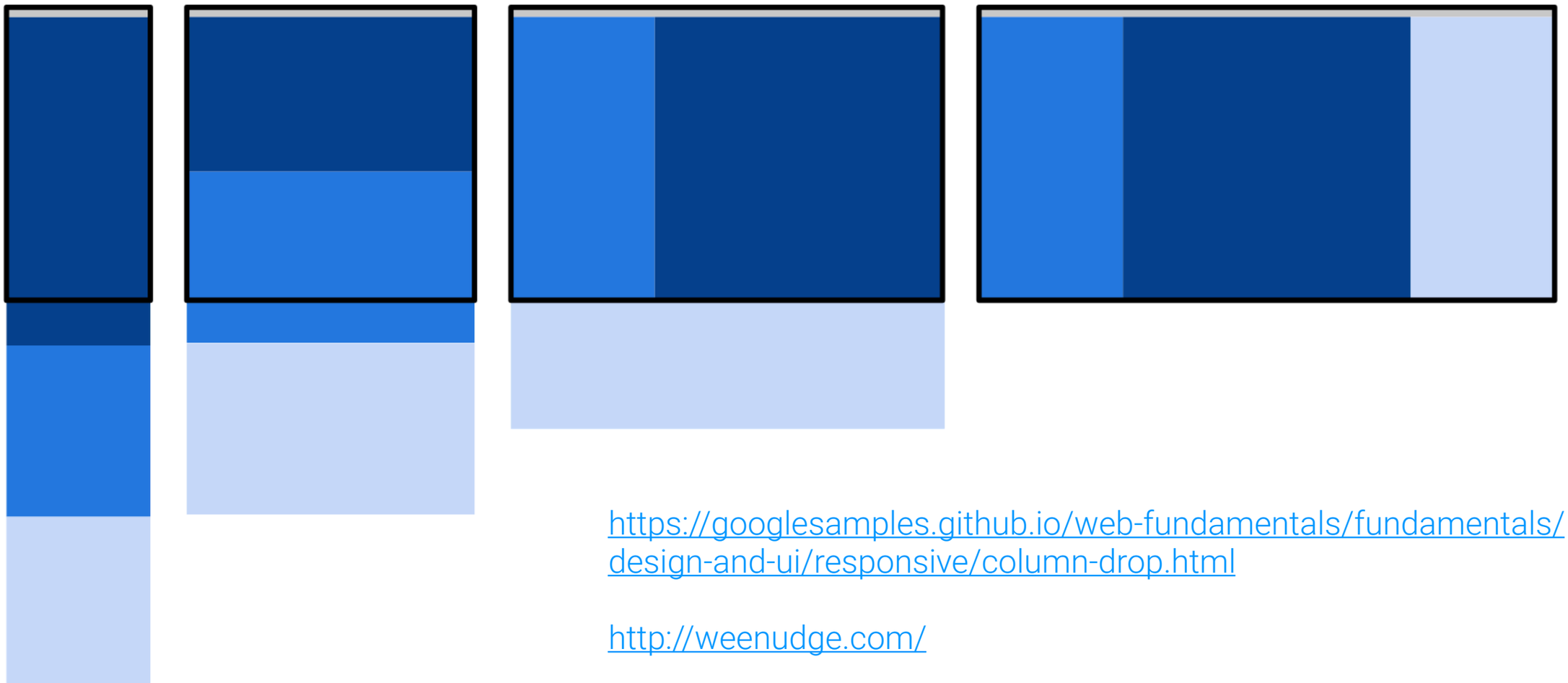


Column Drop

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

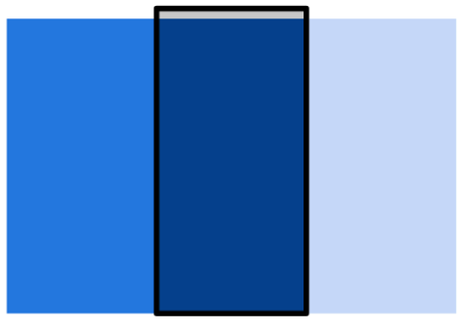
https://googlesamples.github.io/web-fundamentals/fundamentals/design-and-ui/responsive/tiny-tweaks.html

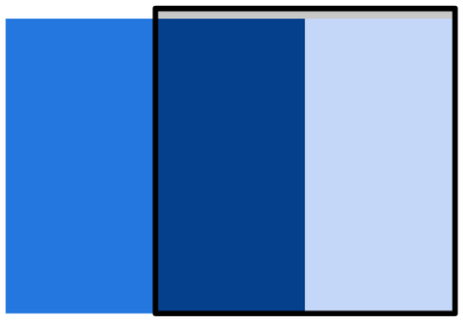
http://futurefriendlyweb.com/

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.

mobile

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 ”

mobile

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

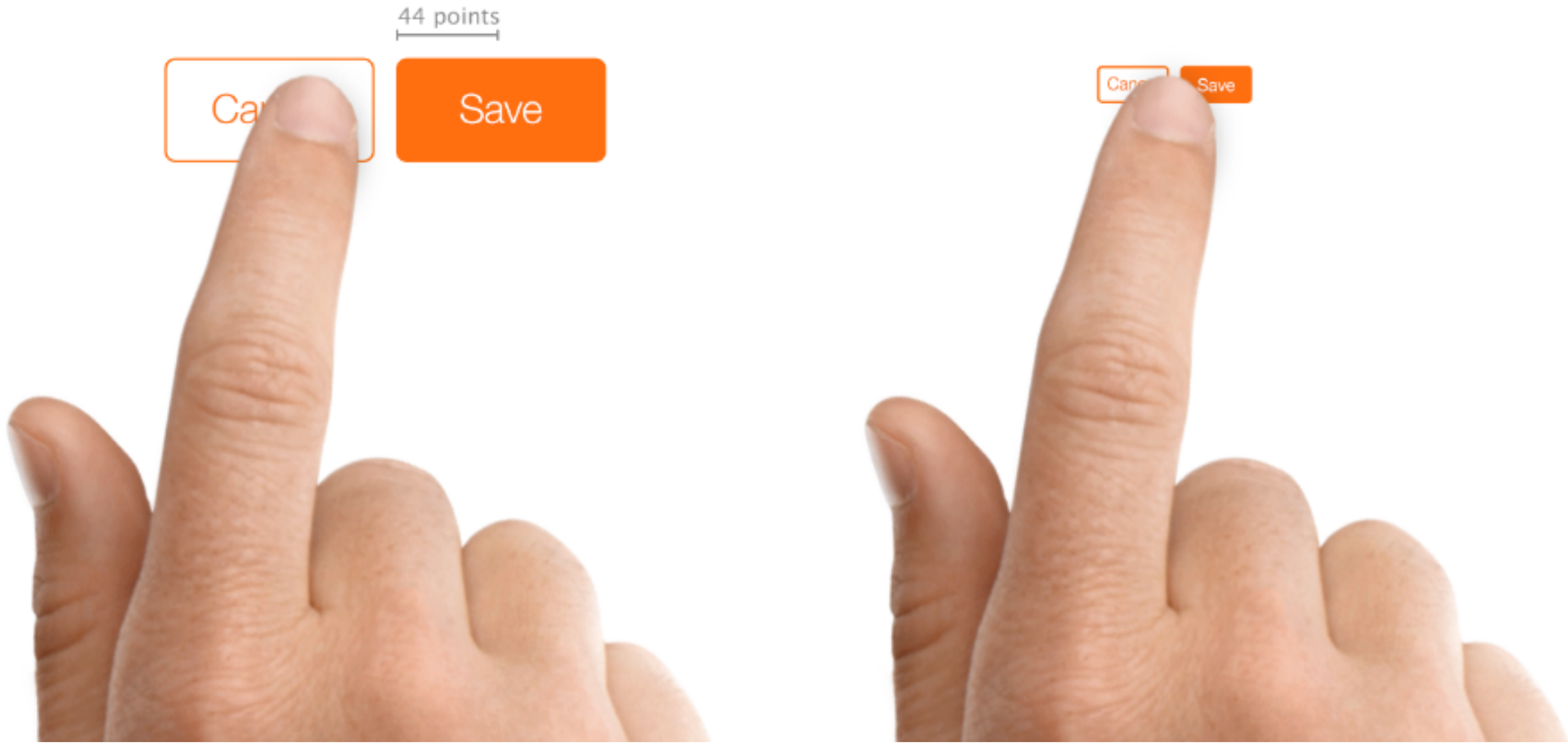
we look at the impact these differences have when developing the structure and functionality

mobile

2.

ergonomics

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

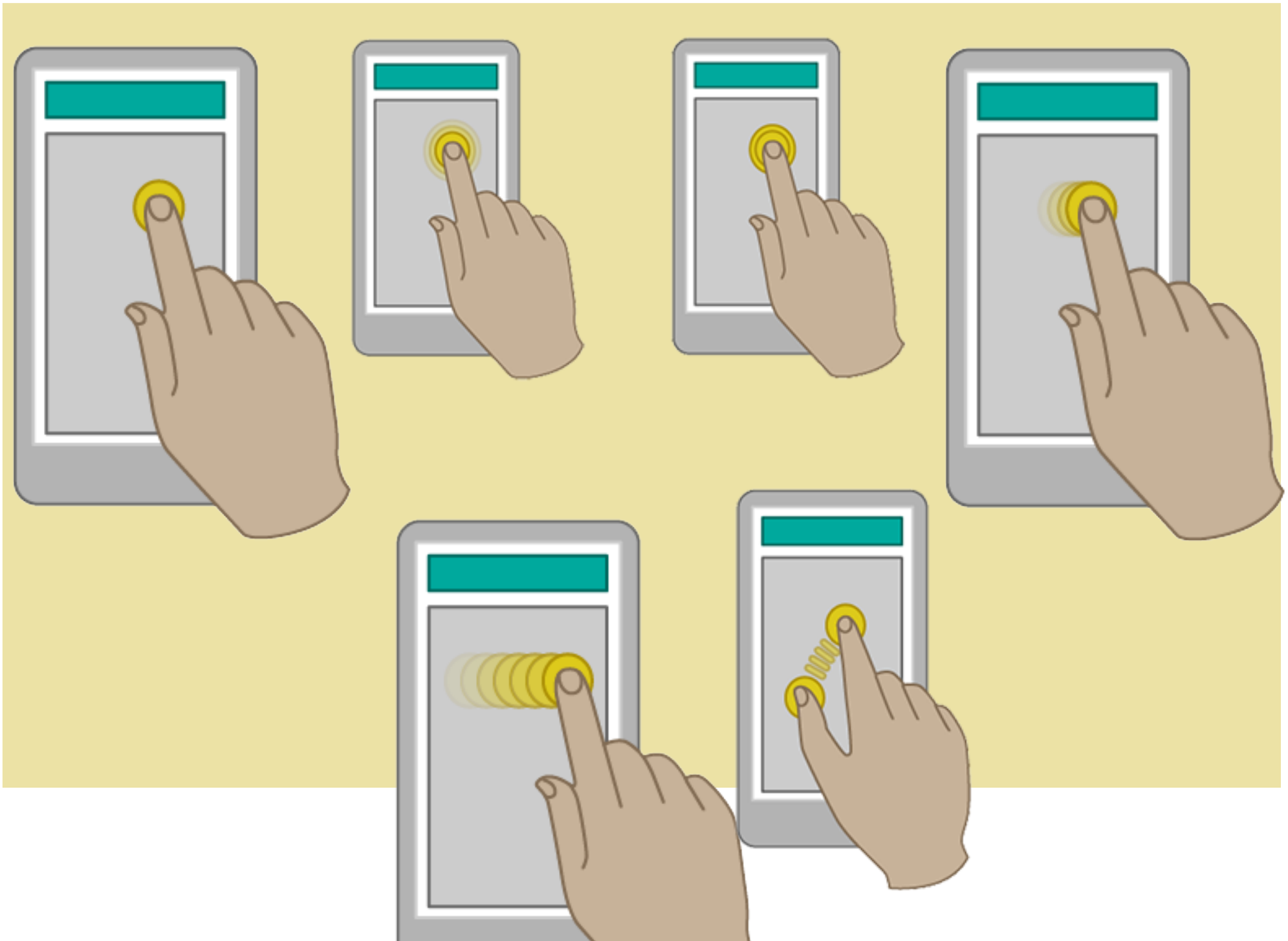


For mobile-friendly designs, make sure you use at least 48px

mobile

2.

gestures



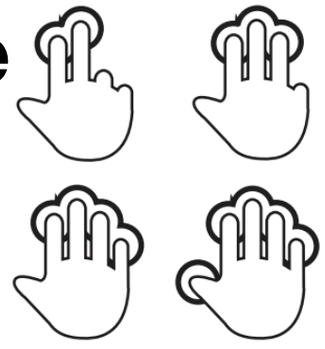
2.

touch gesture guide

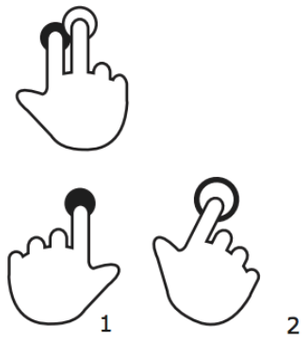
Tap

Press

Multi-finger tap



Press and tap

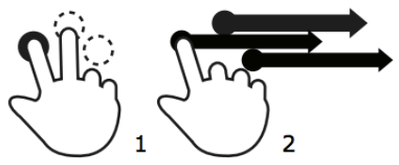


Double tap

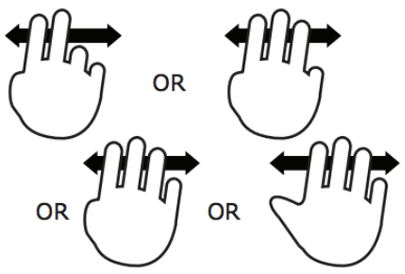
Press and drag

Drag

Press and tap, then drag



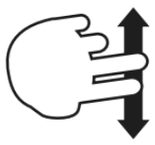
Multi-finger drag



Pinch

Squeeze

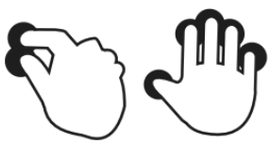
Two-finger drag



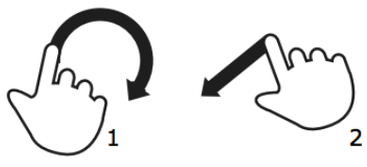
Spread

Flick

Splay



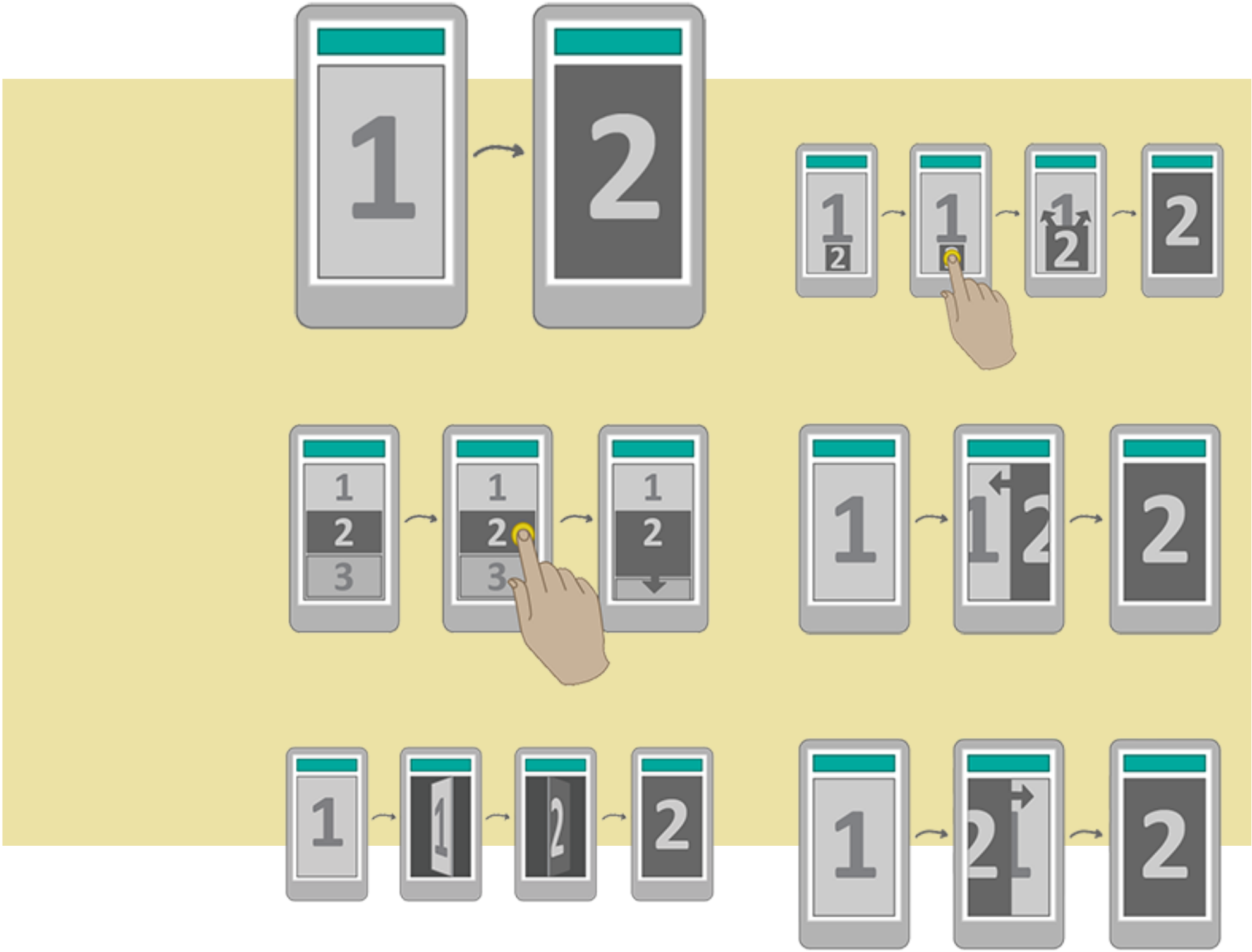
Lasso and cross



mobile

2.

transitions



mobile

2.

common patterns

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

mobile

2.

selecting content

- press to move forward
- coverflow
- flipover peel back

mobile

2.

signing in

- auto sign in
- remember details
- pin codes

mobile

2.

using forms

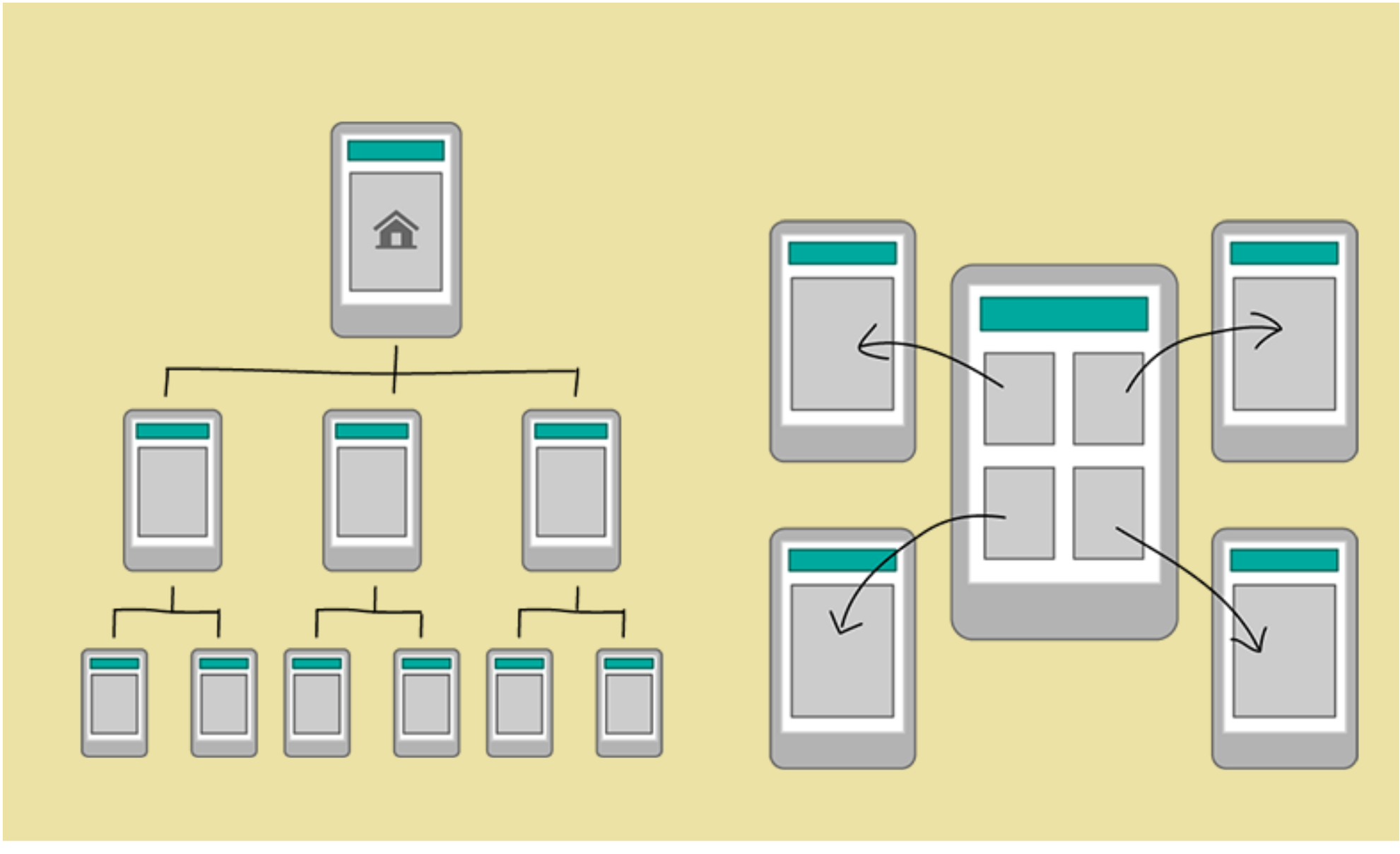
- save user details
- provide the right keyboard
- progress bars

visual design

mobile

3.

information architecture

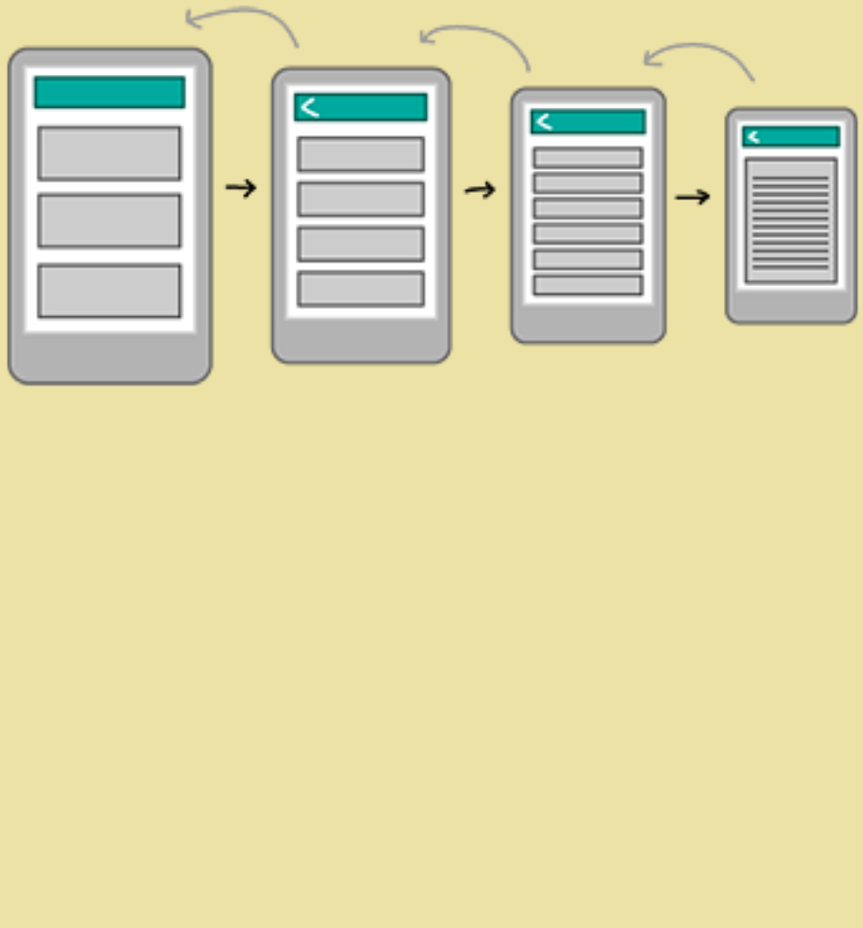


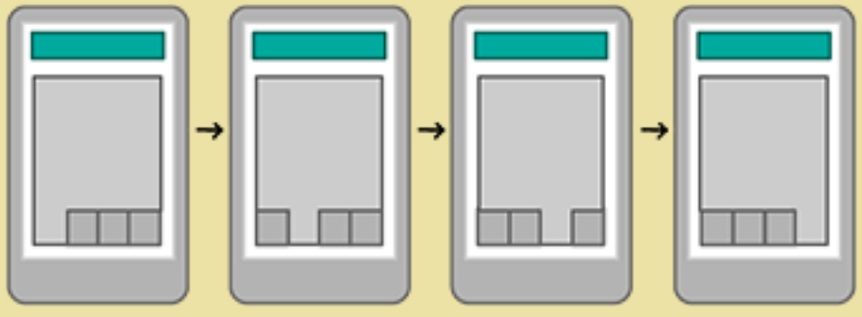
visual design

mobile

3.

information architecture



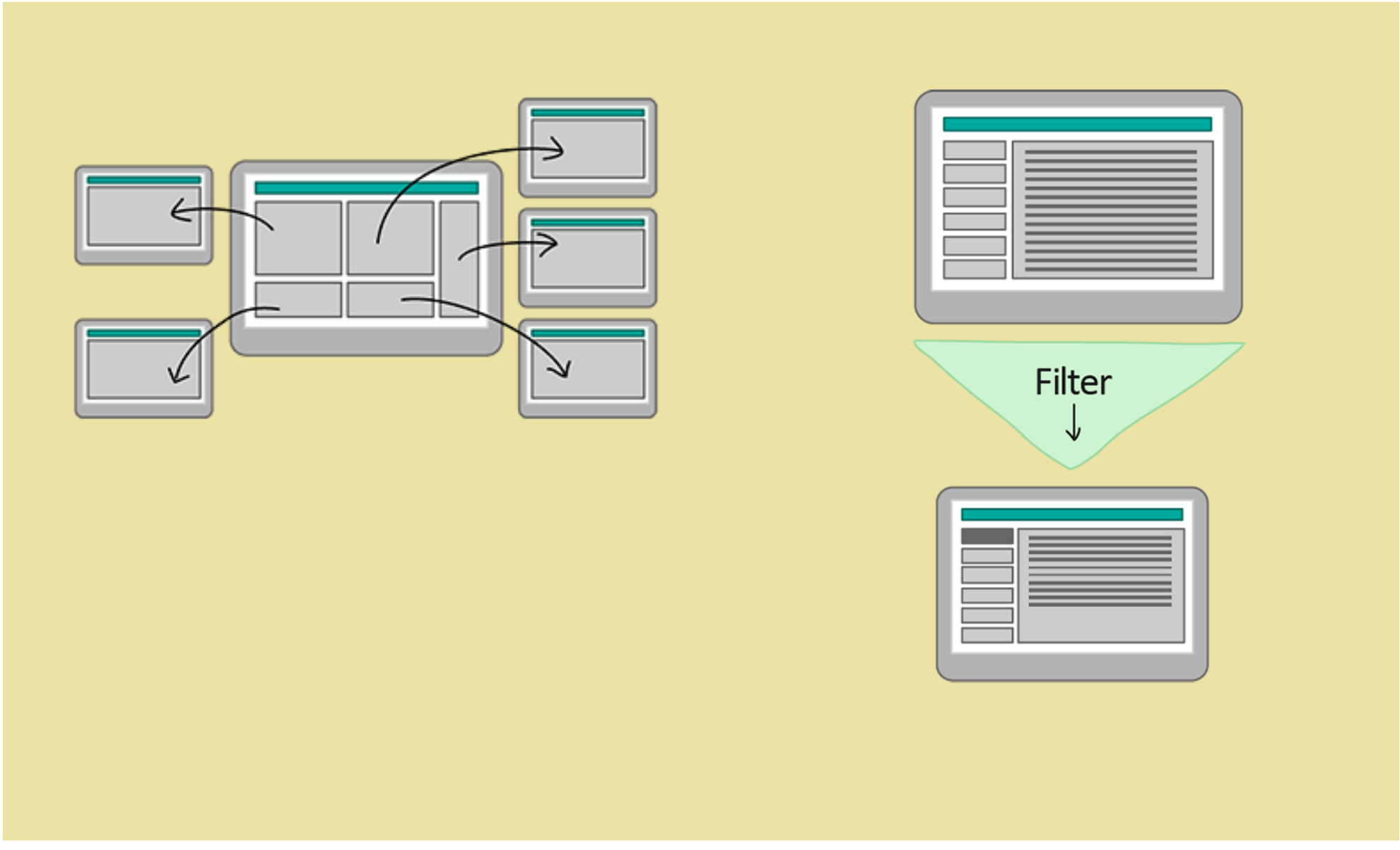


visual design

mobile

3.

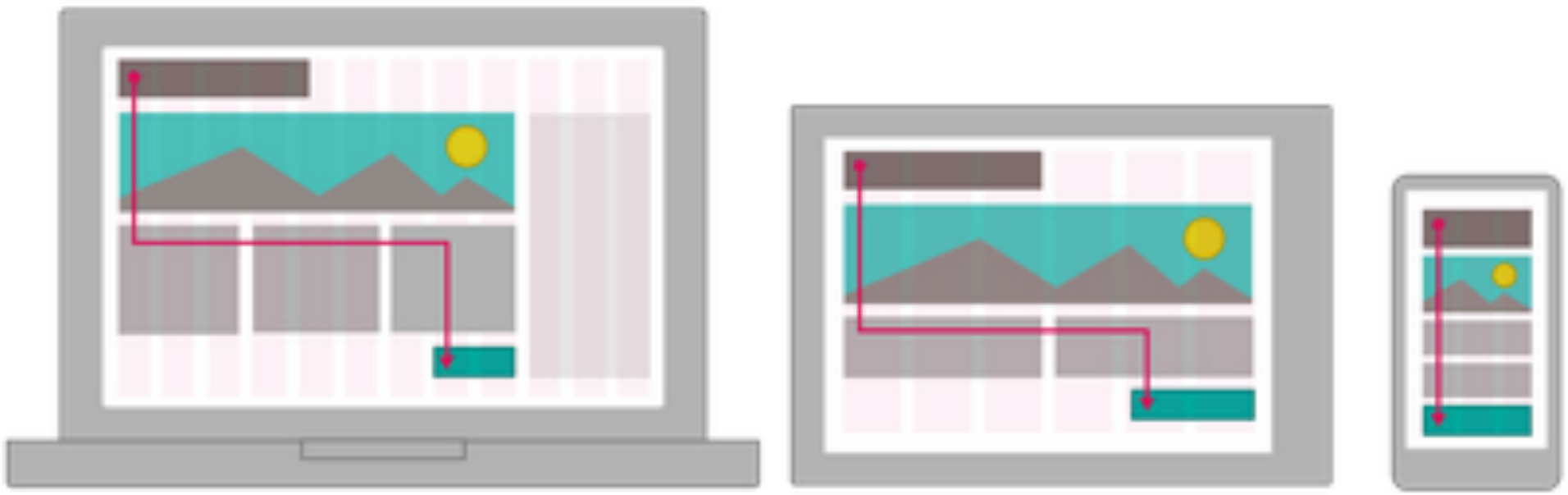
information architecture



3.

usable layouts

limited space



Tips

1.

2.

visual design
mobile

3.

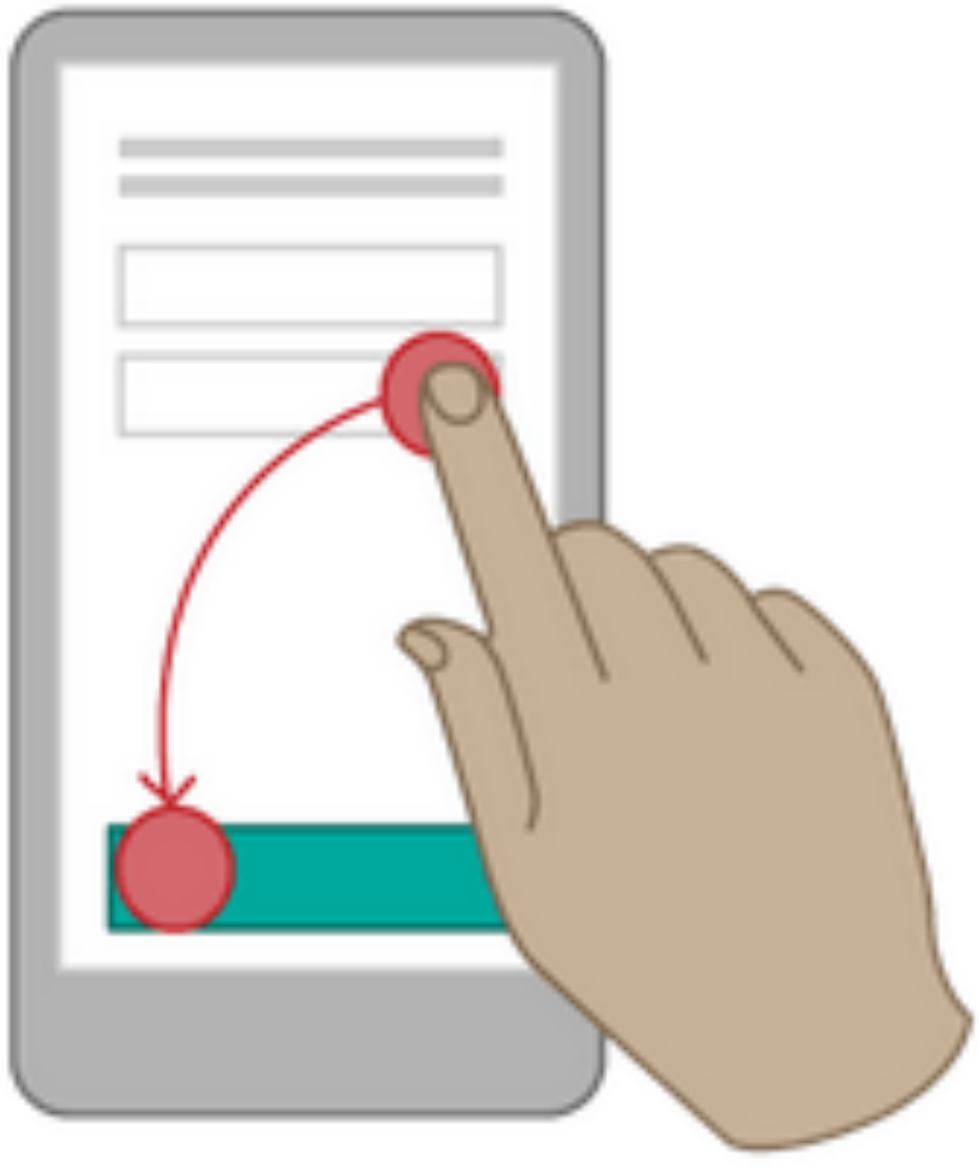
user friendly interactions

target areas

Tips

1.

2.



3.

designing for communication

vertical rhythm

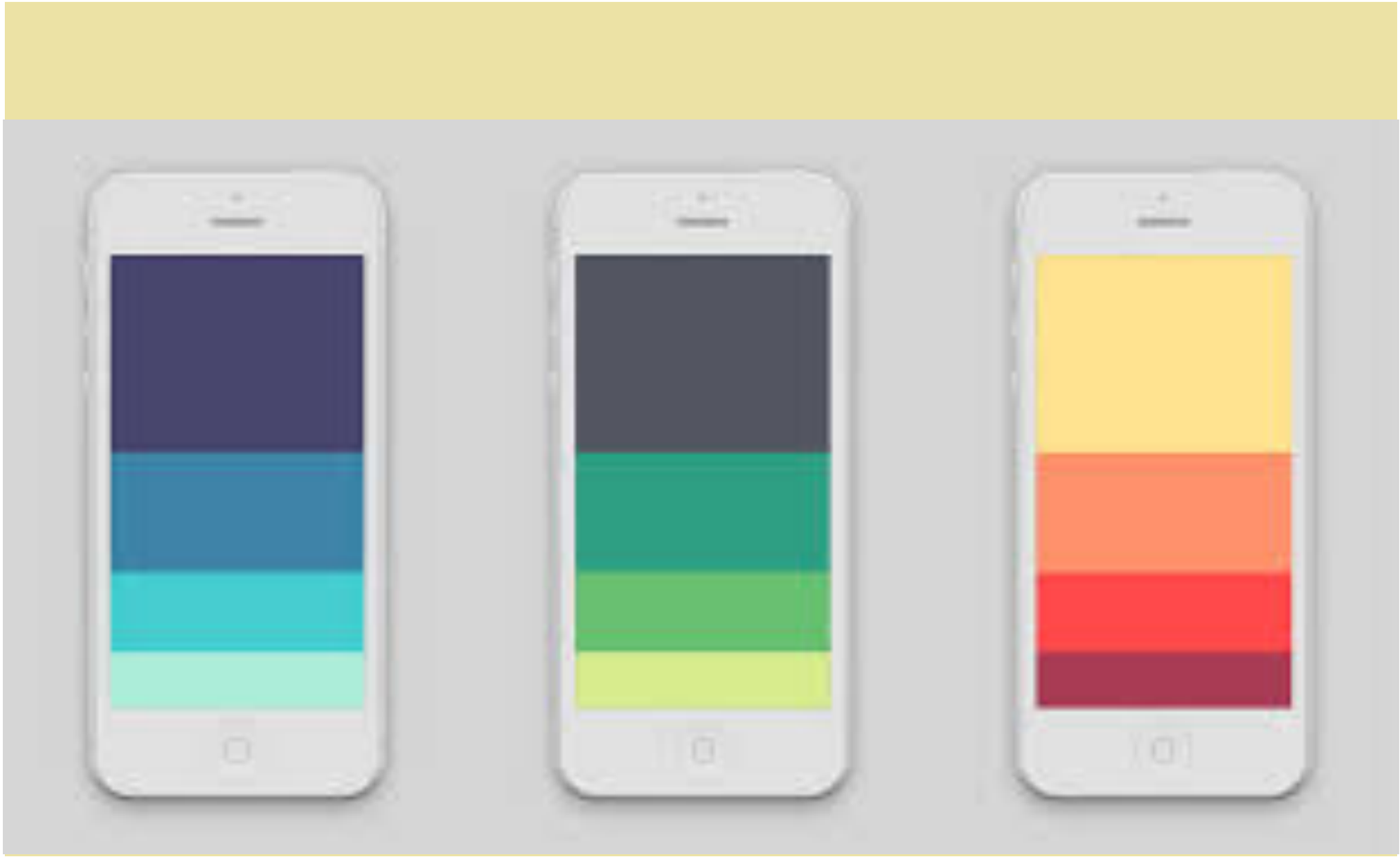
Tip

3.

designing for communication

using color

Tip

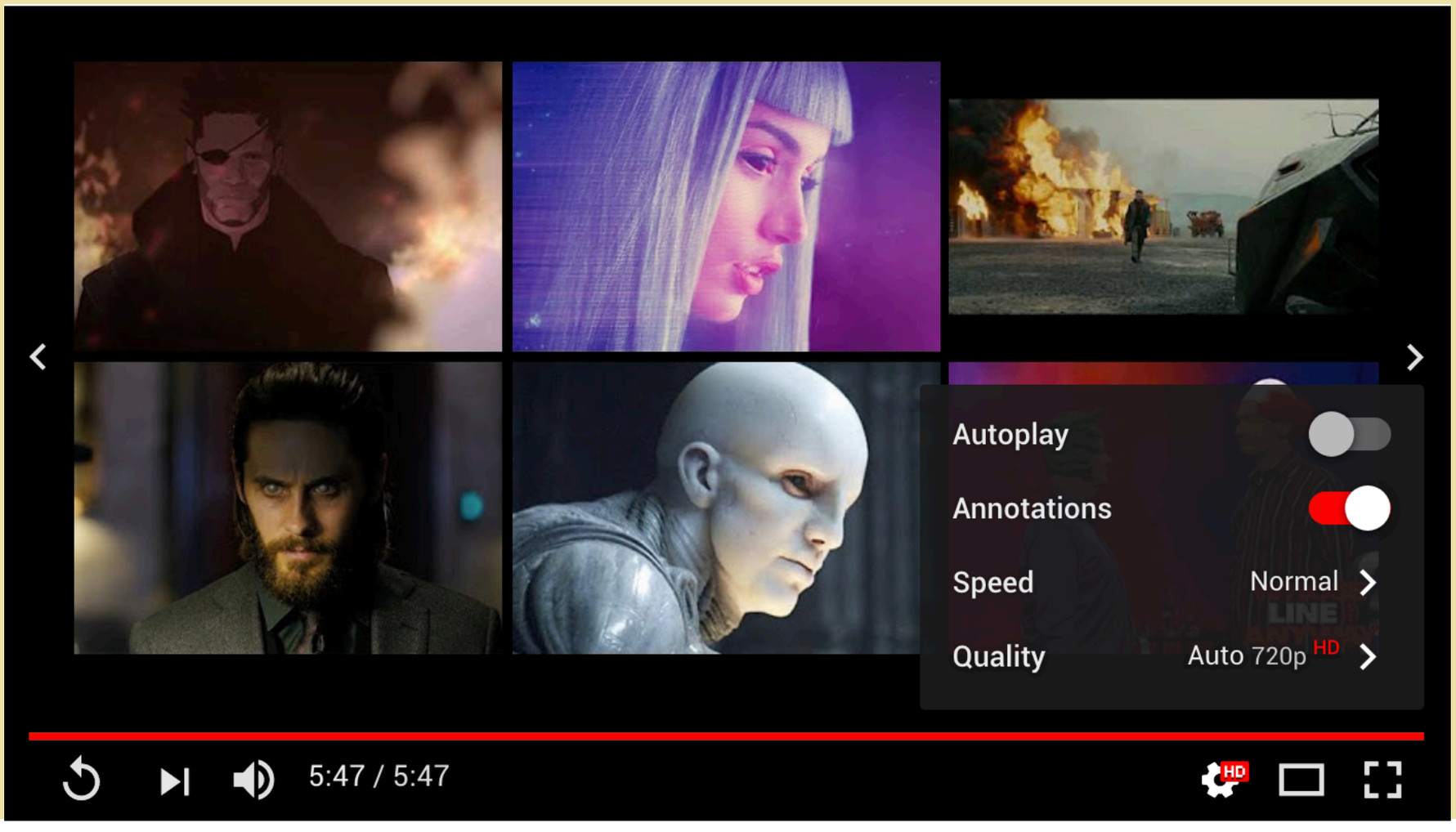


3.

designing for communication

using color as a visual language

?



3.

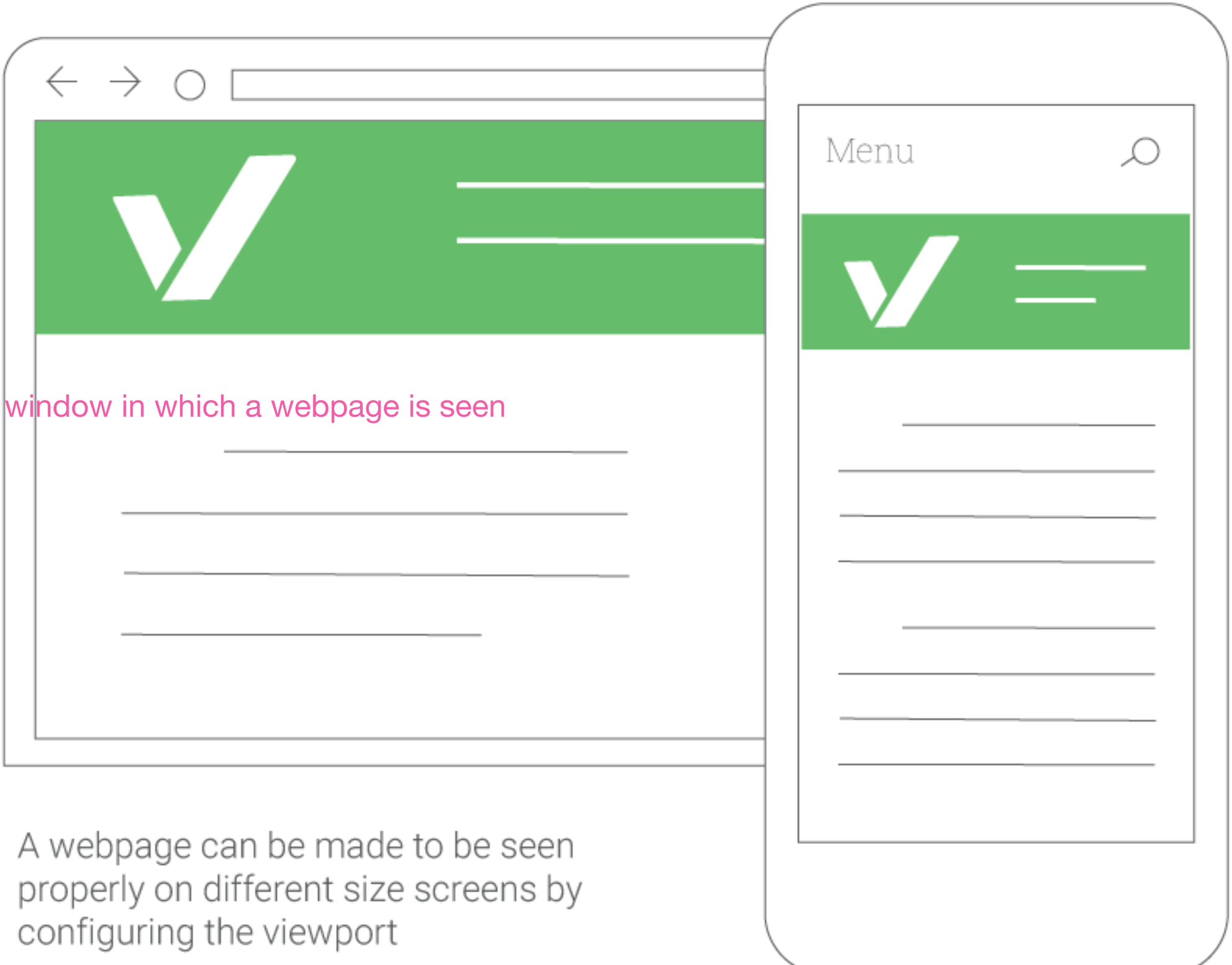
designing for communication

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.

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

Tip 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

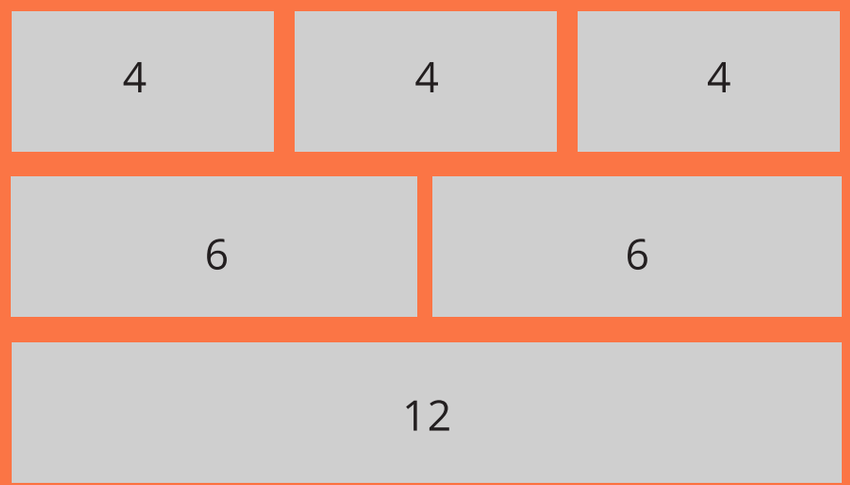
#3

FLEXIBLE MEDIA

FLUID GRIDS

#1

MODULAR

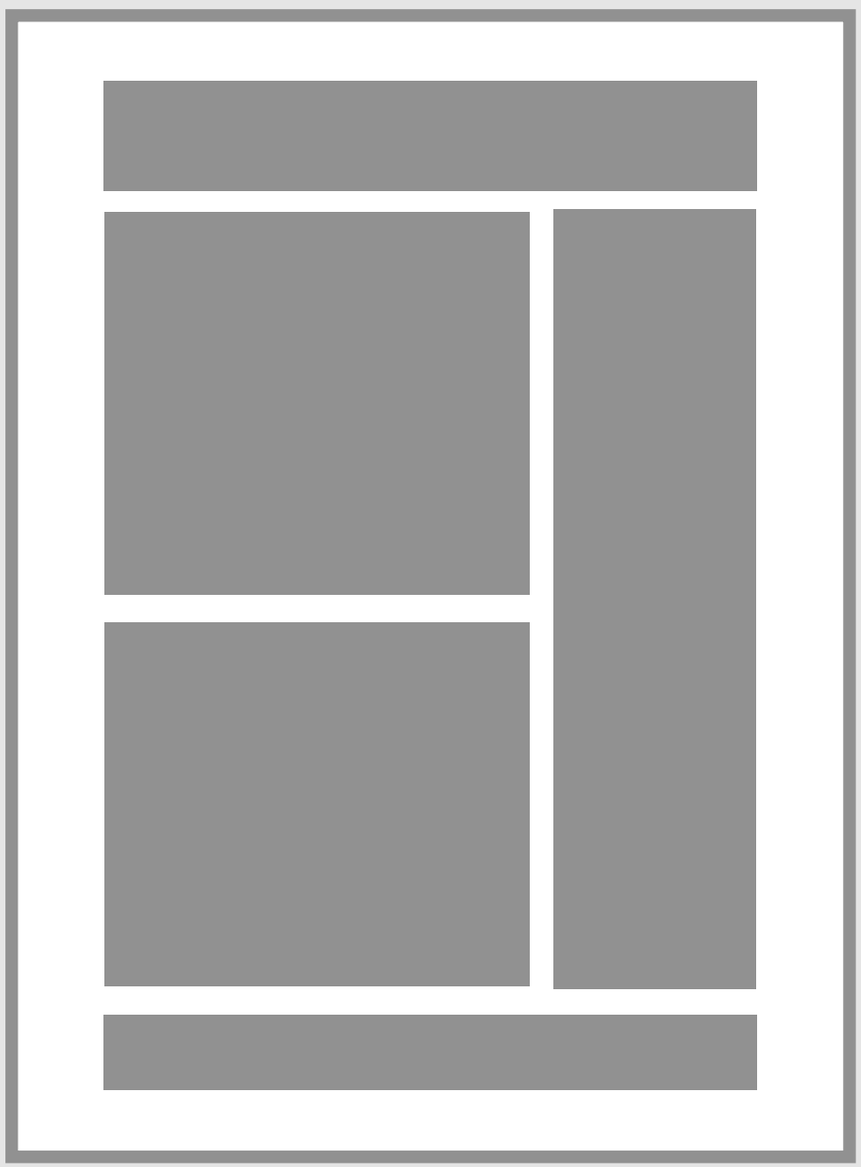


Building a site using modular grids that can flow and adapt to various screen sizes by using percentages, not pixels (as well as a little help from some media queries).

FIXED VS. FLUID

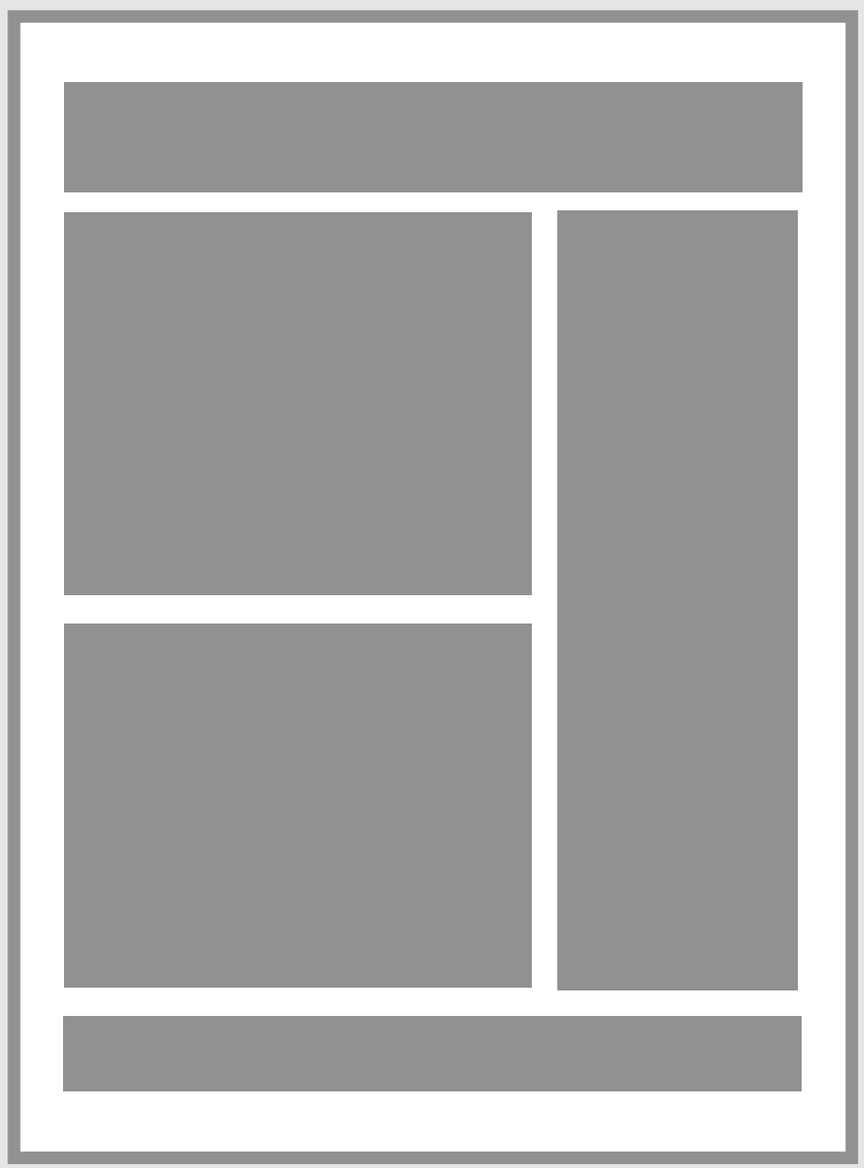
FIXED GRID

960px



FLUID GRID

100%



FLUID GRIDS SCALE & FLEX

And still allow you to align your content.

Logo

Navigation

Article

Sidebar

Footer

Log

Navigation

Article

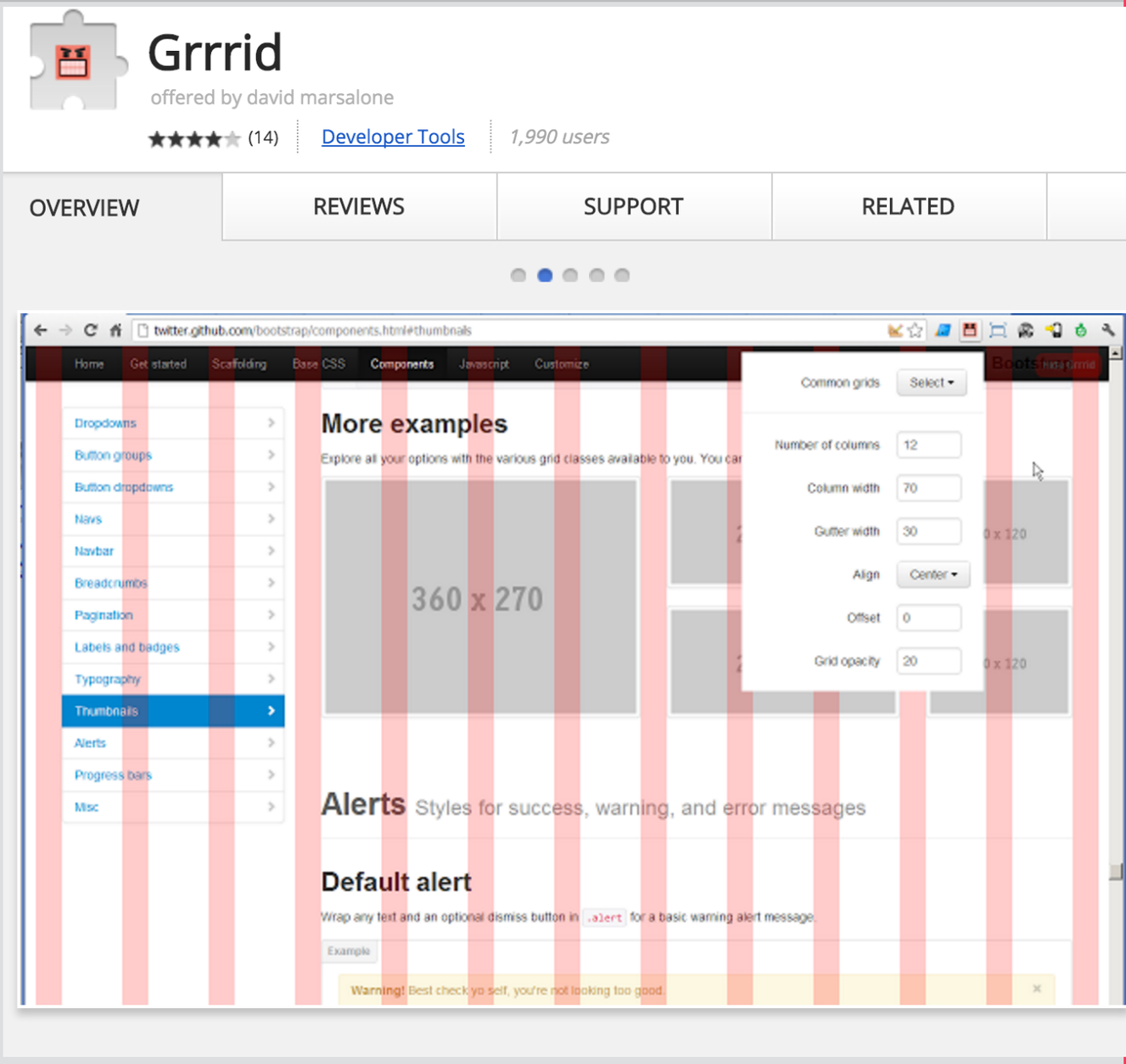
Sidebar

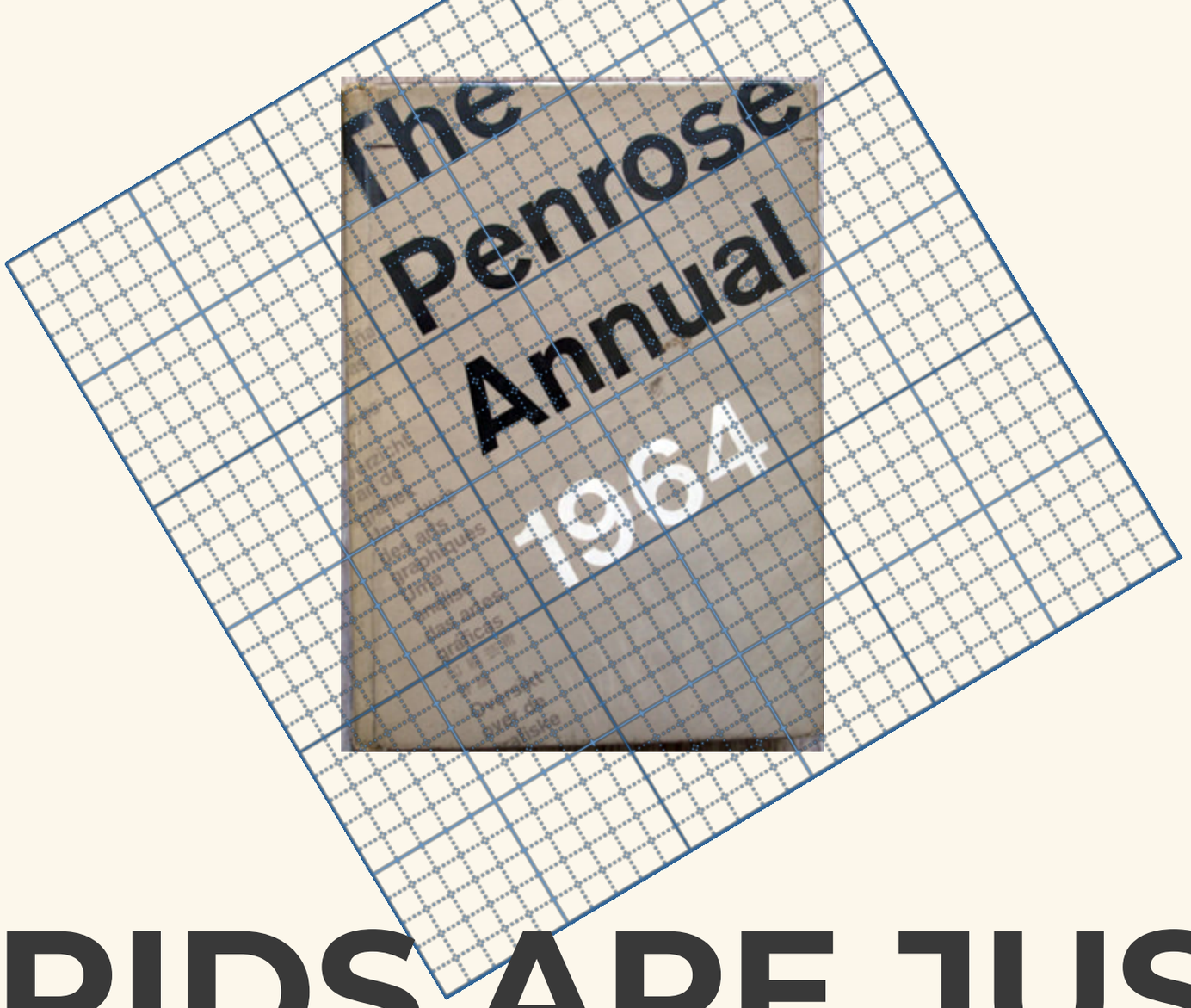
Footer

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.

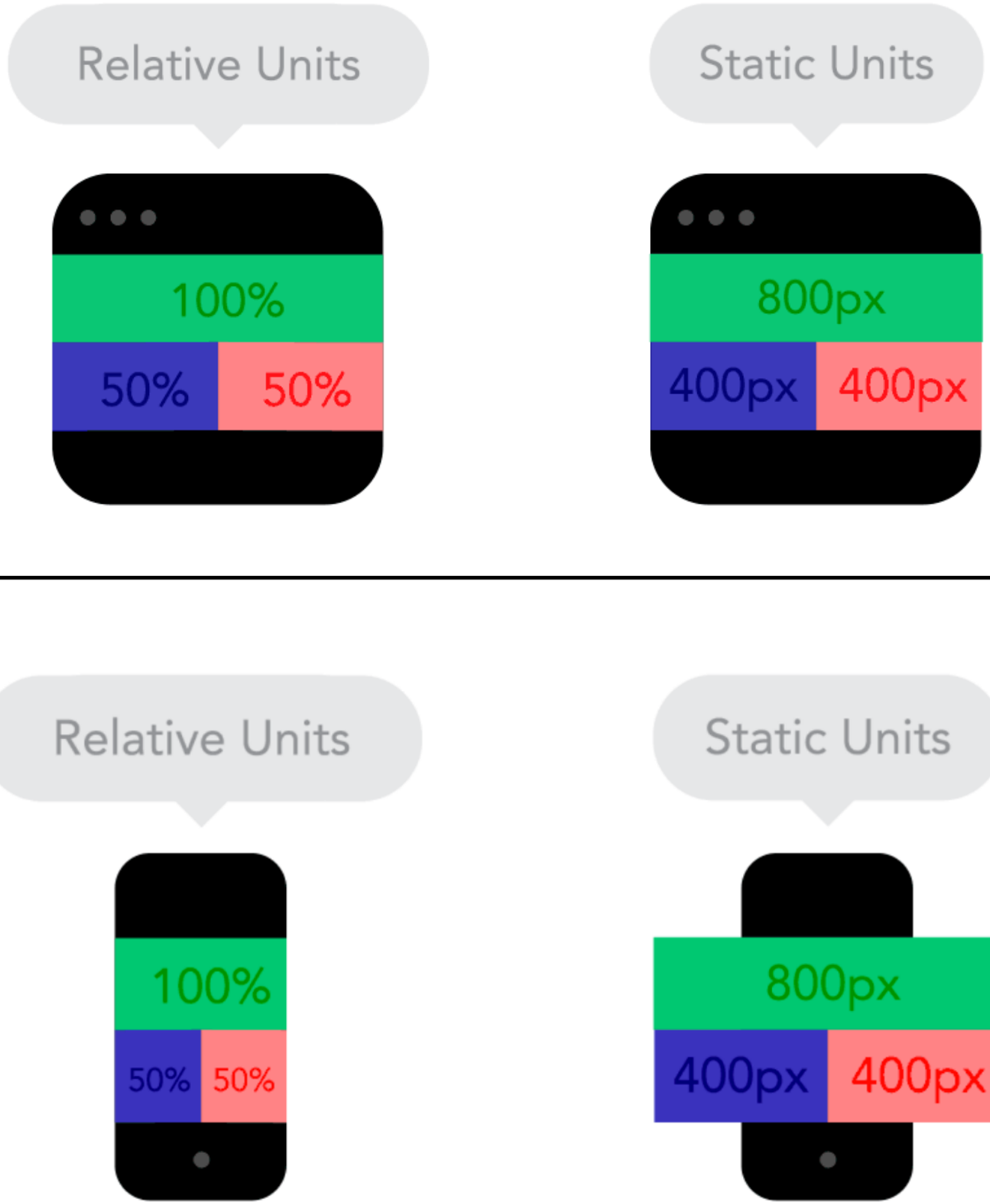




GRIDS ARE JUST GUIDES TO HELP ALIGN

They help you to align elements and provide structure.

They’re not perfect.

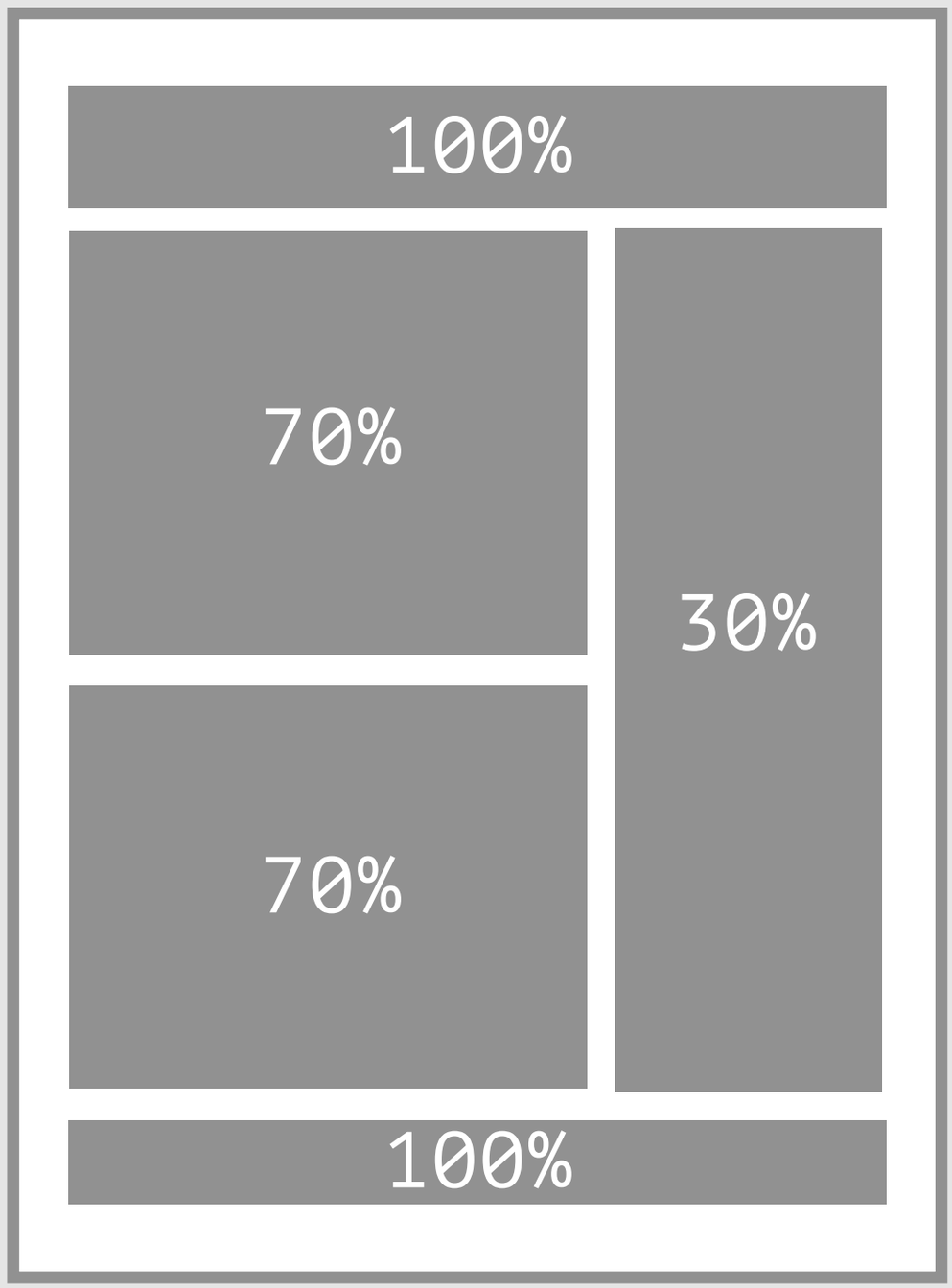


USE %

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

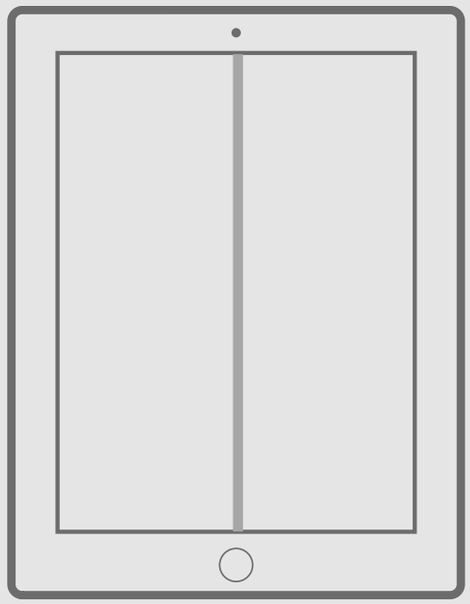


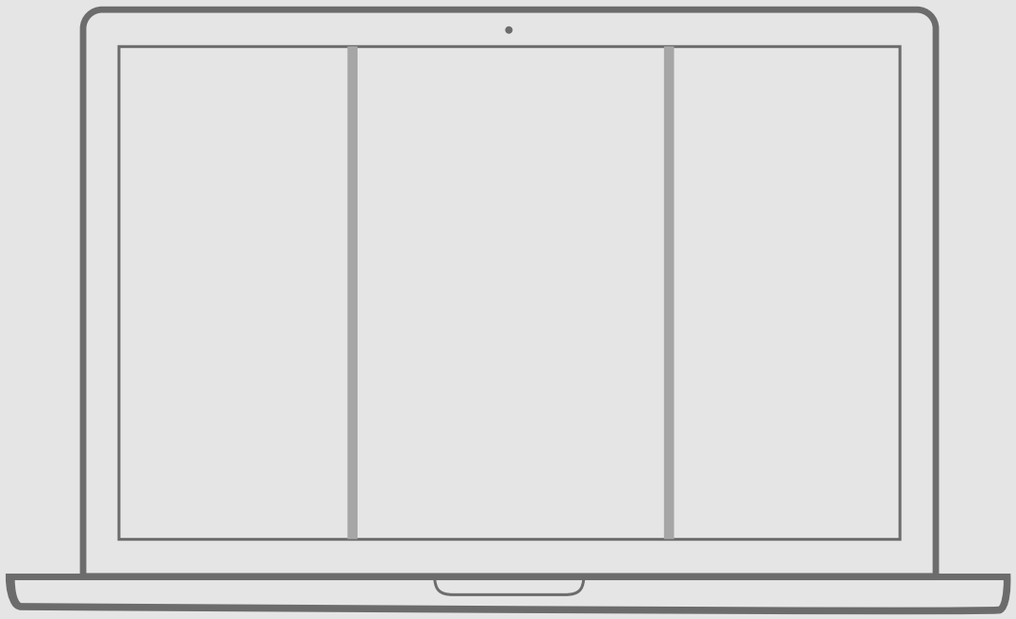

```
#header {
  width: 100%;
}
#side-bar {
  float: right;
  width: 30%;
}
#content {
  float: left;
  width: 70%;
}
```

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 :)





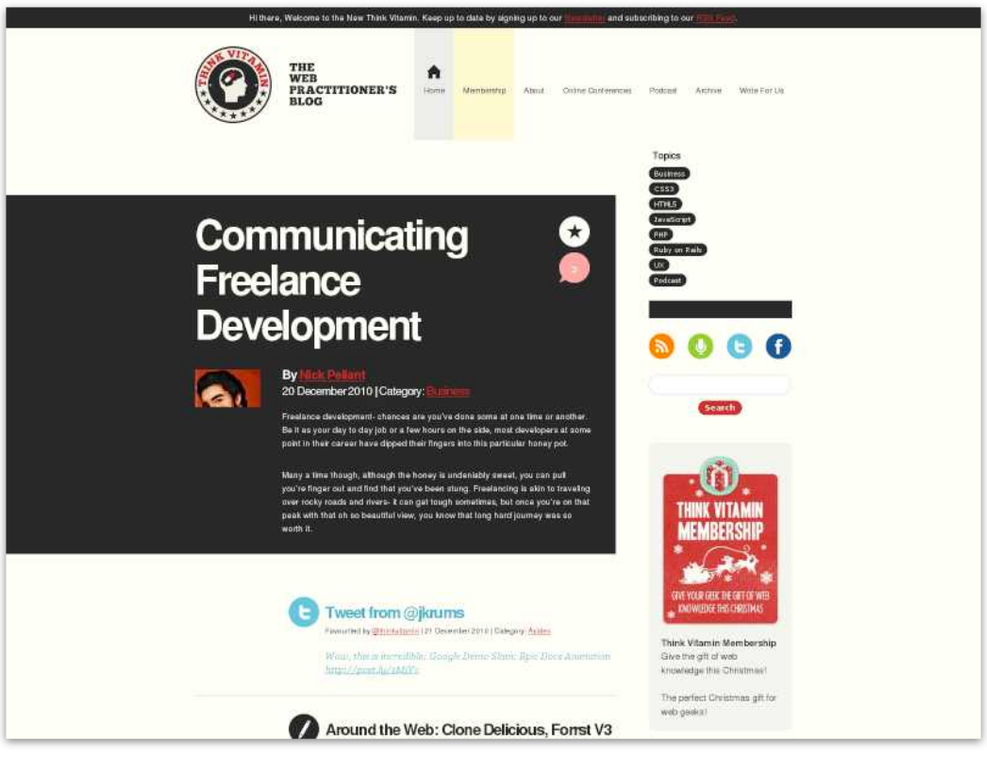
BREAKPOINTS

#2

http://mediaqueri.es/popular/

Think Vitamin

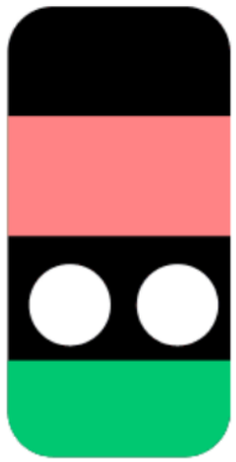
92



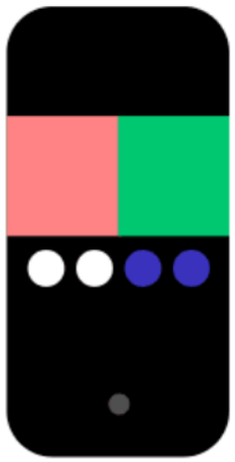
With Breakpoints

Without Breakpoints

With Breakpoints



Without Breakpoints



BREAK-POINTS

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

#3

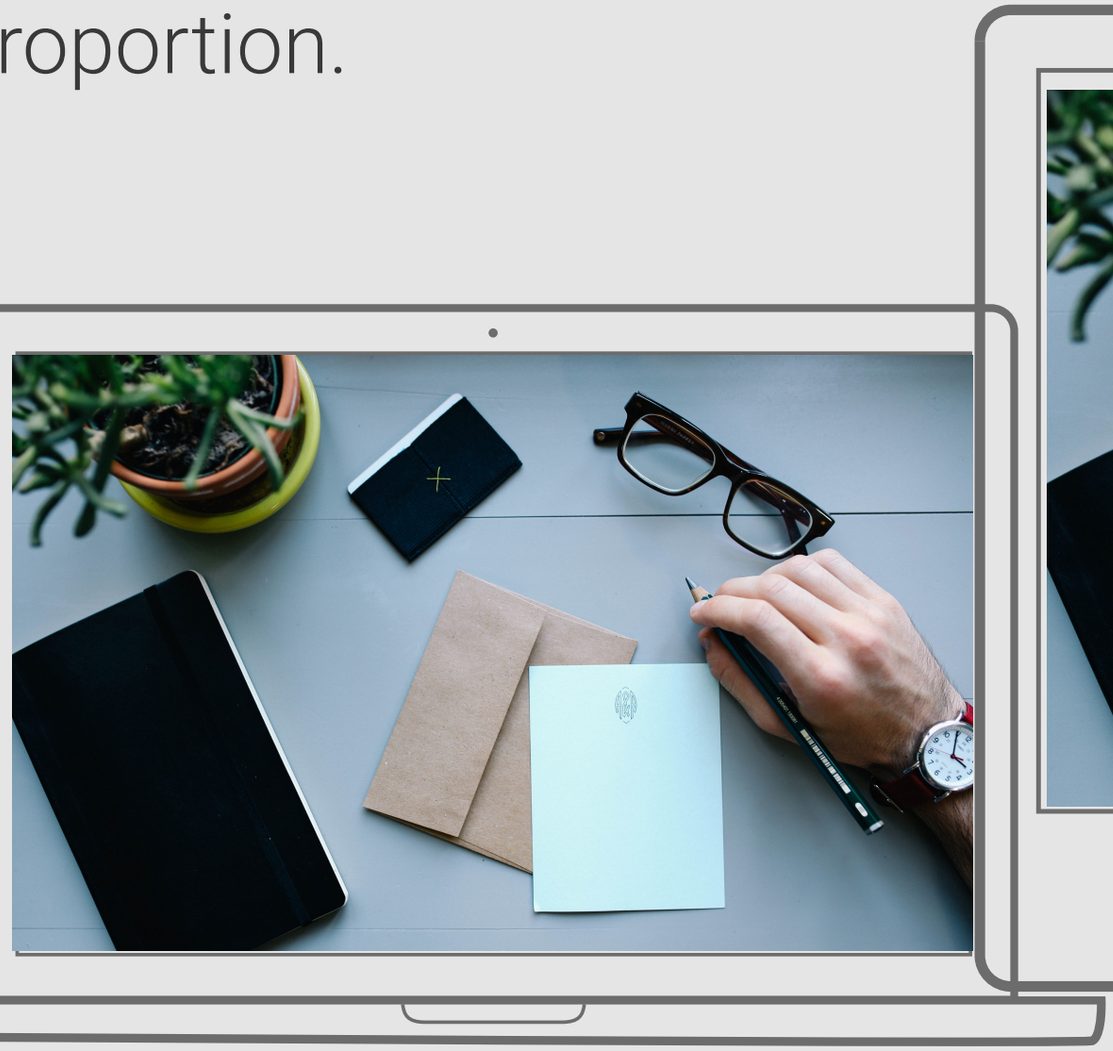
FLEXIBLE MEDIA

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.



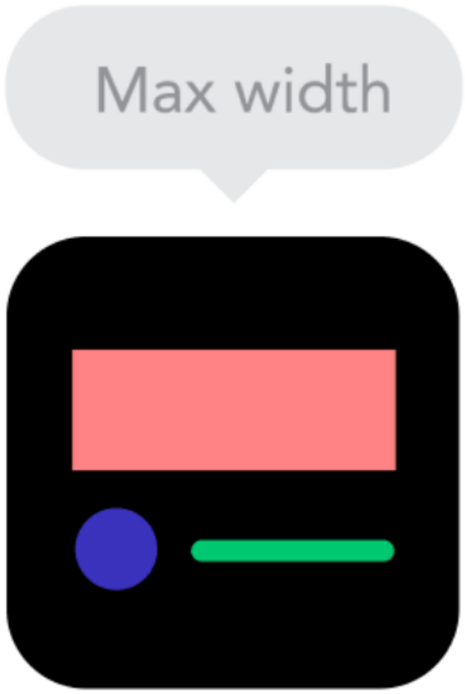
SIMPLE LINE OF CODE

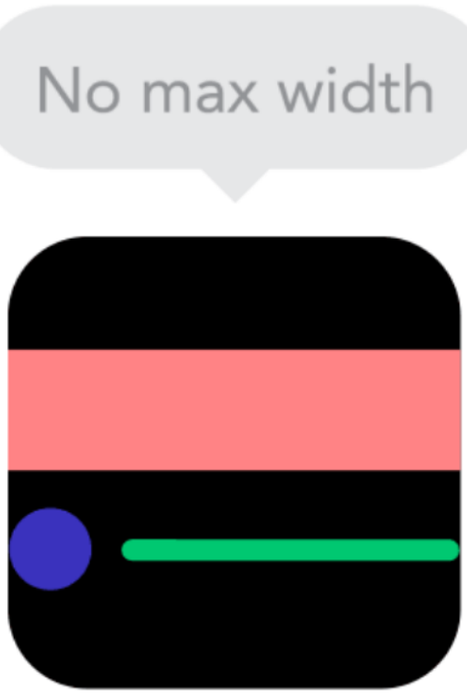
```
img {
    max-width: 100%;
}
```

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.



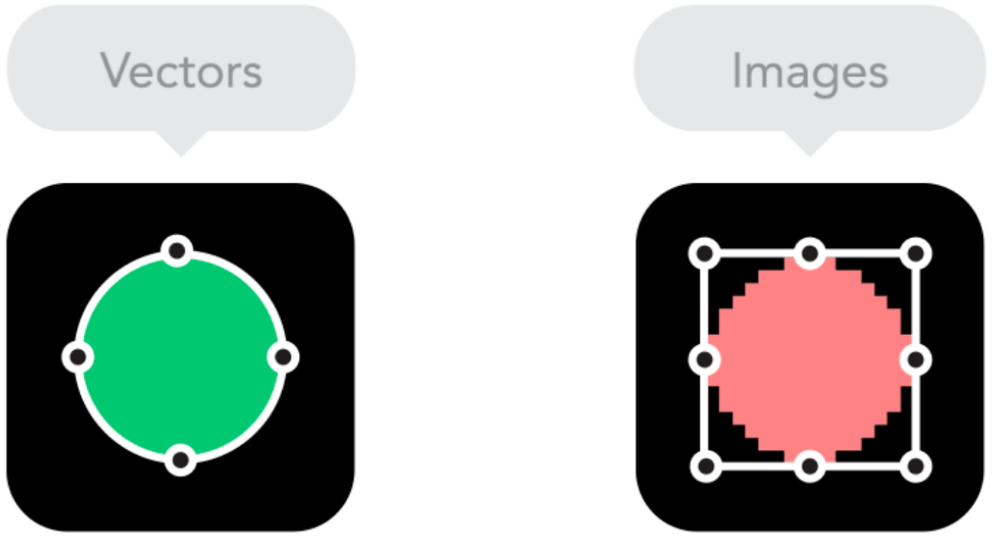



```
/* Small screens (default) */
html { font-size: 100%; }

/* 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

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

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

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

4. 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)

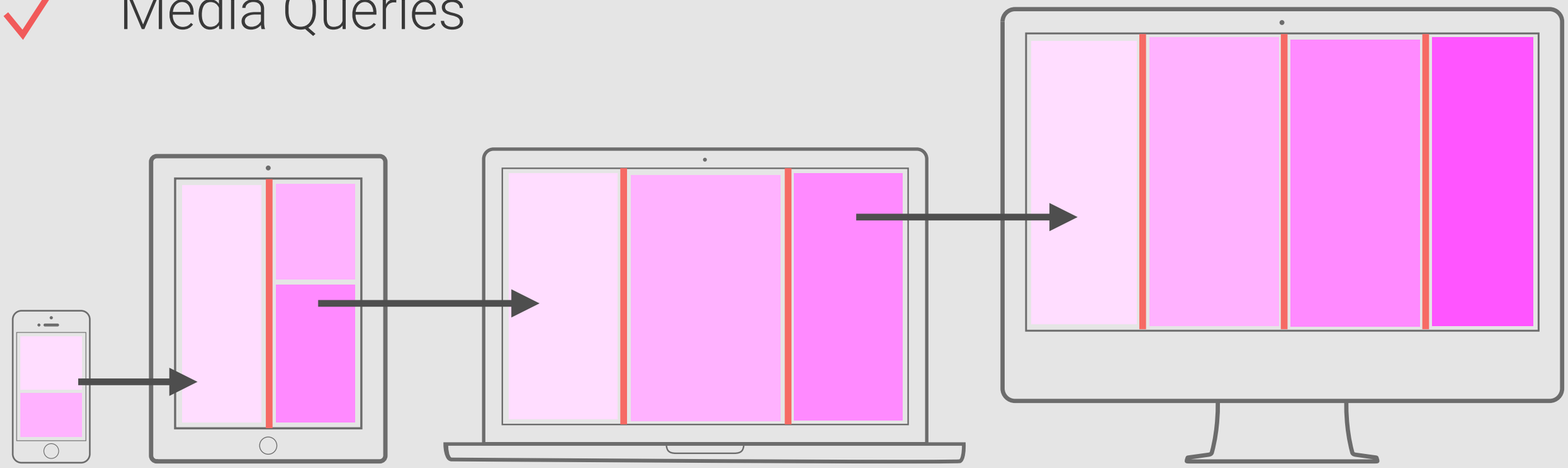
```
/* ...mobile styles here...
*/ @media screen and (min-width: 600px) {
/* ...tablet styles here... */ }

@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



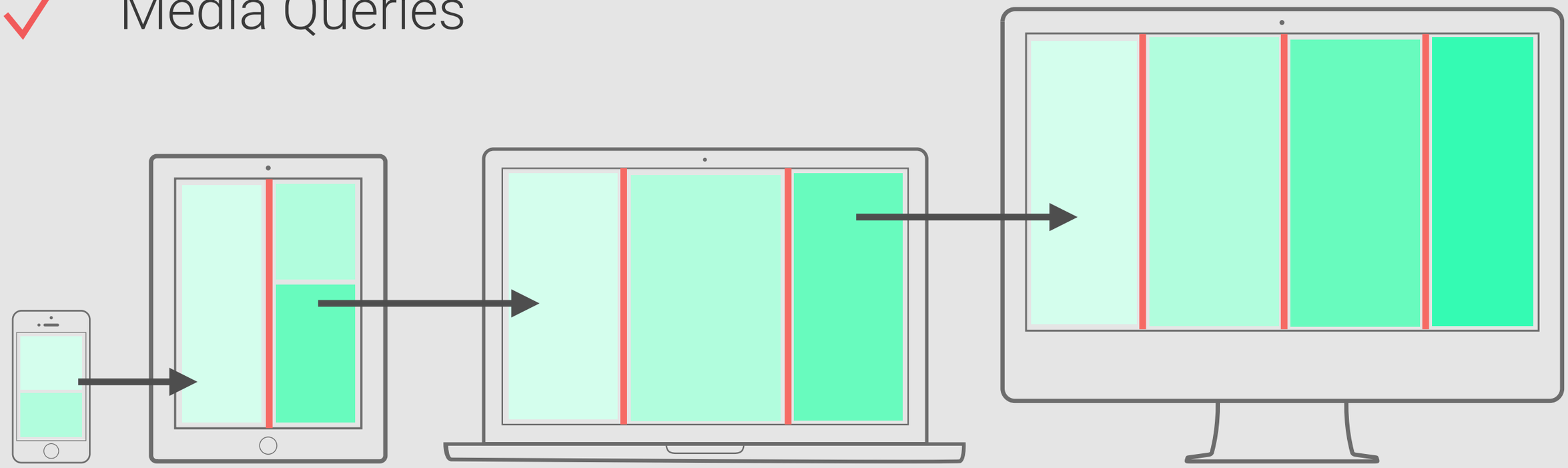
ADAPTIVE DESIGN

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

~~Fluid Grids~~

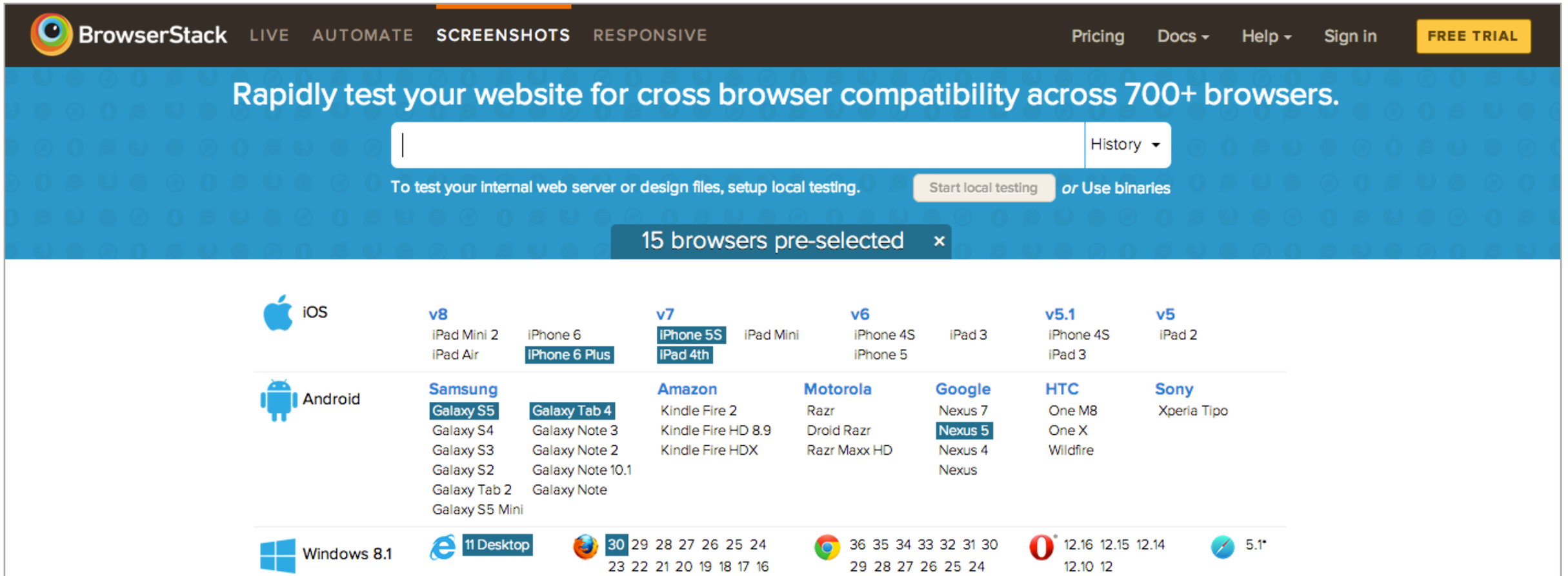
~~Flexible Media~~

✓ Media Queries



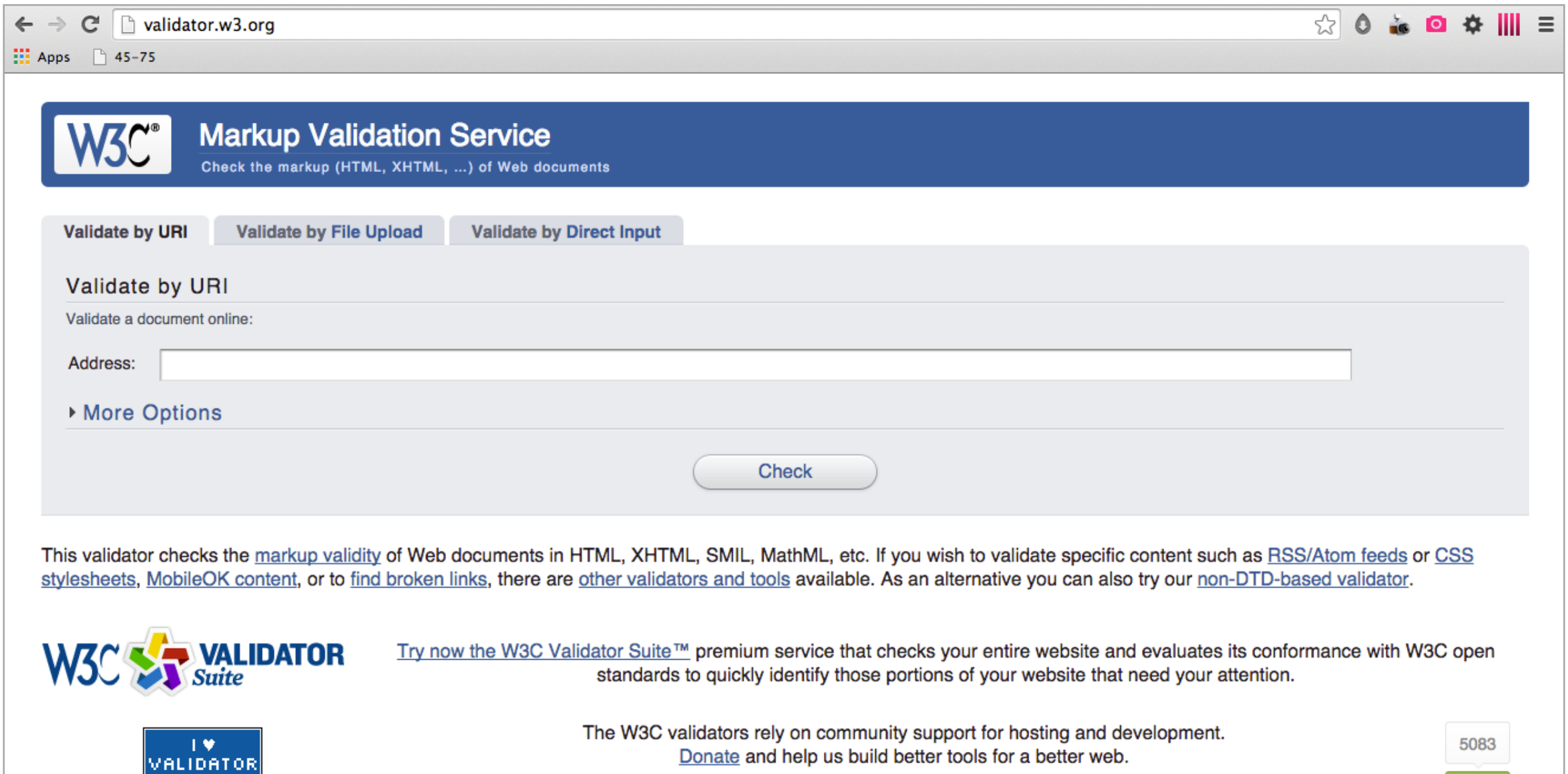
CROSS-BROWSER TESTING

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



VALIDATE YOUR CODE

Check your code to check for errors.



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/

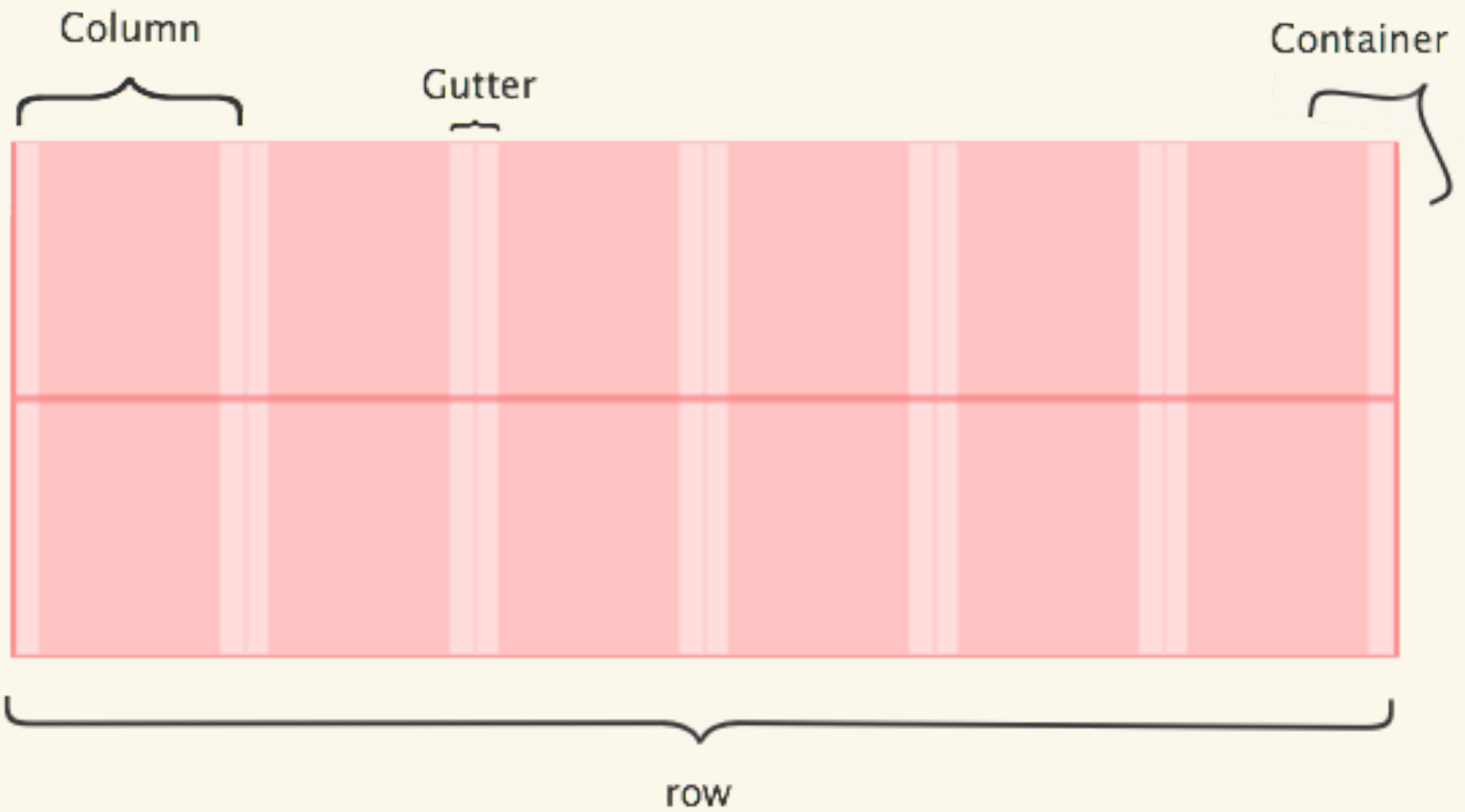
9 Basic Principles of Responsive Web Design
http://blog.froont.com/9-basic-principles-of-responsive-web-design/

Wild Experiments in the Latest Coding
http://beta.theexpressiveweb.com/#!/css3-media-queries

frameworks

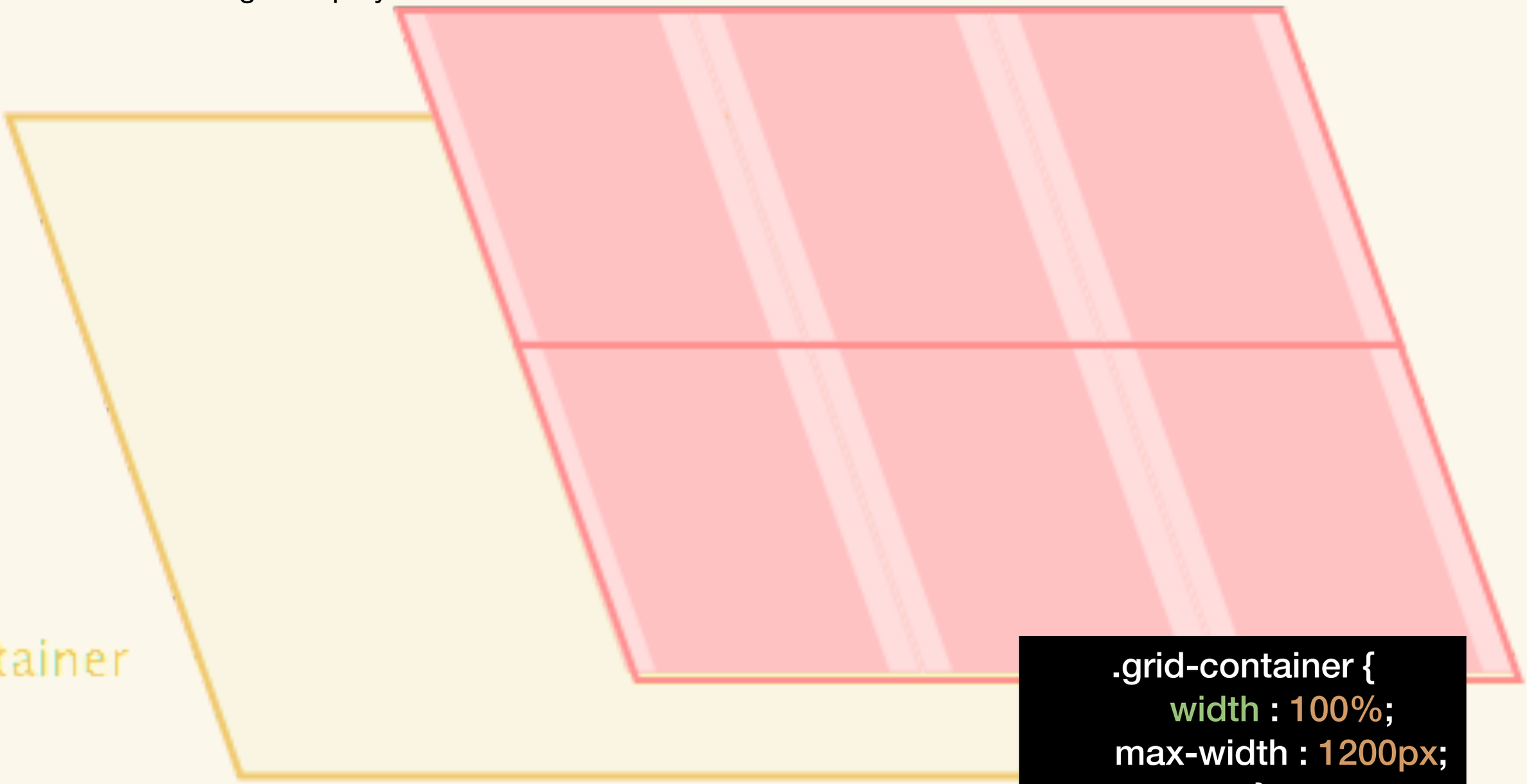
- A FLUID GRID LAYOUT
- RESPONSIVE DESIGN
- CUSTOM FORM ELEMENTS
- TYPOGRAPHY
- JAVASCRIPT INTERACTION
- CROSS BROWSER COMPATABLILITY
- IT IS FAST
- IT IS CUSTOMIZABLE

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.



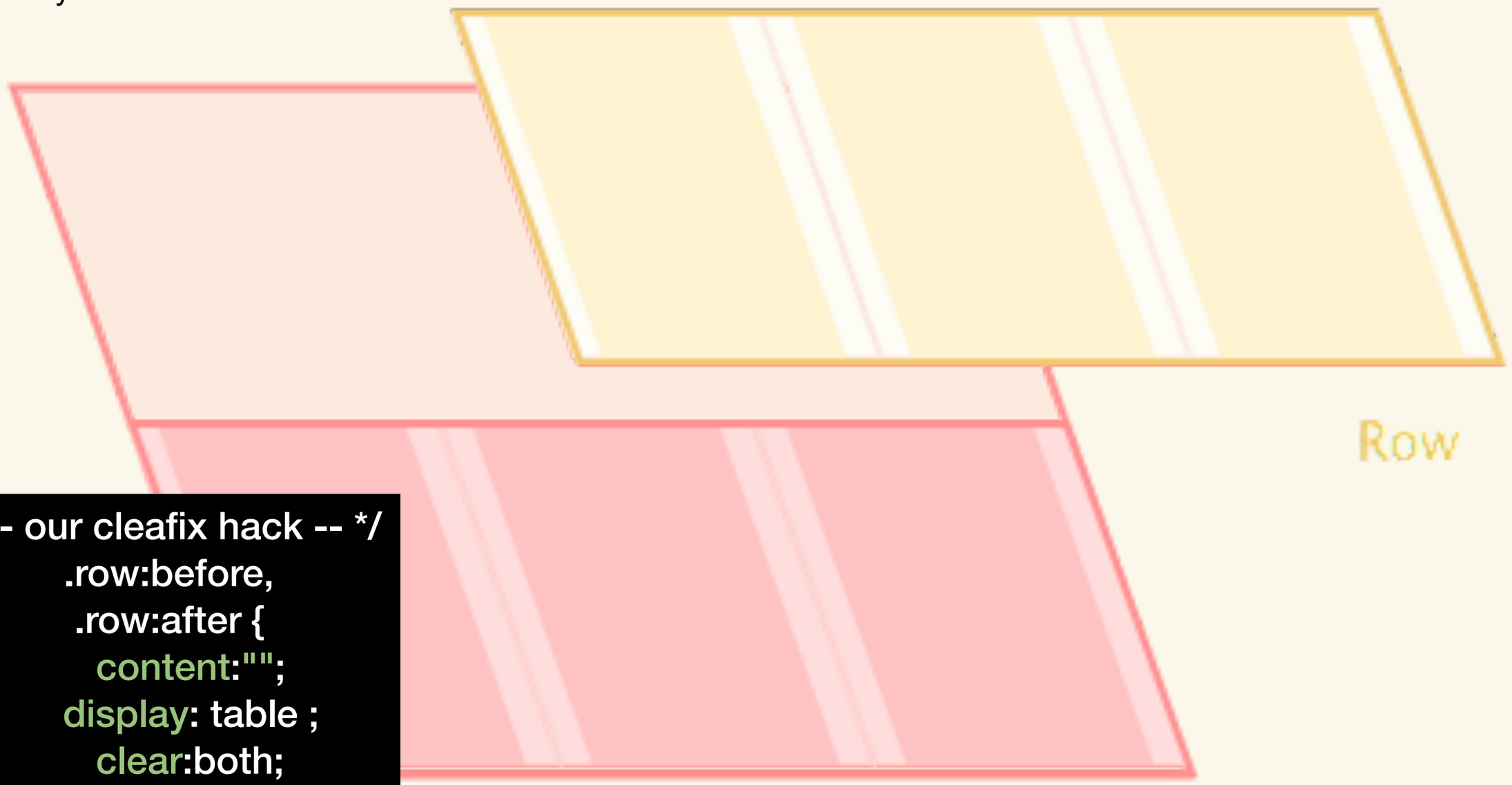

```
.grid-container {
    width : 100%;
max-width : 1200px;
}
```

BOX SIZING BORDER-BOX

```
html {
  box-sizing: border-box;
}

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

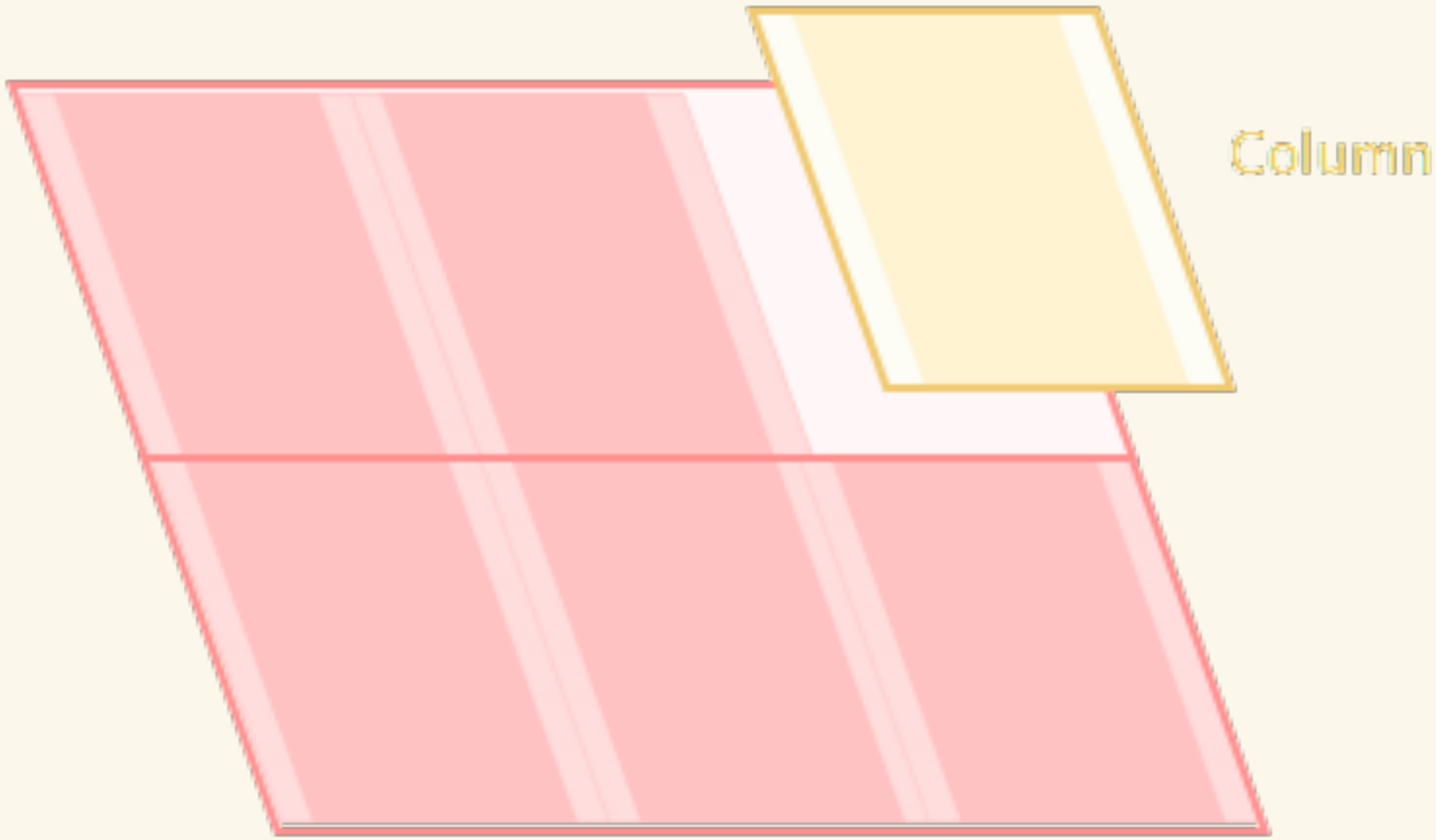
ROW

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 posit
columns in CSS. From my personal experience, the least error prone and most wide
of these methods is the 'float' method. If our columns are empty however, our floate
columns will stack on top of each other. To prevent this, we'll give our columns a mi
height of 1px as well as float them.

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

COLUMN WIDTHS

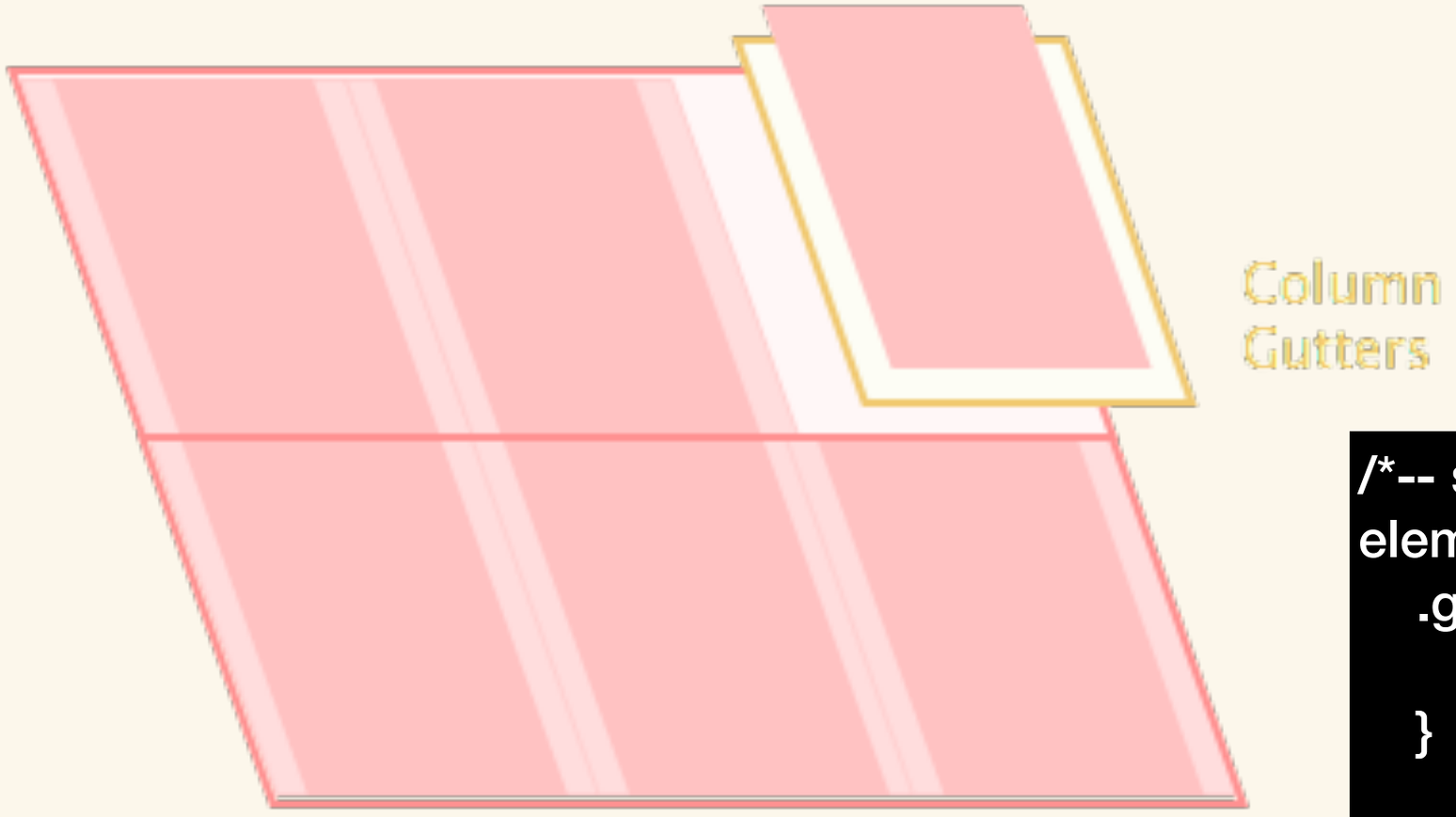
```
[class*='col-'] {
    float: left;
  min-height: 1px;
   width: 16.66%;
   }
```

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: 66.664%;
  }
.col-5{
  width: 83.33%;
  }
.col-6{
  width: 100%;
  }
```

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.

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

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

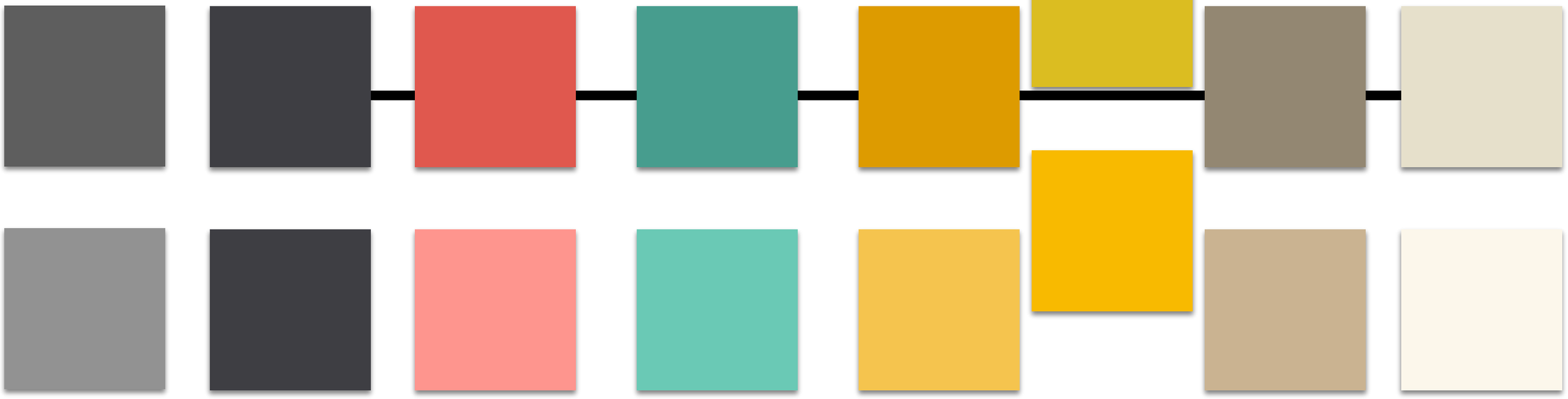
http://j4n.co/blog/Creating-your-own-css-grid-system

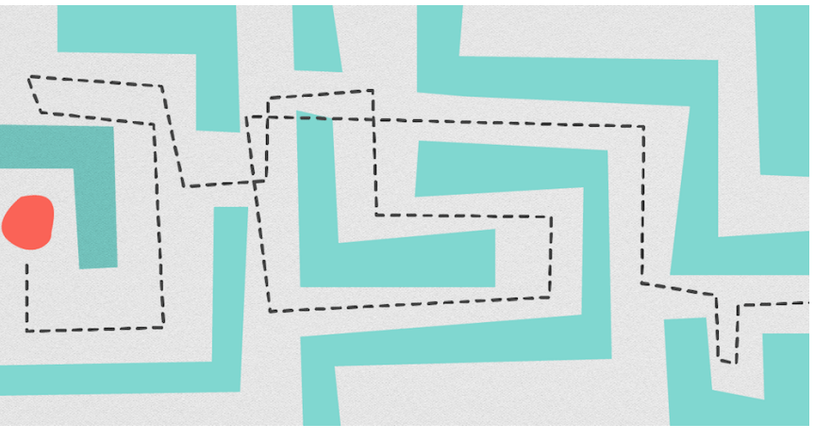
https://zellwk.com/blog/from-html-grids-to-css-grids/

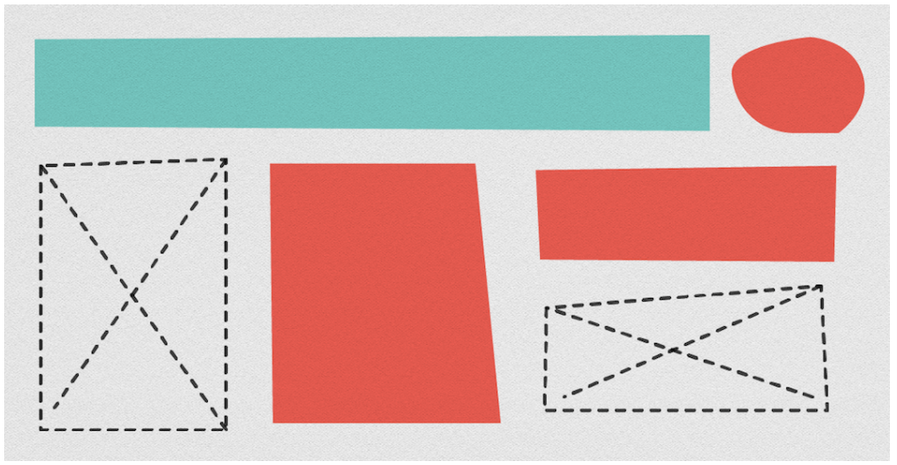
CSS GRID SYSTEM

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

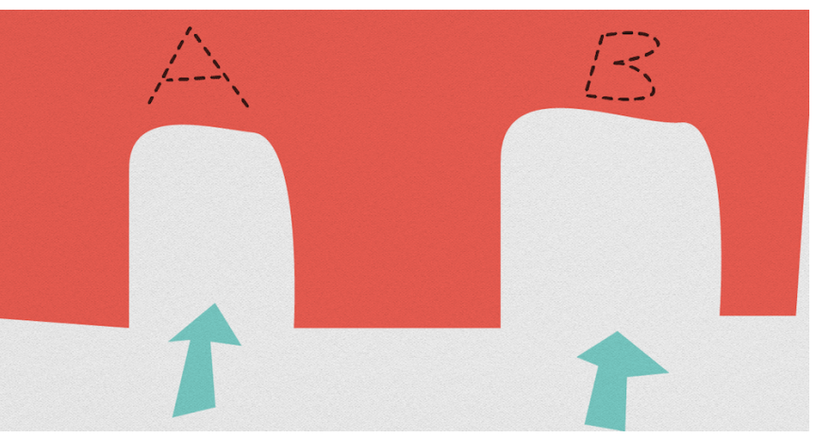
https://www.digitalocean.com/community/tutorials/an-introduction-to-jquery

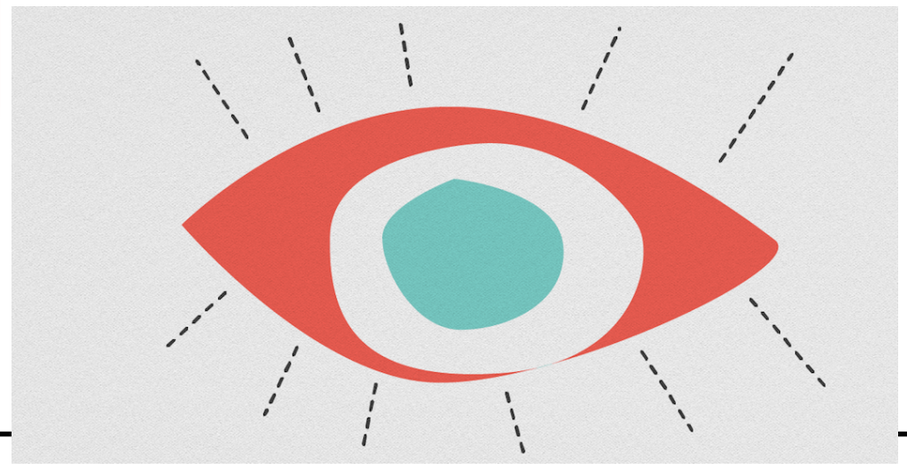


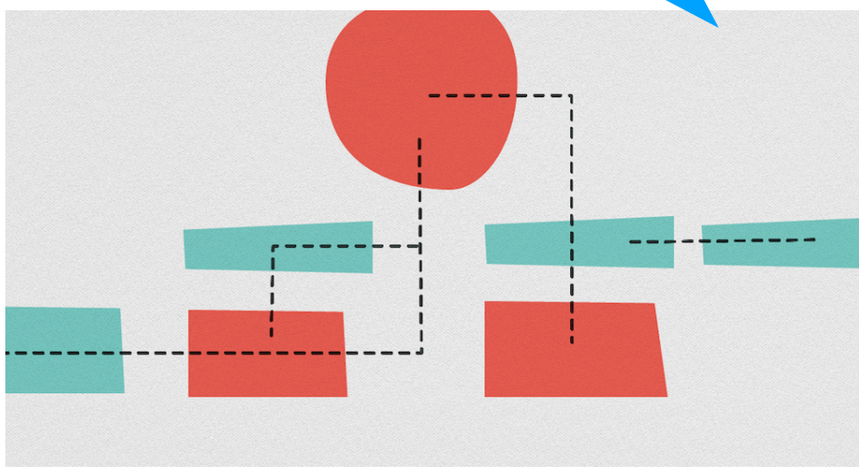




Tip 1







Rule of Thumb Across Most Browsers:

1EM = 16PX= 100%

Is the default

vertical space between lines of type

LEADING

leading

print

line-height

leading

css code

Design Elements

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

Instructor: Angela Wyman

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

Instructor: Angela Wyman

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
Paragrapn text
- Bullet
- Text

Tip 1

Intro to C2

Digital Asset & File Supply Images