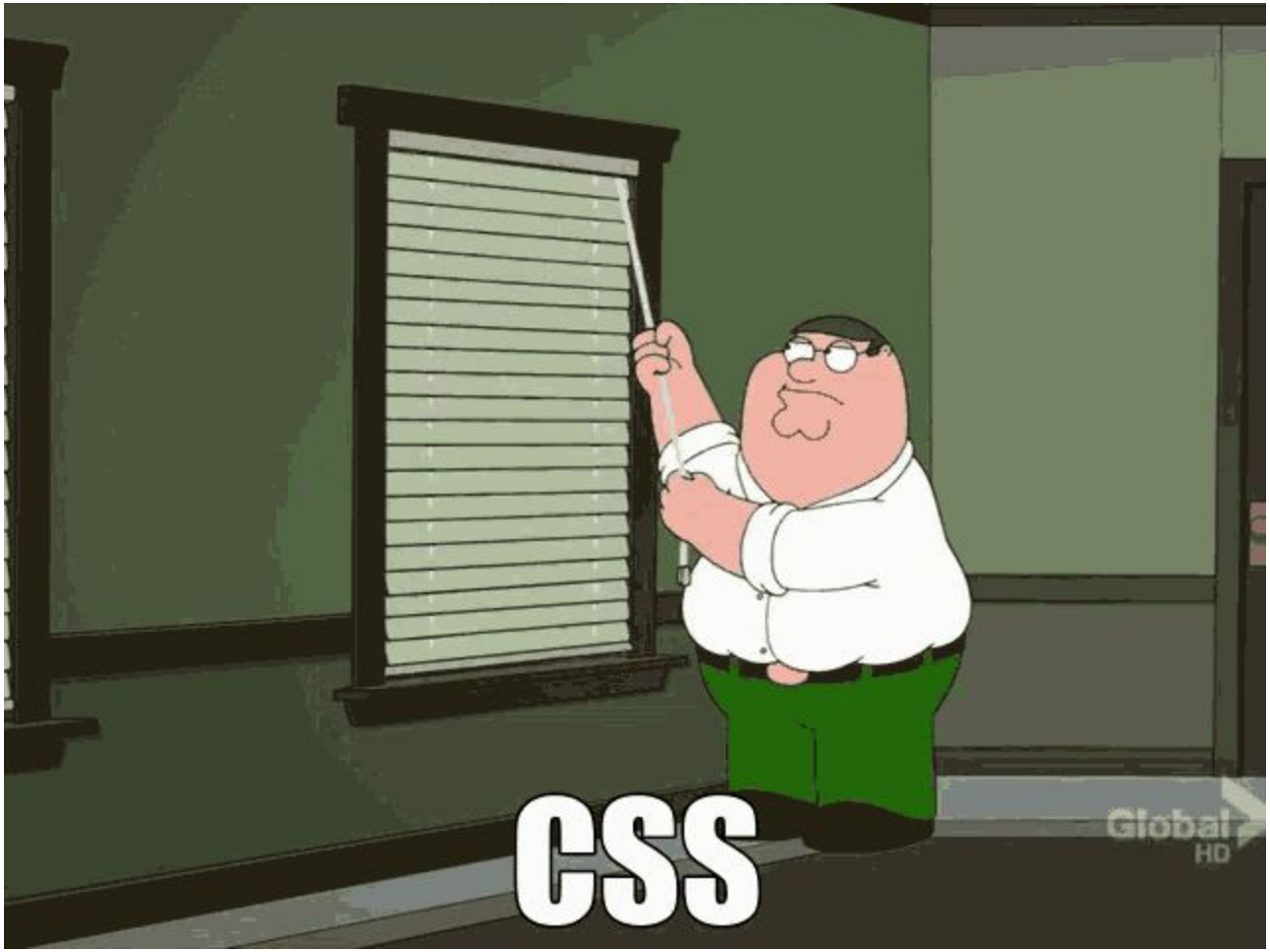
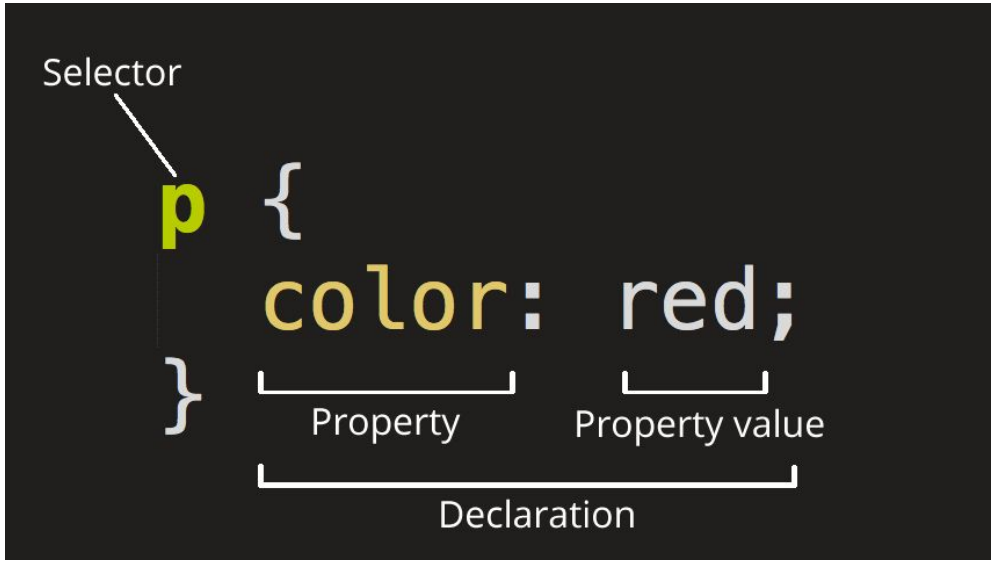
CSS
For beginners

CSS



CSS - Cascading Style Sheets

Describe the presentation of a document written in HTML.



First Example

HTML:

```
<h1>Tommy the cat</h1>

<p>I remember as if it were a meal
ago...</p>

<p>Said Tommy the Cat as he reeled
back to clear whatever foreign matter
 may have nestled its way into his
mighty throat.</p>
```

CSS:

```
p {
  color: red;
  font-family: Helvetica, Arial, serif;
}
```

RESULT:

https://jsfiddle.net/gocode/74vcmn5e/

Use CSS on HTML file - 3 Ways

Inline CSS - On specific element

```
<h1 style="color:blue;">Blue Heading</h1>
```

Internal CSS

```
<!DOCTYPE html>
<html>
<head>
<style>
  body {background-color: powderblue;}
  h1   {color: blue;}
  p    {color: red;}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

External CSS File

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Tag vs. Class vs. ID

Tag

Use a tag name when writing a rule for a tag

In HTML

```
<ul><li>Home</li><li>About Us</li></ul>
```

In CSS

```
ul {
  list-style-type: none; margin: 0; padding: 0;
}
```

Class

Use a period when writing a rule for a class

In HTML

```
<a class="pdf" href="brochure.pdf">Brochure</a>
```

In CSS

```
.pdf {
  background: url(images/pdf.gif) no-repeat left 50%
}
```

ID

Use a pound sign when writing a rule for a id

In HTML

```
<div id="wrapper">Main Content</div>
```

In CSS

```
#wrapper { width: 750px; margin; 0 auto; }
```

Grouping Selectors

You can group all the selectors of same style to minimize the code. The selectors should be separated with comma.

For Example:

```
h2 { text-align: center; color: red }
p { text-align: center; color: red }
```

Grouped into:

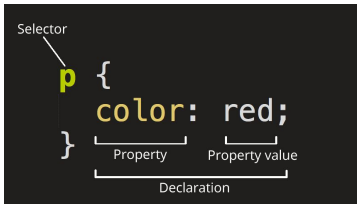
```
h2, p { text-align: center; color: red }
```

Selectors List

div	all DIV tags
div, span	all DIV tags and all SPAN tags
div span	all SPAN tags inside DIVs
#content	element with ID “content”
.box	all elements with CLASS “box”
ul#box	UL tag with ID “box”
span.box	all SPAN tags with CLASS “box”
*	all elements
#box *	all elements inside #box
a:link, a:active, a:visited	links in normal state, in clicked state, and in visited state
a:hover	link with mouse over it
div > span	all SPANs one-level deep in a DIV

Cheatsheet:

https://gist.github.com/magiczny leszek/809a69dd05e1d5f12d01



Pseudo-element selectors

::first-letter

selects the first letter of the specified .foo element, commonly used with :first-child to target first paragraph

```
.foo::first-letter {
    font-size: 30px;
}
```

::before ::after

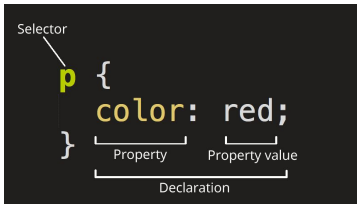
adds generated content before/after the .foo element when used with content property

```
.foo::before {
    color: red;
    content: 'baz';
}
```

```
.foo::after {
    color: red;
    content: 'baz';
}
```

Cheatsheet:

https://gist.github.com/magicznyleszek/809a69dd05e1d5f12d01



Properties

Text

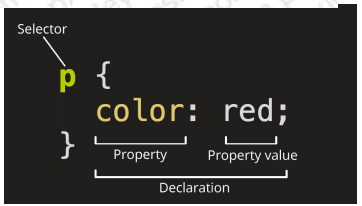
font-family	font used, e.g. Helvetica, Arial
font-size	text size, e.g. 60px, 3em
color	text color, e.g. #000, #abcdef
font-weight	how bold the text is, e.g. bold
font-style	what style the text is, e.g. italic
text-decoration	sets a variety of effects on text, e.g. underline, overline, none
text-align	how text is aligned, e.g. center
line-height	spacing between lines, e.g. 2em
letter-spacing	spacing between letters, e.g. 5px
text-indent	indent of the first line, e.g. 2em
text-transform	applies formatting to text, e.g. upper-case, lowercase, capitalize
vertical-align	align relative to baseline, e.g. text-top

Borders and Lists

border	sets border style for all borders, in the format: border: (solid, dashed, dotted, double) (width) (color), e.g. border: solid 1px #000
border-top border-bottom border-left border-right	sets border style for a specific border (same property syntax used for padding and margin, e.g. margin-left)
list-style-type	sets style of bullets, e.g. square
list-style-position	sets how text wraps when bulleted, e.g. outside, inside
list-style-image	sets an image for a bullet, e.g. list-style-image:url(bullet.png)

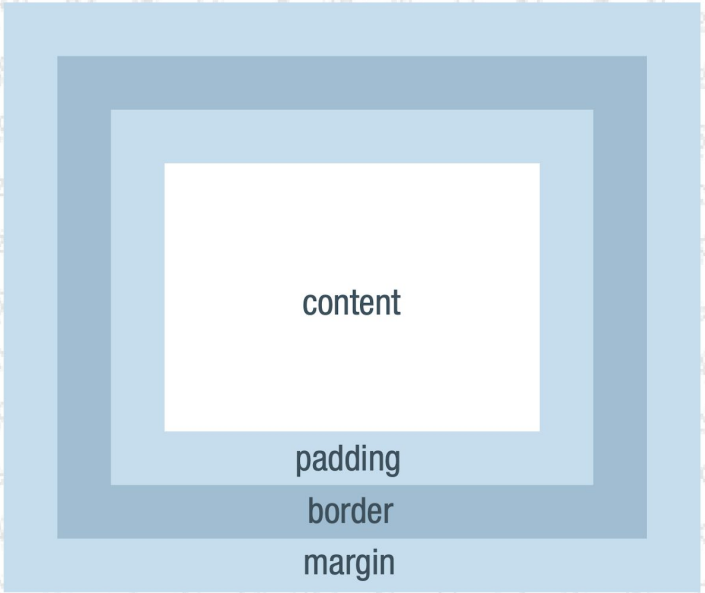
Everything Else

background	sets background of an element, in the format: background: (color) (image) (repeat) (position), e.g. background: #000 url(bg.png) repeat-x top left
cursor	sets shape of cursor, e.g. pointer
outline	a border drawn around an element that doesn't affect the box model
border-collapse	sets how borders within tables behave, e.g. collapse
clear	sets on what side a new line starts in relation to nearby floated elements, e.g. left, right, both



Box Model & Positioning

Box Model



Positioning

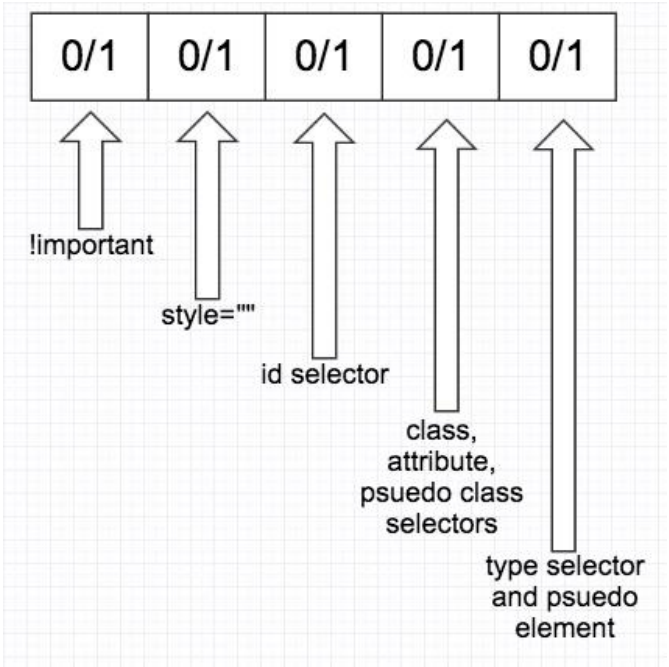
position	places elements on screen, e.g. absolute, fixed, relative
float	stacks elements horizontally in a particular direction, e.g. left
top, left, right, bottom	specifies the offsets used in absolute, fixed, and relative positions, e.g. top:10px;left:10px
display	sets how the element is placed in the doc flow, e.g. block, inline, none
z-index	sets the stacking order of elements, e.g. z-index of 1 is below z-index of 2
overflow	sets what happens to content outside of container, e.g. auto, hidden

JSFiddle - CSS Box Model: http://jsfiddle.net/gocode/68t72xzr/ JSFiddle - z-index: http://jsfiddle.net/gocode/53n86hsz/

CSS Specificity

```
#myDivId .myDivClass
{
    font-size: 20px;
}
```

1 id selector and 1 class selector = 00110




```
/*00030*/
.a .b .c
{
    color: green;
}

/*00020*/
.a .b
{
    color: red;
}
```

```
<div class="a">
    <div class="b">
        <div class="c">
            Some Text
        </div>
    </div>
</div>
```

The first expression had the value 00030 and second expression had value 00020, therefore second expression couldn't override the first expression.

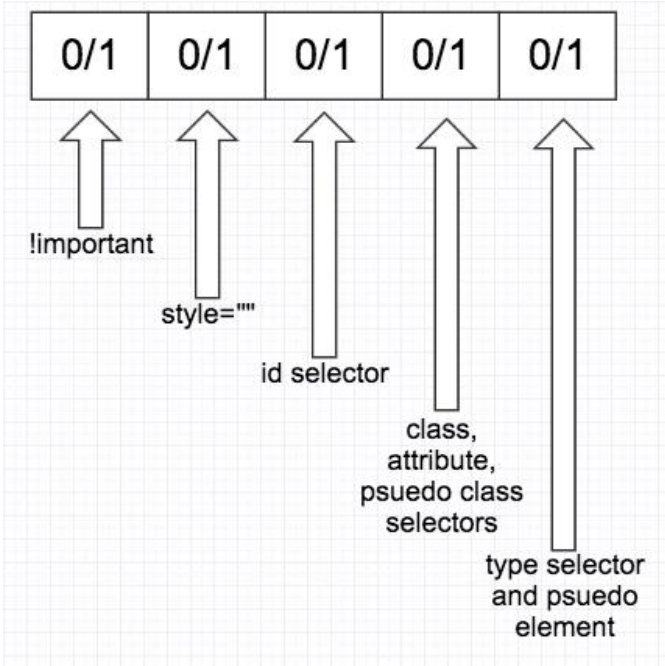
CSS Specificity

Try to reduce use of !important directive

```css
/*00020*/
.a .b
{
    color: red;
}

/*00010*/
.b
{
    /*now becomes 10010*/
    color: yellow !important;
}
```

```html
<div class="a">
    <div class="b">
        Some Text
    </div>
</div>
```



CSS Inheritance

Some property values applied to an element will be inherited by that element's children, and some won't.

Inherited Properties

- font (font-family, font-size etc..)
- color
- text-align
- text-ident
- letter-spacing
- line-height
- list-style-image
- list-style-position
- list-style-type
- list-style
- etc..

Non Inherited Properties

- border (border-width, border-color etc..)
- margin
- padding
- outline
- etc..

Note that browsers set the default color for links ("a" element) to blue automatically.

JSFiddle: https://jsfiddle.net/gocode/sd7x51w8/

CSS Units

em	**Calculated relative to the current font-size of the parent element.** For example, 2em indicates 2 times larger size of the current element's font-size.
ex	Calculated relative the **height** of the current font-size.
px	Pixels size is calculated relative to the viewing device. For low dpi devices 1px is one dot in the display. For high-resolution screens, 1px may indicate multiple device pixels.
%	Percentage value relative to any element. For example, 50% width of the container.
rem	Relative to the font-size of the root element.
vw	Measured as a percentage value of the viewport's width. If the viewport width is 100cm then 1vw=1cm.
vh	Measured as a percentage value of the viewport's height. If the viewport height is 100cm then 1vh=1cm.

Percentages Values vs. Pixels Values

https://jsfiddle.net/gocode/m6tejvbr/

The effect of this is that the first div always has the same width, even if the viewport is resized (it will start to disappear off screen when the viewport becomes narrower than the screen), whereas the second div's width keeps changing when the viewport size changes so that it will always remain 75% as wide as its parent.

Display property - **Block** vs. **Inline-Block** vs. **Inline**

https://jsfiddle.net/gocode/97xn84pq/

block
The element generates a block element box, generating line breaks both before and after the element when in the normal flow.

inline
The element generates one or more inline element boxes that do not generate line breaks before or after themselves. In normal flow, the next element will be on the same line if there is space

inline-block
The element generates a block element box that will be flowed with surrounding content as if it were a single inline box (behaving much like a replaced element would).

visibility: hidden vs. display: none

https://jsfiddle.net/gocode/ucvf5mtn/

Display: none

Completely strips an element from the page.

This means that if you apply display: none to an element, it won't appear on your website and there will be no visible evidence of it ever having existed – meaning that the surrounding elements will treat the element as empty space and adapt accordingly.

Also good for performance saving.

Visibility: hidden

Simply hides an element from the page, while still rendering the tag in the viewport.

This means that even though the element is invisible, there is still space allocated for it on the page, and the surrounding HTML elements will respect that space.

Also, The browser still need to render this element behind the scene.

Position: **static , relative, absolute, fixed**

— — —

http://jsfiddle.net/gocode/vqbo4cuk/

static
The element is positioned according to the normal flow of the document. The top, right, bottom, left, and z-index properties have no effect. This is the default value.

relative
The element is positioned according to the normal flow of the document, and then offset relative to itself based on the values of top, right, bottom, and left. The offset does not affect the position of any other elements.

absolute
The element is removed from the normal document flow, and no space is created for the element in the page layout. It is positioned relative to its closest positioned ancestor, if any; otherwise, it is placed relative to the initial containing block. Its final position is determined by the values of top, right, bottom, and left.
This value creates a new stacking context when the value of z-index is not auto. The margins of absolutely positioned boxes do not collapse with other margins.

fixed
The element is removed from the normal document flow, and no space is created for the element in the page layout. It is positioned relative to the initial containing block established by the viewport, except when one of its ancestors has a transform, perspective, or filter property set to something other than none (see the CSS Transforms Spec), in which case that ancestor behaves as the containing block. (Note that there are browser inconsistencies with perspective and filter contributing to containing block formation.) Its final position is determined by the values of top, right, bottom, and left.
This value always creates a new stacking context. In printed documents, the element is placed in the same position on every page.

Position: **sticky**

https://jsfiddle.net/gocode/40a29euo/

sticky

The element is positioned according to the normal flow of the document, and then offset relative to its *nearest scrolling ancestor* and containing block (nearest block-level ancestor), including table-related elements, based on the values of top, right, bottom, and left. The offset does not affect the position of any other elements.

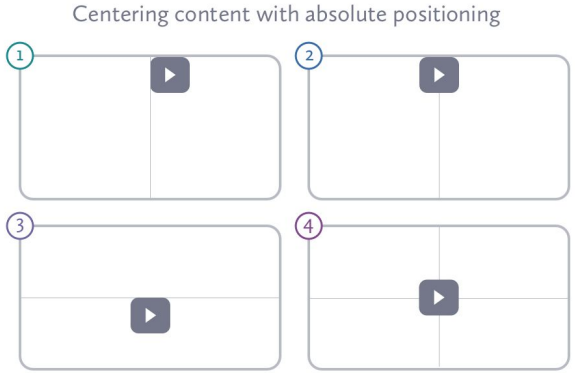
This value always creates a new stacking context. Note that a sticky element "sticks" to its nearest ancestor that has a "scrolling mechanism" (created when overflow is hidden, scroll, auto, or overlay), even if that ancestor isn't the nearest actually scrolling ancestor. This effectively inhibits any "sticky" behavior (see the Github issue on W3C CSSWG).

Use **Position: Absolute** to center element

https://jsfiddle.net/gocode/8a6d2e1z/

1. Add left: 50% to the element that you want to center. You will notice that this aligns the left edge of the child element with the 50% line of the parent.
2. Add a negative left margin that is equal to half the width of the element. This moves us back onto the halfway mark.
3. Next, we'll do a similar process for the vertical axis. Add top: 50% to the child
4. And then add a negative top margin equal to half its height.

But.. we always need to know the centered element width & height...



Original post:
https://robots.thoughtbot.com/positioning

Use **Transform: Translate** to center element

---

https://jsfiddle.net/gocode/13yjwrdu/

Use **position: relative, top: 50%, left: 50% , transform: translate(-50%, -50%)** in the centered element.

Big Advantage - You don't need to know the element's width and height.

There's a simpler way to do this without changing the position using **'flex'**, but it's for another presentation...

Shortends

```
background: url(example.gif);
background-color: #eaeaea ;
background-repeat: repeat-x;
background-position: top left;
```

```
background: #eaeaea url(example.gif) repeat-x top left;
```

```
border-color: red;
border-width: 1px;
border-style: solid;
```

```
border: 1px solid red;
```

```
list-style-position: outside;
list-style-image: none;
list-style-type: disc;
```

```
list-style: disc outside;
```

```
font-family: Arial, Helvetica;
font-weight: bold;
font-style: italic;
font-size: 1em;
line-height: 1.5em;
```

```
font: bold italic 1em/1.5em Arial, Helvetica;
```

Shortends

```
margin-top: 10px;
margin-right: 5px;
margin-bottom: 15px;
margin-left: 20px;
```

```
/* top=10px, right=5px, bottom=15px, left=20px */
margin: 10px 5px 15px 20px;
```

```
padding-top: 10px;
padding-right: 5px;
padding-bottom: 15px;
padding-left: 20px;
```

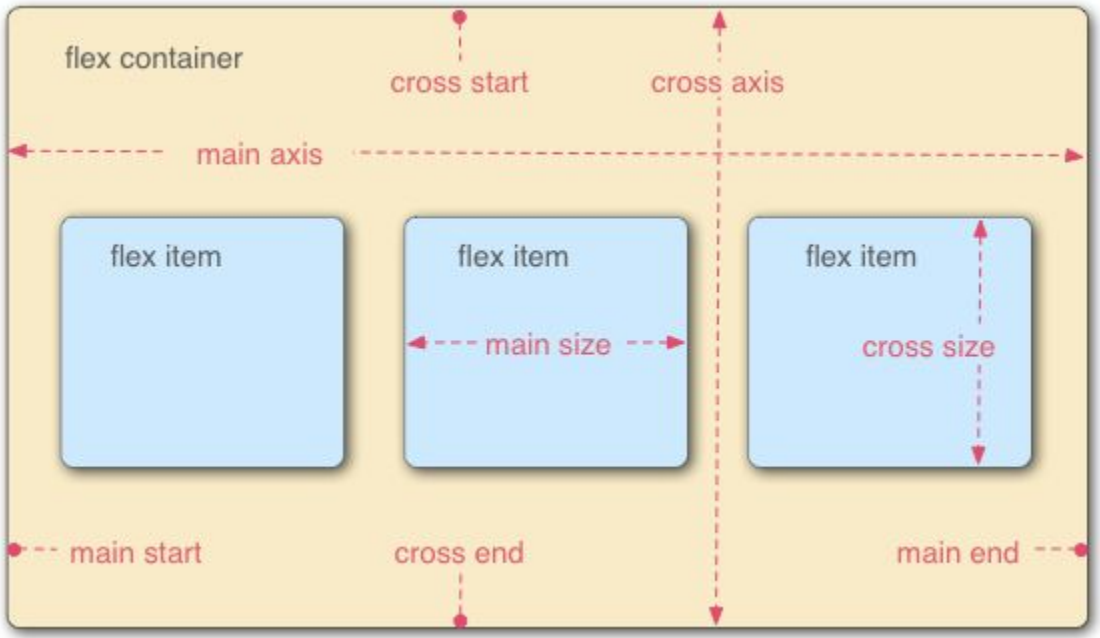
```
/* top=10px, right=5px, bottom=15px, left=20px */
padding: 10px 5px 15px 20px;
```

Flexbox

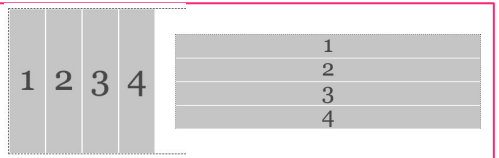
https://yoksel.github.io/flex-cheatsheet

One-dimensional layout method for laying out items in rows or columns

Flexbox Model



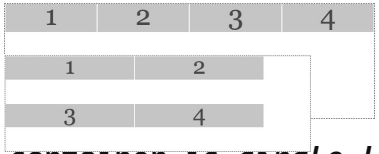
Flexbox - Orientation




```
flex-direction: row;
flex-direction: column;
```

Direction of the flex container's main axis

https://yoksel.github.io/flex-cheatsheet/#flex-direction




```
flex-wrap: nowrap;
flex-wrap: wrap;
```

Controls whether the flex container is *single-line* or *multi-line*

https://yoksel.github.io/flex-cheatsheet/#flex-wrap

```css
.parent {
  display: flex;
}

.child {
  position: relative;
  min-width: 2.5rem;
  min-height: 2.5rem;
  padding: 0.5rem;
  background-color: #bcc8df;
  background-repeat: no-repeat;
  background-position: 50% 50%;
  border: 1px solid #fff;
}
```

Flexbox - Orientation

```
flex-direction: row;
flex-wrap: wrap;
```

→

```
flex-flow: row wrap;
```

Shorthand for setting the flex-direction and flex-wrap
https://yoksel.github.io/flex-cheatsheet/#flex-flow

```
order: -1;
order: 1;
order: 0;
```

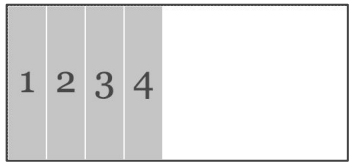
Controls the order in which children of a flex container appear within the flex container
https://yoksel.github.io/flex-cheatsheet/#order

Flexbox - Aligning

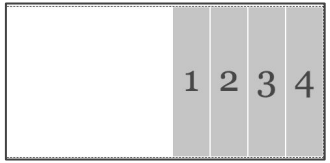
justify-content

Aligns flex items along the main axis of the current line of the flex container
https://yoksel.github.io/flex-cheatsheet/#justify-content

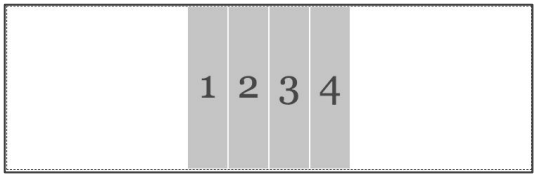
flex-start



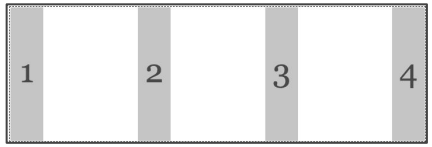
flex-end



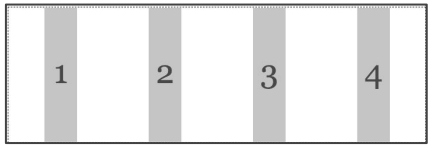
center



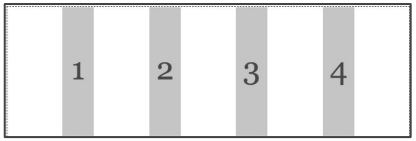
space-between



space-around



space-evenly

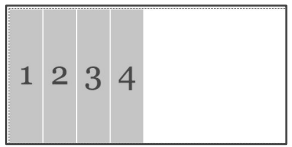


Flexbox - Aligning

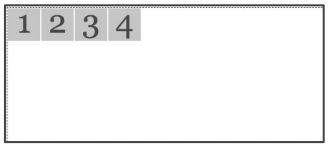
align-items

Controls the alignment of items on the cross-axis
https://yoksel.github.io/flex-cheatsheet/#align-items

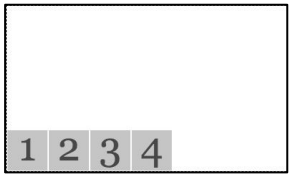
stretch



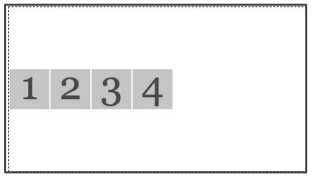
flex-start



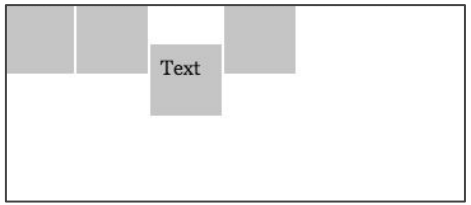
flex-end



center



baseline



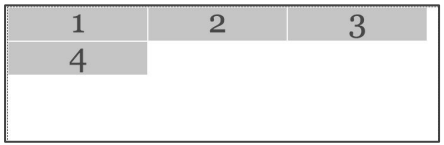
Flexbox - Aligning

align-content (For multiply lines)

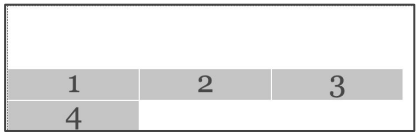
Sets the distribution of space between and around content items
https://yoksel.github.io/flex-cheatsheet/#align-content

stretch

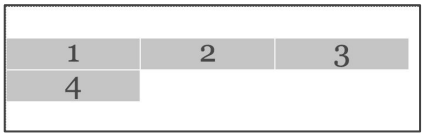
flex-start



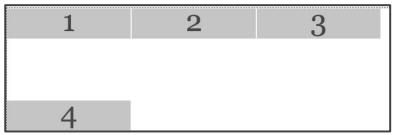
flex-end



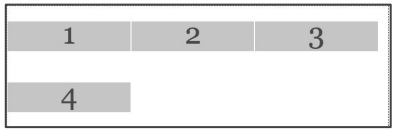
center



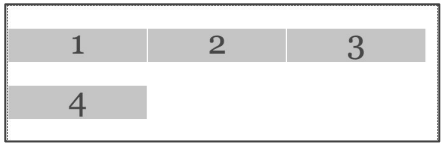
space-between



space-around



space-evenly



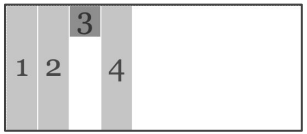
Flexbox - Aligning

align-self (For specific item cross axis)

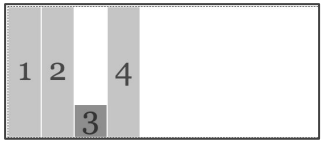
Overrides a flex item's align-items value
https://yoksel.github.io/flex-cheatsheet/#align-self

stretch

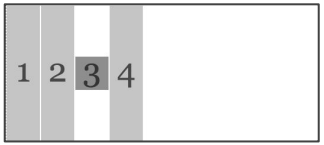
flex-start



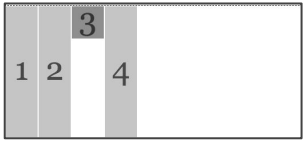
flex-end



center



baseline

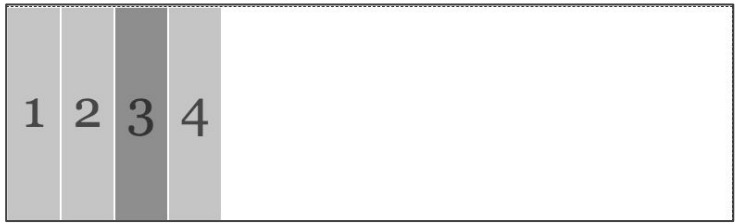


Flexbox - Alignment

flex-grow

Sets the flex grow factor of a flex item main size (width/height according to flex-direction)
https://yoksel.github.io/flex-cheatsheet/#flex-grow

0



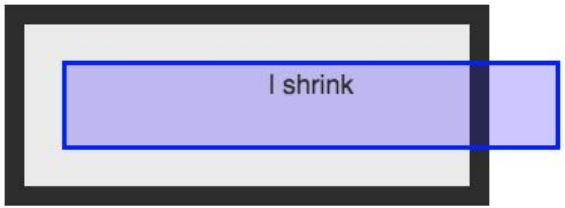
1

Flexbox - Alignment

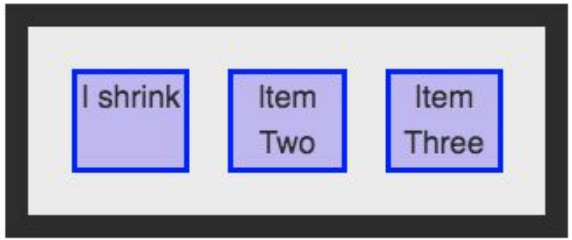
`flex-shrink`

Sets the flex shrink factor of a flex item. If the size of all flex items is larger than the flex container, items shrink to fit according to flex-shrink
https://developer.mozilla.org/en-US/docs/Web/CSS/flex-shrink

0



1

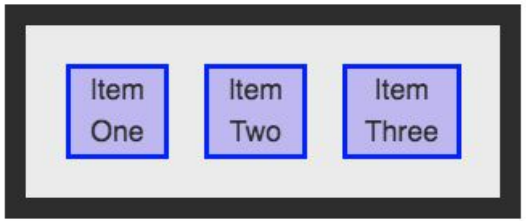


Flexbox - Alignment

`flex-basis`

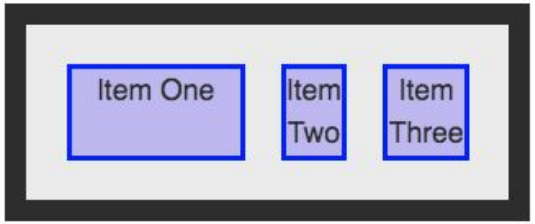
Sets the initial main size of a flex item
https://developer.mozilla.org/en-US/docs/Web/CSS/flex-basis

`auto`



`0`

`200px`



Flexbox - Shortend

On item:

flex: <flex-grow> <flex-shrink> <flex-basis>

Initial: flex: 0 1 auto

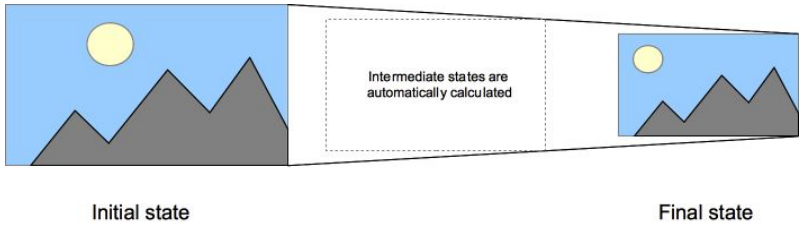
flex: <positive-number> ⟶ **flex:** <positive-number> **1 0;**

Common usage:

flex: 1

Summary:
https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Transition



Transitions enable you to define the **transition between two states of an element**. Different states may be defined using pseudo-classes like **:hover** or **:active** or **dynamically** set using JavaScript.

Shorthand property for transition-property, transition-duration, transition-timing-function, and transition-delay

```
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 2s;
}
```

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

We can also write 2 property transitions: `transition: width 2s, height 4s;`

Transition

```
div {

  transition: <property> <duration> <timing-function> <delay>;

}
```

transition-property
Specifies the name or names of the CSS properties to which transitions should be applied. (like **'width'**)

transition-duration
Specifies the duration over which transitions should occur. (like **'4s'**)

transition-timing-function
Specifies a function to define how intermediate values for properties are computed. (like **'linear'**)

transition-delay
Defines how long to wait between the time a property is changed and the transition actually begins. (like **'1s'**)

Animation

```
p {
  animation-duration: 3s;
  animation-name: slidein;
}

@keyframes slidein {
  from {
    margin-left: 100%;
    width: 300%;
  }

  to {
    margin-left: 0%;
    width: 100%;
  }
}
```

CSS animations make it possible to animate transitions from one CSS style configuration to another. Animations consist of two components, a style describing the CSS animation and a set of keyframes that indicate the start and end states of the animation's style, as well as possible intermediate waypoints.

Animation

```
p {
  animation-duration: 3s;
  animation-name: slidein;
}

@keyframes slidein {
  from {
    margin-left: 100%;
    width: 300%;
  }

  75% {
    font-size: 300%;
    margin-left: 25%;
    width: 150%;
  }

  to {
    margin-left: 0%;
    width: 100%;
  }
}
```

We can tell the browser that 75% of the way through the animation sequence, the header should have its left margin at 25% and the width should be 150%.

Animation - Shortend

```css
div {
  animation-name: example;
  animation-duration: 5s;
  animation-timing-function: linear;
  animation-delay: 2s;
  animation-iteration-count: infinite;
  animation-direction: alternate;
}
```



```css
div {
  animation: example 5s linear 2s infinite alternate;
}
```

Transform

The transform property allows you to visually manipulate an element by skewing, rotating, translating, or scaling.

original

transform: rotate(0.5turn);

transform: skew(10deg, 20deg);

Don't forget to check for browsers support

CanIUse.com

For example, the :in-range and :out-of-range CSS pseudo-classes support can be found here: https://caniuse.com/#feat=css-in-out-of-range

Exercises

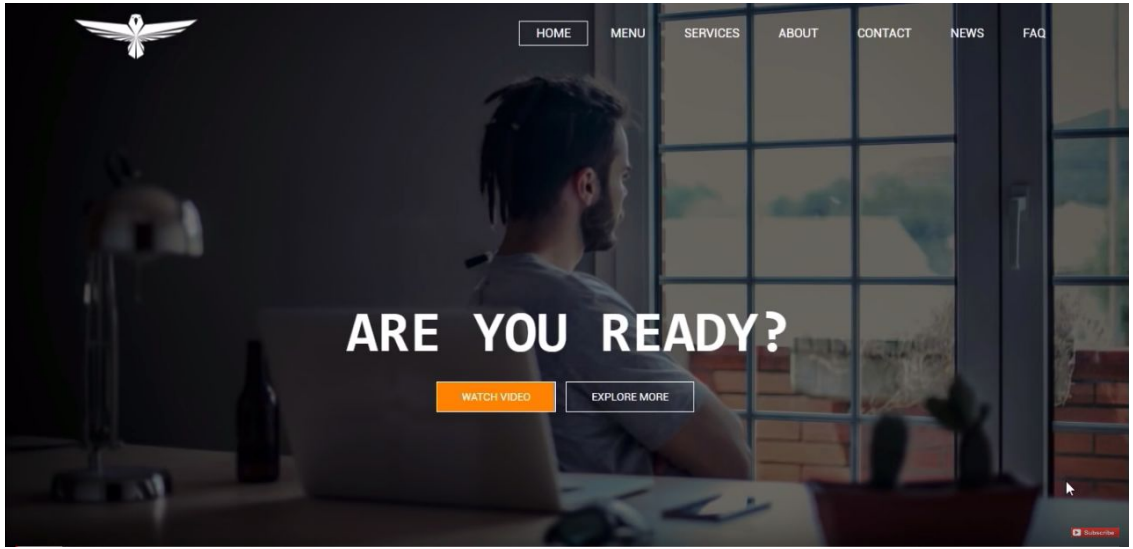
1. Answer this quiz (4 questions):
   https://www.khanacademy.org/computing/computer-programming/html-css/css-text-properties/e/quiz--text-properties
2. Read about CSS Gradients and try to recreate some of the following examples. Don't worry about matching the colors perfectly.

3. Create a **nav** element with **ul** and **li** items for menu links (For example: Home, Articles, About Us). Use CSS to show them nicely in the same row. Demo:

4. **Advanced:** Take any page from a site you like (maybe a simple one), put it on screenshot and try to "convert" it to HTML & CSS.

Exercises in class

1. What cause this problem and how to solve it?
   https://codepen.io/chriscoyier/pen/ClGcF
2. CSS Selectors Exercises - https://flukeout.github.io/
3. Create a chess board using the minimum HTML & CSS code.
4. Try to design this site:



Exercises in class - Advanced

1. http://davidshariff.com/quiz/
2. http://pixact.ly/
3. http://htmlacademy.org/

More Info

1. https://www.khanacademy.org/computing/computer-programming/html-css#intro-to-css
2. https://www.youtube.com/watch?v=wNX7lWzchow
3. https://www.youtube.com/watch?v=yfoY53QXEnI
4. https://benhowdle.im/cssselectors/