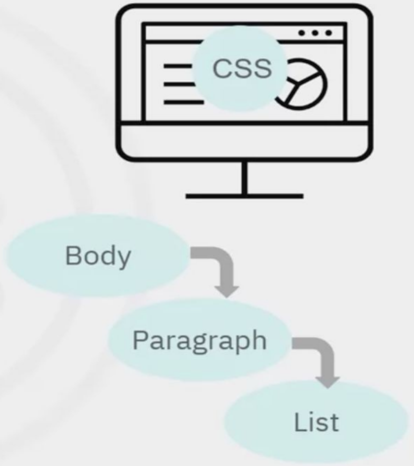
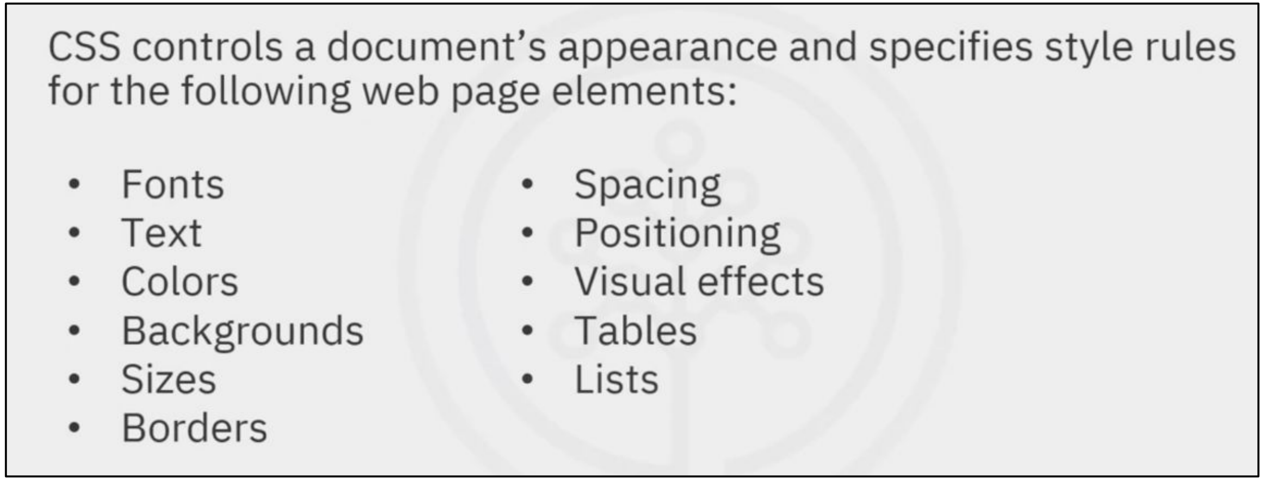
CSS
(Cascading Style Sheets)

What is CSS?

- Design that is layered over the top of an HTML Web page
- Describes how HTML elements are displayed
- Creates a uniform look throughout each element of each page of the website
- Child and descendant elements often inherit styles that are defined for parent elements



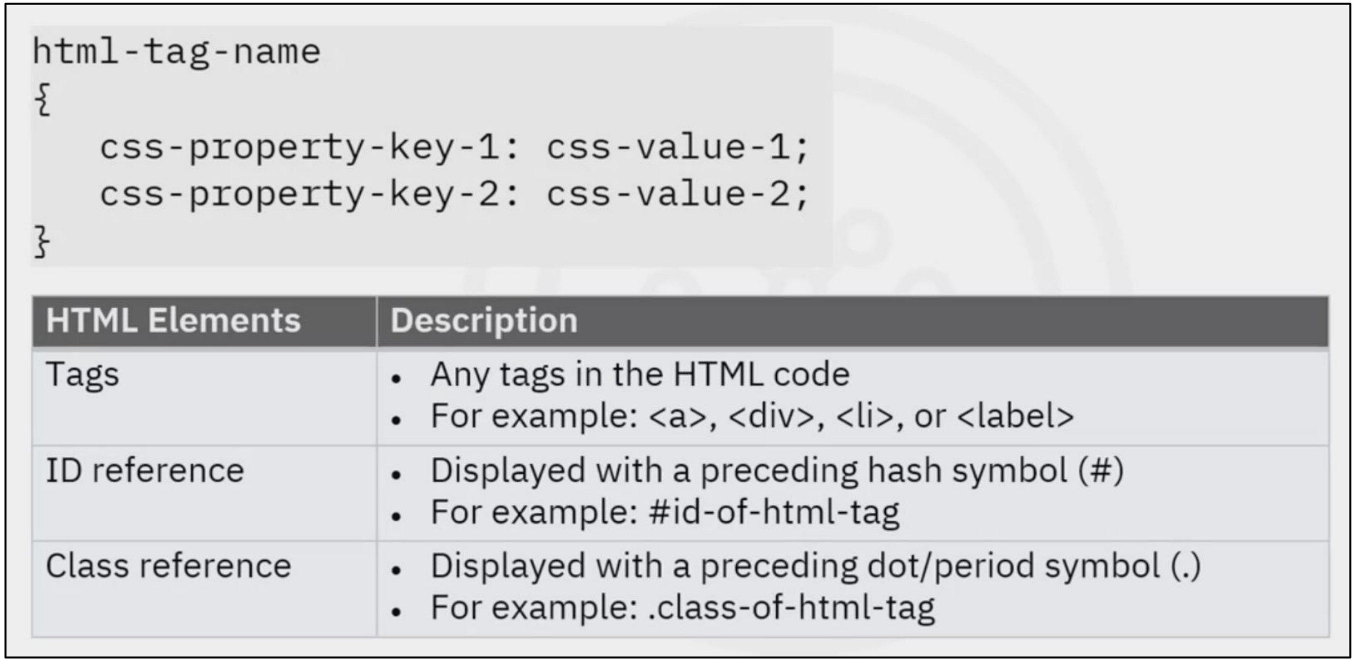
What Elements Can CSS Control?



CSS controls a document's appearance and specifies style rules for the following web page elements:

- Fonts
- Text
- Colors
- Backgrounds
- Sizes
- Borders
- Spacing
- Positioning
- Visual effects
- Tables
- Lists

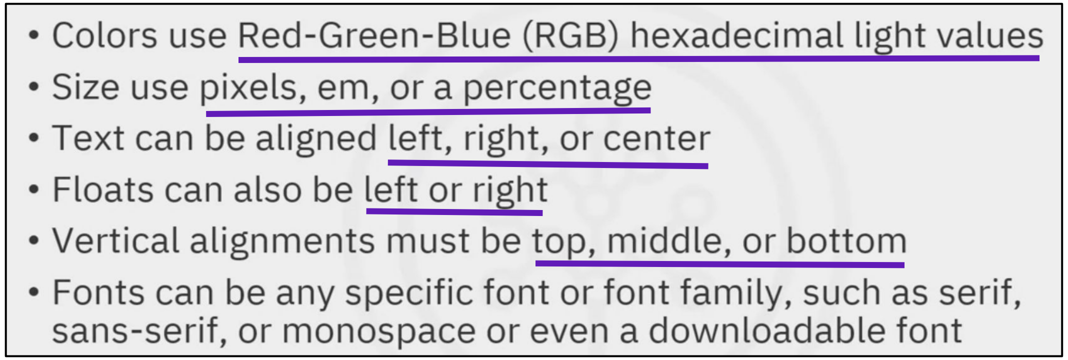
CSS Format



```
html-tag-name
{
   css-property-key-1: css-value-1;
   css-property-key-2: css-value-2;
}
```

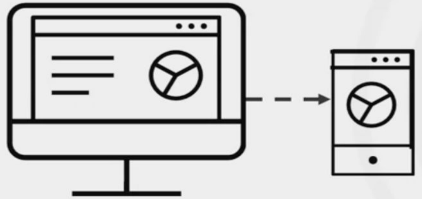
HTML Elements	Description
Tags	• Any tags in the HTML code • For example: <a>, <div>, <li>, or <label>
ID reference	• Displayed with a preceding hash symbol (#) • For example: #id-of-html-tag
Class reference	• Displayed with a preceding dot/period symbol (.) • For example: .class-of-html-tag

Guidelines



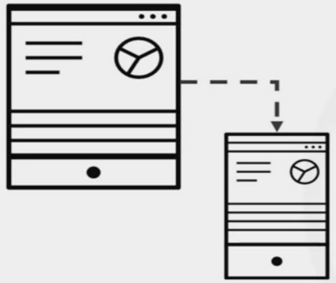
- Colors use Red-Green-Blue (RGB) hexadecimal light values
- Size use pixels, em, or a percentage
- Text can be aligned left, right, or center
- Floats can also be left or right
- Vertical alignments must be top, middle, or bottom
- Fonts can be any specific font or font family, such as serif, sans-serif, or monospace or even a downloadable font

Choosing A Layout



In a fluid layout:

- The height and width of elements are flexible
- The expansion or contraction is based on screen size
- The elements are specified using percentages and ems

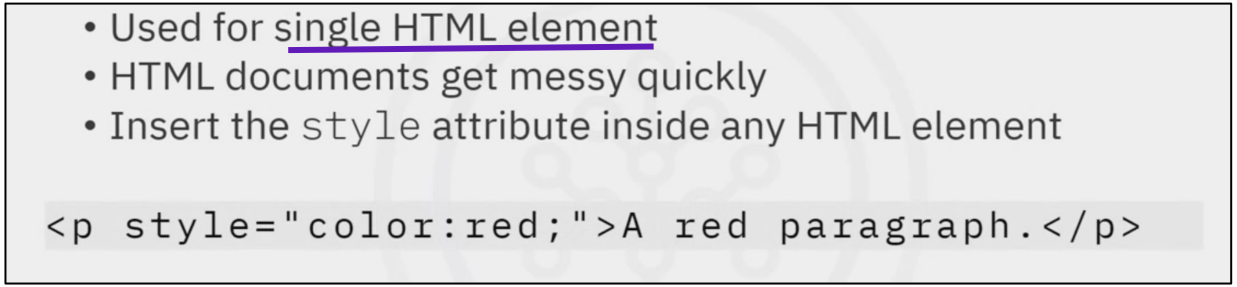


In a fixed layout:

- You specify the height and width of elements
- Values remain the same regardless of screen size
- You specify elements using pixels

Applying CSS to HTML

1. **Inline CSS**

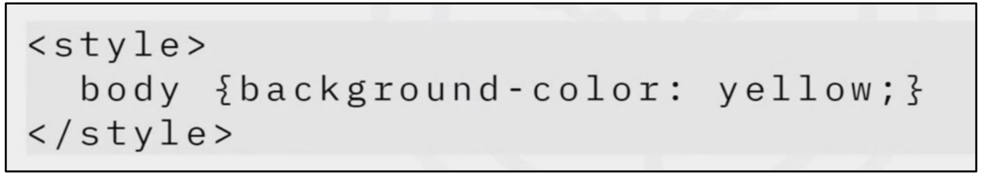


- Used for single HTML element
- HTML documents get messy quickly
- Insert the `style` attribute inside any HTML element

```
<p style="color:red;">A red paragraph.</p>
```

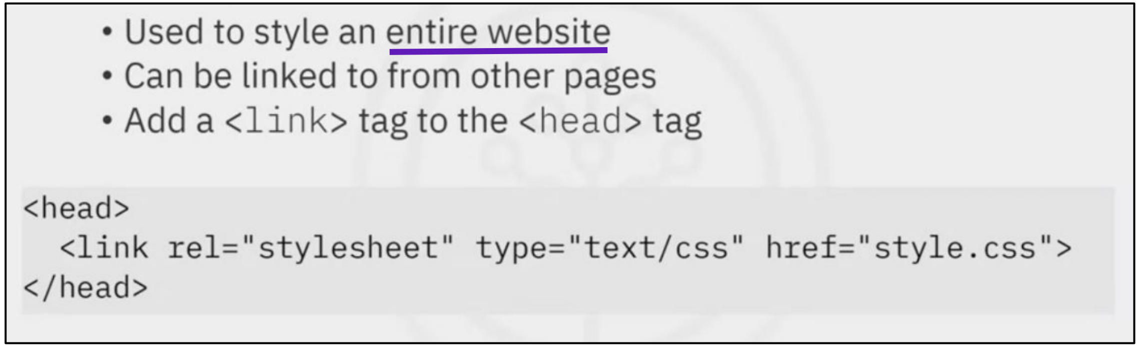
2. Internal CSS

- Used for **a single page**
  - However It “dirties” the page with a non-HTML code If you copy and paste this style on each page.
  - It will increase the load time of each page, which causes the user to wait longer
- To use this method, the <style> tag must be used, with your CSS code inside.



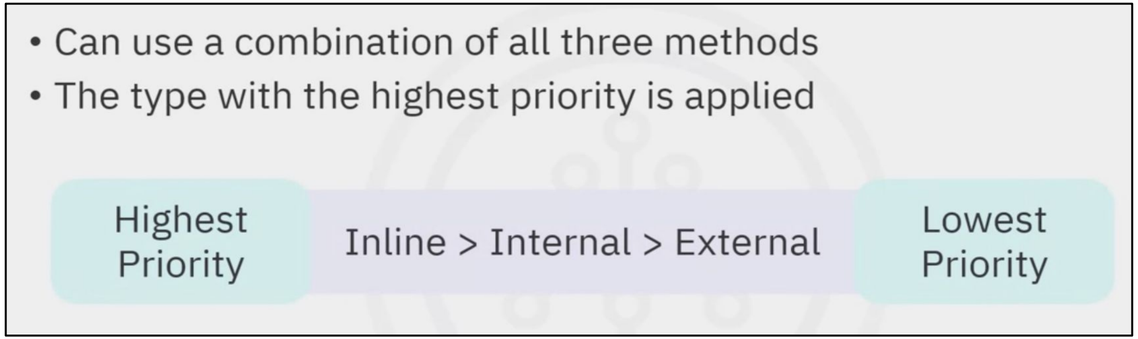
```
<style>
  body {background-color: yellow;}
</style>
```

3. External CSS



- Used to style an entire website
- Can be linked to from other pages
- Add a `<link>` tag to the `<head>` tag

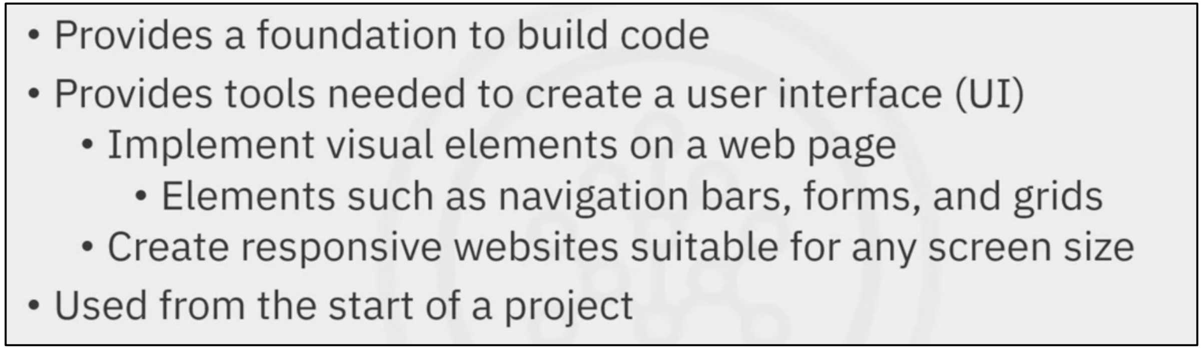
```
<head>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
```



- Can use a combination of all three methods
- The type with the highest priority is applied

Highest Priority | Inline > Internal > External | Lowest Priority

CSS Frameworks



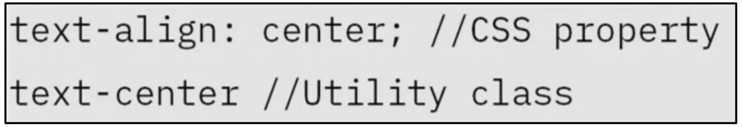
- Provides a foundation to build code
- Provides tools needed to create a user interface (UI)
  - Implement visual elements on a web page
    - Elements such as navigation bars, forms, and grids
  - Create responsive websites suitable for any screen size
- Used from the start of a project

Types of Frameworks

- Using no framework at all and just using plain CSS (also called **Vanilla CSS**) requires you to write all the styling on your own.
  - This gives you the freedom to style everything exactly as you want it, but also requires a lot of time and effort, as you must style every component.
- An alternative to this is to use a
  1. **Utility based frameworks**
     - Gives "utility" classes that scope to a single CSS property.
     - This makes it easier to apply CSS properties directly in your HTML code, which can save a lot of time while still giving you the freedom to style components as you wish.
  2. **Component based frameworks**
     - Provides pre-styled components and templates which are easy to add to any website.
     - This requires little knowledge of CSS and makes it easy to keep consistent styles, but also limits you to only the components made available by the framework.

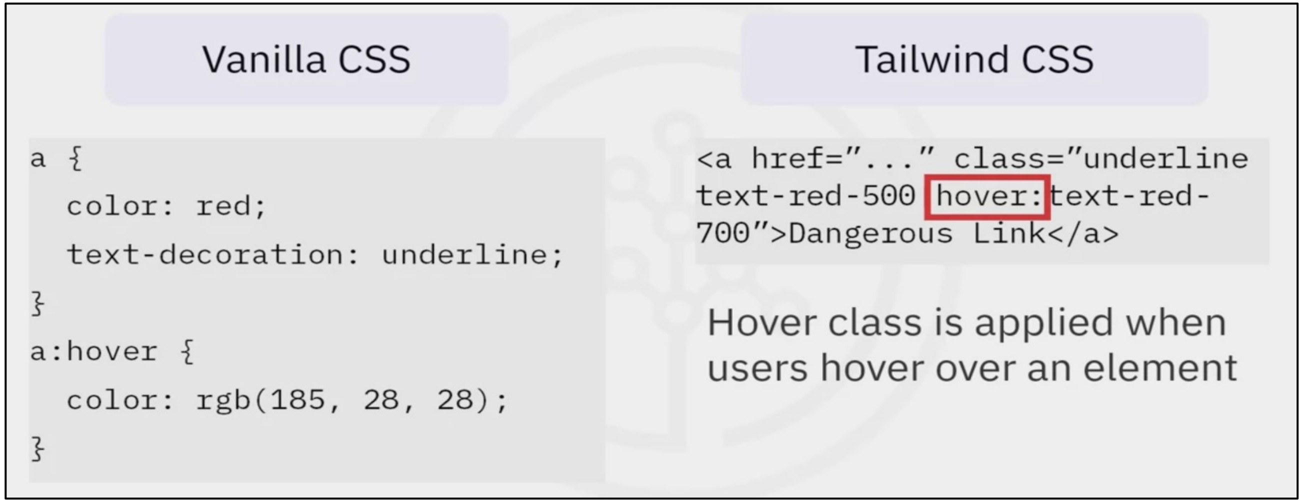
1. Utility First Frameworks (Advantages/Disadvantages)

- These typically come in the form of **classes** which scope to single-purpose CSS classes.
- Instead of having to write out the entire CSS property, it allows you to use a property by referencing its corresponding class within the “class” attribute of your desired HTML element.
- For example:
    - Instead of using the ***“text-align: center;”*** CSS property in your code, a utility-first framework might have a self-descriptive class, such as ***“text-center”***, which does the same thing when added to the ”class” attribute of an HTML element.
- Makes it easy to be consistent with color choices, spacing, typography, shadows, and everything else that makes up a well-engineered design system.
- Since it involves adding many classes to your HTML markup, this often causes the download size to increase, and potentially slows down your web pages.



```
text-align: center; //CSS property
text-center //Utility class
```

Example: TailWind CSS



Vanilla CSS

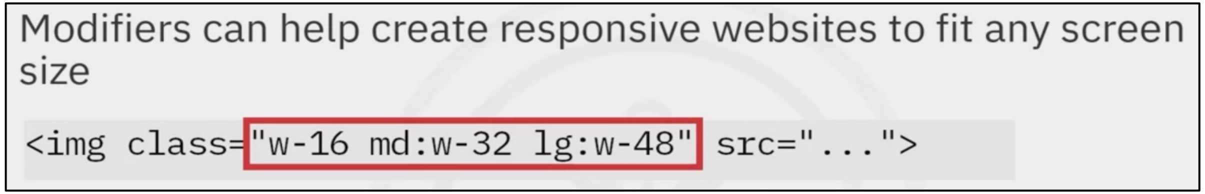
```
a {
  color: red;
  text-decoration: underline;
}
a:hover {
  color: rgb(185, 28, 28);
}
```

Tailwind CSS

```
<a href="..." class="underline
text-red-500 hover:text-red-
700">Dangerous Link</a>
```

Hover class is applied when users hover over an element

- Example:
  - Adding “md:” before a class will only apply the class when a user’s screen size is greater than 768 pixels wide, which is defined by Tailwind as a medium size screen.
  - This code will display an image with a width of 16 (64px) by default, a width of 32 (128px) on medium screens, and a width of 48 (192px) on large screens.



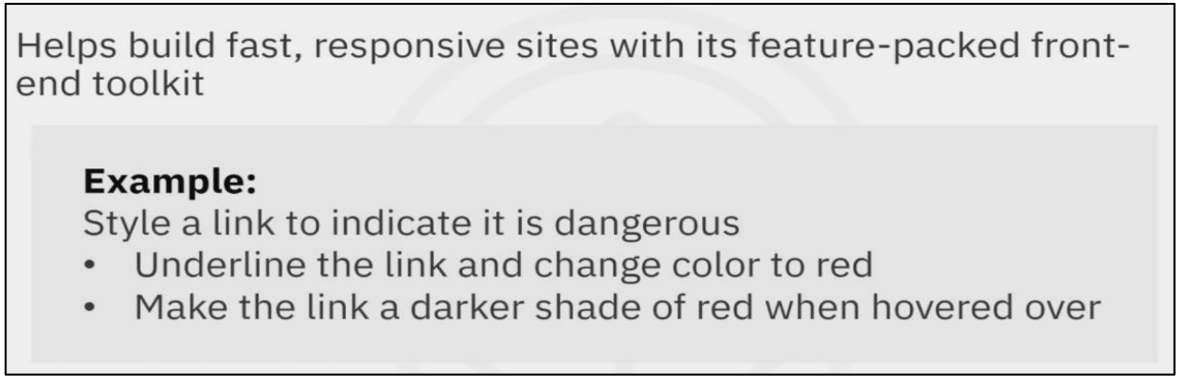
Modifiers can help create responsive websites to fit any screen size

```
<img class="w-16 md:w-32 lg:w-48" src="...">
```

2. Component frameworks (advantages/disadvantages)

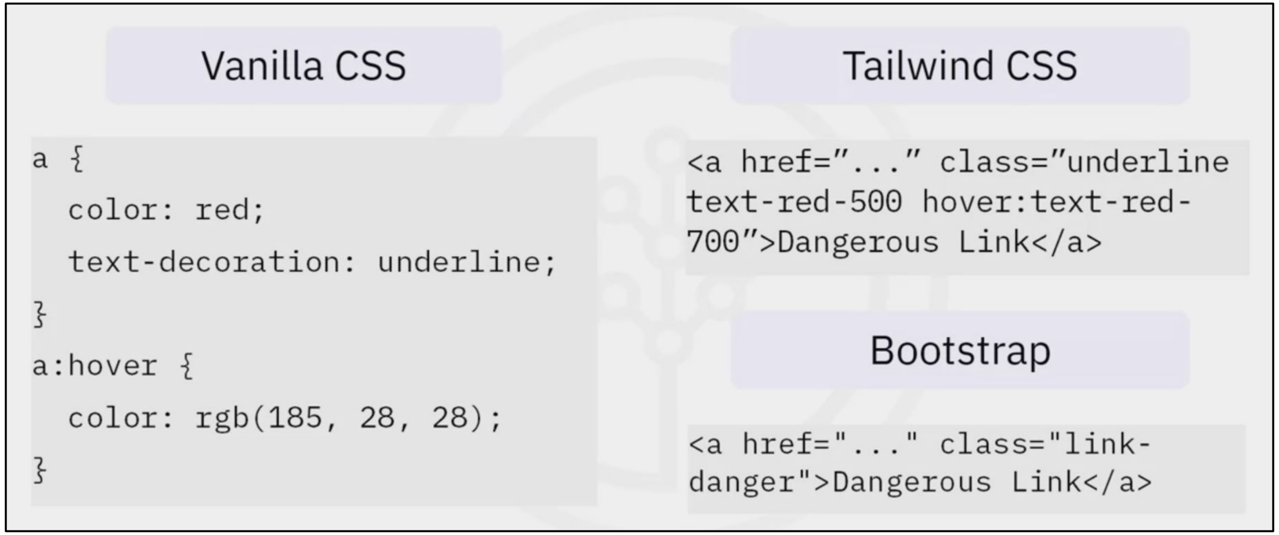
- Provides pre-styled components which can be easily added to your code.
- This results in the ability to develop well-styled websites rapidly, as significantly less time needs to be spent styling each element. It also makes it easier to keep all related elements styled uniformly, as you can simply choose the same or similar styles each time.
- However, having all these pre-defined styles limits you only to what the framework provides, and doesn’t give you the freedom of customizing everything exactly as you want it. They also provide a lot of overhead code that you wouldn’t otherwise get if you choose not to use any frameworks, as component frameworks will often provide you with more components than what you’ll use.

Example: Bootstrap


Helps build fast, responsive sites with its feature-packed front-end toolkit

Example:
Style a link to indicate it is dangerous

- Underline the link and change color to red
- Make the link a darker shade of red when hovered over



Vanilla CSS

```
a {
  color: red;
  text-decoration: underline;
}
a:hover {
  color: rgb(185, 28, 28);
}
```

Tailwind CSS

```
<a href=”...” class=”underline
text-red-500 hover:text-red-
700”>Dangerous Link</a>
```

Bootstrap

```
<a href="..." class="link-
danger">Dangerous Link</a>
```