CSS: Cascading Style Sheets

RESPONSIVE DESIGN

POLITECNICO
DI TORINO

e-Lite

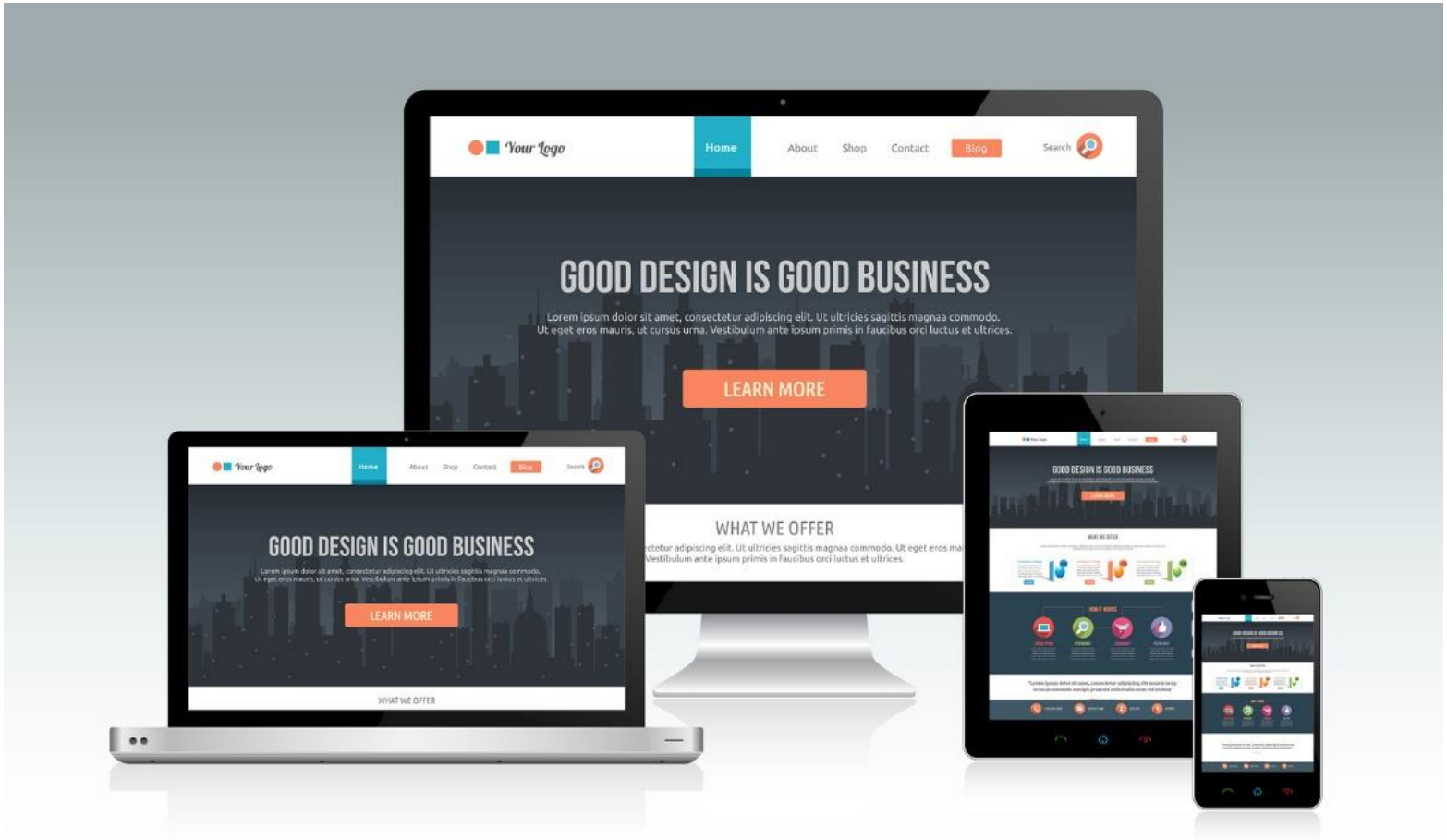
Summary

- CSS media queries and responsive web design
- CSS responsive images
- CSS speech module

- Some more properties

CSS RESPONSIVE DESIGN AND MEDIA QUERIES

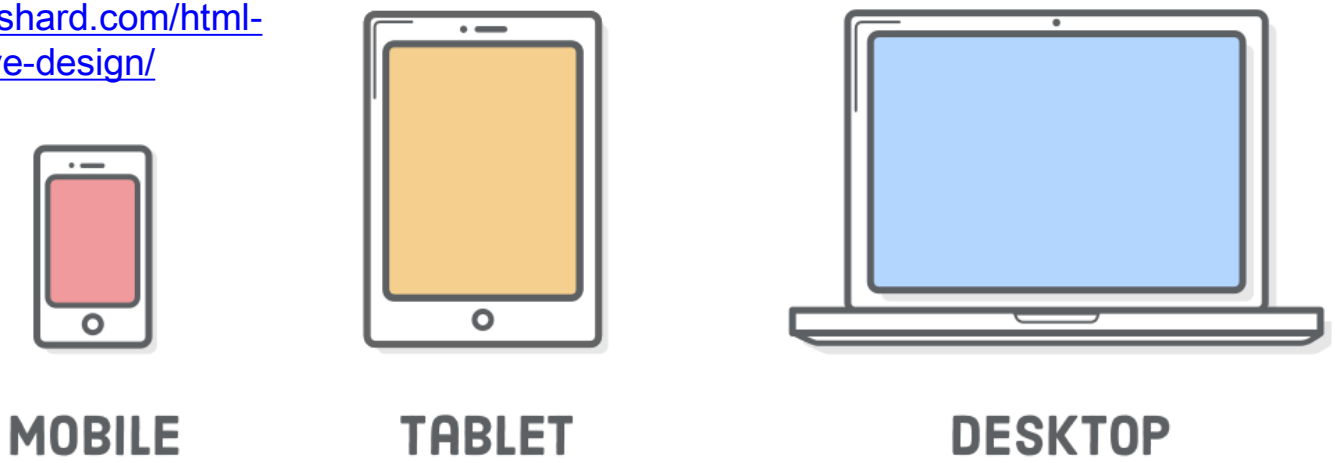
Responsive web design



Responsive design

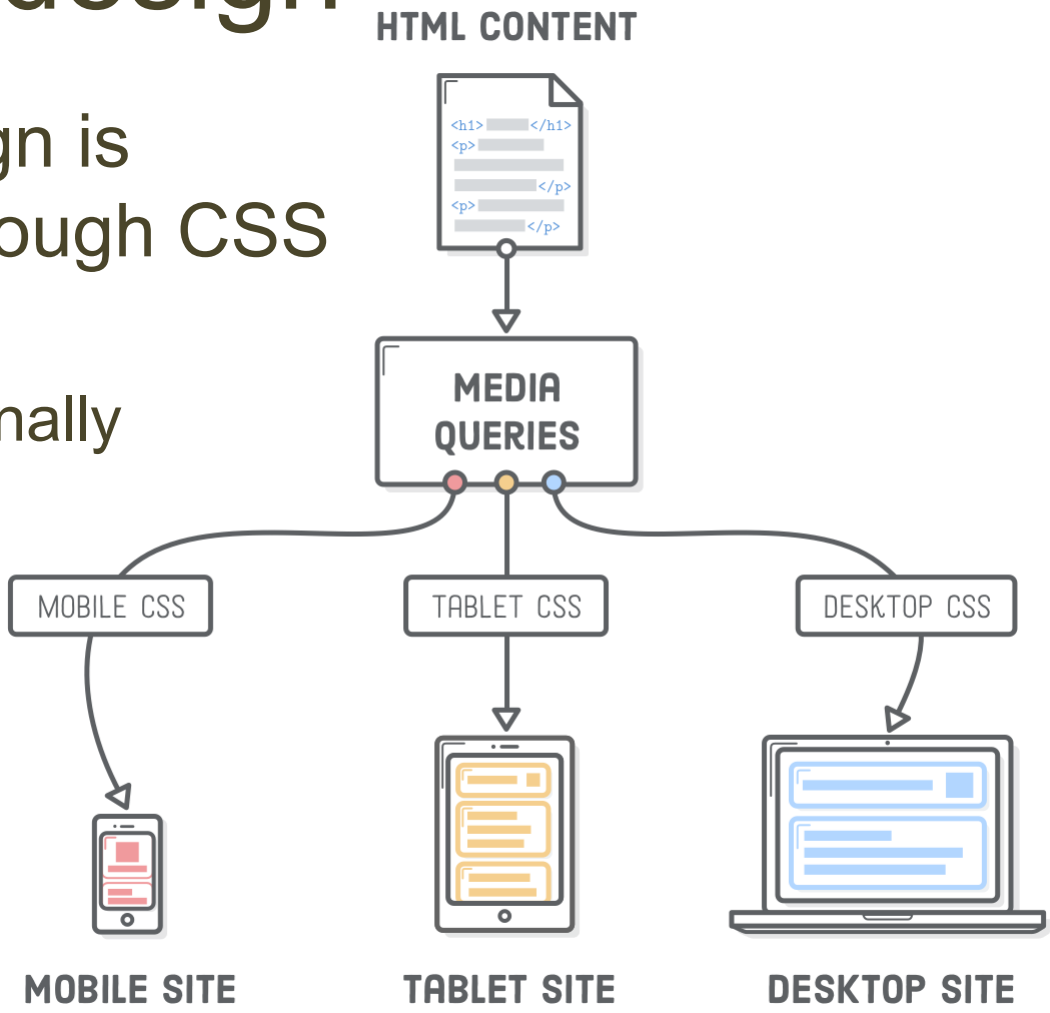
- Your website should display equally well in everything from widescreen monitors to mobile phones
- An approach to web design and development that eliminates the distinction between the mobile-friendly version of your website and its desktop counterpart
  - With responsive design they're the same thing

https://internetingishard.com/html-and-css/responsive-design/



Responsive design

- Responsive design is accomplished through CSS "media queries"
  - A way to conditionally apply CSS rules



Media queries

- A media query consists of a media type and can contain one or more expressions, which resolve to either true or false
- The result of the query is true if the specified media type matches the device the document is being displayed on and if all expressions in the media query are true
- When a media query is true, the corresponding style sheet or style rules are applied, following the normal cascading rules

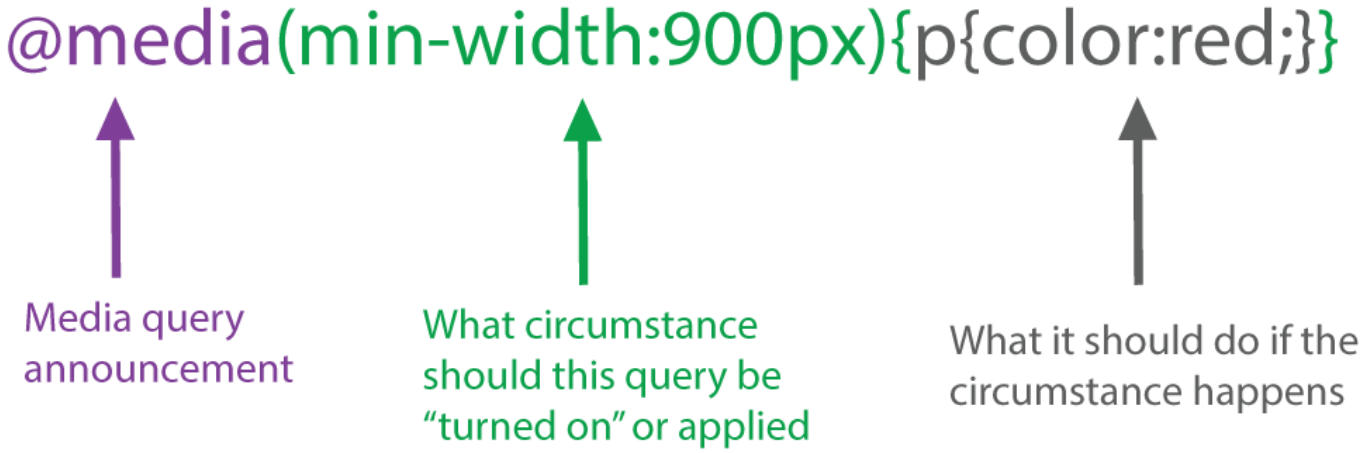
```
@media not|only mediatype and (expressions)
{ CSS-Code; }
```

- Unless you use the not or only operators, the media type is optional and the “all” media type is implied

http://www.w3schools.com/cssref/css3_pr_mediaquery.asp

Media queries

- Example: if the screen is wider than 900 pixels then the color of the text should be red



- Example: MediaQueryDemo.htm

Media types

- Can be used to specify how a document is presented on different media

CSS3 Media Types

Value	Description
all	Used for all media type devices
print	Used for printers
screen	Used for computer screens, tablets, smart-phones etc.
speech	Used for screenreaders that "reads" the page out loud

- CSS 2.1 defined ten media types (all, aural, braille, embossed, handheld, print, projection, screen, tty, tv), but they never got a real support by devices, other than the print media type, and they are now deprecated

Media queries

http://www.w3schools.com/cssref/css3_pr_mediaquery.asp

- Media queries look at the capability of the device, and can be used to check many things, such as
  - Width and height of the viewport
  - Width and height of the device
  - Orientation (is the tablet/phone in landscape or portrait mode?)
  - Resolution
  - … and much more
- List of supported media features

Media features (some)

Value	Description
aspect-ratio	The ratio between the width and the height of the viewport
color	The number of bits per color component for the output device
color-index	The number of colors the device can display
device-aspect-ratio	The ratio between the width and the height of the device
device-height	The height of the device, such as a computer screen
device-width	The width of the device, such as a computer screen
grid	Whether the device is a grid or bitmap
height	The viewport height
max-aspect-ratio	The maximum ratio between the width and the height of the display area
max-color	The maximum number of bits per color component for the output device
max-color-index	The maximum number of colors the device can display
max-device-aspect-ratio	The maximum ratio between the width and the height of the device
max-device-height	The maximum height of the device, such as a computer screen
max-device-width	The maximum width of the device, such as a computer screen

Media queries

- Most media features accept “min-” or “max-” prefixes

```
screen and (min-height: 20em)
```

- Media features can often be used without a value

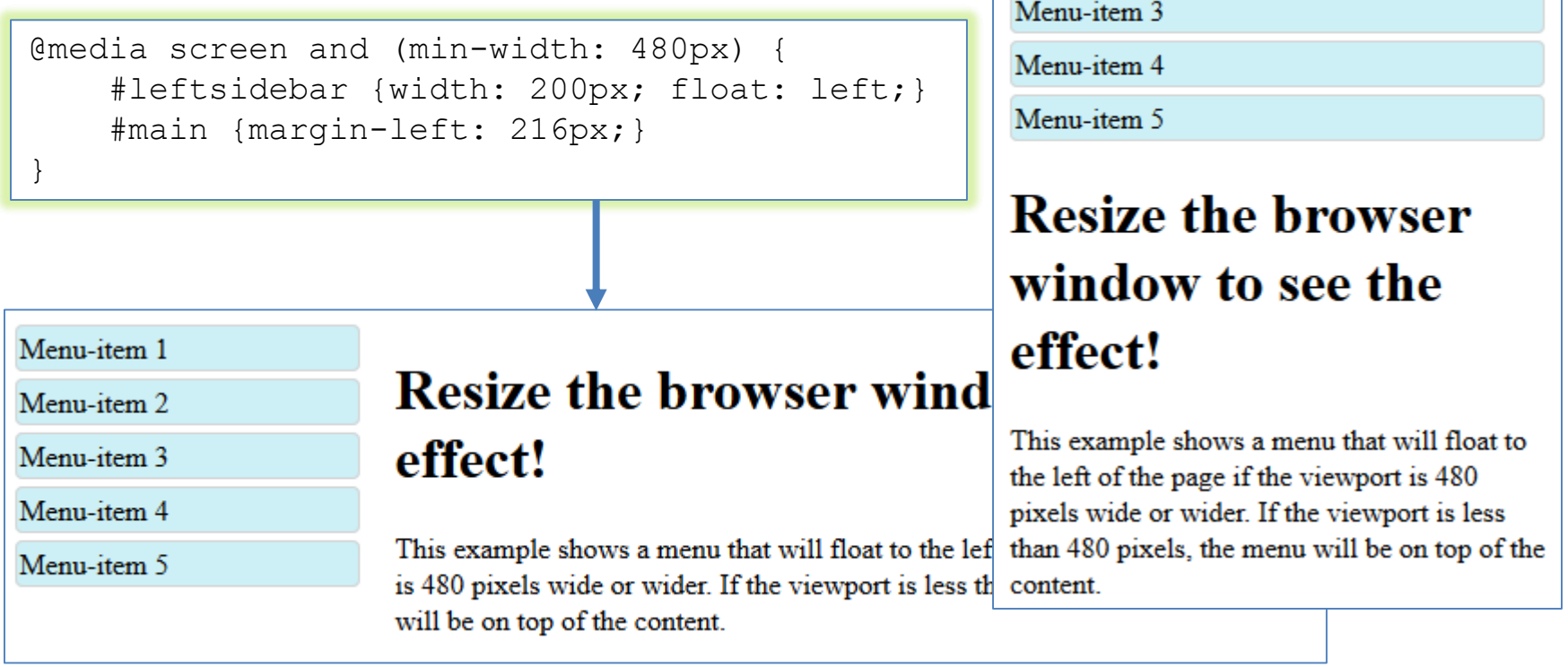
```
screen and (color)
```

- Media features only accept single values: one keyword, one number, or a number with a unit identifier

```
(orientation: portrait)
(min-width: 20em)
(min-color: 2)
(device-aspect-ratio: 16/9)
```

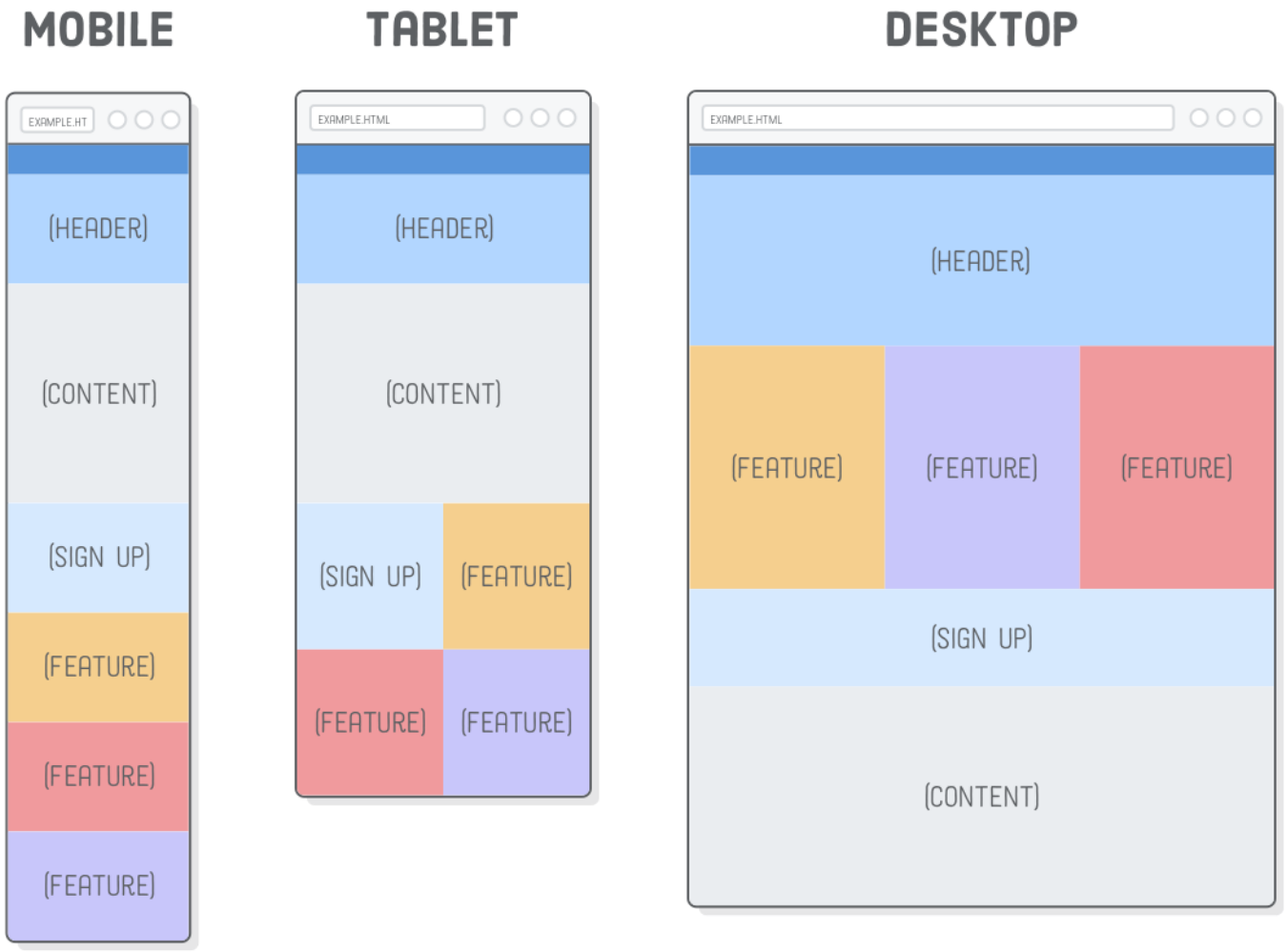
Example

```
@media screen and (min-width: 480px) {
    #leftsidebar {width: 200px; float: left;}
    #main {margin-left: 216px;}
}
```



https://www.w3schools.com/css/tryit.asp?filename=trycss3_media_queries2

Layout for responsive design



The viewport

- The viewport is the user's visible area of a web page
- The viewport varies with the device
  - Smaller on a mobile phone than on a computer screen
- Before tablets and mobile phones, web pages were designed only for computer screens, and web pages had a static design and a fixed size
- When users started surfing the internet using tablets and mobile phones, fixed size web pages were too large to fit the viewport
  - To fix this, browsers on those devices scaled down the entire web page to fit the screen.
  - This was not perfect…

The viewport

- The HTML5 <meta> tag allows to take control over the viewport
- A <meta> viewport element gives the browser instructions on how to control the page dimensions and scaling
- All web pages should include

```
<meta name="viewport"
  content="width=device-width, initial-scale=1.0">
```

- The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device)
- The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser

Size content to the viewport

- Users are used to scroll websites vertically on both desktop and mobile device
- If the user is forced to scroll horizontally, or zoom out, to see the whole web page it results in a poor user experience
- Some rules
  - Do NOT use large fixed width elements (e.g. images): remember to adjust this content to fit within the width of the viewport
  - Do NOT let the content rely on a particular viewport width to render well: screen dimensions and width in CSS pixels vary widely between devices
  - Use CSS media queries to apply different styling for small and large screens: consider using relative width values (percentages)

Layout for responsive design

- Most of responsive design patterns have similar behavior
  - Fluid layouts for mobile/tablet devices to target a range of screen widths instead of specific mobile devices: designing a fluid layout that looks good anywhere between 300 pixels and 500 pixels (or whatever)
  - Fixed-width layouts for wider screens
- “Breakpoints” for a responsive web site
  - The points a which your sites content will “respond” to provide the user with the best possible layout to consume the information

Breakpoints

- Strategies for definition
  - Use device resolution: the smart phone (usually the iPhone at 320px and 480px), the tablet (usually the iPad at 768px and 1024px) and finally anything above 1024px
  - Mobile first: the best practice is to design for your smallest viewport first… as you expand that view there will come a point at which the design looks terrible… this is where you add a breakpoint
- From device specific breakpoints to content specific breakpoints, i.e. adding a breakpoint when the content is no longer easy to consume
- Besides, mobile-first approach force to really identify what is the most important information on your site

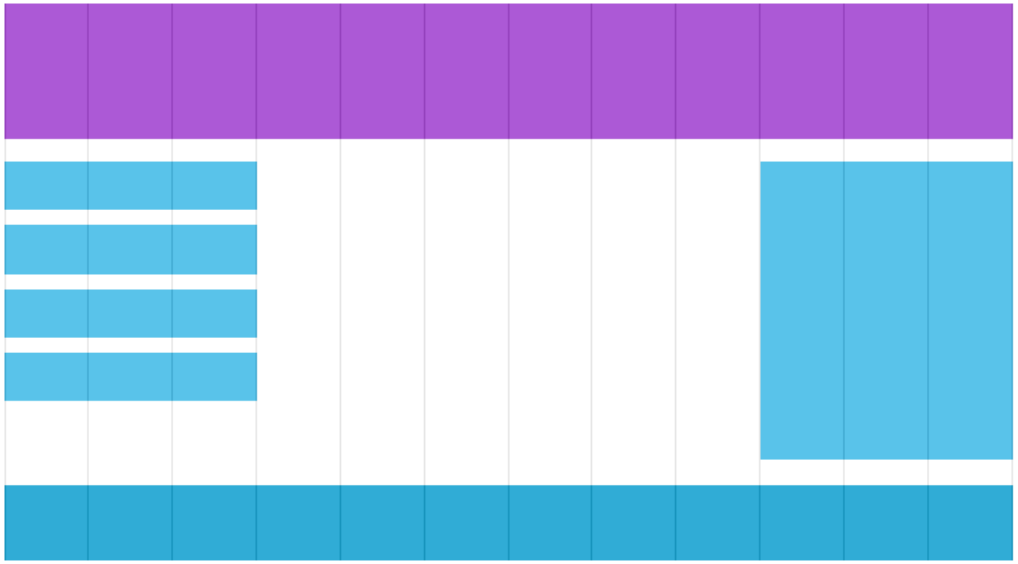
Example: device breakpoints

```
/* Smartphones (portrait and landscape) ----------- */
@media only screen and (min-device-width : 320px) and
(max-device-width : 480px)
{ /* Styles */ }

/* Smartphones (landscape) ----------- */
@media only screen and (min-width : 321px)
{ /* Styles */ }

/* Smartphones (portrait) ----------- */
@media only screen and (max-width : 320px)
{ /* Styles */ }

/* iPads (portrait and landscape) ----------- */
@media only screen and (min-device-width : 768px) and
(max-device-width : 1024px)
{ /* Styles */ }

...
```

https://responsivedesign.is/develop/browser-feature-support/media-queries-for-common-device-breakpoints/

Grid-view

- Many web pages are based on a grid-view, i.e. the page is divided into columns
  - Very helpful when designing web pages: easier to place elements on the page
- A responsive grid-view often has 12 columns, a total width of 100%, and will shrink and expand as you resize the browser window



Building a responsive grid-view

https://www.w3schools.com/cssref/tryit.asp?filename=trycss3_box-sizing

- All HTML elements should have the box-sizing property set to border-box
  - This makes sure that the padding and border are included in the total width and height of the elements
- Responsive grid-view with 12 columns
  - Percentage for one column: 100% / 12 columns = 8.33%
- Then we make one class for each of the 12 columns
  - class = “col-n”, where n defines how many columns the section should span

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

```
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}
```

Building a responsive grid-view

- All the columns should float to the left, and have a padding
- Each row should be wrapped in a
  - The number of columns inside a row should always add up to 12

```
[class*="col-"] {
    float: left;
    padding: 15px;
}
```

```
<div class="row">
  <div class="col-3">...</div> <!-- 25% -->
  <div class="col-9">...</div> <!-- 75% -->
</div>
```

- The columns inside a row are all floating to the left
  - The other elements will be placed as if the columns do not exist
  - Add a style that clears the flow

```
.row::after {
    content: "";
    clear: both;
    display: table;
}
```

https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_styles

Responsive design with media queries

- Breakpoint at 768px
- When the screen (browser window) gets smaller than 768px, each column should have a width of 100%

https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_breakpoints

```
/* For desktop: */
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}

@media only screen and (max-width: 768px) {
    /* For mobile phones: */
    [class*="col-"] {
        width: 100%;
    }
}
```

Design for mobile first

- To make the page display faster on smaller devices

```
/* For mobile phones: */
[class*="col-"] {
    width: 100%;
}
@media only screen and (min-width: 768px) {
    /* For desktop: */
    .col-1 {width: 8.33%;}
    .col-2 {width: 16.66%;}
    .col-3 {width: 25%;}
    .col-4 {width: 33.33%;}
    .col-5 {width: 41.66%;}
    .col-6 {width: 50%;}
    .col-7 {width: 58.33%;}
    .col-8 {width: 66.66%;}
    .col-9 {width: 75%;}
    .col-10 {width: 83.33%;}
    .col-11 {width: 91.66%;}
    .col-12 {width: 100%;}
}
```

More breakpoints

- For tablets: one more media query (at 600px), and a set of new classes for devices larger than 600px (but smaller than 768px)

```
@media only screen and (min-width: 600px)
{
    /* For tablets: */
    .col-m-1 {width: 8.33%;}
    .col-m-2 {width: 16.66%;}
    .col-m-3 {width: 25%;}
    .col-m-4 {width: 33.33%;}
    .col-m-5 {width: 41.66%;}
    .col-m-6 {width: 50%;}
    .col-m-7 {width: 58.33%;}
    .col-m-8 {width: 66.66%;}
    .col-m-9 {width: 75%;}
    .col-m-10 {width: 83.33%;}
    .col-m-11 {width: 91.66%;}
    .col-m-12 {width: 100%;}
}
```

```
<div class="row">
<div class="col-3 col-m-3">...</div>
<div class="col-6 col-m-9">...</div>
<div class="col-3 col-m-12">...</div>
</div>
```

https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_col-s

Orientation: portrait / landscape

- Media queries can also be used to change layout of a page depending on the orientation of the browser
- Set of CSS properties that will only apply when the browser window is wider than its height
  - "Landscape" orientation

```
body {
    background-color: lightgreen;
}

@media only screen and (orientation: landscape) {
    body {
        background-color: lightblue;
    }
}
```

https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_mediaquery_orientation

Responsive website with flexbox

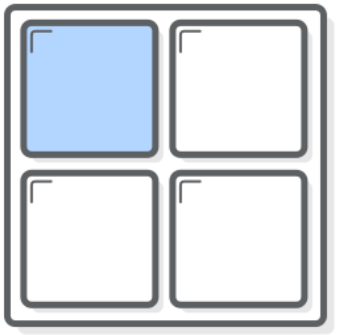
- https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_website2

- To simulate a mobile device in the desktop browser
  - Google Chrome: View > Developer > Developer Tools in the menu bar; then click the Toggle Device Toolbar icon
  - Mozilla Firebox: Strumenti > Sviluppo web > Modalità visualizzazione flessibile

CSS RESPONSIVE IMAGES

Images

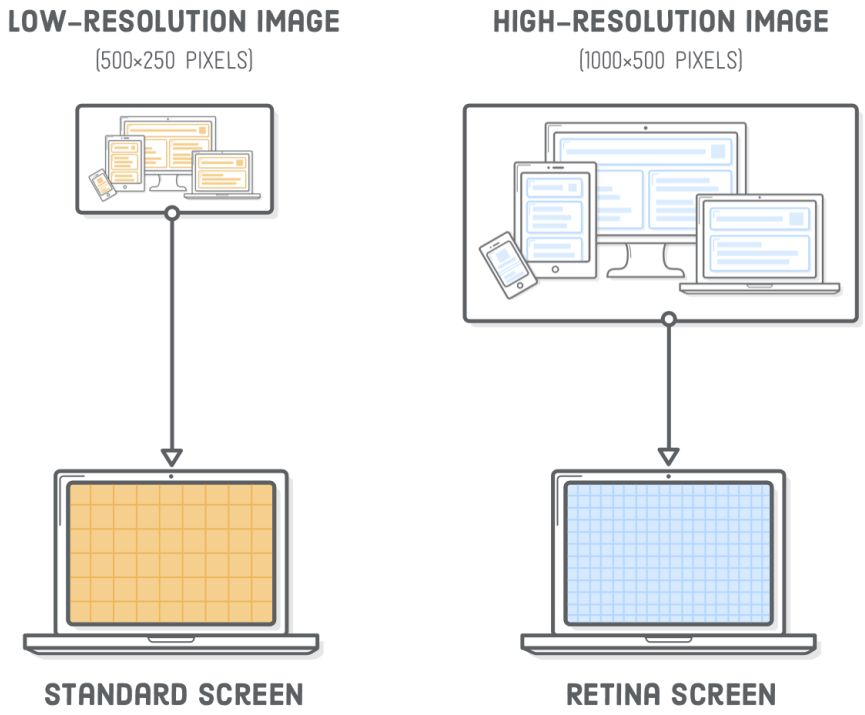
- Images have inherent dimensions
  - We can't stretch a photo that's 500×250 pixels to anything beyond 500 pixels wide because it'll get pixelated
- Three characteristics to consider to make images responsive
  - The device's dimensions
  - The image's dimensions
  - The device's screen resolution
- E.g., to render correctly on a retina device, an image needs to be twice as big as its final display dimensions



STANDARD RESOLUTION

RETINA RESOLUTION

Images



- Vectorial images: no resolution problem
  - They scale by definition
  - SVG (Scalable Vector Graphic) images
- Raster images: sensible to resolution
  - PNG, GIF, JPG images

Images

- If the width property is set to 100%, the image will be responsive and scale up and down

```
img {
    width: 100%;
    height: auto;
}
```

- If the max-width property is set to 100%, the image will scale down if it has to, but never scale up to be larger than its original size

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

https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_image

Different images for different devices

- A large image can be perfect on a big computer screen, but useless on a small device
  - Why load a large image when you have to scale it down anyway?
- To reduce the load, or for any other reasons, you can use media queries to display different images on different devices

```
/* For devices smaller than 400px: */
body {
    background-image: url('img_smallflower.jpg'); }

/* For devices 400px and larger: */
@media only screen and (min-device-width: 400px) {
    body {
        background-image: url('img_flowers.jpg'); }
}
```

https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_image_mediaq

The <picture> element

- HTML5 introduced the <picture> element
  - Similar to the <audio> and <video> elements: allows to define different sources, and the first source that fits the preferences is the one being used
- The media attribute is optional, and accepts the media queries

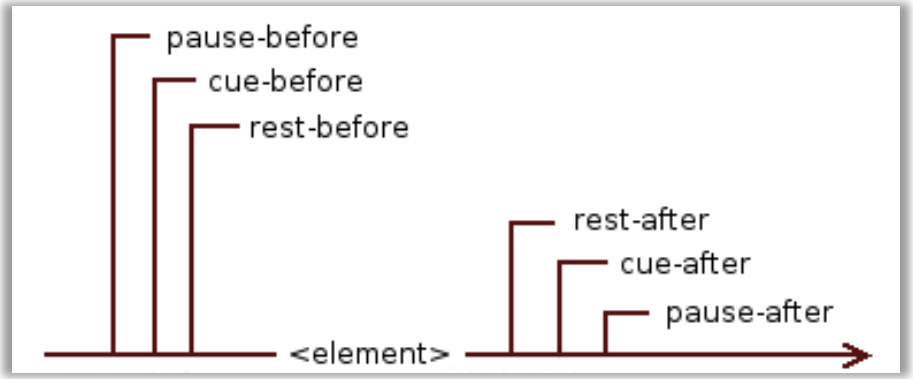
```
<picture>
  <source srcset="img_smallflower.jpg" media="(max-width: 400px)">
  <source srcset="img_flowers.jpg">
  <img src="img_flowers.jpg" alt="Flowers">
</picture>
```

CSS SPEECH MODULE

CSS speech module

- Evolved version of the aural media type
- Describes the various properties that allow web content developers to declare the rendering of web documents via speech synthesis
- In addition to this, the specification also defines optional audio cue which can be used in web documents
  - https://www.w3.org/TR/css3-speech/
  - https://www.w3schools.com/cssref/css_ref_aural.asp

A new “box model”



- Pauses and rests
  Set a pause or rest
  before or after an element’s content is spoken. You can use either time units (for example, “2s” for 2 seconds) or keywords (none, x-weak, weak, medium, strong and x-strong)
- Cues
  Use sounds to delimit particular elements and control their volume
- Voice-rate
  Control the speed at which elements are spoken. This can be defined as a percentage or keyword: x-slow, slow, medium, fast and x-fast
- Voice-stress
  Indicate any emphasis that should be applied, using different keywords: none, moderate, strong and reduced

Speech related CSS3 properties

- Voice-volume
  - Sets a volume using a number from 0 to 100 (0 being silence), percentages or a keyword (silent, x-soft, soft, medium, loud and x-loud)
- Voice-family
  - Sets specific types of voices and voice combinations (as you do with fonts)
- Voice-balance
  - Controls which channel sound comes from (if the user's sound system supports stereo)
- Speak
  - Instructs the screen reader to spell out particular words, digits or punctuation. Available keywords are none, normal, spell-out, digits, literal-punctuation, no-punctuation and inherit

Media type speech

- Examples

```
@media speech
{ body { voice-family: Paul } }
```

```
h2
{ voice-family: female;
  voice-balance: left;
  voice-volume: soft;
  cue-after: url(sound.au);
}
```

Tells a screen reader to read all h2 tags in a female voice, from the left speaker, in a soft tone and followed by a particular sound

```
h1 { voice-family: announcer, old male }
p.part.romeo { voice-family: romeo, young male }
p.part.juliet { voice-family: juliet, female }
p.part.mercutio { voice-family: male 2 }
p.part.tybalt { voice-family: male 3 }
p.part.nurse { voice-family: child female }
```

SOME MORE PROPERTIES

RGBA color and opacity

- RGBA color
  - Like RGB color definitions, but allows a fourth field, defining the alpha value of the color being applied
  - Like opacity, the alpha value is between 0.0 (fully transparent) and 1.0 (fully opaque)

```
div { color: rgb(0,255,0); }
```

```
div { color: rgba(0,255,0,0.5); }
```

HSLA color and opacity

- HSLA color
  - Like HSL color, but allows a fourth field, defining the alpha value of the color being applied

```
div { color: hsl(240,50%,50%); }
```

```
div { color: hsla(240,50%,50%,0.5); }
```

Color and opacity

- The difference between RGBA or HSLA and opacity is that the former applies transparency only to a particular element, whereas the latter affects the element we target and all of its children

- Example:

```
div
{ color: #f00;
  opacity: 0.5; }
```


```
#alpha-example
{ background: hsla(324, 100%, 50%, .5);
  border: 1em solid rgba(0, 0, 0, .3);
  color: rgba(255, 255, 255, .8);}
```

Multiple backgrounds

- It is possible to apply multiple layered backgrounds to an element using multiple properties such as background-image, background-repeat, background-size, background-position, background-origin and background-clip

```
background:
  url(body-top.png) top left no-repeat,
  url(body-bottom.png) bottom left no-repeat,
  url(body-middle.png) left repeat-y;
```

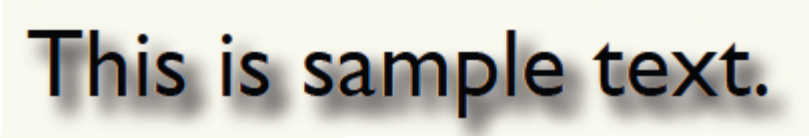
Multiple backgrounds

- Example


```
#multiple_background
{ width:400px;
  height:150px;
  border:2px solid #CCC;
  background:
    url(uccello.jpg) no-repeat 30px top,
    url(lumaca.jpg) no-repeat right 105px,
    url(logo.jpg) no-repeat 60px 55px,
    url(fiore.jpg) no-repeat 5px 55px,
    url(erba.jpg) repeat-x bottom,
    url(cielo.jpg) repeat-x top;
}
```

Text shadows

- Arguments: horizontal offset, vertical offset, blur radius and color to be used as the shadow
- Multiple shadow definitions may be separated using commas
- Examples




```
text-shadow: 10px 10px 10px #333;
```


```
body
{ background: #ccc; }
p
{ margin: 0 0 1em;
  font-size: 60px;
  font-weight: bold;
  color: #ccc;letter-spacing: 1px;
  text-shadow: -1px -1px 0px #333, 1px 1px 1px #fff; }
```

Text shadows

- Examples


```
text-shadow: 0 0 .2em white, 0 0 .5em white;
```


```
color: transparent:
text-shadow: 0 0 .2em white;
```

```
text-shadow:
        0 0 4px white,
        0 -5px 4px #fff,
        2px -10px 6px #fd3,
        -2px -15px 11px #f80,
        2px -25px 18px #f20;
```

Word wrap

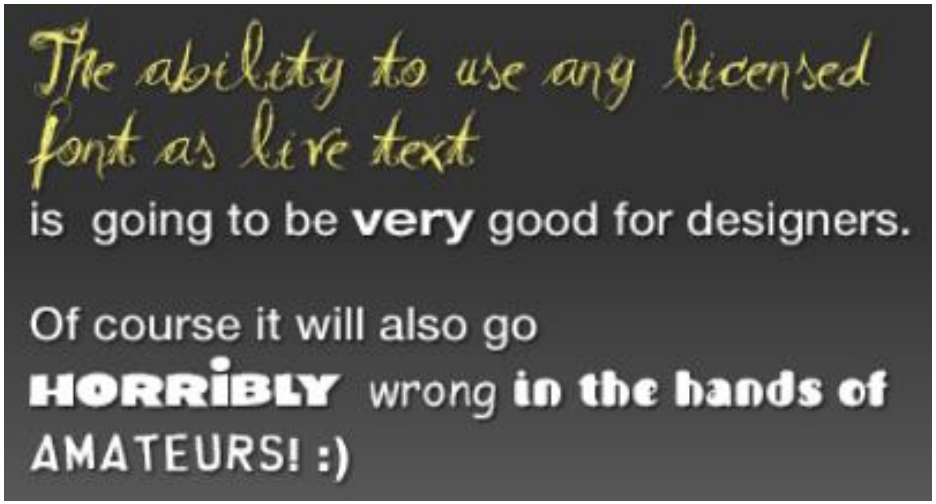
- Specifies whether the current rendered line should break if the content exceeds the boundary of the specified rendering box for an element
- Example

```
div
{ word-wrap: break-word }
```

Value	Description
normal	Content will exceed the boundaries of the specified rendering box.
break-word	Content will wrap to the next line when necessary, and a word-break will also occur if needed.

@font-face

- Simple technique for allowing designers to use their own fonts for display on the web, eliminating the constrictions that currently exist

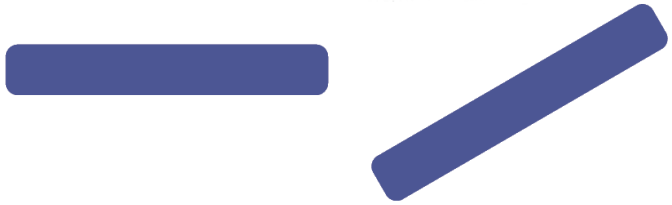



```
@font-face
{ font-family: 'DroidSans';
  src: url('droidsans.ttf') format('truetype');
}

h2
{ font-family: 'DroidSans', Impact, sans-serif;
}
```

Transformations

- Rotate
  - Rotates the selected element at the defined angle, defined in degrees
  - The rotation doesn't affect layout, and elements that are transformed are treated similarly to position:relative

```
transform: rotate(30deg);
```



- Scale
  - Scales the element in question based on the specified unit-less numbers given for the X and Y axes
  - If only one number is given, it is applied to both axes

```
transform: scale(0.5,2.0);
```

Transformations

- Skew
  - Skews the element around the X and Y axes by the specified angles, in degrees
  - If it's only one number, the Y axis is assumed to be zero

```
transform: skew(-30deg);
```

- Translate
  - Moves the object along each axis by the length specified
  - The unit can be anything accepted as a length in CSS, such as px, em, percentages, …

```
transform: translate(30px, 0);
```

Transitions

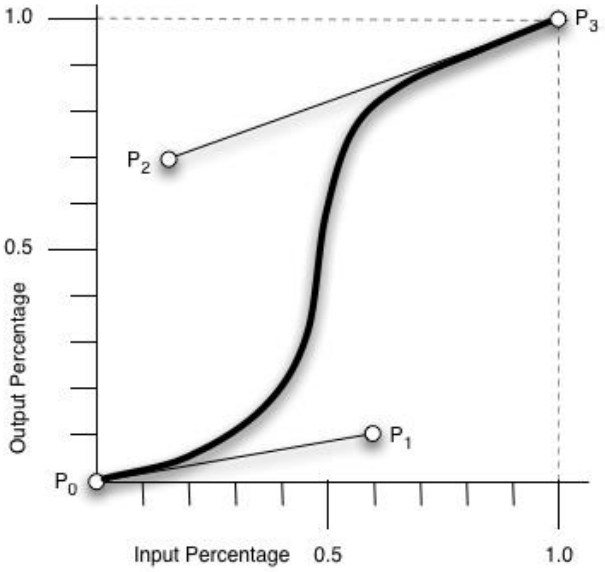
- Create an effect when changing from one style to another

```
div
{
  width:100px;
  height:100px;
  background:red;
  transition:width 2s;
  -moz-transition:width 2s; /* Firefox 4 */
  -webkit-transition:width 2s; /* Safari and Chrome */
-  o-transition:width 2s; /* Opera */
}

div:hover
{
width:300px;
}
```

http://www.w3schools.com/css/css3_transitions.asp

Transitions

- Funzione di timing
  - Il modo di calcolare i valori intermedi durante la transizione: permette di cambiare la velocità della transizione durante la sua durata
  - Valori: ease, ease-in, ease-out, ease-in-out, linear (equivalenti a specifiche curve di Bezier), cubic-bezier (P_0, P_1, P_2, P_3)



http://www.w3.org/TR/css3-transitions/#transition-timing-function_tag

Transformations and transitions

foto.htm

License