(Slide 0)

Hi! I’m gonna tell about Tailwind CSS, how it differs from usual use of css and from other frameworks.

(Slide1.1)

So, what is Tailwind?

(Slide 1.2)

Tailwind is a utility-first CSS framework packed with classes that can be composed to build any design.

In other words this prepared classes just need to be applied to the elements in order to get the desired result.

Technically it is similar to using inline styleЫs, when you describe the design of an element inside its tag.

(Slide 1.3)

However Tailwind helps you to simplify the styling of content in your apps.

As an example, I’ll show how the same block of code can be styled using regular css. It takes about 40 lines of code.

(Slide 1.5)

And the same code using Tailwind. Just 9 lines.

(Slide 1.6)

Also, according to The State of CSS 2022 survey, most developers in the world using CSS frameworks are now interested in learning and applying Tailwind CSS.

(Slide 2.1)

There are several ways to implement Tailwind into a workflow.

(Slide 2.2)

By using CDN. Just add the script to the head of your project.

(Slide 2.3)

By using Tailwind CLI tool. You need to install Tailwind into dependences of your project. Configure tailwind.config.js file. Add the Tailwind directives to your CSS. Run the CLI tool to scan your template files for classes and build your CSS. Add compiled CSS file to the <head> of your project.

(Slide 2.4)

Another way is to connect Tailwind to js framework according to the documentation.

(Slide 3)

As I’ve already noted Tailwind is about service classes.

There are plenty of these classes, but of course you don't need to remember them all.

Firstly, because there is documentation.

Secondly, names of these classes are mostly intuitive and you can guess about the purpose of a particular class.

For example, if you take a look at the presented button code, you can see that **px-6** means the size of the paddings on the x axis, **h** is the height of the button, then text–transform **uppercase** property, **semibold** font, **borders** 4 units thick, **border color**, **roundness**, **background color**, button **text color**. As you can see, everything is simple and clear.

And since there are a lot of classes, of course not all of them get into production, all unused classes are removed from the build, so there remains a minimum set of only necessary classes.

(Slide 4)

Tailwind has a basic configuration file, and by default it includes a large collection of classes needed to style the application.

But you can also create and add your own classes, which can contain colors, fonts, sizes and other data.

(Slide 5)

Classes for different screen sizes are supported out of the box. With the help of breakpoint prefixes, you can change the appearance of components.

(Slide 6)

All pseudo-classes are also available. Hover, focus, active etc. Just add to the desired class a prefix with the desired state.

Different themes are also supported. There are different ways to switch themes, either manually or depending on the system settings.

(Slide 7)

IDEs can also work with Tailwind. For VS code there is a Tailwind CSS IntelliSense extension with intelligent autocomplete suggestions. If you hover the cursor over some utility class, you can immediately see what it is.

(Slide 9)

I would also like to focus on how tailwind differs from bootstrap.

Bootstrap is primarily a set of ready-made components, i.e. the component already consists of different parts with applied styles.

And Tailwind is a tool that allows you to create your own sets of components using utility classes.

Bootstrap also has the ability to use utility classes, but tailwind offers a more convenient way to use and configure them.

(Slide 10.1)

And I would also like to focus on the downsides of tailwind. And must admit its advantages are its disadvantages.

(Slide 10.2)

For example. Code written with Tailwind can get messy with a long list of classes.

This is an input with 17 CSS classes. Which is easier, to read through the classes horizontally one by one, or to scan them vertically?

(Slide 10.3)

Here is how the above will look in a CSS file. I think it looks much better.

(Slide 10.4)

You should always give names to the elements.

In a design system, it’s hard to discuss a specific component without agreeing on a name for it.

(Slide 10.4)

Names of some classes are confusing.

When I first started using Tailwind, I needed to add the class that is responsible for align-items: center property. My first thought was to use align-center, but it didn’t work. I looked up at the docs and I was confused.

The class items-center will add the CSS property align-items: center, where the class align-middle will be for vertical-align: middle. It needs a bit of muscle memory to memorize them.

Summing up, I would note that Tailwind can fit well to many developers.

However, it is not an ideal solution for every project.

It is worth to estimate its advantages for the codebase and the potential impact on the team before implementing it to the project.