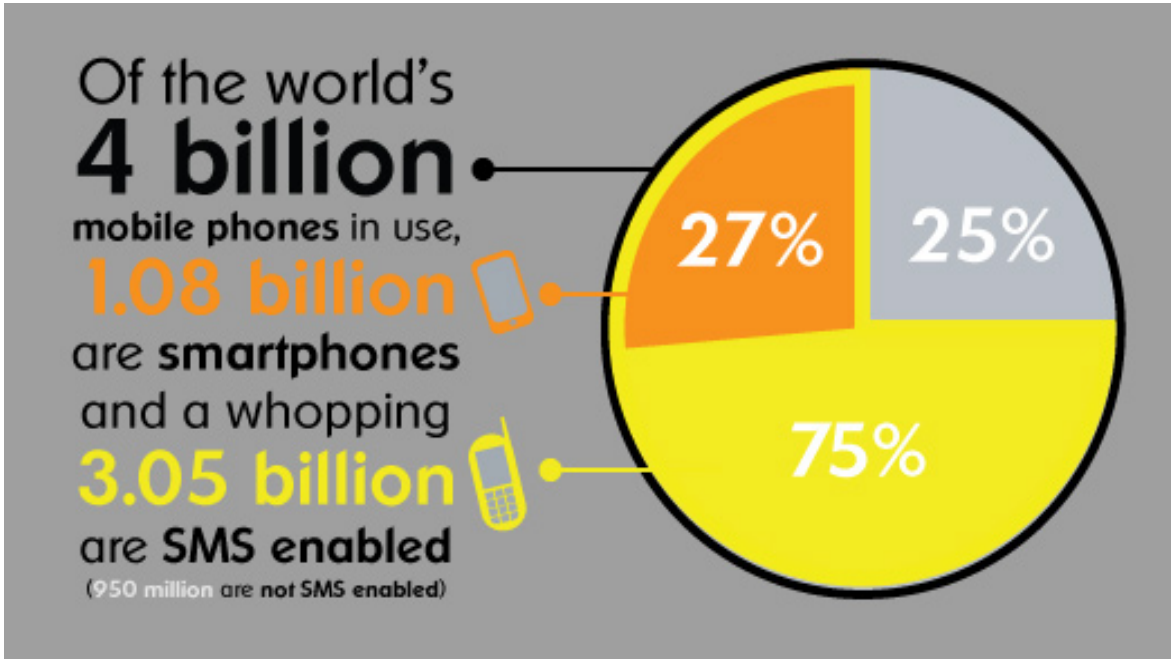
Responsive Web Design

Outline

- The Need - Mobile Growth
- What is Responsive Web Design
- How to Design Responsively
- Major Technology Features
  - media queries
  - fluid grids
  - scalable images
- More Examples of Responsive Websites

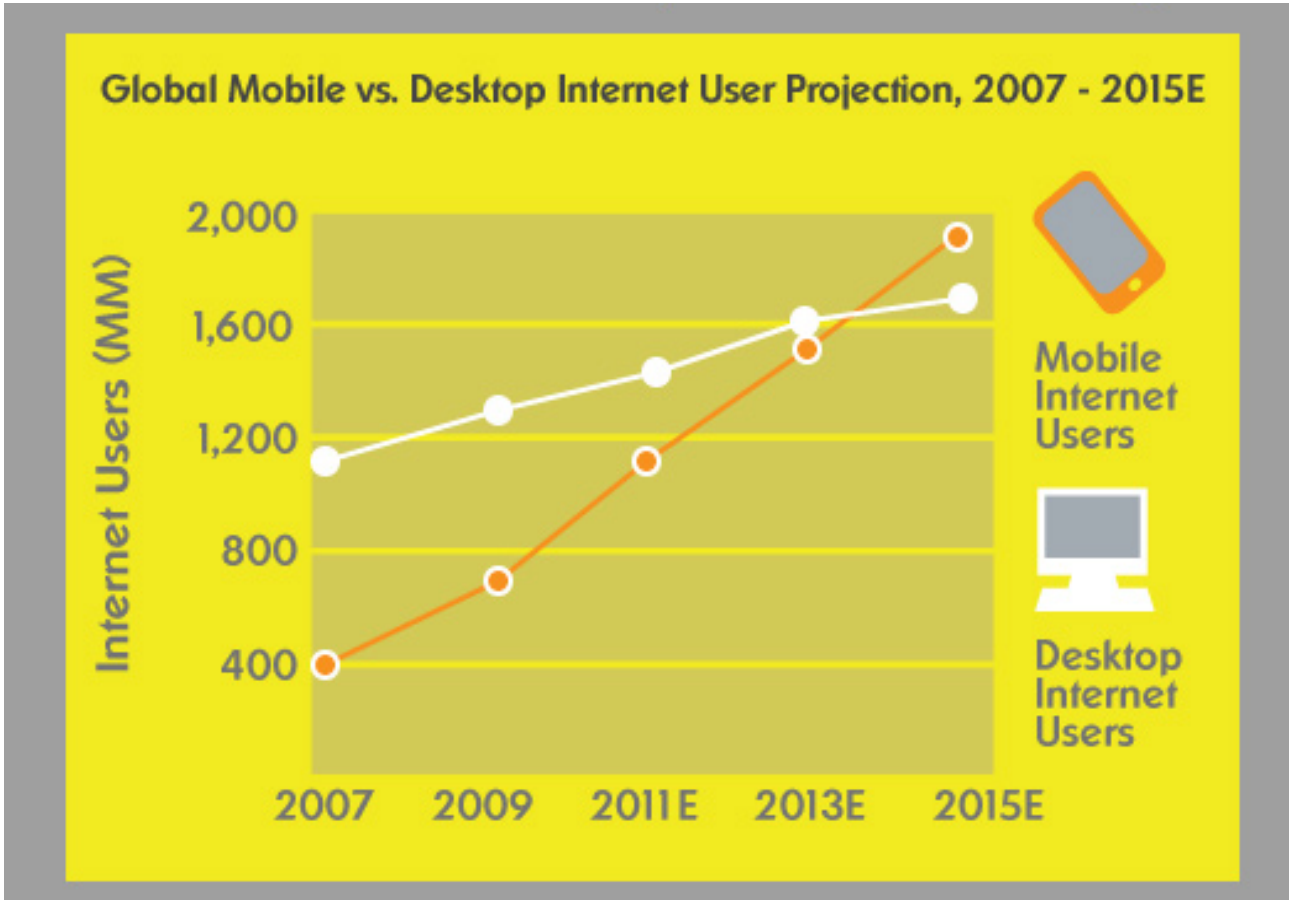
Size of the Mobile Market



The Growth of Mobile Marketing and Tagging by Microsoft Tag

- There is a proliferation of mobile devices worldwide.
- More and more people access Internet only through mobile devices.
- Estimated by the year 2020, 12 billion mobile subscriptions.
- Websites must be designed to make sure the mobile viewer has an excellent experience.

How Fast is the Mobile Internet Growing?



The Growth of Mobile Marketing and Tagging by Microsoft Tag

As of 2014, mobile internet has overtaken desktop internet usage

Mobile Forces You to Focus

- Requires the design team to focus on the most important content, data, and functions.
- There is no space in a 320 by 480 pixel screen (original 2007 iPhone) for extraneous, unnecessary elements
- More recent devices have resolutions of 1,000 ppi, but the screen size is still between 4 and 6 inches
- One is forced to focus on content that's most important to users
  - What features and functionality are essential
  - It may be nice to have, but does it belong on your page
- Designers must prioritize

Building for Mobile Can Extend Your Capabilities

- The design technique of "Mobile first" encourages designers to use the full capabilities of the mobile device.

"Saying mobile design should have less is like saying paperbacks have smaller pages, so we should remove chapters." (Clark)

- Mobile devices offer
  - Use of **geo-location** to optimize the experience.
  - Require **Switch layouts** depending on the way they're held.
  - Need to support rich, **multi-touch** interfaces
    - input devices that recognize two or more simultaneous touches
      - e.g. two finger tap, two finger scroll, pinch, zoom
    - some devices also recognize differences in pressure and temperature (i.e. Apple Watch)

Design for the Mobile Web

- There are three main approaches:
  1. Build an entirely separate .mobi mobile site
     - The domain name **mobi** is a top-level domain. Its name is derived from the adjective *mobile*, indicating it is used by mobile devices for accessing Internet resources via the Mobile Web.
     - The domain was approved by ICANN on 11 July 2005, and is managed by the mTLD global registry
     - To date only 0.06% of web TLDs:

     http://w3techs.com/technologies/details/tld-mobi-/all/all
  2. Host the mobile site within your current domain (m.mycompany.com)
  3. Configure your current website for mobile display using *responsive web design* techniques

Reasons for not using m.mycompany.com websites

1. Redirects can hinder/annoy search engines
2. Redirects take time
3. If you offer an m.website for iPhone, what about for iPad, Android, etc
4. Sharing a mobile m.website will not work for people on laptops, as they will end up with a site designed for a small screen
5. Philosophical: every web resource should live at one URL!

- In conclusion, building a *single responsive website* is the preferable way to go

Responsive Web Design

- **RWD** is the concept of developing a website in a way that **allows the layout to automatically adjust** according to the user's screen resolution (called its *viewport*).

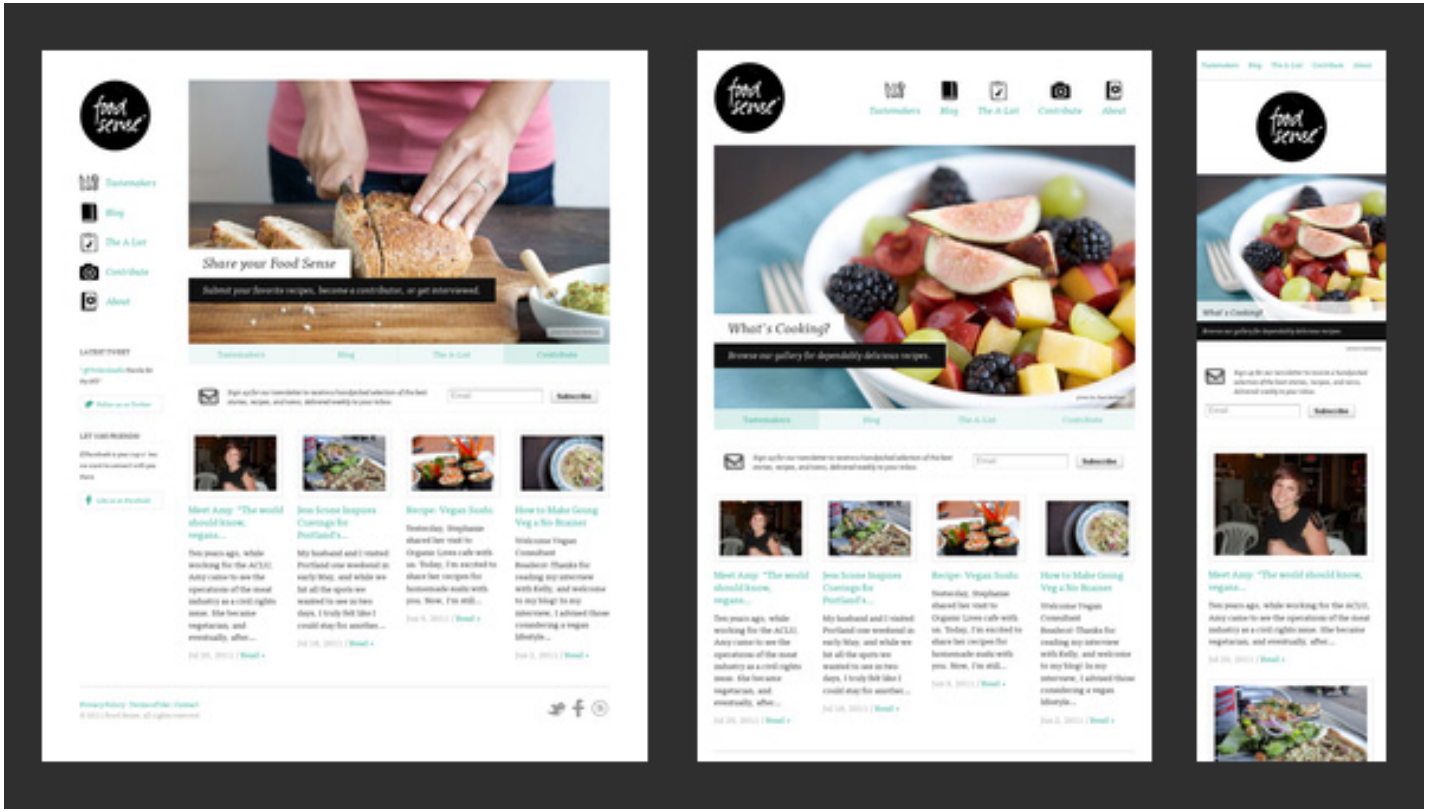
- The viewport meta tag lets you set the width and initial scale of the viewport.
- For example

<meta name="“viewport” content="“width=590”">

- See viewport sizes at

http://viewportsizes.com/

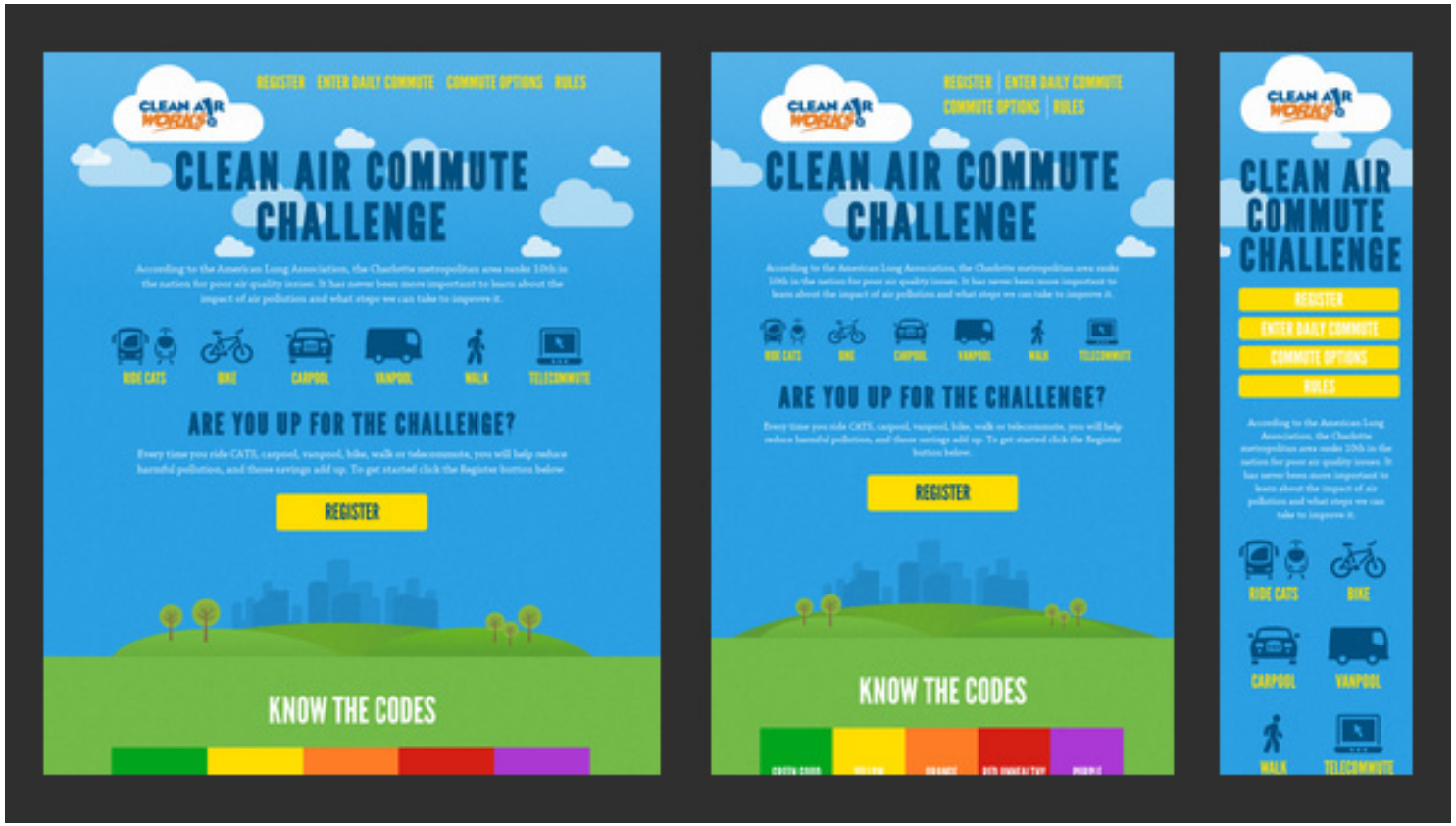
Responsive Web Site Example: Food Sense



Use your laptop/ desktop, tablet and smartphone to check out the website

http://foodsense.is/

Responsive Web Site Example
Clean Air Commute Challenge



http://clearairchallenge.com/

Popular Viewport Sizes

• Smartphones:	Screen Size (pixels)	Pixels/Inch	Screen Size (inch on diag)
iPhone 7 Plus	1920x1080	401 ppi	5.5in
iPhone 7	1334x750	326 ppi	4.7in
Samsung Galaxy S7	2560x1440	534 ppi	5.5in
• Tablets:			
iPad Air 2	2048x1536	264 ppi	9.4in
iPad mini 3	2048x1536	326 ppi	7.9in
Samsung Galaxy Tab S	1600x2560	360 ppi	8.4in, 10.5in
• Notebooks:			
MacBook Air	1440x900	128 ppi	13.3in
MacBook Pro Retina	2880x1800	220 ppi	15.4in
• Desktops:			
iMac	2560x1440	109 ppi	27in
HP XR30W	2560x1600	101 ppi	29.7in

Responsive Web Design

- The term "Responsive Web Design" was coined by Ethan Marcotte in an article on May 25, 2010.
  - for his article see http://www.alistapart.com/articles/responsive-web-design
  - for his website see http://ethanmarcotte.com/
- Responsive web design (RWD) is a web design approach that tries to achieve an ideal viewing experience, which means:
  - easy reading and navigation with a minimum of resizing, panning, and scrolling
  - across a wide range of devices (from mobile phones to desktop monitors)
- A site designed with RWD adapts the layout to the viewing environment by using
  1. fluid, proportion-based grids,
  2. flexible images, and
  3. CSS3 *media queries* (you provide different CSS based upon the viewport size).
     - see the W3C documentation of Media Queries at

     http://www.w3.org/TR/css3-mediaqueries/

Media Queries

- W3C created *media queries* as part of the CSS3 specification.
  - http://www.w3.org/TR/css3-mediaqueries/
- A *media query* consists of a *media type* (`screen`) and the actual query enclosed within parentheses, containing a particular *media feature* (`max-device-width`) to inspect, followed by the *target value* (`480px`). The query can be zero or more expressions that check for the conditions of particular media features
  - **Example below**: Style sheet (example.css) applies to devices of a certain media type (screen) with certain feature (color screen).

```
<link rel="stylesheet" type="text/css"
  media="screen and (color)"
  href="example.css" />
```

- Note: the keyword "`all`" applies to all media types and is the default
- Here are two equivalent pair of media queries

```
<style> ...
@media all and (min-width:500px) { … }
@media (min-width:500px) { … }
</style>
```

Media Queries (cont'd)

- Enhanced media types allows targeting of specific physical characteristics of the device, e.g.

```
<link rel="stylesheet" type="text/css" media="screen and (max-device-width: 480px)" href="min.css" />
```

- A media type (screen), and
- the actual query enclosed within parentheses, containing a particular *media feature* (max-device-width) to inspect, followed by the target value (480px).
- the expression is asking the device if its horizontal resolution (max-device-width) is equal to or less than 480px.

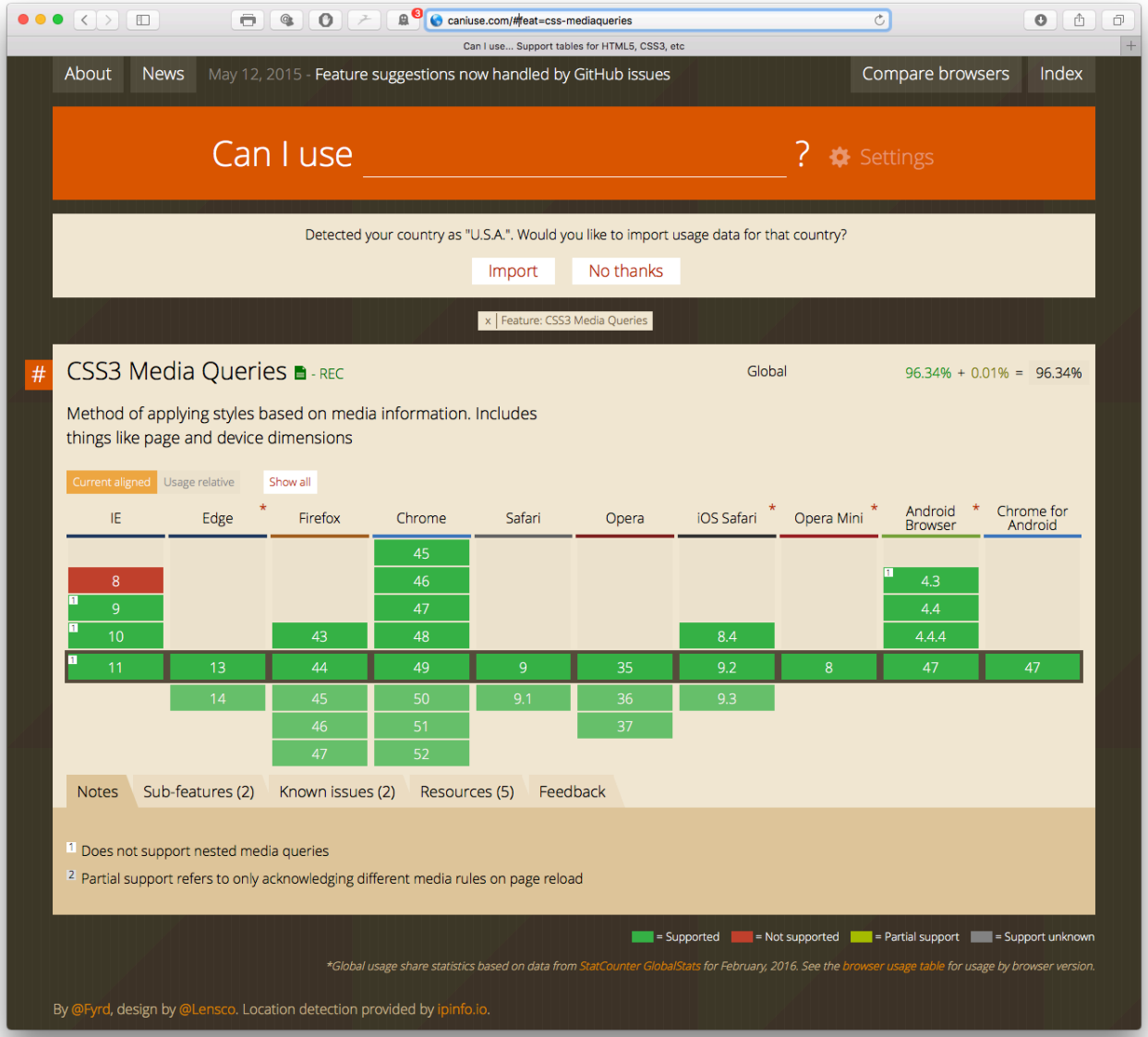
Media Queries (cont'd)

- Multiple property values in a single query can occur by chaining them together with the ***and*** keyword

```
<link rel="stylesheet" media="only screen and (min-width:200px) and (max-width: 500px)" href="small.css">
```

```
<link rel="stylesheet" media="only screen and (min-width:501px) and (max-width: 1100px)"
    href="large.css">
```

caniuse.com

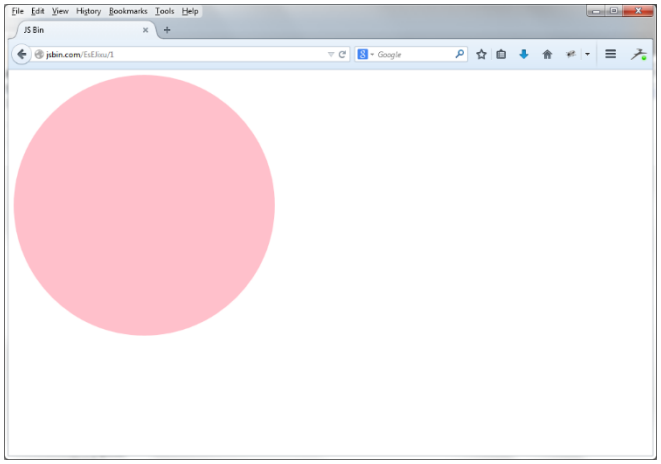


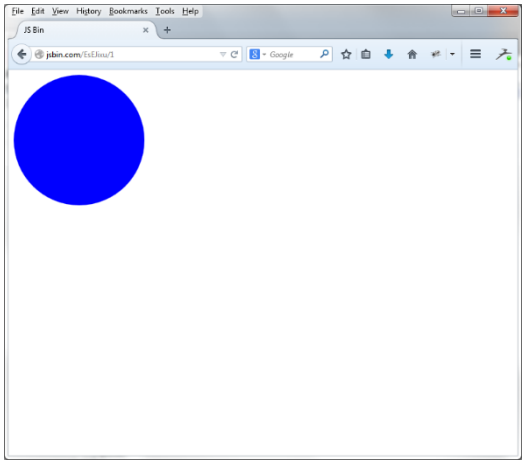
"Can I use" provides up-to-date browser support tables for support of front-end web technologies on desktop and mobile web browsers. The site was built and is maintained by Alexis Deveria".

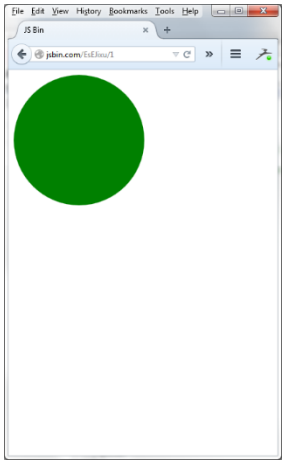
All recent browser versions support media queries.

A Simple Example

- Create a circle that is green for mobile phones, blue for tablets, and pink for desktops (watch with Chrome resizing)







find the code here http://jsbin.com/EsEJixu/1/

A Simple Example: the Source Code

```
<style>
.myCircle {
  width:200px;
  height:200px;
  -webkit-border-radius: 50%;
  -moz-border-radius: 50% ;
  border-radius: 50% ;
  background:blue;
}

@media (max-width: 480px) {
  .myCircle {
      background:red;
  }
}
```

```
@media (max-width: 768px) {
  .myCircle {
      background:green;
  }
}


@media (min-width: 960px) {
  .myCircle {
      background:pink;
      width:400px;
      height:400px;
  }
}
</style>
```

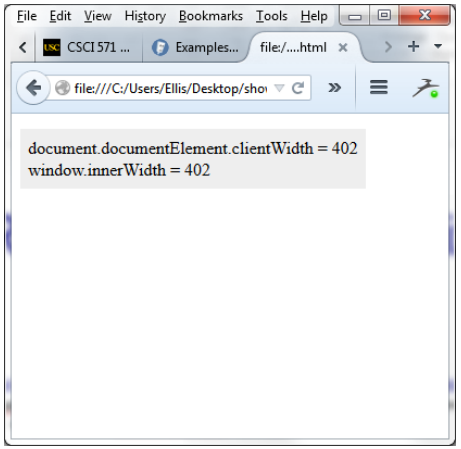
```
<body>
  <div class="myCircle"></div>
</body>
```

Note: to create a circle, width and height must be equal and border-radius set to 50%, which causes the to wrap around

Reporting Screen Size

- Below is a JavaScript program that detects and prints the width of the browser

```
<div style="background:#EEEEEE; position:absolute; top:2ex;
  padding:1ex; z-index:10;">
   document.documentElement.clientWidth =
                  <span id="clientWidth"></span><br />
   window.innerWidth = <span id="innerWidth"></span>
</div>

<script type="text/javascript">
function showViewportSize()   {
   document.getElementById("clientWidth").innerHTML =
  document.documentElement.clientWidth;
   document.getElementById("innerWidth").innerHTML =
  window.innerWidth;
   setTimeout("showViewportSize()", 1250);
}
showViewportSize()
//</script>
```



iPad Specific Media Query Property

- Stephen Gilbert has published specific media queries for iPad, iPad mini and iPhone, see http://stephen.io/mediaqueries/, here are some samples:
- All 5 different iPads (iPads 1-5 and iPad mini) can be targeted with just one CSS media query. Recently updated with Retina iPad and iPad mini, and iPhone 6 & 6 Plus.
- **iPad 3 & 4 in portrait & landscape uses the same media query**

```
@media only screen
and (min-device-width : 768px)
and (max-device-width : 1024px)
and (-webkit-min-device-pixel-ratio: 2)  { /* STYLES GO HERE */}
```

- **iPad mini in landscape**

```
@media only screen
and (min-device-width : 768px)
and (max-device-width : 1024px)
and (orientation : landscape)
and (-webkit-min-device-pixel-ratio: 1)  { /* STYLES GO HERE */}
```

- **iPhone 6/6s in landscape**

```
@media only screen
and (min-device-width : 375px)
and (max-device-width : 667px)
and (orientation : landscape) { /* STYLES GO HERE */}
```

Using JavaScript to Handle Older Browsers

- Many older browsers do not support media queries;
- **css3-mediaqueries.js** by Wouter van der Graaf is a JavaScript library to make IE 5+, Firefox 1+ and Safari 2 transparently parse, test and apply CSS3 Media Queries.
- Firefox 3.5+, Opera 7+, Safari 3+ and Chrome already offer native support.
- Retrieve the library from here:
  - https://css3-mediaqueries-js.googlecode.com/svn/trunk/css3-mediaqueries.js
- P.S.: The percentage of users still using those browsers is **less than 1%**. **No longer recommended.**

Combining Media Queries with JavaScript

- Consider the simple media query below;

```
.sidebar { float: right; width: 250px; }
@media all and (max-device-width: 600px) {
.complicatedFunctionality { display: none; }
.sidebar { float: none; width: auto; }
}
```

- If the width of the device is 600px at maximum, then the styles (`complicatedFunctionality` and `sidebar`) are executed
- However, there are two main problems
  1. even if the media query is true, the browser will still download all of the scripts associated with the website
  2. even if images are hidden from mobile browsers, or low source ones are used, the browser still downloads the full-source variants
- In other words, media queries do NOT stop the browser from downloading assets that will not be used on a mobile phone;
- ***Solution***: use JavaScript to solve the problem

Combining Media Queries with JavaScript

- All browsers have paired the `width` and `device-width` media queries with the values of `document.documentElement.clientWidth` and `screen.width`, respectively, so one can use the following code snippets to download the desired content:

```
if (screen.width >= 600) {
    // download complicated script
    // swap in full-source images for low-source ones
}

or

if (screen.width < 600) {
    // don't download complicated script
    // use low-source images instead of full-source ones
}
```

Hiding Content on Smaller Screens

- In some cases it may be advisable to hide content on the mobile device that would otherwise be visible on the desktop
  - this is done so user's with small screens can avoid long scrolls
  - the usual way to handle this is to provide a link to "additional content"
- CSS allows us to show and hide content using

```
display: none;
```

- Either declare `display: none` for the HTML block element that needs to be hidden in a specific style sheet or detect the browser width and do it through JavaScript.
- **Note** that `visibility: hidden` just hides the content (although it is still there), whereas the display property gets rid of it altogether.
- See Google on Building Smartphone-Optimized Websites

`https://developers.google.com/webmasters/smartphone-sites/details`

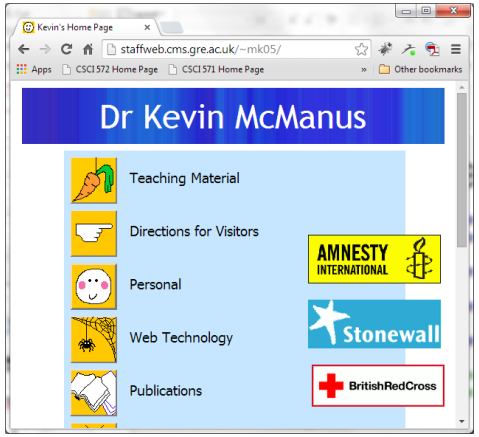
Usability Guidelines for Websites on Mobile Devices

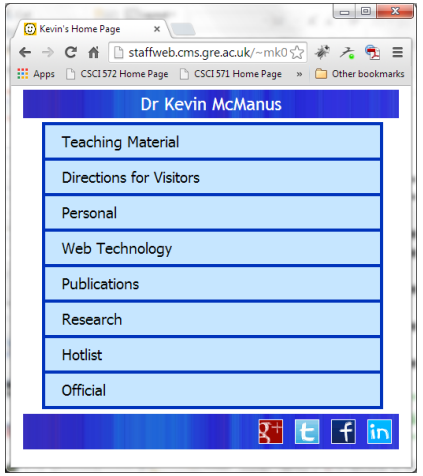
1. Reduce the amount of content
2. Single column layouts work best
3. Present the navigation differently (in a menu, dropdown, or at the bottom)
4. Minimize text entry
   - request PIN instead of password
5. Decide how many mobile sites you need
   - consider users with slower connections and possibly make a lower resolution site
6. Design for touchscreen and non-touchscreen users
7. Take advantage of smartphone built-in functionality
   - see an address on a map
   - find the nearest

See: http://www.webcredible.com/blog-reports/web-usability/mobile-guidelines.shtml

Another Example: http://staffweb.cms.gre.ac.uk/~mk05/

```
<link rel="stylesheet" type="text/css" href="index.css"/>
<link rel="stylesheet" type="text/css" media="screen and (max-width: 710px)"
href="indexSmall.css" />
<link rel="stylesheet" type="text/css" media="screen and (max-width: 610px)"
href="indexTiny.css" />
<link rel="stylesheet" type="text/css" media="screen and (max-width: 570px)"
href="indexMobile.css" />
<link rel="stylesheet" type="text/css" media="screen and (min-width: 830px)"
href="indexLarge.css" />
<!-- <link rel="stylesheet" media="screen and (min-device-width: 800px)" href="index800.css"
/>
<link rel='stylesheet' media='screen and (min-width: 701px) and (max-width: 900px)'
href='css/medium.css' /> -->
```





Contents of the CSS Files

- ***IndexLarge.css***

```
body { width:800px; margin:4px auto 0px auto; }
```

Allows the main table to shrink at 800px

- ***IndexSmall.css***

```
#campaigns { position:relative; top:-60ex; text-align:right; z-
  index:4; }
#mainMenu { background: #BBEEFF; width:80%; margin-left:10%;
  padding:0px; position:relative; top:0px; z-index:2; }
#menuColA { width:100%; float:none; padding:4px 0px 0px 4px; }
#menuColB { width:100%; padding:0px 0px 4px 4px; }
```

Switches to a single column main table at 710px

- ***IndexTiny.css***

```
#conformance { display:none; }
```

Removes the W3C conformance badges at 610px

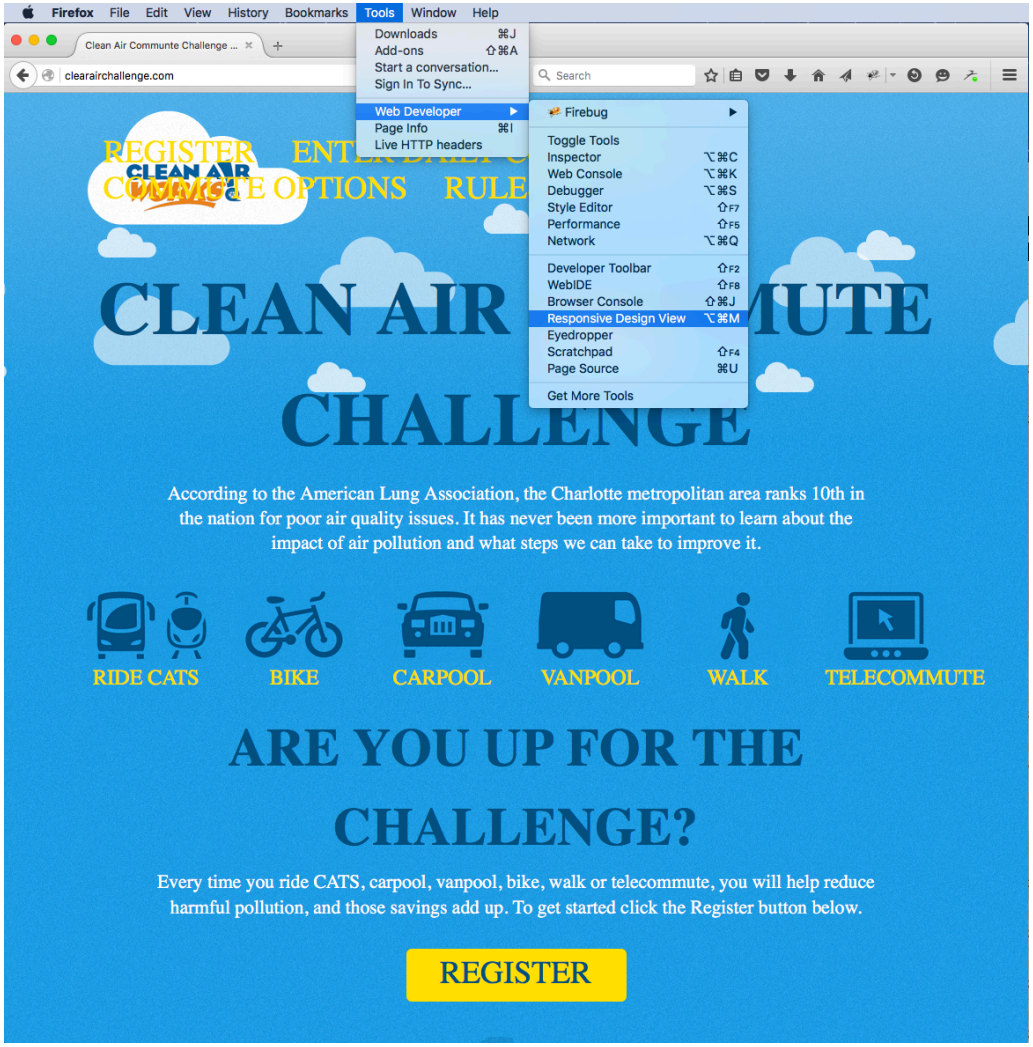
Contents of the CSS Files (cont'd)

- ***IndexMobile.css***

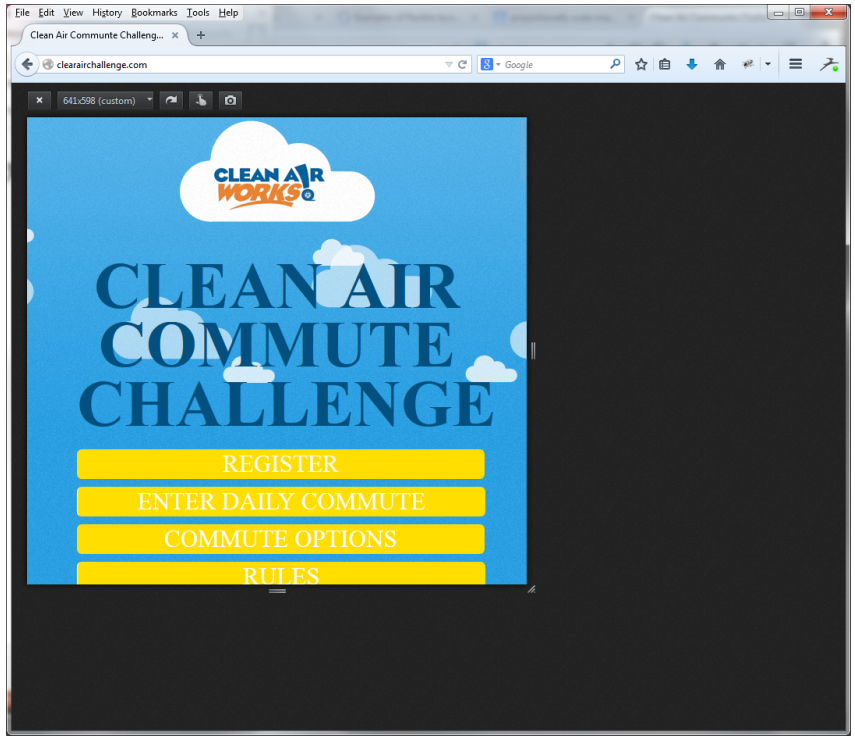
```
h1 {  font-size:1.4em; }

img.button { display:none; }

#mainMenu { width:90%; margin:0px 5% 0px 5%; padding:0px; background:blue;
   }
div.menuPanel { background:#C6E6F0; margin:0px; padding:0px; border:solid
   2px #0033BB; }
#menuColA { padding:0px; border-bottom:none; margin:0px; }
#menuColB { padding:0px; border-top:none; margin:0px; }
div.menuItem { padding:1ex; margin:0px; border:solid 2px #0033BB;  }

#copy { display:none; }
#sig { display:none; }
#campaigns { display:none; }
```

Removes the buttons and campaigns (leaves social media icons), removed padding and changes to a single column menu

Firefox Includes a Tool to Check for Responsive Design View



Firefox Responsive Design View



Creating Fluid Grids

- In adaptive grids, we define pixel-based dimensions.
  - Hence we have to adjust the widths and heights manually for certain device viewports.
- In fluid grids we define relative-based dimensions.
  - Since fluid grids flow naturally within the dimensions of its parent container, limited adjustments will be needed for various screen sizes and devices.
- In fluid grids we
  1. Define a maximum layout size for the design.
  2. The grid is divided into a specific number of columns to keep the layout clean and easy to handle.
  3. Then we design each element with proportional widths and heights instead of pixel based dimensions.
- So whenever the device or screen size is changed, elements will adjust their widths and heights by the specified proportions to its parent container.

Converting Fixed Pixel Sizes to Percentages

- To transform pixel-based column widths into percentage-based, *flexible* measurements use the formula: `target ÷ context = result`
- If the initial value of the page's title is 700px — but it's contained within a designed width of 988px, divide 700px (the target) by 988px (the context) like so:
- `700 ÷ 988 = 0.7085`
- And 0.7085 translates into 70.85%, a width one can drop directly into the stylesheet:

```
h1 { width: 70.85%;              /* 700px / 988px = 0.7085 */  }
```

- we can do the same with the margin of 144px (288px ÷ 2):
- `144 ÷ 988 = 0.14575`
- the 0.14575 becomes 14.575%, and add that directly to the style rule as a value for the title's margin-left:

```
h1 { margin-left: 14.575%;     /* 144px / 988px = 0.14575 */
     width: 70.85%;           /* 700px / 988px = 0.7085 */   }
```

Flexible Images

- To avoid having an image deformed due to the screen size one should avoid specific definitions of width and height and instead use CSS's `max-width` property setting it to 100%

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

- As long as no other width-based image styles override this rule, every image will load in its original size, unless the viewing area becomes narrower than the image's original width.
  - With the maximum width of the image set to 100% of the screen or browser width, if the screen becomes narrower, so does the image.
- The browser will resize the images as needed using CSS to guide their relative size.

An Example of Scaled Images

Scaled to the size of the containing element

ul#image-icons li img { max-width:100% }

More Examples of Responsive Web Sites

For some additional examples of responsive web design see:

- http://hicksdesign.co.uk/
- http://2011.uxlondon.com/
- http://www.stpaulsschool.org.uk/
- http://css-tricks.com/
- http://yiibu.com/
- http://sickdesigner.com/
- http://ethanmarcotte.com/
- http://foodsense.is
- http://earthhour.fr
- http://w3conf.org
- http://mediaqueri.es
- http://fourkitchens.com
- http://achieveinternet.com

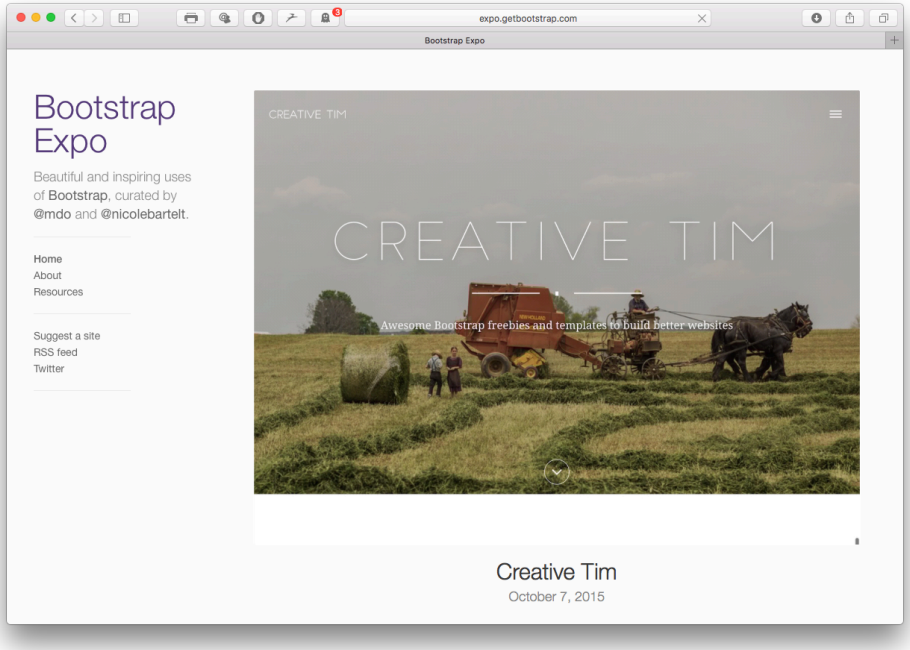
Bootstrap

- Bootstrap is a powerful front-end framework for faster and easier responsive web development.
- It includes HTML and CSS based design templates for common user interface components like Typography, Forms, Buttons, Tables, Navigations, Dropdowns, Alerts, Modals, Tabs, Accordion, Carousel and many other as well as optional JavaScript extensions.
- Bootstrap also gives you ability to create responsive layout with much less effort
- Bootstrap responsive features make your web pages to appear more appropriately on different devices and screen resolutions without any change in markup

Getting Started

- There are two versions available for download, **compiled Bootstrap** and **Bootstrap source** files. Current version is 3.3.6.
- Go to http://getbootstrap.com/ and click on Download Bootstrap
- Select the compiled and minified CSS, JavaScript and fonts
- See samples of uses of Bootstrap at:

  http://expo.getbootstrap.com



Supported Browsers

- Specifically, Bootstrap support the latest versions of the major browsers and platforms. On Windows, it fully supports Internet Explorer 11. Partial support is available for IE 8-10.

Mobile devices

Generally speaking, Bootstrap supports the latest versions of each major platform's default browsers. Note that proxy browsers (such as Opera Mini, Opera Mobile's Turbo mode, UC Browser Mini, Amazon Silk) are not supported.

	Chrome	Firefox	Safari
Android	✔ Supported	✔ Supported	N/A
iOS	✔ Supported	✔ Supported	✔ Supported

Desktop browsers

Similarly, the latest versions of most desktop browsers are supported.

	Chrome	Firefox	Internet Explorer	Opera	Safari
Mac	✔ Supported	✔ Supported	N/A	✔ Supported	✔ Supported
Windows	✔ Supported	✔ Supported	✔ Supported	✔ Supported	✖ Not supported

The Bootstrap file structure

- bootstrap/
- |——— css/
- | |——— bootstrap.css
- | |——— bootstrap.css.map
- | |——— bootstrap.min.css
- | |——— bootstrap.min.css.map
- | |——— bootstrap-theme.css
- | |——— bootstrap-theme.css.map
- | |——— bootstrap-theme.min.css
- | |——— bootstrap-theme.min.css.map
- |——— js/
- | |——— bootstrap.js
- | |——— bootstrap.min.js
- |——— fonts/
- | |——— glyphicons-halflings-regular.eot
- | |——— glyphicons-halflings-regular.svg
- | |——— glyphicons-halflings-regular.ttf
- | |——— glyphicons-halflings-regular.woff
- | |——— glyphicons-halflings-regular.woff2

Compiled CSS and JS files, bootstrap.* as well as Compiled and minified CSS and JS (bootstrap.min.*)

Four font files (glyphicons-halflings-regular.* in the fonts folder)

Creating a Basic Bootstrap Template File

```
<!DOCTYPE html>
   <html>
   <head>
       <meta charset="utf-8">
       <title>Basic Bootstrap Template</title>
       <meta name="viewport" content="width=device-width, initial-scale=1.0">
       <link rel="stylesheet" type="text/css" href="css/bootstrap.min.css">
   </head>
   <body>
       <h1>Hello, world!</h1>
   <script src="http://code.jquery.com/jquery.min.js"></script>
   <script src="js/bootstrap.min.js"></script>
   </body>
   </html>
```

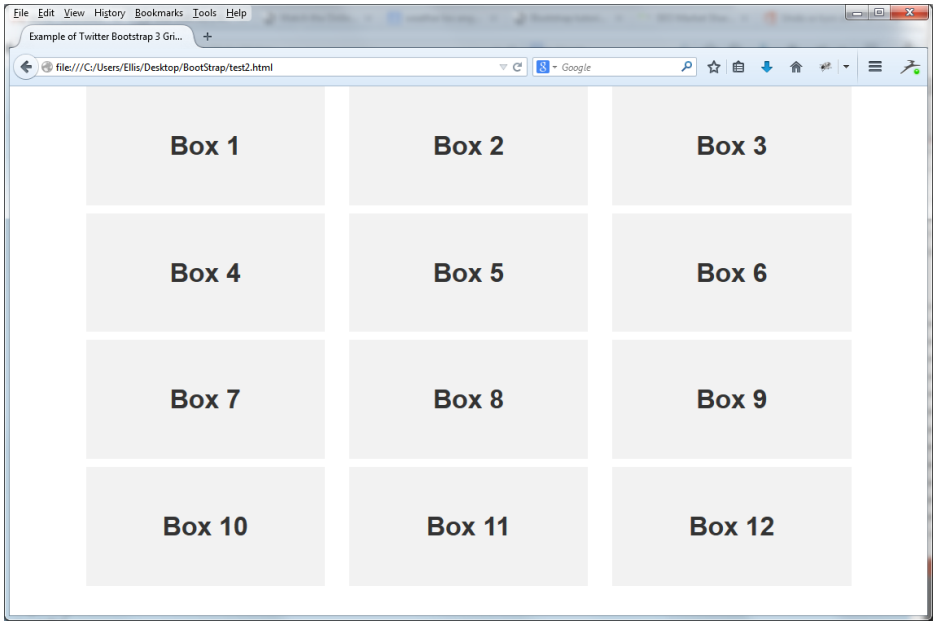
Bootstrap Grid System

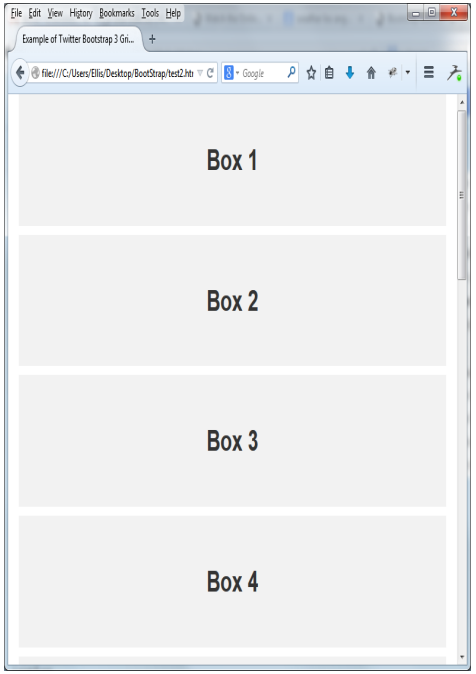
- Bootstrap 3 introduces the responsive mobile first ***fluid grid system*** that appropriately scales up to 12 columns as the device or viewport size increases.
- Bootstrap 3 includes predefined grid classes for quickly making grid layouts for different types of devices like phones (xs), tablets (sm), desktops (md),and larger desktops (lg). For example,
  - you can use the ***.col-xs-*** class to create grid columns for extra small devices like smartphones;
  - similarly the ***.col-sm-*** for small screen devices like tablets;
  - the ***.col-md-*** class for medium size devices like desktop, and
  - the ***.col-lg-*** for large desktop screens.

Creating Layouts with Bootstrap Grid System

- In the above illustration there are total 12 content boxes in all devices, but its placement vary according to the device screen size,
- E.g. the mobile device layout is rendered as one column grid layout , with 1 column and 12 rows placed above one another,
- in tablet it is rendered as two column grid layout which has 2 columns and 6 rows.
- in medium screen size devices like laptop and desktop it is rendered as three column grid layout which as 3 columns and 4 rows, and
- in large screen devices like large desktop it is rendered as four column grid layout which has 4 columns and 3 rows.

Browser Output





In a laptop or desktop having screen or viewport width greater than or equal to 992px and less than 1200px;
it has 4 rows where each row has 3 equal columns resulting in 3x4 grid layout.

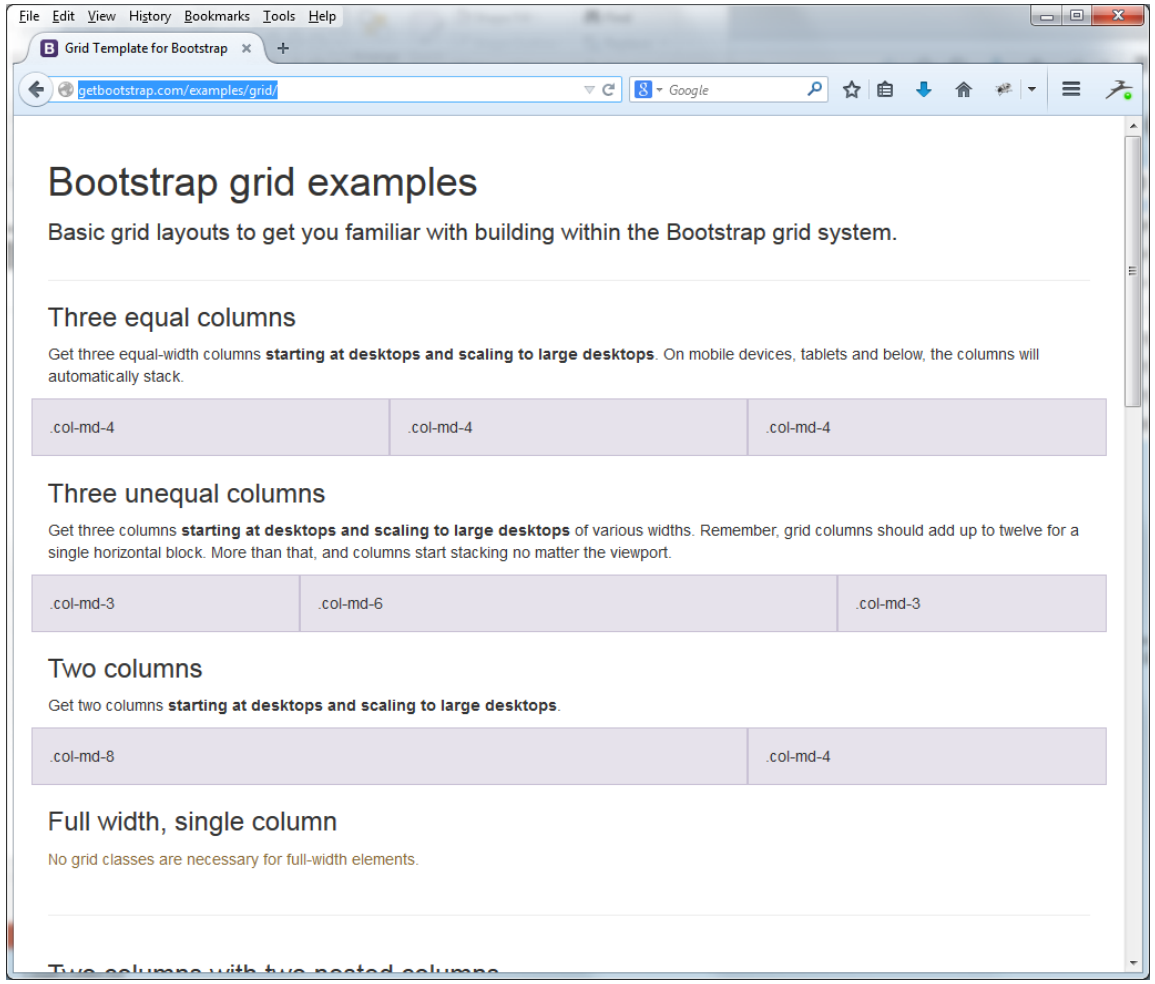
Bootstrap Grid System

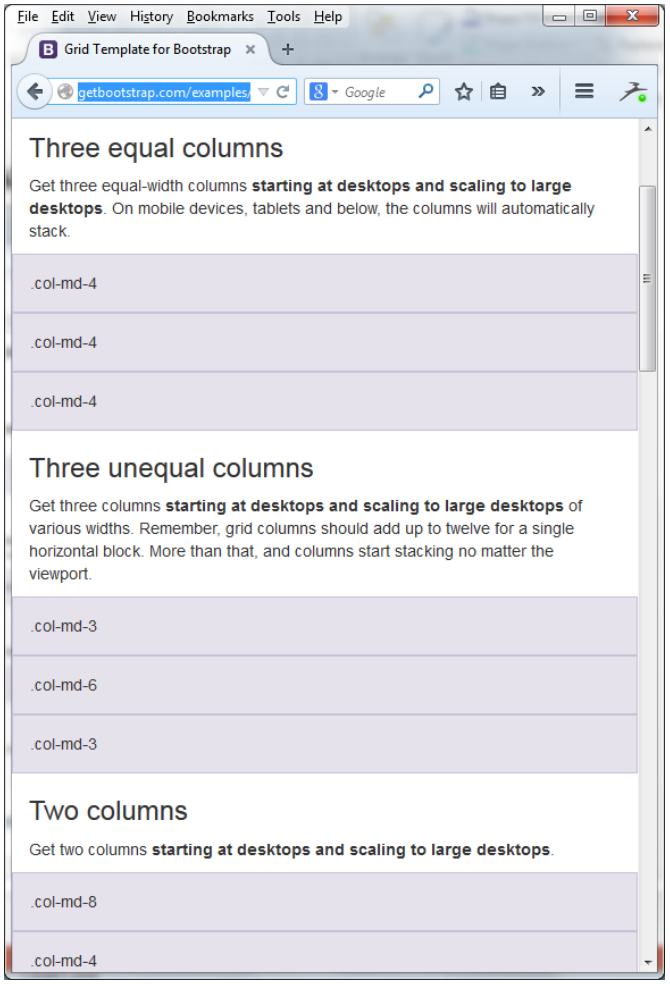
Applying any .col-sm- class to an element will not only affect its styling on small devices like tablets, but also on medium and large devices having screen size greater than or equal to 768px (i.e. ≥768px) if .col-md- and .col-lg- class is not present.

Similarly the .col-md- class will not only affect the styling of elements on medium devices, but also on large devices if a .col-lg- class is not present

Features Bootstrap 3 Grid System	**Extra small devices** Phones (<768px)	**Small devices** Tablets (≥768px)	**Medium devices** Desktops (≥992px)	**Large devices** Desktops (≥1200px)
Max container width	None (auto)	750px	970px	1170px
Class prefix	.col-xs-	.col-sm-	.col-md-	.col-lg-
Max column width	Auto	~62px	~81px	~97px
Gutter width	15px on each side of a column (i.e. 30px)			

Five Bootstrap Grid Examples



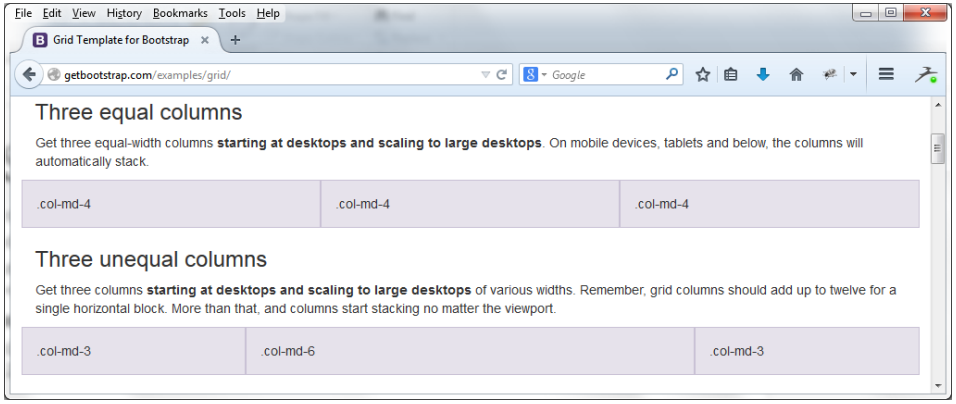


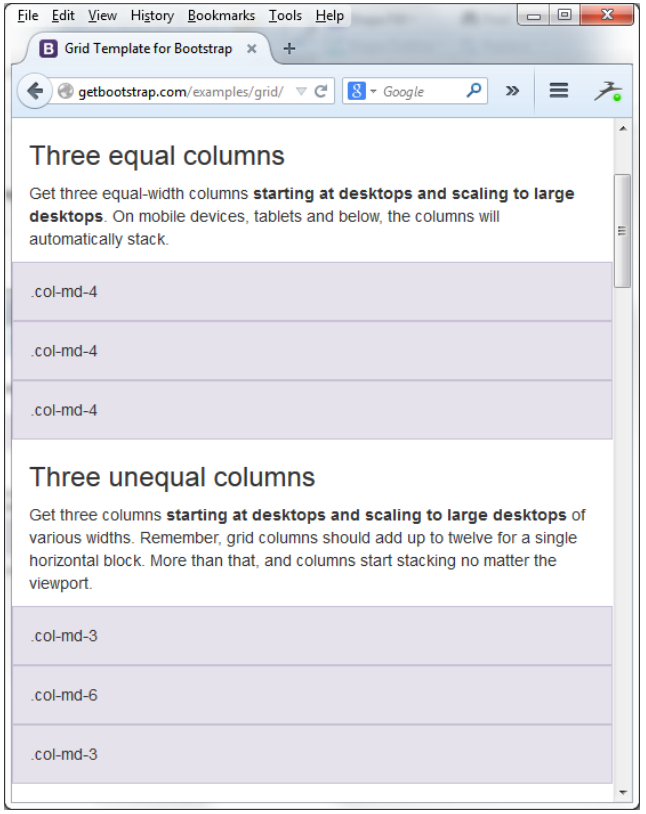
http://getbootstrap.com/examples/grid/

Source Code (1 of 5)

```
<!DOCTYPE html><html lang="en"><head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">
    <link rel="icon" href="../../favicon.ico">
    <title>Grid Template for Bootstrap</title>
    <!-- Bootstrap core CSS -->
    <link href="../../dist/css/bootstrap.min.css" rel="stylesheet">
    <!-- Custom styles for this template -->
    <link href="grid.css" rel="stylesheet">
    <!-- Just for debugging purposes. Don't actually copy these 2 lines! -->
    <!--[if lt IE 9]><script src="../../assets/js/ie8-responsive-file-
warning.js"></script><![endif]-->
    <script src="../../assets/js/ie-emulation-modes-warning.js"></script>
    <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->
    <script src="../../assets/js/ie10-viewport-bug-workaround.js"></script>
```

Three Equal/Unequal Columns

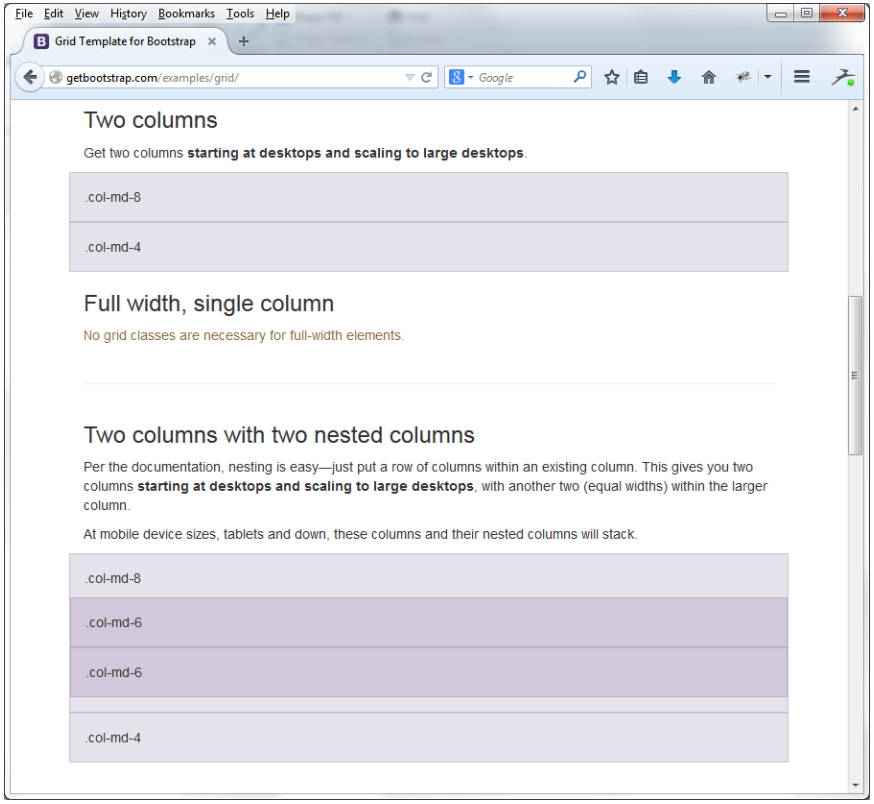


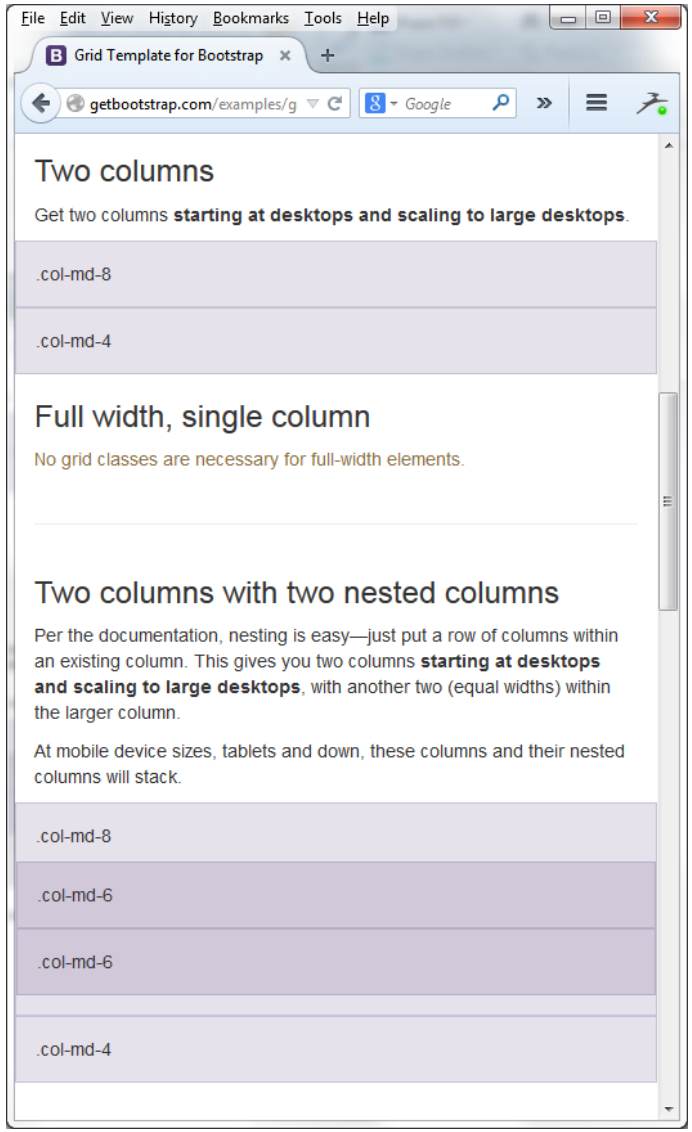


Source Code (2 of 5)

```
</head><body><div class="container">
      <div class="page-header"><h1>Bootstrap grid examples</h1>
        <p class="lead">Basic grid layouts to get you familiar with building within the
Bootstrap grid system.</p></div>
      <h3>Three equal columns</h3>
      <p>Get three equal-width columns <strong>starting at desktops and scaling to large
desktops</strong>. On mobile devices, tablets and below, the columns will automatically
stack.</p>
      <div class="row">
        <div class="col-md-4">.col-md-4</div>
        <div class="col-md-4">.col-md-4</div>
        <div class="col-md-4">.col-md-4</div>
      </div>
       <h3>Three unequal columns</h3>
      <p>Get three columns <strong>starting at desktops and scaling to large
desktops</strong> of various widths. Remember, grid columns should add up to twelve for a
single horizontal block. More than that, and columns start stacking no matter the
viewport.</p>
      <div class="row">
        <div class="col-md-3">.col-md-3</div>
        <div class="col-md-6">.col-md-6</div>
        <div class="col-md-3">.col-md-3</div>
      </div>
```

Two Columns, Two with Nested Columns

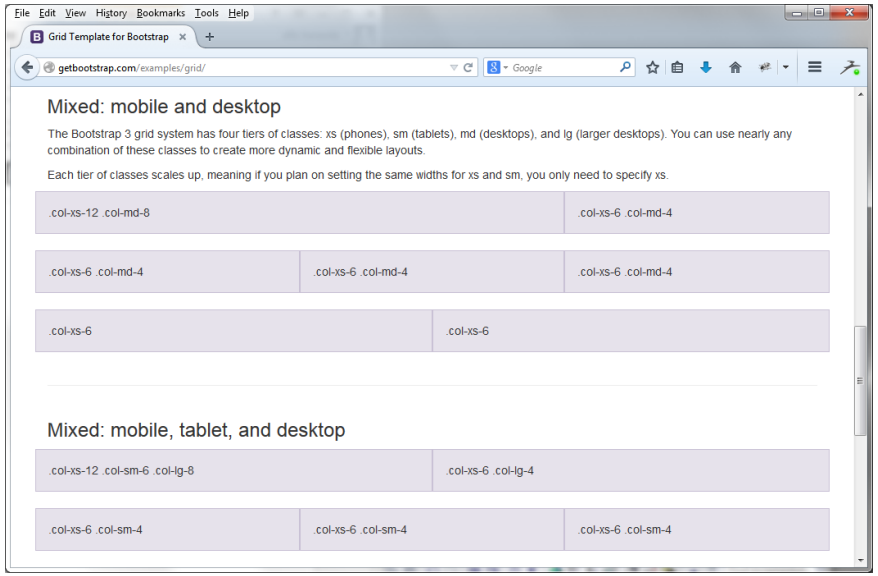


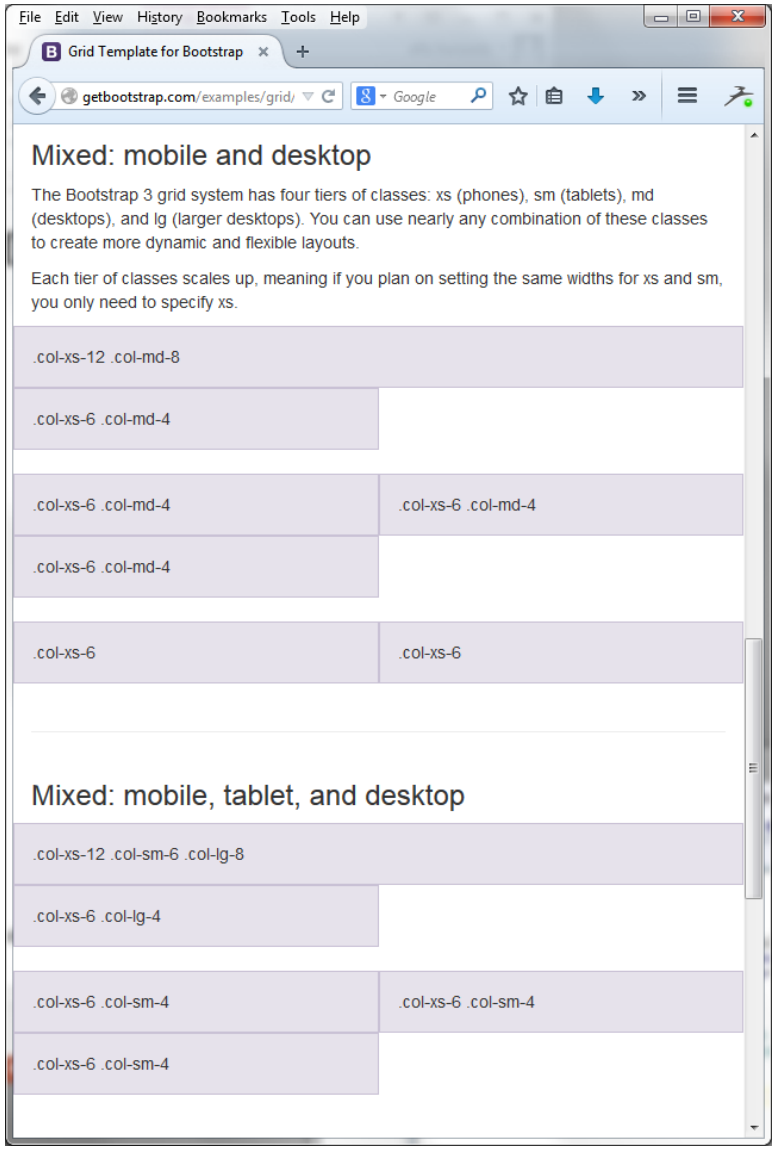


Source Code (3 of 5)

```
<h3>Two columns</h3>
      <p>Get two columns <strong>starting at desktops and scaling to large
desktops</strong>.</p>
      <div class="row">
        <div class="col-md-8">.col-md-8</div>
        <div class="col-md-4">.col-md-4</div>
      </div>
      <h3>Full width, single column</h3>
      <p class="text-warning">No grid classes are necessary for full-width elements.</p>
      <hr><h3>Two columns with two nested columns</h3>
      <p>Per the documentation, nesting is easy—just put a row of columns within an
existing column. This gives you two columns <strong>starting at desktops and scaling to
large desktops</strong>, with another two (equal widths) within the larger column.</p>
      <p>At mobile device sizes, tablets and down, these columns and their nested columns
will stack.</p>
      <div class="row">
        <div class="col-md-8">
          .col-md-8
          <div class="row">
            <div class="col-md-6">.col-md-6</div>
            <div class="col-md-6">.col-md-6</div>
          </div>
        </div>
        <div class="col-md-4">.col-md-4</div>
      </div>
      <hr>
```

Mixed Mobile, Desktop, Tablet

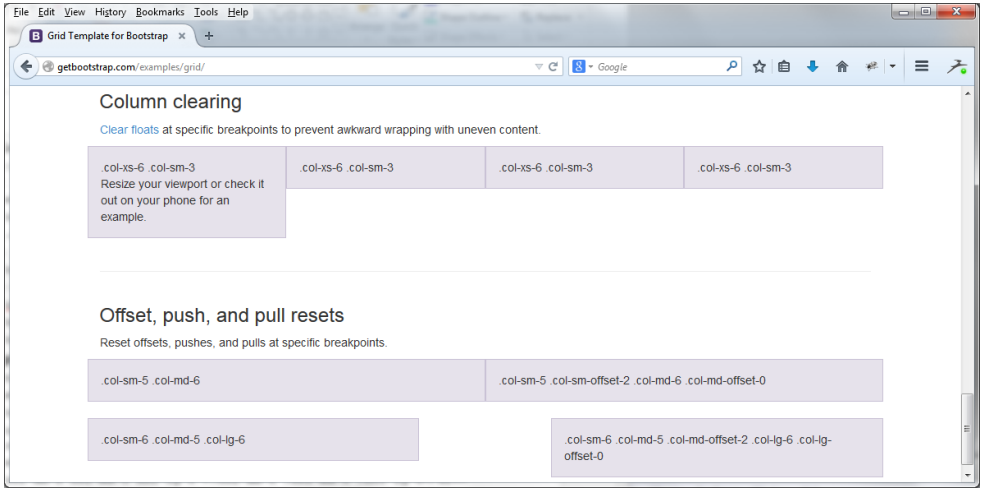


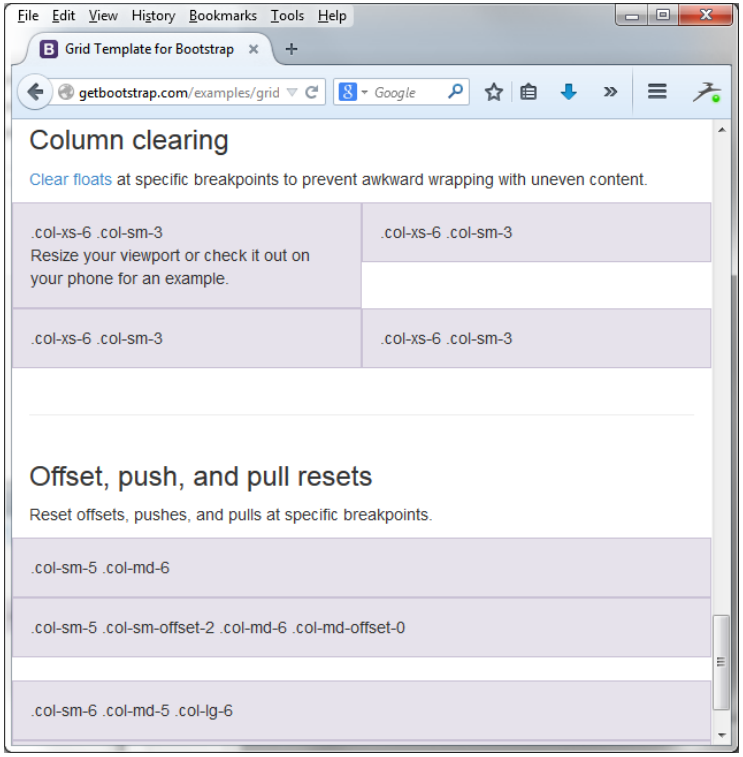


Source Code (4 of 5)

```
<h3>Mixed: mobile and desktop</h3>
      <p>The Bootstrap 3 grid system has four tiers of classes: xs (phones), sm
(tablets), md (desktops), and lg (larger desktops). You can use nearly any combination of
these classes to create more dynamic and flexible layouts.</p>
      <p>Each tier of classes scales up, meaning if you plan on setting the same widths
for xs and sm, you only need to specify xs.</p>
      <div class="row">
        <div class="col-xs-12 col-md-8">.col-xs-12 .col-md-8</div>
        <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div></div>
      <div class="row">
        <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
        <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
        <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div></div>
      <div class="row">
        <div class="col-xs-6">.col-xs-6</div>
        <div class="col-xs-6">.col-xs-6</div></div>
<hr><h3>Mixed: mobile, tablet, and desktop</h3>
      <p></p>
      <div class="row">
        <div class="col-xs-12 col-sm-6 col-lg-8">.col-xs-12 .col-sm-6 .col-lg-8</div>
        <div class="col-xs-6 col-lg-4">.col-xs-6 .col-lg-4</div></div>
      <div class="row">
        <div class="col-xs-6 col-sm-4">.col-xs-6 .col-sm-4</div>
        <div class="col-xs-6 col-sm-4">.col-xs-6 .col-sm-4</div>
        <div class="col-xs-6 col-sm-4">.col-xs-6 .col-sm-4</div></div>
```

Column Clearing; Offset





Source Code (5 of 5)

```
<hr><h3>Column clearing</h3>
      <p><a href="http://getbootstrap.com/css/#grid-responsive-resets">Clear floats</a> at
specific breakpoints to prevent awkward wrapping with uneven content.</p>
      <div class="row">
        <div class="col-xs-6 col-sm-3">
          .col-xs-6 .col-sm-3
          <br>Resize your viewport or check it out on your phone for an example.</div>
        <div class="col-xs-6 col-sm-3">.col-xs-6 .col-sm-3</div>
        <!-- Add the extra clearfix for only the required viewport -->
        <div class="clearfix visible-xs"></div>
        <div class="col-xs-6 col-sm-3">.col-xs-6 .col-sm-3</div>
        <div class="col-xs-6 col-sm-3">.col-xs-6 .col-sm-3</div></div><hr>
<h3>Offset, push, and pull resets</h3>
      <p>Reset offsets, pushes, and pulls at specific breakpoints.</p>
      <div class="row">
        <div class="col-sm-5 col-md-6">.col-sm-5 .col-md-6</div>
        <div class="col-sm-5 col-sm-offset-2 col-md-6 col-md-offset-0">.col-sm-5 .col-sm-
offset-2 .col-md-6 .col-md-offset-0</div></div>
      <div class="row">
        <div class="col-sm-6 col-md-5 col-lg-6">.col-sm-6 .col-md-5 .col-lg-6</div>
        <div class="col-sm-6 col-md-5 col-md-offset-2 col-lg-6 col-lg-offset-0">.col-sm-6
.col-md-5 .col-md-offset-2 .col-lg-6 .col-lg-offset-0</div>
      </div></div> <!-- /container --><!-- Bootstrap core JavaScript
    ================================================== -->
    <!-- Placed at the end of the document so the pages load faster --></body></html>
```

Introduction to Brackets

- Brackets is an open-source code editor focused on web development and built with HTML, JavaScript, and CSS.
  - It was originally released in July 2012 on GitHub (http://github.com/adobe/brackets).
  - Download Brackets here, http://brackets.io/
  - While launched by Adobe, the commiters behind Brackets include folks from numerous sources.
- Brackets supports web development and provides code hinting for HTML, CSS, and JavaScript
- The ***Live Preview*** feature connects your Brackets editor to your browser. As you edit CSS, updates happen in real time providing instant feedback.
- The ***Quick Editing*** feature let's you select an HTML tag and instantly get to the CSS code that applies to that part of the DOM.
- Brackets offers an API that can be used by developers to add whatever feature they want.
  - Extensions have been created for CSS linting, HTML validation, GitHub integration, and more.

Some Brackets Features

- Live connect:
  - As you modify your HTML code the browser will update in real time.
  - You will also see highlights in the DOM for the area you're modifying
- Integration with Theseus
  - Theseus is an open source project created by folks from both Adobe and MIT. It is focused on providing debugging support for both Chrome *and* Node.js applications.
  - Imagine being able to debug a Node.js application made up of server-side JavaScript as well as client-side code.
  - Theseus is now integrated into Brackets and can be used within the editor itself
- See videos on Brackets at

https://www.youtube.com/watch?v=rvo3Mv1Z4qU&feature=youtu.be

http://www.lynda.com/Brackets-tutorials/Adobe-Edge-Code-Brackets-First-Look/149704-2.html