

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>

Tailwind CSS

PREPARATION / REQUIREMENTS

Editor: Visual Studio Code

Node.js

Installation & Basic Skills

Install Tailwindcss in 5 steps

Get styles and workflow up and running

Mini Project

Container + Breakpoints + Debug Screen Plugin

Format and preset headings

Define your own heading font

Add your own color

Background: Free icons

Introduction to responsive design with Tailwindcss

Background patterns

Quote above with quotation marks and other color

Create "Dark Mode"

Brief introduction to Alpine.js + Darkmode button

Dynamic display of the dark mode button

Previously used directives in Alpine.js

Outsource button styles

Shrink CSS with Purgecss

This E-Book belongs to the Tailwind-Css-Course on Udemy.

The Link to the course is: <https://www.udemy.com/course/tailwindcss-with-examples>

PREPARATION / REQUIREMENTS

In this course section you learn which software you should install and use on your computer to work efficiently with TailwindCSS.

Editor: Visual Studio Code

[Go to lecture >](#)

In this course we use **Visual Studio Code**. It is free and you [can download it here](#) for all operating systems.

Extensions for VSC:

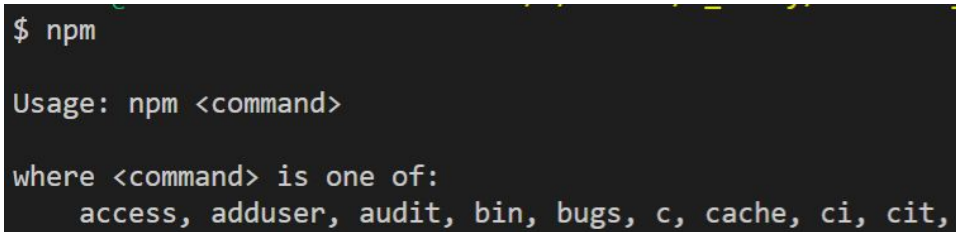
- Headwind
- Live Server
- Tailwind CSS IntelliSense
- Alpine.js IntelliSense

Node.js

[Go to lecture >](#)

You can download Node here: <https://nodejs.org/en/download/>

After the installation you can test Node by simply typing “npm” into a terminal window:

A terminal window with a dark background and light green text. The prompt is '\$ npm'. Below it, the text 'Usage: npm <command>' is shown. Then, 'where <command> is one of:' is displayed, followed by a list of commands: 'access, adduser, audit, bin, bugs, c, cache, ci, cit,'. The list is truncated with an ellipsis.

```
$ npm

Usage: npm <command>

where <command> is one of:
  access, adduser, audit, bin, bugs, c, cache, ci, cit,
  etc.
```

etc.

[GO UP](#)

Copyright: The creator and owner of this document is Martin Krebs Eberth, masterclassudemy@gmail.com

You can copy this document and spread this far and wide as long as you mention the link to the according course on Udemy.

Installation & Basic Skills

[Go to lecture >](#)

Install Tailwindcss in 5 steps

Step 1: *Specify the installation location*

First of all, you have to define a folder on your computer where you want to put the examples from this course.

We will do several Tailwindcss installations in this course, so I suggest you give this folder a general name, such as. "Udemy_tailwindcss" or something similar. In this folder, please create a subfolder "miniproject".

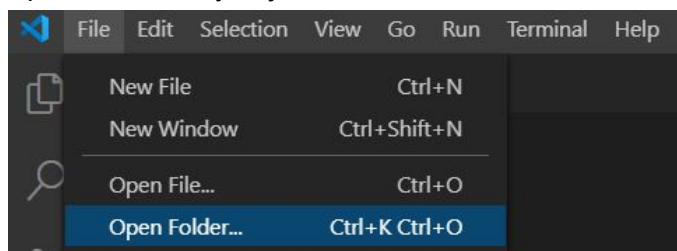
You can find the official instructions for Tailwindcss here:

<https://tailwindcss.com/docs/installation>

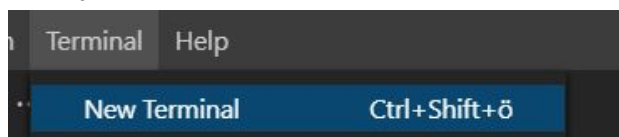
But be careful! We will deviate slightly from the official version because I have found a solution that is more practical for me.

Step 2: *Enter the folder in the VSC and open the terminal*

Open the folder you just created in Visual Studio Code:



Then you open a terminal window:



Step 3: *initialize Git and generate .gitignore*

Before you do anything, first create a Git repository:

```
git init
```

Then you create a **.gitignore** file and open it in the editor

There you insert:

[node_modules/](#)

Then save and close the .gitignore again.

[GO UP](#)

Copyright: The creator and owner of this document is Martin Krebs Eberth, masterclassudemy@gmail.com

You can copy this document and spread this far and wide as long as you mention the link to the according course on Udemy.

This E-Book belongs to the Tailwind-Css-Course on Udemy.

The Link to the course is: <https://www.udemy.com/course/tailwindcss-with-examples>

Step 4: Install Tailwindcss + Tools

In the terminal you execute the following 3 commands:

```
npm init -y
```

> This creates the **package.json** file

```
npm install tailwindcss postcss postcss-cli autoprefixer
```

Attention: The postcss-cli is not mentioned in the official documentation. But we need it! (see below)

```
npx tailwindcss init -p
```

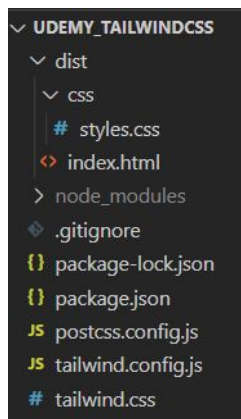
- This creates the **tailwind.config.js**
(The file is important, otherwise autocomplete will not work in VS Code)
- The -p ensures that the **postcss.config.js** file is created at the same time.

Step 5: Create folder structure and files

Create:

- *tailwind.css*
- Folder “*dist/css*” (everything goes together in VSC!)
> The code for publishing is then in the “dist” folder
- A file “**styles.css**” in the css folder
- A file **index.html** in /dist/
- In the index.html with “! + TAB” create an HTML frame

Your structure should now look like this:



[GO UP](#)

Copyright: The creator and owner of this document is Martin Krebs Eberth, masterclassudemy@gmail.com

You can copy this document and spread this far and wide as long as you mention the link to the according course on Udemy.

Get styles and workflow up and running

[Go to lecture >](#)

[View Commit of this lecture on GitHub >](#)

Step 1: the 3 parts of Tailwindcss

Copy the 3 lines into tailwind.css:

```
@tailwind base;  
@tailwind components;  
@tailwind utilities;
```

Step 2: embed styles.css in HTML

In the */dist/index.html*

```
<link rel="stylesheet" href="css/styles.css">
```

Step 3: set up compilation scripts

In the */package.json*

```
"scripts": {  
  "build": "postcss tailwind.css -o dist/css/styles.css",  
  "watch": "postcss tailwind.css -o dist/css/styles.css --watch"  
},
```

Attention: Set the correct input / output files if you are using a different folder structure.

Note: These scripts only work with the “**postcss-cli**”. That is the reason why we installed this before - different from the official Tailwindcss documentation!
(CLI = Comand Line Interface)

Step 4: Compile the CSS

In the terminal you now run either `npm run build` or `npm run watch` to fill your styles.css with Tailwind styles.

The difference is that `npm run build` is a one-time compilation, while `npm run watch` “waits” for changes in the background.

The result is a */dist/styles.css* with all the tailwind.css styles and a file size of **almost 4 MB!**
But don't worry: we will dramatically reduce the CSS later! :-)

Step 5: test the tailwind styles

First we have to restart the editor (VSC) for the Tailwind Intellisense plugin to work!

In index.html we can now give the body a background color:

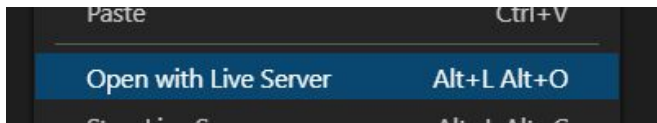
```
<body class="bg-blue-400">
```

Then we open the page with the **live server** (right-click in the index.html):

[GO UP](#)

This E-Book belongs to the Tailwind-Css-Course on Udemy.

The Link to the course is: <https://www.udemy.com/course/tailwindcss-with-examples>



Then we should see a blue page!

Step 6: Make the first *git* commit

We execute in the command line:

git status

This shows us the changed files (in red):

```
Untracked files:
(use "git add <file>..." to include in what will be committed)
.gitignore
dist/
package-lock.json
package.json
postcss.config.js
tailwind.config.js
tailwind.css
```

As we can see, the **node_modules** are NOT included, what we achieved with the **.gitignore**.

git add -A

That adds us all of the files to Git (now in green):

git status

```
Changes to be committed:
(use "git rm --cached <file>..." to unstage)
new file:   .gitignore
new file:   dist/css/styles.css
new file:   dist/index.html
new file:   package-lock.json
new file:   package.json
new file:   postcss.config.js
new file:   tailwind.config.js
new file:   tailwind.css
```

git commit -m 'Get styles and workflow up and running'

Warning: Use single quotation marks for the commit message!

git status

```
$ git status
On branch master
nothing to commit, working tree clean
```

Now we've done a first commit = "Save" in Git.

[GO UP](#)

Copyright: The creator and owner of this document is Martin Krebs Eberth, masterclassudemy@gmail.com

You can copy this document and spread this far and wide as long as you mention the link to the according course on Udemy.

This E-Book belongs to the Tailwind-Css-Course on Udemy.

The Link to the course is: <https://www.udemy.com/course/tailwindcss-with-examples>

Mini Project

Container + Breakpoints + Debug Screen Plugin

[Go to lecture >](#)

[View Commit of this lecture on GitHub >](#)

You can start many websites by first creating a **centered** container that spans the **entire height** of the screen:

```
<div class="container min-h-screen mx-auto bg-gray-100">
</div>
```

As you can see, this container is already responsive, if you change the width of the browser window, you can observe how the ratio of edge and container width adapts at certain breakpoints.

To make the breakpoints visible during the development of the website, there is the “**Tailwind Debug Screens**” plug-in:

Installation

```
npm install tailwindcss-debug-screens --save-dev
```

Insert in *tailwind.config.js*

```
module.exports = {
  //...
  plugins: [
    require('tailwindcss-debug-screens'),
  ]
}
```

Optional - place at the **top left** (in the tailwind.config.js)

```
theme: {
  debugScreens: {
    position: ['top', 'left'],
  },
}
```

Apply in the HTML document

```
<body class="debug-screens">
```

Important: run `npm run build` for activation.

[GO UP](#)

Copyright: The creator and owner of this document is Martin Krebs Eberth, masterclassudemy@gmail.com

You can copy this document and spread this far and wide as long as you mention the link to the according course on Udemy.

This E-Book belongs to the Tailwind-Css-Course on Udemy.

The Link to the course is: <https://www.udemy.com/course/tailwindcss-with-examples>

Now you can see the 5 breakpoints “**sm, md, lg, xl and 2xl**” during development.

When you go live later, you can simply remove the “debug-screens” class from the body tag.

Here is the link to the plugin's github:

page: <https://github.com/jorenvanhee/tailwindcss-debug-screens>

Format and preset headings

[Go to lecture >](#)

[View Commit of this lecture on GitHub >](#)

Of course, we don't want to have to specify the size of every single heading on a large website. That's why we can preset certain things in **tailwind.css**.

First, however, we start `npm run watch` in the terminal so that we don't have to run `npm run build` every time a change is made.

Then we insert into the tailwind.css:

```
@layer base {  
  h1 {  
    @apply text-5xl;  
  }  
  h2 {  
    @apply text-3xl;  
  }  
  h3 {  
    @apply text-2xl;  
  }  
}
```

The **@layer base** means that our definitions in the compiled stylesheets are inserted at the end of the base styles.

As you can see, you can access the utility classes with **@apply**.

Now we can **remove** all size classes from the headings in the *index.html*.

But, **IMPORTANT!**

The sizes can still be overwritten manually in the HTML!

So we can e.g. still overwrite the preset size of the H1 still with e.g. text-8xl. This corresponds to overwriting styles in HTML with the **style = "..."** attribute!

More information in the Tailwindcss documentation:

<https://tailwindcss.com/docs/adding-base-styles>

[GO UP](#)

Copyright: The creator and owner of this document is Martin Krebs Eberth, masterclassudemy@gmail.com

You can copy this document and spread this far and wide as long as you mention the link to the according course on Udemy.

Define your own heading font

[Go to lecture >](#)

[View Commit of this lecture on GitHub >](#)

To make a website more interesting, you can add your own fonts. We now want to download the [“Oswald” font from Google](#) for our headlines.

Step 1: Download the font and save it locally

You go to the **Download Family** button at the top right and you will get a zip or folder with different font weights.

From what we downloaded, we move the file *static/Oswald-Bold.ttf* to a new folder */dist/fonts/Oswald*, which we create.

Step 2: Import the font locally

In the *tailwind.css* within @layer base and above the headings we add:

```
@font-face {  
  font-family: Oswald;  
  src: url(/dist/fonts/Oswald/Oswald-Bold.ttf) format("truetype");  
}
```

What is important is the format specification **“truetype”**, and NOT **“ttf”**, as you might think if you don't know your way around and read the [official Tailwindcss documentation](#).. ;-)

If we now run `npm run watch`, the font-family is immediately compiled into *styles.css*, and if we search for “Oswald” we get:

```
@font-face {  
  font-family: Oswald;  
  
  src: url(/dist/fonts/Oswald/Oswald-Bold.ttf) format("truetype");  
}
```

Tip: If something doesn't work, you can click on the link in the URL while holding down the “CTRL” key to see if the path is correct!

Step 3: Extend the Tailwindcss theme

In the *tailwind.config.js* we add under **“theme”** and **“extend”**:

```
theme: {  
  extend: {  
    fontFamily: {  
      headline: ['Oswald']  
    }  
  },  
},
```

The name **“headline”** can be chosen freely here!

[GO UP](#)

This E-Book belongs to the Tailwind-Css-Course on Udemy.

The Link to the course is: <https://www.udemy.com/course/tailwindcss-with-examples>

As a result, the “font-headline” class can now be selected in the HTML! How cool is that?

Step 4: Assign the new class “font-headline” automatically

Now let's go one step further and automatically give the headings H1, H2 and H3 the new “font-headline” class. That is why we give the headings the new class in **tailwind.css** via **@apply**:

```
h1 {  
  @apply text-5xl font-headline;  
}  
h2 {  
  @apply text-3xl font-headline;  
}  
h3 {  
  @apply text-2xl font-headline;  
}
```

Now we no longer have to manually add the “font-headline” style to the headings!

In the compiled **/dist/styles.css** you can now see:

```
h1 {  
  font-family: Oswald;  
  font-size: 3rem;  
  line-height: 1;  
}
```

[GO UP](#)

Copyright: The creator and owner of this document is Martin Krebs Eberth, masterclassudemy@gmail.com

You can copy this document and spread this far and wide as long as you mention the link to the according course on Udemy.

Add your own color

[Go to lecture >](#)

[View Commit of this lecture on GitHub >](#)

As we saw at the beginning in the project description, the customer would like to add their own color to the website: **#212f49**.

Nothing easier than this:

Step 1: Add color to the tailwind theme

In the **tailwind.config.js** we add under “**theme**” and “**extend**”:

```
colors: {  
  mainColor: '#212f49'  
}
```

The name “mainColor” can be chosen freely.

Now we can e.g. say in the body tag of index.html:

```
class="bg-mainColor debug-screens"
```

Also if we look in the compiled **dist/styles.css**, we can find the “mainColor” in all possible variants, e.g. :

- border-mainColor
- bg-mainColor
- divider-mainColor
- etc.

Step 2: Assign the new main color to the headings

In the **tailwind.css** we can now assign the main color to the headings **by default**:

```
h1 {  
  @apply text-5xl font-headline text-mainColor;  
}  
h2 {  
  @apply text-3xl font-headline text-mainColor;  
}  
h3 {  
  @apply text-2xl font-headline text-mainColor;  
}
```

Now - as expected - our headings have the main color without having to assign them separately.

[GO UP](#)

Background: Free icons

[Go to lecture >](#)

Great selection in different styles:

<https://material.io/resources/icons>

Size and line width adjustable when downloading:

<https://feathericons.com/>

Very nice, too! Recommended by Tailwindcss:

<http://www.zondicons.com/icons.html>

Available in 2 different sizes = versions - very practical!

<https://heroicons.com>

Great Background SVG Patterns! Adjustable in color. Super!!!

<https://www.heropatterns.com/>

Introduction to responsive design with Tailwindcss

[Go to lecture >](#)

We have the following breakpoints in Tailwind:

sm	md	lg	xl	2xl
ab 640px	ab 768px	ab 1024px	ab 1280px	ab 1536px

The abbreviations mentioned above can be prefixed to all utility classes with colons,

e.g. **lg:bg-blue-500**

would mean that **from “lg” onwards** we have a background of the color bg-blue-500 for a viewport width of 1024px and more.

If we now define, for example, a background color of **bg-gray-100 lg:bg-blue-500**, this means that a slightly gray background applies to small screens up to a maximum width of 1023px, and then from lg = 1024px there will be a blue background.

[GO UP](#)

Background patterns

[Go to lecture >](#)

[View Commit of this lecture on GitHub >](#)

We will now use a background pattern for both the blue background and the light gray background in the container.

To do this, we go to the page: <https://www.heropatterns.com/>

There we enter the color value `#212f49` as the background color, which we defined as the “mainColor” (see *tailwind.config.js*)

Then we may set the foreground color and play with the slider for the opacity, and then click on the desired pattern.

In the window we get a code which we can apply directly to the body element. From there we only copy the area “background-image...” and then insert this style as “body” into **tailwind.css**:

```
body {  
  background-image: url("data:image/svg+xml,  
}
```

Attention! Don't forget the `npm run build` or `npm run watch`!

We now do the same with the **.container** class. (Select pattern + insert in *tailwind.css*)
I used “Jigsaw” for the background and “Topography” for the container.

Quote above with quotation marks and other color

[Go to lecture >](#)

[View Commit of this lecture on GitHub >](#)

For the second line of the heading we want to use a different color (indigo), we want to put the text in italics, set it slightly transparent and present it as a quote (**<q>**).

It is interesting that the HTML element **<q> Text </q>** displays the quotation marks differently depending on the country code in the HTML tag:

[GO UP](#)

This E-Book belongs to the Tailwind-Css-Course on Udemy.

The Link to the course is: <https://www.udemy.com/course/tailwindcss-with-examples>

en Martin Luther King said: "I have a dream!"

de Martin Luther King sagte: „Ich habe einen Traum!“

de-CH Martin Luther King sagte: «Ich habe einen Traum!»

pl Martin Luther King powiedział: „Mam marzenie!”

ru Мартин Лютер Кинг сказал: «У меня есть мечта!»

Overall, our new block for the second part of the heading now looks like this:

```
<span class="text-indigo-900 italic text-4xl sm:text-5xl lg:text-7xl opacity-80">  
  <q>&nbsp;I'll get you in shape!&nbsp;</q>  
</span>
```

As you can see, we have adjusted the text sizes of the second heading line depending on the resolution.

Create “Dark Mode”

[Go to lecture >](#)

[View Commit of this lecture on GitHub >](#)

The so-called Dark Mode is there to avoid dazzling with dark background lighting, e.g. when surfing the Internet at night with the lights switched off.

Step 1: Activation of dark mode

For our example we choose the so-called “**Manual Mode**” which we activate as follows in **tailwind.config.js**:

```
module.exports = {  
  darkMode: 'class', // or 'media' or 'class'  
  // ...  
}
```

Please be aware that the property ‘darkMode’ already exists and is **preset to false!**

Step 2: Activate dark mode in the document

For this we simply have to insert the class “dark” in the HTML tag:

```
<html class="dark" lang="en">
```

Step 3: Include alternative dark classes in the document

We don't see any difference yet, but if we add the **prefix “dark:”** to a class somewhere in the document, it will be activated as long as the class “dark” is in the HTML tag.

[GO UP](#)

Copyright: The creator and owner of this document is Martin Krebs Eberth, masterclassudemy@gmail.com

You can copy this document and spread this far and wide as long as you mention the link to the according course on Udemy.

This E-Book belongs to the Tailwind-Css-Course on Udemey.

The Link to the course is: <https://www.udemy.com/course/tailwindcss-with-examples>

Example:

In **tailwind.css** we have assigned the color class “text-mainColor” for the headings H1, which colors the H1 in a very dark blue.

We can now overwrite this **directly in the document**, e.g. by using the class “**dark: text-yellow-200**” for the first H1.

Now the heading is in a light yellow.

We can also do this centrally by adding in tailwind.css directly at H1, H2 and H3:

dark: text-yellow-200 Now the class is applied to all headings.

You can also **combine** “dark” and “hover” with each other. Example:

dark: hover: bg-gray-500

Here is the official link for dark mode: <https://tailwindcss.com/docs/dark-mode>

Two more brief comments on dark mode:

1. The dark mode does not necessarily have to be “dark”, you could also optimize your site e.g. for red / green - blindness etc.
2. You can also **use** dark mode **only for parts** of your HTML document. Just use the class “dark” in the according section where you need it.

But now we want to ensure that you can **dynamically** switch between “dark” and “non-dark”, and we do that with **Javascript**.

You can use “pure” = “Vanilla” JavaScript, jQuery, or a JavaScript framework such as Vue.js.

But we want to use the relatively new “mini framework” [alpine.js](#).

Alpine.js is awesome!

A brief introduction follows.

[GO UP](#)

Copyright: The creator and owner of this document is Martin Krebs Eberth, masterclassudemy@gmail.com

You can copy this document and spread this far and wide as long as you mention the link to the according course on Udemey.

Brief introduction to Alpine.js + Darkmode button

[Go to lecture >](#)

[View Commit of this lecture on GitHub >](#)

Alpine.js can be found on GitHub: <https://github.com/alpinejs/alpine>

Step 1: Include Alpine in your HTML

```
<script src="https://cdn.jsdelivr.net/gh/alpinejs/alpine@v2.x.x/dist/alpine.min.js"
defer></script>
```

Step 2: Extension: Alpine.js Intellisense

Bitte installiere diese Extension in Deinem Visual Studio Code.

Step 3: Create variables with x-data

You can bind a variable to an HTML element.

This is done with the directive **x-data="{ foo: 'bar' }"**

We now create a variable for the entire HTML document, which says that dark mode is switched off by default by inserting in the HTML tag:

```
<html x-data="{ dark: false }" class="dark" lang="de">
```

This variable is now valid in the entire HTML document! **Please note** that a variable is only valid within the element in which it was declared!

Step 4: Evaluate variables with x-bind

Now we can - also in the HTML document - evaluate whether the variable “dark” is either true or false, and set the class “dark” accordingly:

```
<html x-data="{ dark: false }" x-bind:class=" dark ? 'dark' : " " lang="de">
```

The colon before “class” is the short form of x-bind:

```
<html x-data="{ dark: false }" :class=" dark ? 'dark' : " " lang="de">
```

Simple on / off switch with @click

For a button we now add the @click event:

```
<button @click="dark = !dark">
```

And now you can switch the dark mode on and off with the button.

[GO UP](#)

Dynamic display of the dark mode button

[Go to lecture >](#)

[View Commit of this lecture on GitHub >](#)

Dynamic content with **x-text**

Example:

```
<span x-text="dark ? 'Day Mode': 'Night Mode' "></span>
```

“Day Mode” is output here **if the variable dark = true**, **otherwise** the word “Night Mode” is output.

Dynamic content with **x-show**

Example:

```
<svg x-show="dark" ...
```

Here the SVG is only displayed **if the variable dark = true**.

Previously used directives in Alpine.js

- **x-data** > to define variables
- **x-bind**, or short form “.” > to make attributes dynamic, e.g. the classes
- **@click** > to change variables with a click
- **x-show** > to show or hide elements dynamically
- **x-text** > to change text dynamically

[GO UP](#)

Outsource button styles

[Go to lecture >](#)

[View Commit of this lecture on GitHub >](#)

In order to set up a cascading button system [similar to Bootstrap](#), the styles for buttons can be outsourced to **tailwind.css** according to the following system:

Basic button class

```
.btn {  
  @apply rounded shadow-sm transition hover:shadow-none px-3 py-2 dark:text-white  
}
```

Button Icon Class

```
.btn-icon {  
  @apply flex items-center  
}
```

Button color classes

```
.btn-gray {  
  @apply bg-gray-300 hover:bg-gray-400 dark:bg-gray-400 dark:hover:bg-gray-500  
}  
.btn-indigo {  
  @apply bg-indigo-300 hover:bg-indigo-400 dark:bg-indigo-400 dark:hover:bg-indigo-500  
}
```

With this system different buttons are possible. Examples:

- **btn** > makes a colorless button without an icon
- **btn btn-icon** > makes a colorless button with an icon
- **btn btn-gray** > makes a gray button without an icon
- **btn btn-icon btn-indigo** > makes a purple button with an icon.

[GO UP](#)

This E-Book belongs to the Tailwind-Css-Course on Udemey.

The Link to the course is: <https://www.udemy.com/course/tailwindcss-with-examples>

Shrink CSS with Purgecss

[Go to lecture >](#)

[View Commit of this lecture on GitHub >](#)

Official page:

<https://purgecss.com/>

Step 1: Installation

```
npm install @fullhuman/postcss-purgecss
```

Step 2: Activation and set folders + file types

In the end you get purgecss to work with the following lines in **tailwind.config.js**:

```
purge: {  
  enabled: true,  
  content: ['./dist/**/*.html'],  
},
```

If you replace “**enabled: true**” with “enabled: false”, then it no longer works. You should only set it to “true” if you want to publish the project.

You should also temporarily set “enabled” back to false if you want to make subsequent changes to the *tailwind.css* or the Tailwind configuration (new plugins or changes in the *tailwind.config.js*).

Background info:

<https://tailwindcss.com/docs/optimizing-for-production>

[GO UP](#)

Copyright: The creator and owner of this document is Martin Krebs Eberth, masterclassudemy@gmail.com

You can copy this document and spread this far and wide as long as you mention the link to the according course on Udemey.