



Frontend

CSS

What's CSS?

Cascading Style Sheet

Basic chunk of CSS

```
.header {  
  color: blue;  
  font-size: 2em;  
}
```

Basic chunk of CSS

```
.header {  
  color: blue;  
  font-size: 2em;  
}
```

selector

Targets the object to be styled.

Basic chunk of CSS

```
.header {  
  color: blue;  
  font-size: 2em;  
}
```

properties

key: value relations that apply the actual styles.

Selectors

```
/* Target an HTML element directly */
```

```
p
```

```
/* Target an element with a class name(s) */
```

```
.class
```

```
.class.class2
```

```
/* Target the element with the specific id */
```

```
#id
```

```
/* Target an element with a specific attribute value */
```

```
.class[title="Nicolas Cage"]
```

Selectors

- Try to avoid targeting HTML tags directly.
- An element can have multiple .classes and classes may be repeated across the page.
- .classes can't start with numbers.
- #ids can only appear once in a page and you should avoid styling them with CSS, stick to the .classes if possible.

Properties and values

[Full list](#)

:active, ::after (:after), align-content, align-items, align-self, all, <angle>, animation, animation-delay, animation-direction, animation-duration, animation-fill-mode, animation-iteration-count, animation-name, animation-play-state, animation-timing-function, @annotation, annotation(), attr(), ::backdrop, backface-visibility, background, background-attachment, background-blend-mode, background-clip, background-color, background-image, background-origin, background-position, background-repeat, background-size, <basic-shape>, ::before (:before), <blend-mode>, blur(), border, border-bottom, border-bottom-color, border-bottom-left-radius, border-bottom-right-radius, border-bottom-style, border-bottom-width, border-collapse, border-color, border-image, border-image-outset, border-image-repeat, border-image-slice, border-image-source, border-image-width, border-left, border-left-color, border-left-style, border-left-width, border-radius, border-right, border-right-color, border-right-style, border-right-width, border-spacing, border-style, border-top, border-top-color, border-top-left-radius, border-top-right-radius, border-top-style, border-top-width, border-width, bottom, box-decoration-break, box-shadow, box-sizing, break-after, break-before, break-inside, brightness(), calc(), caption-side, ch, @character-variant, character-variant(), @charset, :checked, circle(), clear, clip, clip-path, cm, color, <color>, columns, column-count, column-fill, column-gap, column-rule, column-rule-color, column-rule-style, column-rule-width, column-span, column-width, content, contrast(), <counter>, counter-increment, counter-reset, @counter-style, cubic-bezier(), cursor, <custom-ident>, :default, deg, :dir(), direction, :disabled, display, @document, dpcm, dpi, dppx, drop-shadow(), element(), ellipse(), em, :empty, empty-cells, :enabled, ex, filter, :first, :first-child, ::first-letter (:first-letter), ::first-line (:first-line), :first-of-type, flex, flex-basis, flex-direction, flex-flow, flex-grow, flex-shrink, flex-wrap, float, :focus, font, @font-face, font-family, font-feature-settings, @font-feature-values, font-kerning, font-language-override, font-size, font-size-adjust, font-stretch, font-style, font-synthesis, font-variant, font-variant-alternates, font-variant-caps, font-variant-east-asian, font-variant-ligatures, font-variant-numeric, font-variant-position, font-weight, <frequency>, :fullscreen, grad, <gradient>, grayscale(), grid, grid-area, grid-auto-columns, grid-auto-flow, grid-auto-position, grid-auto-rows, grid-column, grid-column-start, grid-column-end, grid-row, grid-row-start, grid-row-end, grid-template, grid-template-areas, grid-template-rows, grid-template-columns, height, :hover, hsl(), hsla(), hue-rotate(), hyphens, hz, <image>, image(), image-rendering, image-resolution, image-orientation, ime-mode, @import, in, :indeterminate, inherit, initial, :in-range, inset(), <integer>, :invalid, invert(), isolation, justify-content, @keyframes, khz, :lang(), :last-child, :last-of-type, left, :left, <length>, letter-spacing, linear-gradient(), line-break, line-height, :link, list-style, list-style-image, list-style-position, list-style-type, margin, margin-bottom, margin-left, margin-right, margin-top, marks, mask, mask-type, matrix(), matrix3d(), max-height, max-width, @media, min-height, minmax(), min-width, mix-blend-mode, mm, ms, @namespace, :not(), :nth-child(), :nth-last-child(), :nth-last-of-type(), :nth-of-type(), <number>, object-fit, object-position, :only-child, :only-of-type, opacity, opacity(), :optional, order, @ornaments, ornaments(), orphans, outline, outline-color, outline-offset, outline-style, outline-width, :out-of-range, overflow, overflow-wrap, overflow-x, overflow-y, padding, padding-bottom, padding-left, padding-right, padding-top, @page, page-break-after, page-break-before, page-break-inside, pc, <percentage>, perspective, perspective(), perspective-origin, pointer-events, polygon(), position, <position>, pt, px, quotes, rad, radial-gradient(), <ratio>, :read-only, :read-write, rect(), rem, repeat(), ::repeat-index, ::repeat-item, repeating-linear-gradient(), repeating-radial-gradient(), :required, resize, <resolution>, rgb(), rgba(), right, :right, :root, rotate(), rotatex(), rotatey(), rotatez(), rotate3d(), ruby-align, ruby-merge, ruby-position, s, saturate(), scale(), scalex(), scaley(), scalez(), scale3d(), :scope, scroll-behavior, ::selection, sepia(), <shape>, shape-image-threshold, shape-margin, shape-outside, skew(), skewx(), skewy(), steps(), <string>, @styleset, styleset(), @stylistic, stylistic(), @supports, @swash, swash(), symbols(), table-layout, tab-size, :target, text-align, text-align-last, text-combine-upright, text-decoration, text-decoration-color, text-decoration-line, text-decoration-style, text-indent, text-orientation, text-overflow, text-rendering, text-shadow, text-transform, text-underline-position, <time>, <timing-function>, top, touch-action, transform, transform-origin, transform-style, transition, transition-delay, transition-duration, transition-property, transition-timing-function, translate(), translatex(), translatey(), translatez(), translate3d(), turn, unicode-bidi, unicode-range, :unresolved, unset, <uri>, url(), <user-ident>, :valid, ::value, var(), vertical-align, vh, @viewport, visibility, :visited, vmax, vmin, vw, white-space, widows, width, will-change, word-break, word-spacing, word-wrap, writing-mode, X Y Z, z-index

Comments

```
/* This is a CSS comment */
```

```
/*
```

```
I can have  
multiple lines
```

```
*/
```

CSS syntax best practices

Indent your CSS

```
.selector,  
.selector-secondary,  
.selector[type="text"] {  
  ..padding: 15px;  
  ..margin-bottom: 15px;  
  ..background-color: rgba(0,0,0,0.5);  
  ..box-shadow: 0 1px 2px #ccc, inset 0 1px 0 #fff;  
}
```

This makes your code far easier to read for fellow coders

Each selector in a single line

```
.selector,  
.selector-secondary,  
.selector[type="text"] {  
    padding: 15px;  
    margin-bottom: 15px;  
    background-color: rgba(0,0,0,0.5);  
    box-shadow: 0 1px 2px #ccc, inset 0 1px 0 #fff;  
}
```

When grouping selectors, keep individual selectors to a single line.

Use of space

```
.selector,  
.selector-secondary,  
.selector[type="text"] {  
    padding: 15px;  
    margin-bottom: 15px;  
    background-color: rgba(0,0,0,0.5);  
    box-shadow: 0 1px 2px #ccc, inset 0 1px 0 #fff;  
}
```

Include one space before the opening brace of declaration blocks for legibility. Place closing braces of declaration blocks on a new line. Include one space after : for each declaration.

End all declarations with a semi-colon

```
.selector,  
.selector-secondary,  
.selector[type="text"] {  
    padding: 15px;  
    margin-bottom: 15px;  
    background-color: rgba(0,0,0,0.5);  
    box-shadow: 0 1px 2px #ccc, inset 0 1px 0 #fff;  
}
```

The last declaration's is optional, but your code is more error prone without it.

Comma-separated property values

```
.selector,  
.selector-secondary,  
.selector[type="text"] {  
    padding: 15px;  
    margin-bottom: 15px;  
    background-color: rgba(0,0,0,0.5);  
    box-shadow: 0 1px 2px #ccc,inset 0 1px 0 #fff;  
}
```

They should include a space after each comma (e.g., `box-shadow`). But don't separate values inside `rgb()`, `rgba()`, `hsl()` or `hsla()`.

Use of 0

```
.selector,  
.selector-secondary,  
.selector[type="text"] {  
  padding: 15px;  
  margin-bottom: 15px;  
  background-color: rgba(0,0,0,0.5);  
  box-shadow: 0 1px 2px #ccc, inset 0 1px 0 #fff;  
}
```

When a value is 0, don't add a unit. This reduces clutter and improves readability.

HEX color values

```
.selector,  
.selector-secondary,  
.selector[type="text"] {  
  padding: 15px;  
  margin-bottom: 15px;  
  background-color: rgba(0,0,0,0.5);  
  box-shadow: 0 1px 2px #ccc, inset 0 1px 0 #fff;  
}
```

Lowercase all hex values, e.g., #fff. Lowercase letters are much easier to discern when scanning a document. Also, use the shorthand form when possible (e.g., #fff instead of #ffffff).

Quote attribute values in selectors

```
.selector,  
.selector-secondary,  
.selector[type="text"] {  
  padding: 15px;  
  margin-bottom: 15px;  
  background-color: rgba(0,0,0,0.5);  
  box-shadow: 0 1px 2px #ccc, inset 0 1px 0 #fff;  
}
```

They're the only way to guarantee code renders the same in any environment.

Advanced CSS selectors

Pseudo-selectors

```
/* State pseudo-selectors */
```

```
a:hover
```

```
input:focus
```

```
/* CSS generated content */
```

```
.class::before
```

```
.class::after
```

Selector combinators

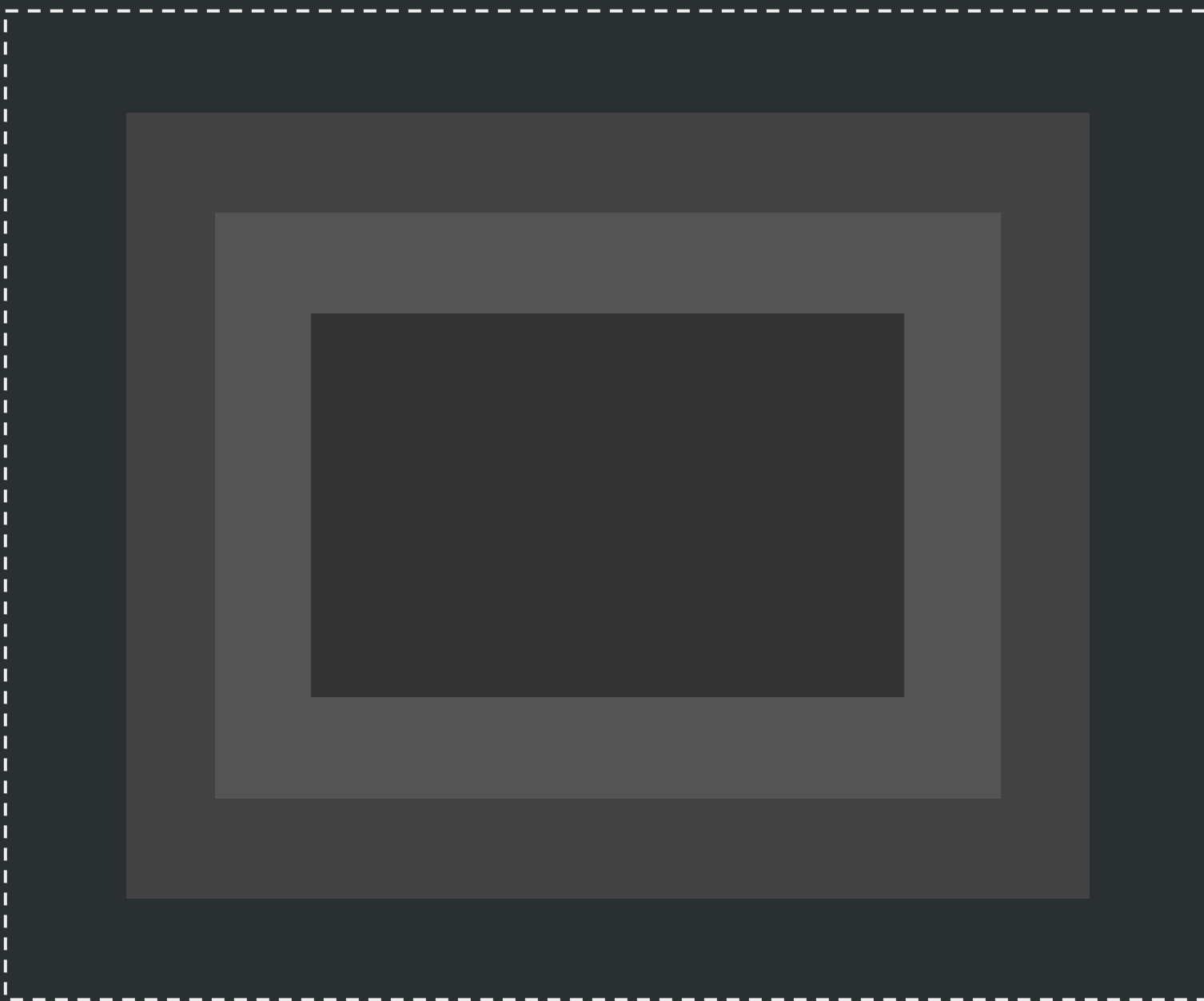
```
/* Targets all <a> elements inside the ones with .class */  
.class a
```

```
/* Targets only the direct children of type <a> */  
.class > a
```

```
/* Targets the <a> elements that come right after a .class  
element */  
.class + a
```

```
/* Targets all <a> elements that come after a .class  
element inside the same parent */  
.class ~ a
```

CSS box model





The diagram illustrates the CSS box model with four nested rectangles. The innermost rectangle is blue and contains the text 'content' and 'it has a width and height properties'. It is surrounded by a dark gray border, which is further surrounded by a lighter gray border, and finally by a dashed white border. The text 'content' is in bold white font, while the rest is in regular white font.

content

it has a width and height
properties

CSS box model

```
.box {  
    width: 400px;  
    height: 300px;  
    background: #ccc;  
}
```



padding

Defined using the padding
properties

CSS box model

```
.box {
```



```
padding-top: 30px;  
padding-right: 30px;  
padding-bottom: 30px;  
padding-right: 30px;
```

```
}
```

CSS box model

/* Shorthand notation */

.box {



padding: 30px;

}



border

Defined using the border
properties

CSS box model

```
.box {
```



```
border-width: 10px;
```

```
border-style: solid;
```

```
border-color: red;
```

```
}
```

CSS box model

/* Shorthand notation */

.box {



border: 10px solid red;

}



CSS box model

/* Shorthand notation */

.box {



margin: 50px;

}

CSS box model

```
/*  
The shorthand notation also works for multiple values in  
margins and paddings.  
*/  
.box {  
  ...  
  margin: 50px; /* all sides */  
  margin: 50px 0; /* top-bottom / left-right */  
  margin: 50px 0 0; /* top / left-right / bottom */  
  margin: 50px 0 0 10px; /* top / right / bottom / left */  
}
```

Exercise

Clone the HTML file here: [CSSPlayground](#)

- Follow the directions for each div.
- Use the correct selector for each div to change their background color, #red should be red, .blue should be blue.
- Add a margin of 10px to all of the boxes.
- Add a red border to blue divs.
- Add a blue border to red divs.
- Give the direction H3 a padding of 20% on all sides using short hand notation. See what happens.

Properties

Display and box model

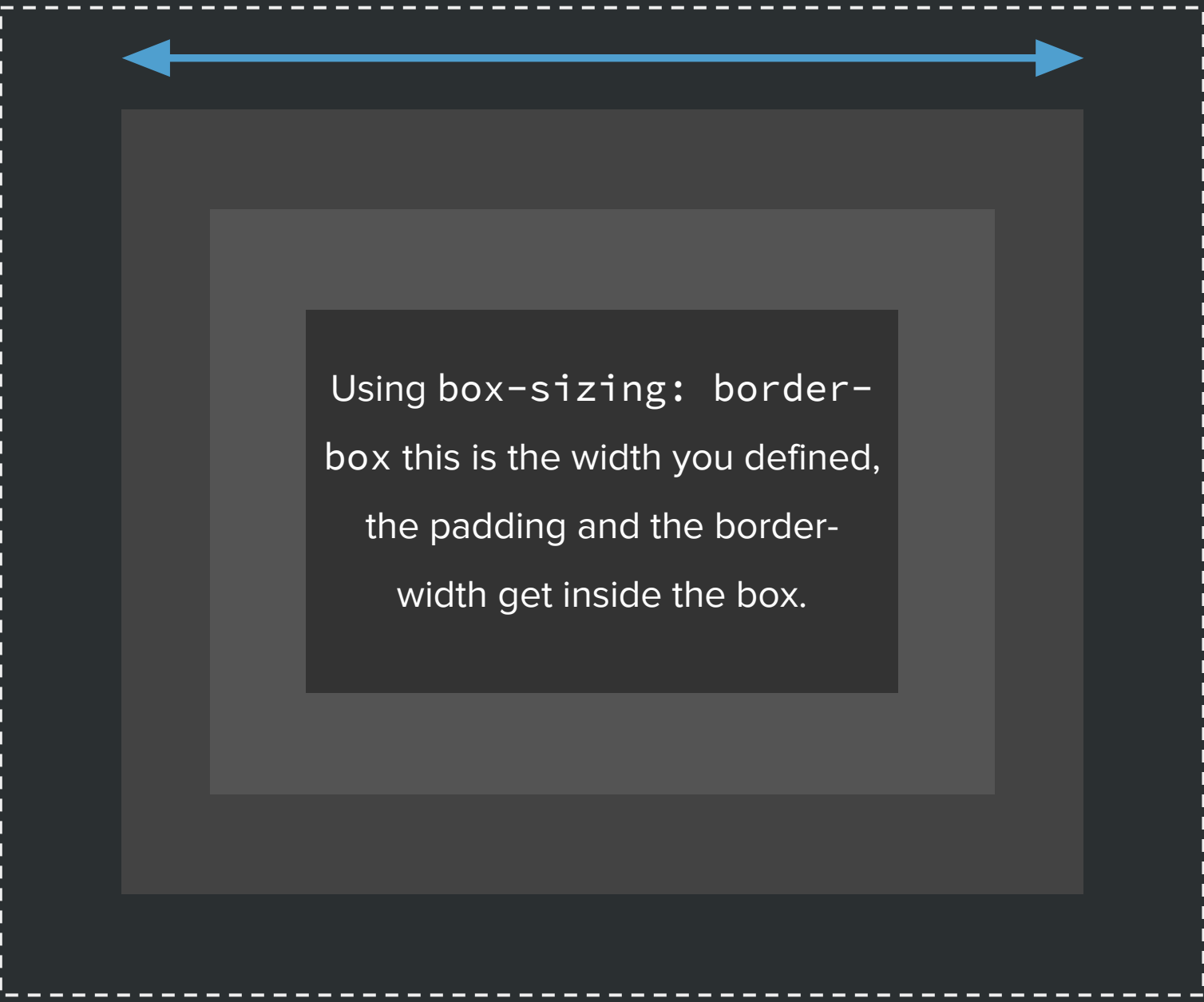
box-sizing

```
/* content-box is the default setting but  
it's usually overridden with border-box */  
box-sizing: content-box;
```

```
/* makes padding and border sizing values  
become a part of the width and height of the  
element */  
box-sizing: border-box;
```



Using `box-sizing:`
`content-box` this is the width
you defined, the padding and the
border-width sum up.



The diagram illustrates the `border-box` model. It features three concentric rectangles. The outermost rectangle is defined by a dashed white border. A blue double-headed arrow spans the width of this dashed border. Inside this is a medium-gray rectangle, and inside that is a dark-gray rectangle. The text is centered within the dark-gray rectangle.

Using `box-sizing: border-box`: `border-box` this is the width you defined, the padding and the border-width get inside the box.

display

[Try it](#)

```
/* inline elements occupy just the size of its contents
and can't have sizing properties */
display: inline;
/* inline-block elements occupy just the size of its
contents and can have sizing properties */
display: inline-block;
/* block elements occupy 100% of the horizontal space fit
to its content's height, it can have fixed sizing */
display: block;
```

HTML elements have a native display property. Keep it in mind when you give them styles.

float

[Try it](#)

```
/* floated items allow siblings to flow besides them,  
useful for images in articles. Bootstrap uses these in the  
.pull-left and .pull-right classes. */  
float: left;  
float: right;
```

Floated items can be messy to deal with, the parent element loses track of their height unless you apply a fix.

Properties

Position

position

`/* all elements are position static by default */`

`position: static;`

`/* relative elements can be positioned using top and left properties and are positioned relative to their parent */`

`position: relative;`

`/* fixed elements are positioned relative to the viewport and don't move while scrolling through the page */`

`position: fixed;`

`/* absolute elements are positioned relative to the first parent which is positioned relative, absolute or fixed.`

`They become layers outside of the current flow */`

`position: absolute;`

top/right/bottom/left

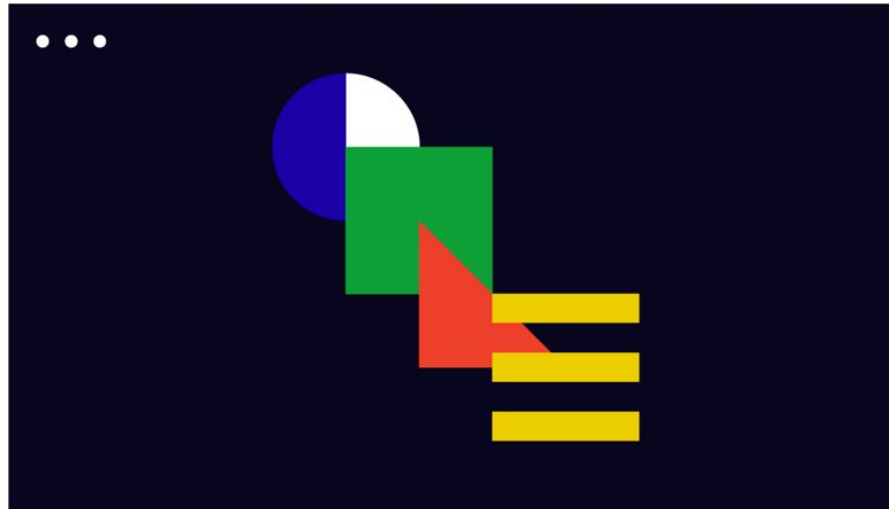
```
/* elements with position: relative/absolute/fixed may  
start using these, no shorthand for this, sorry */  
top: 10px;  
right: 100px;  
left: 100px;  
bottom: 10px;
```

z-index

[Try it](#)

```
/* elements with position: relative/absolute/fixed can  
have a depth value, kind of like the order of the layers  
in a Photoshop/Illustrator/Sketch file */
```

```
z-index: 1;  
z-index: 2; /* this would be in front */
```



Thursday 22 Jan 2015

Positioning in web design

Understanding how positioning in the web works is crucial for responsive web design, as it allows us speak the same language with developers and helps make better design decisions. Compared to static design tools (Photoshop, Illustrator,

Property units

em, rem, px, pt, cm, vw, vh...

CSS units

```
html {  
  /* font-size: 16px; default value */  
}  
  
.box {  
  font-size: 14px;  
  width: 400px;  
  width: 25rem; /* 1rem = 16px */  
  width: 25em; /* 1em = 14px */  
  width: 50vw; /* 50% of the viewport width */  
  height: 50vh; /* 50% of the viewport height */  
}
```


Property colors

hex, rgb, rgba, hsl, hsla...

CSS colors

```
.box {  
  color: white;  
  color: #fff;  
  color: rgb(255,255,255);  
  color: rgba(255,255,255,0.5); /* white color, 50% opaque */  
  color: hsl(0,100%,100%);  
  color: hsla(0,100%,100%,0.5); /* white color, 50% opaque */  
}  
  
/* here's a great color picker to learn about color */
```

Properties

Typography

font-size

`/* defines the size of a font */`

`font-size: 1.5em /* 1.5 * 16 = 24px */`

line-height

```
/* defines the line-height of the text */
```

```
line-height: 1.5 /* 150% of the current font-size */
```

text-align

```
text-align: left /* default */
```

```
text-align: center
```

```
text-align: right
```

color

[Try it](#)

```
/* sometimes CSS is weird, why not font-  
color? or text-color? */
```

```
color: rgb(230,230,230)
```

font-family

/* define a font-family to be used in order of preference, in this example "Helvetica Neue" will be used, if it's not present in the system, Arial will do, if it doesn't, then the default sans-serif font from the system */

font-family: "Helvetica Neue", Arial, sans-serif

/* here you have some cool font-stacks */

@font-face

```
/* load a font file from the local machine and start using it as a font-family */
```

```
@font-face {  
  font-family: 'MyWebFont';  
  src: url('webfont.woff2') format('woff2'),  
       url('webfont.woff') format('woff'),  
       url('webfont.ttf') format('truetype')  
}
```

```
/* luckily we have some services that can help a lot */
```

Properties

Visual

box-shadow

```
/*  
properties in order:  
1 inset?  
2 x offset  
3 y offset  
4 blur  
5 spread(units)?  
6 color  
*/  
box-shadow: inset 0 1px 2px 0 rgba(0,0,0,0.3);
```

text-shadow

/*

properties in order:

1 x offset

2 y offset

3 blur

4 color

*/

text-shadow: 0 1px 2px rgba(0,0,0,0.8)

border-radius

```
/* 5px radius on every corner */
```

```
border-radius: 5px
```

```
/* 5px radius on the top corners */
```

```
border-radius: 5px 5px 0 0 /* tl tr br bl */
```

background

```
background-color: #ccc;  
background-image: url("../images/background.png");  
background-repeat: no-repeat/repeat-x/repeat-y;  
background-position: left top/center center/50px 0;
```

background

[Try it](#)

```
/* shorthand notation */
```

```
background: #ccc url(some.png) no-repeat left top;
```

Exercise, part two

Continue working on the previous playground

- Select the box class and give it a left float, watch what happens.
- Position `#top` and `#bottom` as directed.
- Select the green class to give a green background to the `#top` div
- Give `#top` a z-index so that it appears over the top of the other divs
- Give `#top` a box shadow of your choice