Tailwindcss - A great CSS system that you have to be able to do in the 2020's!
This document is part of the online TailwindCss course on Udemy:
https://www.udemy.com/course/tailwindcss-with-examples

Tailwindcss - advanced Configuration

If you plan to work with Tailwind CSS on a professional level, you have to know about the configuration possibilities.

Advanced Configuration of Tailwind CSS

New project and copy Tailwind CSS installation

We are going to start another project now.
If you want you can do another tailwindcss installation, the same way I tought you in the previous course section. You can use my "5-step installation guide".

But there is also another, **faster** possibility, by copying the previous "Mini project" into an entire new project, by usage of the **command line**, which I will show you now:

Step 1: Please **open** the mini project in your Visual Studio Code.

Step 2: Open the command line with **Terminal > New Terminal**

Step 3: Type `cd../` (+ hit Enter) to go one level up in your folder hierarchy. Then you can do an " `ls` " to list the parent folder contents and you should see the folder of the mini project now:

```
Martins-MBP:miniproject martineberth$ cd ../
Martins-MBP:tailwindcss martineberth$ ls
miniproject
Martins-MBP:tailwindcss martineberth$
```

Step 4: Copy the entire "miniproject" folder into a new folder with the name e.g. "part2" (or whatever you like) by saying: `cp -R miniproject part2`
It might take around 2 seconds for your computer to do the copying.
If you do another " `ls` " now, you can see the new folder parallel to the original folder:

```
miniproject
Martins-MBP:tailwindcss martineberth$ cp -R miniproject part2
Martins-MBP:tailwindcss martineberth$ ls
miniproject      part2
Martins-MBP:tailwindcss martineberth$
```

Step 5: Enter your new folder *part2* with `cd part2`, and remove the .git folder with: `rm -rf .git`. Then do a `git init`, to create a new git project.

Step 6: Open the new folder "part2" now in your Visual Studio Code and again open the terminal.

Step 7: Open */dist/index.html,* delete the whole content, and create another empty HTML document with **! + Tab**. In this document, set another title, e.g. "Part 2" and enter some dummy content, in the body e.g. <h1>Hello again</h1>

Tailwindcss - A great CSS system that you have to be able to do in the 2020's!
This document is part of the online TailwindCss course on Udemy:
https://www.udemy.com/course/tailwindcss-with-examples

Step 8: In **/tailwind.css delete all the styles** WITHIN the **body** and the **container** class selector. Then in the body selector you just add a class of **@apply debug-screens;**

Step 9: In the HTML document include styles.css by adding to the body:
<link rel="stylesheet" href="css/styles.css">

Step 10: In the terminal run `npm run watch` and then do a right mouse click somewhere in the *index.html* and choose "Open with live server".

Step 11: In tailwind.config.js change the hex value of "**mainColor**" to **#1E293B**.

Tailwindcss - A great CSS system that you have to be able to do in the 2020's!
This document is part of the online TailwindCss course on Udemy:
https://www.udemy.com/course/tailwindcss-with-examples

Further configuration options

To get an overview of the preset configuration and thus the configuration options, you can have a look here:
https://unpkg.com/browse/tailwindcss@2.0.2/stubs/defaultConfig.stub.js

"Theme" contains the following properties in JavaScript object notation:

screens	colors	spacing	animation	backgroundColor
backgroundImage	backgroundOpacity	backgroundPosition	backgroundSize	borderColor
borderOpacity	borderRadius	borderWitdth	boxShadow	container
cursor	divideColor	divideOpacity	divideWidth	fill
flex	flexGrow	flexShrink	fontFamily	fontSize
fontWeight	gap	gradientColorStops	gridAutoColumns	gridAutoRows
gridColumns	gridColumnEnd	gridColumnStart	gridRow	gridRowStart
gridRowEnd	transformOrigin	gridTemplateColumns	gridTemplateRows	height
inset	keyframes	letterSpacing	lineHeight	listStyleType
margin	maxHeight	maxWidth	minHeight	minWidth
objectPosition	opacity	order	outline	padding
placeHolderColor	placeHolderOpacity	ringColor	ringOffsetColor	ringOffsetWidth
ringOpacity	ringWidth	rotate	skale	skew
space	stroke	strokeWidth	textColor	textOpacity
transitionDuration	transitionDelay	transitionProperty	transitionTimingFunction	
translate	width	zIndex	variantOrder	variants

Each of these 79 properties can be assigned its own values.
There are generally 2 options for this:

Option 1: Overwrite the property directly under **"theme"**, but be careful! Then all other values will be overwritten!, A blatant example would be adding a single color to the theme!, But then there really is only this color!

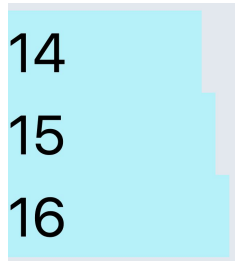**Option 2: Extend** the property under *"extend"*

Example: Expand the spacing with a size "15" = 3.75rem

```
extend: {
  spacing: {
    15 : '3.75rem'
  }
}
```

With this the following could be realized:

```
<div class=" w-14 bg-tuerkis-200 mt-10">14</div>
<div class=" w-15 bg-tuerkis-200">15</div>
<div class=" w-16 bg-tuerkis-200">16</div>
```

Which then leads to the following result:

14

15

16

By the way, there are also points where you have to be careful!
For example, if you were to define a screen size of "largeScreen" or similar within "extend" with 1000px, then the smaller screen sizes "sm" and "md" would still be there, but the breakpoints larger than a 1000px would not be there anymore.
If in doubt, always read the documentation, e.g. here:
https://tailwindcss.com/docs/breakpoints#extending-the-default-breakpoints

Tailwindcss - A great CSS system that you have to be able to do in the 2020's!
This document is part of the online TailwindCss course on Udemy:
https://www.udemy.com/course/tailwindcss-with-examples

Background: Advanced colors

We have a preset color palette in Tailwind CSS. This preset palette is a **subset** of an even larger color palette.

The preset palette has **8 colors** with 10 saturations each:
https://tailwindcss.com/docs/customizing-colors#overview

The extended palette has **22 colors** with 10 saturation levels each:
https://tailwindcss.com/docs/customizing-colors#color-palette-reference

The preset palette has the following colors:

1. Gray = Cool Gray	

Alternatives:

Blue Gray (colder)	
Gray (warmer)	
True Gray (warmer)	
Warm Gray (warmer)	

2. Red = Red	

Alternatives:

Rose	
Orange	

3. Yellow = Amber	

Alternative:

Yellow	

4. Green = Emerald	

Alternatives:

Lime (towards yellow)	
Green (neutral)	
Teal (towards blue)	

5. Blue = Blue	

Alternatives:

Light Blue (slightly towards green)	
Cyan (towards green)	

6. Indigo = Indigo	

7. Purple = Violet	

Alternatives:

Purple	
Fuchsia	

8. Pink = Pink	

Tailwindcss - A great CSS system that you have to be able to do in the 2020's!
This document is part of the online TailwindCss course on Udemy:
https://www.udemy.com/course/tailwindcss-with-examples

Example: Implement your own color palette

I would like to implement a portal for luxury real estate. I already have images for this:
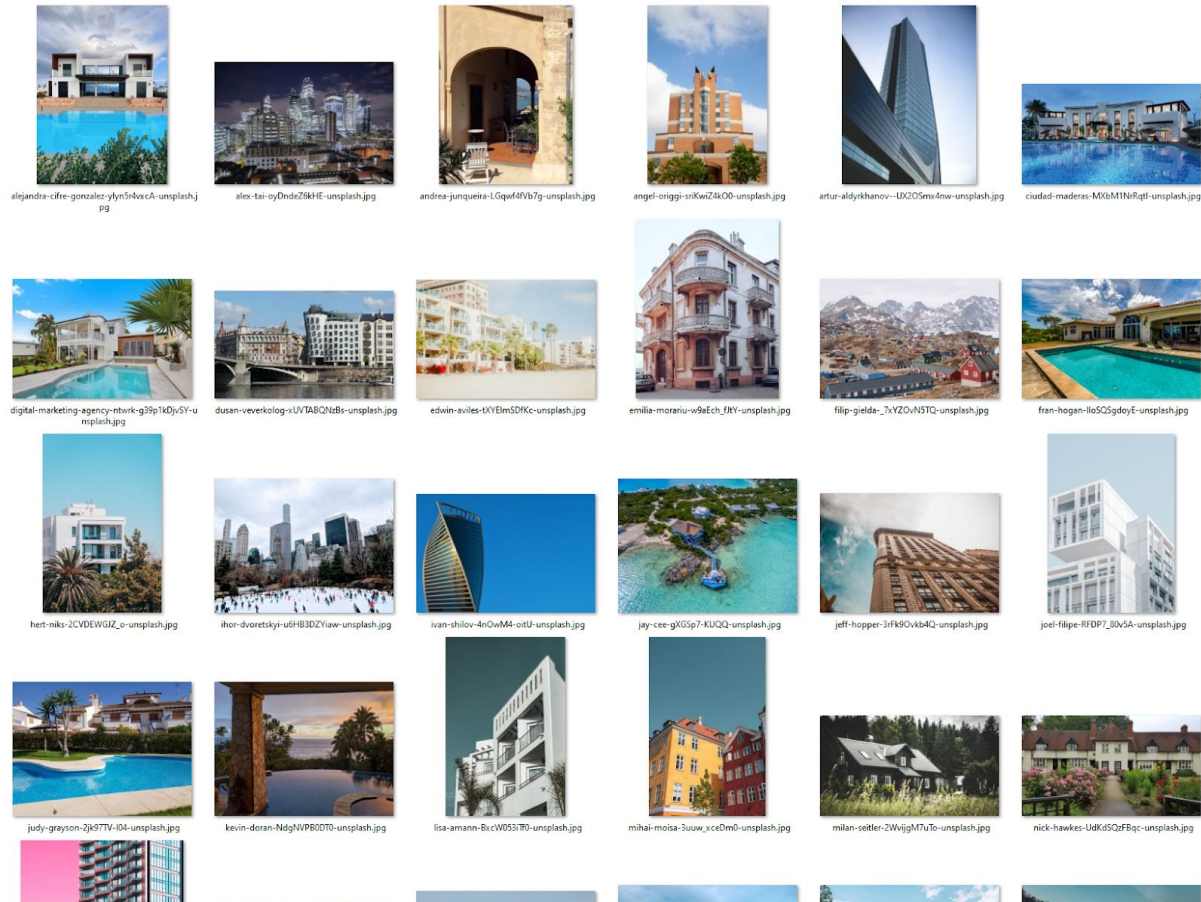
If you look at the pictures, you can see that there are quite a lot of **turquoise tones**
included, e.g. in "pool pictures, beach pictures, but also in the sky)
In addition to a cold **"concrete" gray**, many buildings also have tones between **brown**,
orange and a **warm red**.
If an image contains plants, then the green is more like a **yellowish green**. Occasionally a
neutral blue also occurs.

But I also have to think of the two **main colors** (primary and secondary color). Here I want to
express **seriousness** on the one hand = **blue**, but on the other hand I also want to
emphasize the stability of the **investment** value = **gold = yellow**.

That's why I choose the following colors:

Tailwindcss - A great CSS system that you have to be able to do in the 2020's!
This document is part of the online TailwindCss course on Udemy:
https://www.udemy.com/course/tailwindcss-with-examples

Main color: **blue** (Blue)

Blue colors.blue	50	100	200	300	400	500	600	700	800	900
	#EFF6FF	#DBEAFE	#BFDBFE	#93C5FD	#60A5FA	#3B82F6	#2563EB	#1D4ED8	#1E40AF	#1E3A8A

Second main color: **yellow** (Yellow)

Yellow colors.yellow	50	100	200	300	400	500	600	700	800	900
	#FEFCE8	#FEF9C3	#FEF08A	#FDE047	#FACC15	#EAB308	#CA8A04	#A16207	#854D0E	#713F12

Neutral color: **gray** (Blue Gray)

Blue Gray colors.blueGray	50	100	200	300	400	500	600	700	800	900
	#F8FAFC	#F1F5F9	#E2E8F0	#CBD5E1	#94A3B8	#64748B	#475569	#334155	#1E293B	#0F172A

Secondary color 1: **turquoise** (Cyan)

Cyan colors.cyan	50	100	200	300	400	500	600	700	800	900
	#ECFEFF	#CFFAFE	#A5F3FC	#67E8F9	#22D3EE	#06B6D4	#0891B2	#0E7490	#155E75	#164E63

Secondary color 2: **green** (Lime)

Lime colors.lime	50	100	200	300	400	500	600	700	800	900
	#F7FEE7	#ECFCCB	#D9F99D	#BEF264	#A3E635	#84CC16	#65A30D	#4D7C0F	#3F6212	#365314

Accent color: **red** (Rose)

Rose colors.rose	50	100	200	300	400	500	600	700	800	900
	#FFF1F2	#FFE4E6	#FECDD3	#FDA4AF	#FB7185	#F43F5E	#E11D48	#BE123C	#9F1239	#881337

GO UP

Create your own color theme

To create your own color palette, you do 2 steps in *tailwind.config.js*:

Step 1:
At the very top you add: const colors = require('tailwindcss/colors')

By the way, this corresponds to the file:
/node_modules/tailwindcss/colors.js

In this file colors are there as hex values !! You can use it, because Tailwind only gives you
the RGB values! :-)

Step 2:
Then you define your own colors under "Theme". Example:
```
  colors: {
    transparent: 'transparent',
    current: 'currentColor',
    white: colors.white,
    black: colors.black,
    blue: colors.blue,
    yellow: colors.yellow,
    gray: colors.blueGray,
    turquoise: colors.cyan,
    green: colors.lime,
    red: colors.rose,
  },
```

npm run watch must be running for compilation!

Background: color gradients

https://tailwindcss.com/docs/background-image

We add a slight gradient for the entire "body" from top to bottom for the background. As you can see in the documentation, the gradients go in different directions, we take the variant "**gradient-to-b**" (to bottom).

In total we get:
```
body {
    @apply debug-screens bg-gradient-to-b from-grau-100 to-grau-400 h-screen;
}
```

Now that our body has a gradient that starts at bg-gray-100, and our container also has the background color bg-gray-100, we add a shadow to our container:
```
.container {
    @apply bg-grau-100 h-screen shadow-sm;
}
```

Outsourcing your own theme (presets)

https://tailwindcss.com/docs/presets

Practical approach:
1. Save **tailwind.config.js** under a new name (e.g. company_styles.js)

2. Then everything is removed from this new file that should NOT be variable from project to project and should remain in *tailwind.config.js.*

3. Everything is taken out of the *tailwind.config.js* which is imported via the outsourced file.

4. The outsourced file is **imported** again **via presets:[...]**, example:

```
presets: [
  require('./company_styles')
],
```

Presets for container and body

There are special default settings in Tailwind CSS regarding **centering** and **side padding** for the container class:

```
container: {
    center: true,
     padding: {
      DEFAULT: '0.5rem',
      sm: '1rem',
      lg: '1.5rem',
      xl: '2rem'
    },
  },
```

Übrigens würde ich DIESE SPEZIELLEN Voreinstellungen tatsächlich direkt in *tailwind.config.js* machen und NICHT in einem Preset, weil die Sachen ja von Projekt zu Projekt unterschiedlich sein können.
Wir kopieren also diesen Abschnitt direkt in die *tailwind.config.js* unter theme rein:

```
darkMode: false, // or 'media' or 'class'
theme: {
  debugScreens: {
    position: ['top', 'left'],
  },
  container: {
    center: true,
      padding: {
        DEFAULT: '0.5rem',
        sm: '1rem',
        lg: '1.5rem',
        xl: '2rem'
      },
  },
  extend: {
  },
```

If you are building a project with several sub-pages, it is also advisable to include the background colors for body and container, as well as the "debug" class for the body directly in tailwind.css: (We can then delete the classes in HTML)

```
@layer base {
  body {
    @apply debug-screens bg-gradient-to-b from-grau-100 to-grau-400 h-screen;
  }
  .container {
    @apply bg-grau-100 h-screen;
  }
}
```

Deactivate individual features

If you have received certain style guide specifications from the designer, and you want to make sure that the Frontent developer(s) will definitely NOT do certain things in the future, then you would do well to deactivate the corresponding features in TailwindCSS from the start so to speak, nobody is ever "tempted" to do something wrong.
Of course, when restricting features, you have to be careful not to restrict your options too much on the other hand.

We have already used one of the options for the colors, by defining our own colors in *tailwind.config.js*, but also deactivating all other colors at the same time.

But there is more: In the documentation you can see the last point on the right for every feature page is "**Disabling**"
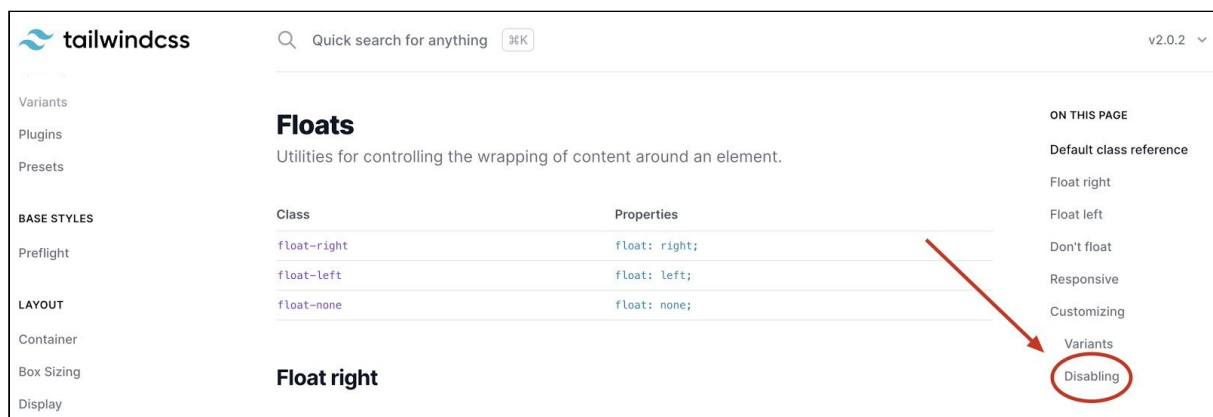
To deactivate a feature, set the property within "**corePlugins**" in tailwind.config.js to **false**.
If "corePlugins" does not yet exist, it must be created first. Example:

```
corePlugins : {
  float: false
}
```

continues on the next page...
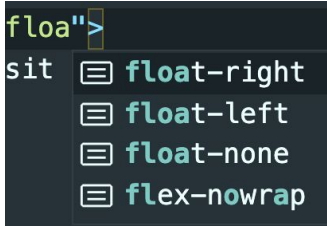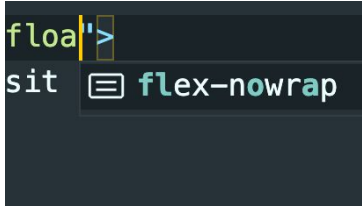
As soon as you have switched off a property, it no longer appears in the suggestions
(after npm compilation):

Before	After

The **names of the properties** are usually given in the **camel case**, for example:
"BoxSizing, objectFit, flexGrow" etc.

However, the name of the underlying CSS property is always used, e.g. in Tailwindcss, as
we know, the property of the line spacing is described as "leading-". But if we wanted to
switch off the line spacing, the corresponding instruction would be: "**lineHeight: false**"

In case of doubt, you should always check the documentation.

Tailwindcss - A great CSS system that you have to be able to do in the 2020's!
This document is part of the online TailwindCss course on Udemy:
https://www.udemy.com/course/tailwindcss-with-examples

Expand / restrict variants

"Variant" means the following: "first, last, odd, even, visited, checked, group-hover, group-focus, focus-within, hover, focus, focus-visible, active, disabled", as well as: "
responsive "and" **dark** ".

Here I would like to refer again to the following document:
https://unpkg.com/browse/tailwindcss@2.0.2/stubs/defaultConfig.stub.js

Here you can see which variants are available (from line 718),
or which variants are preset for the individual properties (from line 734)

Example:
```
<div class="mt-10 mb-10 w-24 hover:w-96 h-24 bg-gelb-700 hover:bg-blau-400"></div>
```
This creates a yellow box, which turns blue when hovering. The width hover does not work, however.
This is because no hover is enabled by default for the width:

```
width: ['responsive'],
```

If we now want the hover variant to also work for the **width**, then we can specify this in
tailwind.config.js as follows:

```
variants: {
  extend: {
    width: ['hover']
  },
},
```

Now the box at Hover is also getting wider.

Now we are adding a change to the width of our div when the ViewPort is changed to size **lg: lg: w-72**

As we can see, this works wonderfully because the "responsive" variant is preset for width, and we
have only **expanded** the variant.

But if we now take the specification *width: ['hover']* from the "extend block":

```
variants: {
  extend: {
  },
  width: ['hover']
},
```

Then only the hover variant works and the responsive variant no longer works!

In order to fix this, we would have to explicitly list the "responsive variant" again when a variant was changed outside of "extend":

```
variants: {
  extend: {
  },
  width: ['responsive','hover']
},
```