CSS Font Syntax Fix

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Fonts, Text, Color Properties</title>

    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Agdasima:wght@400;700&family=Oswald:wght@200..700&display=swap"
        rel="stylesheet">
    <!-- either you can use link(the above three lines) or the below given import statement to include a font from google fonts -->




    <style>
        /* @import url('https://fonts.googleapis.com/css2?family=Agdasima:wght@400;700&family=Oswald:wght@200..700&display=swap'); */
        /* This link single handledly imports two fonts at once...thus only import the link once all the fonts are selected */

        @import url('https://fonts.googleapis.com/css2?family=Tagesschrift&display=swap');



        h1 {
            font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida Sans Unicode', Geneva, Verdana,
sans-serif;
      /*In font-family we have used a main font
      (the first one) followed by subsequent fonts. The
subsequent fonts are known as fall-back fonts. These fonts
will be applied in case the previous font is not available and
this will be followed in a sequence */
      /* try to give a give a generic font family so that the
probablity of the user having that font in their system is
high */

      letter-spacing: 38px;
      text-align: center;
    }

    #hello h1 {
      .tagesschrift-regular {
        font-family: "Tagesschrift", system-ui;
        font-weight: 400;
        font-style: normal;
      }

    }

    p {
      font-family: 'Courier New', Courier, monospace;
      font-style: italic;
      /*mainly used for italic only as bold or underlined
aren't available in this property*/

      font-weight: 1000;
      /*for making the font bold*/

      text-decoration: underline;
      /*mainly for underlining and overlining*/
      /* text-decoration: overline; */
      /*only one property can be used at a time and recent
one always overrides the previous one*/

      font-size: 16px;
      /* if in case you don't give any size then the styling
would be default which would be given by the browser(not
only for font-size but also for other font-properties) */
```

```
        /*PRO TIP:
        we can change/test the value of font-size on website by using style window in inspect element where after selecting the element I can increase or decrease the value of font-size by using arrow keys */
        /* in fact I can check/test how a font looks by temporary tweaking or adding the properties of the font and then copying it to the original source code for permanent changes */

        line-height: 1;
        /* this refers to the gap/spacing between the lines specially in a paragraph which has multiple lines..and this can also be tested using inspect element -> style window using up/down arrow keys*/
        /* The default line height in most browsers is "normal", which is usually between 110% and 120% of the font size. So we can use 1.1 and 1.2 or greater/lesser than those.  1.2=120% */

    }

    h2 {
        font-family: "Agdasima", sans-serif;
        font-weight: 400;
        font-style: normal;
        text-transform: capitalize;
        /*this capitalizes the first letter of each word---->we can also do uppercase, lowercase etc.*/
        text-decoration: underline;
        text-decoration-color: aqua;
        /*applies on the text-decoration not on text*/
        text-decoration-style: wavy;
        /*can make the line/decoration as dashed/dotted/wavy*/
        text-decoration-thickness: 3px;
        text-indent: 23px;
        /*adds indentation to the text from left...it can also be tweaked using arrow keys in the inspect->style window*/

        /*VS CODE PRO TIPS: Using alt+ down key can directly move the above the below without copying it
```

```
            And if you type first two letters of all the words for the property you wanna fetch then it will automatically suggest that property*/
        }

        #test {
            border: 2px solid red;
            width: 90px;
            /*since the width is low so the text will overflow from the border and for catering it we use the below mentioned text-overflow properties*/
            text-overflow: ellipsis;
            /*this can be set to clip too...ellipses hides the lines from the last but clip doesn't ...explore about them more*/
            overflow: hidden;
            /*we can also have scroll (as the value of this attribute) here*/
            word-break: break-all;
            /*it breaks the words to fit in the border otherwise it doesn't break the words*/

        }
    </style>
</head>

<body>
    <div>
        <h1>Fonts</h1>
        <p>This page is on Fonts (Some popular fonts that you can import like Baloo Bhai, Poppins etc.)</p>
        <p><a href="https://codepen.io/web-dot-dev/pen/yLojraG" target="_blank">Click to know more about some fonts</a>
        </p>
    </div>

    <div>
        <h2>Adding custom fonts through Google Fonts</h2>
        Go to Google Fonts <a href="https://fonts.google.com" target="_blank">(ClickMe)</a> -> Select a font you like
        (as many you want but try not to use much)->Perform
```

```
either of the 2 steps mentioned below:
        <ul>
            <li>Either include the fonts in html through link tag
and they also give the code to include it in a
                selector</li>
            <li>Or import the font through style tag</li>
        </ul>
    </div>

    <div>
        <h2>dummy paragraph</h2>
        <p id="test">Lorem ipsum dolor sit amet consectetur,
adipisicing elit. Deleniti, vel soluta natus ipsa ab facere
            sequi aut sed porro laborum odio ipsum eaque! Hic,
minima? Eum amet sed ullam rem, nihil animi at
accusamus
            iste magnam quos hic numquam placeat a soluta
deserunt? Nihil, animi.</p>
    </div>


    <div id="hello">
        <h1>
            I am Vaibhav Mishra
        </h1>
    </div>

</body>

</html>
```

why does my id pf hello doesn't show that text formatting

The reason your `#hello h1` section doesn't apply the **Tagesschrift** font style is because of **incorrect CSS syntax** inside the rule. Specifically, you're using this:

```css
#hello h1 {
    .tagesschrift-regular {
        font-family: "Tagesschrift", system-ui;
        font-weight: 400;
        font-style: normal;
    }
}
```

This is **invalid** in plain CSS. You're writing it as if `.tagesschrift-regular` is a nested selector inside `#hello h1`, which is not valid unless you're using a CSS preprocessor like **SASS** or **LESS** — but in raw CSS, nesting selectors like that is **not supported**.

✅ Correct way:

To apply the "Tagesschrift" font to `#hello h1`, directly write the styles inside the selector like this:

```css
#hello h1 {
    font-family: "Tagesschrift", system-ui;
    font-weight: 400;
    font-style: normal;
}
```

Summary Fix:

Replace this part:

```css
#hello h1 {
    .tagesschrift-regular {
        font-family: "Tagesschrift", system-ui;
        font-weight: 400;
        font-style: normal;
    }
}
```

With this:

```css
```

```
#hello h1 {
    font-family: "Tagesschrift", system-ui;
    font-weight: 400;
    font-style: normal;
}
```

Would you like me to clean up and simplify the full HTML+CSS for better clarity?

/* The specificity of this selector is 1 + 10 = 11 */
h1.cgreen {
color: green;
}

is h1 .cgreen{} would act the same?

Great question! The two selectors are **not the same** and behave **very differently**.

✅ `h1.cgreen { ... }`

- **Meaning:** Selects `<h1>` **elements that also have the class** `cgreen`.
- Example HTML:

```html
<h1 class="cgreen">This will be green</h1>
```

❌ `h1 .cgreen { ... }`

- **Meaning:** Selects **any element with class** `cgreen` **that is a descendant of an** `<h1>`.
- So it matches this:

```html
<h1>
  <span class="cgreen">This will be green</span>
</h1>
```

- It does **not** style the `h1` itself, only its children.

Summary:

Selector	Targets	Example that works
`h1.cgreen`	`<h1>` elements **with class** `cgreen`	`<h1 class="cgreen">Text</h1>`
`h1 .cgreen`	**Any element inside an** `h1` with class `cgreen`	`<h1><span class="cgreen">Text</span></h1>`

Want a quick visual example in a CodePen or want help remembering selector specificity too?

and same goes for id slector too?

Yes, exactly — the **same logic applies to ID selectors** too.

🔹 `h1#hello { ... }`

- **Meaning:** Targets an `<h1>` element **with the ID** `hello`.
- ✅ Matches:

```html
<h1 id="hello">Hi!</h1>
```

🔸 `h1 #hello { ... }`

- **Meaning:** Targets **any element with ID** `hello` that is a **descendant of an** `<h1>`.
- ✅ Matches:

```html
<h1>
  <span id="hello">Hi!</span>
</h1>
```

- ❌ Does **not** apply to:

```html
<h1 id="hello">Hi!</h1>
```

Key Rule:

Selector	Matches
`h1#hello`	`<h1>` **with id="hello"**
`h1 #hello`	Any element **inside `<h1>` with id="hello"`

Would you like a cheat sheet on different types of selectors and their specificity?

thenw hy do we use > and : symbols?

Great follow-up! The `>` and `:` symbols in CSS serve **very specific purposes**, and they are different from class `.` and ID `#` selectors.

Let's break it down:

---

1. 🔹 `>` — Child combinator

- **Usage**: Selects **direct children only**, not grandchildren or deeper descendants.
- Example:

```css
```

```css
div > p {
  color: red;
}
```

```html
<div>
  <p>This is red</p>        <!-- ✅ direct child -->
  <section>
    <p>Not red</p>          <!-- ❌ not direct child -->
  </section>
</div>
```

---

2. 🔸 `:` — Pseudo-classes / pseudo-elements

- **Usage**: Targets elements in a **specific state** or **part of an element**.

🔹 **Pseudo-classes (`:`)**

- Represent **states** of elements.
- Examples:
  - `a:hover` → when mouse is over a link
  - `input:focus` → when an input is focused
  - `li:first-child` → first list item

🔸 **Pseudo-elements (`::`)**

- Represent **parts** of elements.
- Examples:
  - `p::first-line` → styles only the first line of a paragraph
  - `div::before` → insert content before the div
  - `span::after` → insert content after the span

Summary Table:

Symbol	Meaning	Example	Matches
`.`	Class selector	`.btn`	Elements with `class="btn"`
`#`	ID selector	`#main`	Element with `id="main"`
`>`	Direct child combinator	`ul > li`	`li` that's a direct child of `ul`
`:`	Pseudo-class	`a:hover`	`a` when hovered
`::`	Pseudo-element	`p::first-letter`	First letter of a paragraph

Would you like a visual cheat sheet or code example for any of these?

yeah

Absolutely! Let's delve into the roles of the `>` and `:` symbols in CSS, providing clear examples to illustrate their usage.

🔹 `>` — Child Combinator

The `>` symbol is known as the **child combinator**. It selects elements that are **direct children** of a specified parent element.

Example:

```css
div > p {
  color: blue;
}
```

This CSS rule targets all `<p>` elements that are **immediate children** of a `<div>` element.

HTML Illustration:

```html
```

```
<div>
  <p>This paragraph is a direct child and will be blue.</p>
  <section>
    <p>This paragraph is a nested child and will not be blue.</p>
  </section>
</div>
```

In this example:

- The first `<p>` is a direct child of `<div>` and will be styled with blue text.
- The second `<p>` is nested within a `<section>` inside the `<div>`, so it is **not** a direct child and will not be styled.

◆ `:` — Pseudo-Classes

The `:` symbol introduces **pseudo-classes**, which allow you to define the special state of an element.

Common Pseudo-Classes:

- `:hover` — Applies when the user hovers over an element.
- `:focus` — Applies when an element gains focus.
- `:first-child` — Applies to the first child of a parent.
- `:nth-child(n)` — Applies to the nth child of a parent.

Example:

```css
a:hover {
  color: red;
}
```

This rule changes the text color of a link to red when a user hovers over it.

🔹 `::` — Pseudo-Elements

The `::` syntax denotes **pseudo-elements**, which allow you to style specific parts of an element.

Common Pseudo-Elements:

- `::before` — Inserts content before an element's content.
- `::after` — Inserts content after an element's content.
- `::first-letter` — Styles the first letter of a block of text.
- `::first-line` — Styles the first line of a block of text.

Example:

```css
p::first-letter {
  font-size: 200%;
  color: green;
}
```

This rule enlarges and colors the first letter of each `<p>` element.

🧠 Summary Table

Symbol	Name	Usage Example	Description
`>`	Child Combinator	`div > p`	Selects direct `<p>` children of `<div>`
`:`	Pseudo-Class	`a:hover`	Applies style when `<a>` is hovered
`::`	Pseudo-Element	`p::first-letter`	Styles the first letter of `<p>` elements

---

If you'd like to see these examples in action or need further clarification, feel free to ask!

Sources