https://jsmastery.pro

Key Techniques for Debugging CSS

JavaScript Mastery

Debugging your CSS

Debugging your CSS might feel like searching for a needle in a haystack, but with the **right toolkit and a sprinkle of creativity,** those alignment problems or layout shift issues can be resolved quickly and easily!

1 - Use Browser Developer Tools

Modern browsers come equipped with developer tools that are essential for debugging CSS:

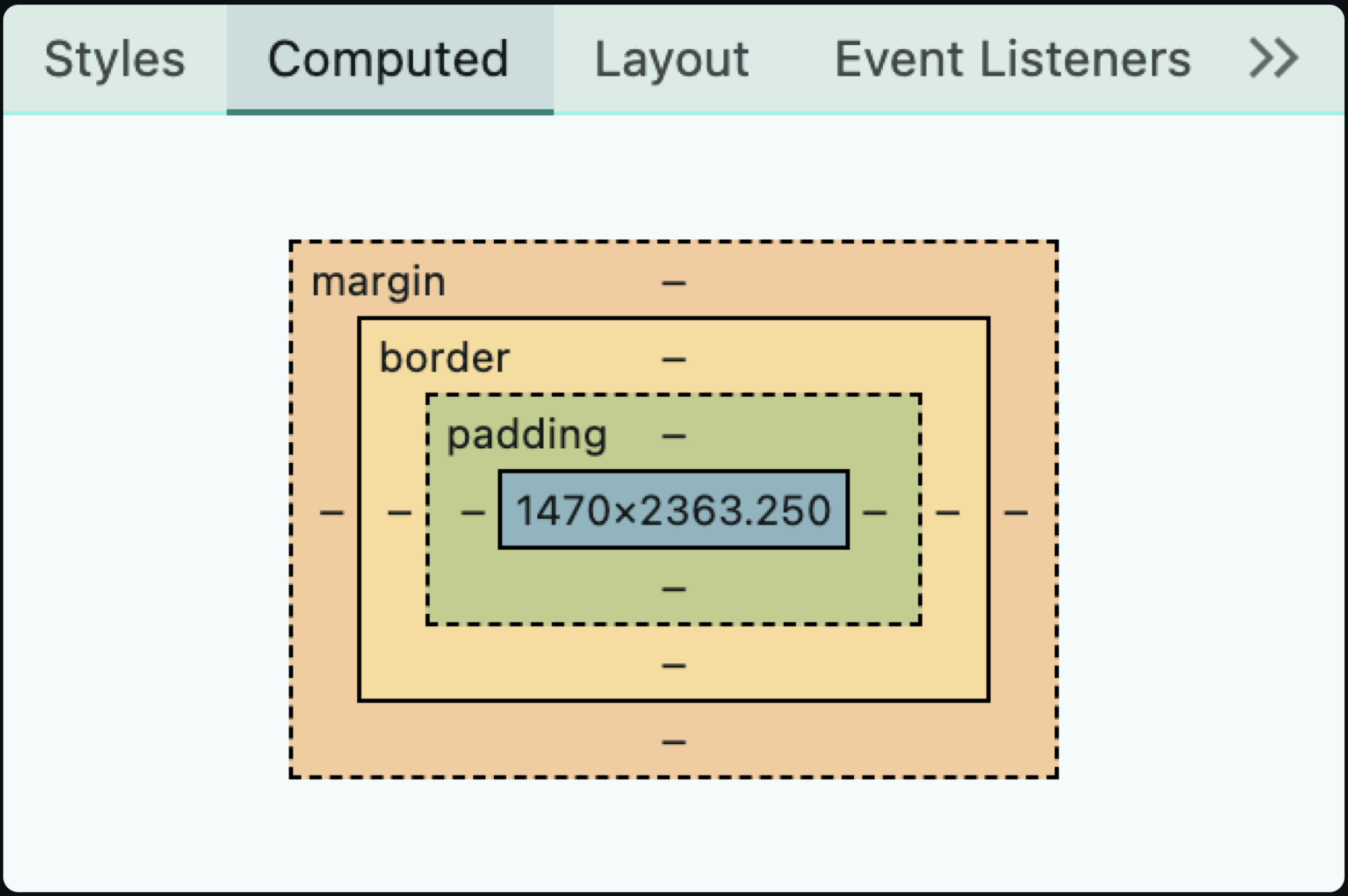
- **Inspect Elements:** Right-click on an element and select "Inspect". This opens the **DevTools** where you can view the CSS styles.

- **Styles Tab:** In the Elements panel of the DevTools, the *Styles* tab shows all CSS rules applied to the selected element.

You can **toggle properties on and off** by unchecking them, which helps in identifying which styles are being overridden.

1 - Use Browser Developer Tools

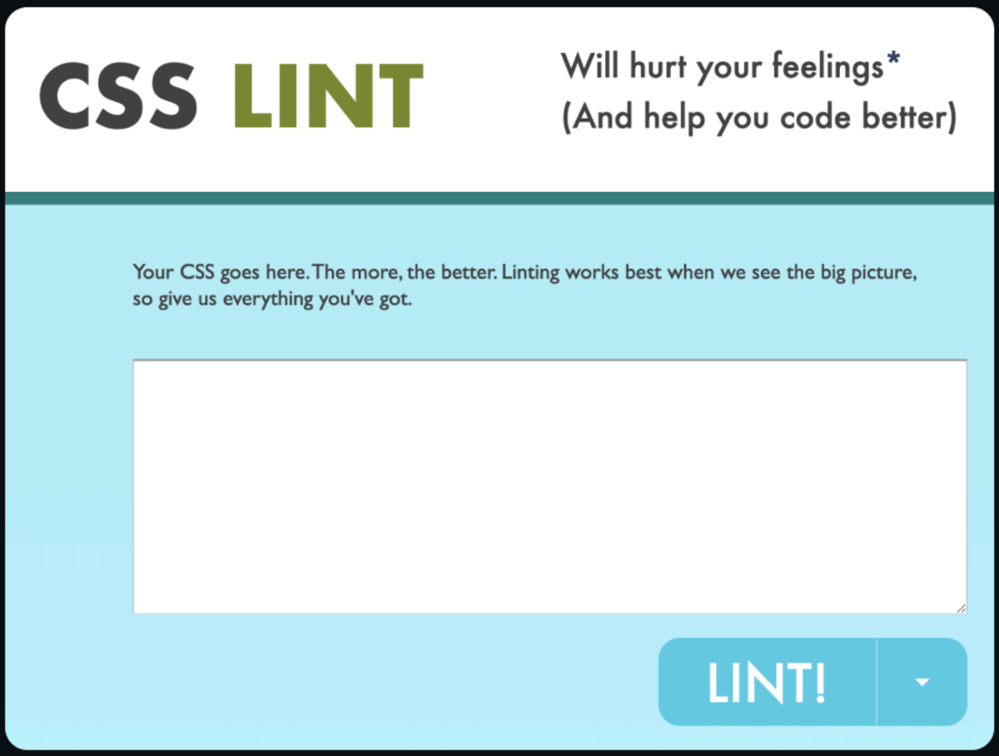
- **Computed Tab:** This tab displays the final computed styles for an element, allowing you to see how various styles are combined and which ones take precedence due to specificity.



2 - Check for Syntax Errors

Always check your CSS for **typos** or **syntax errors.** A missing semicolon or an unclosed bracket can prevent styles from being applied correctly.

Most modern code editors will conveniently highlight these errors for you, but you can also use tools like **CSSLint** or **Stylelint** (a Visual Studio Code extension).



CSS LINT

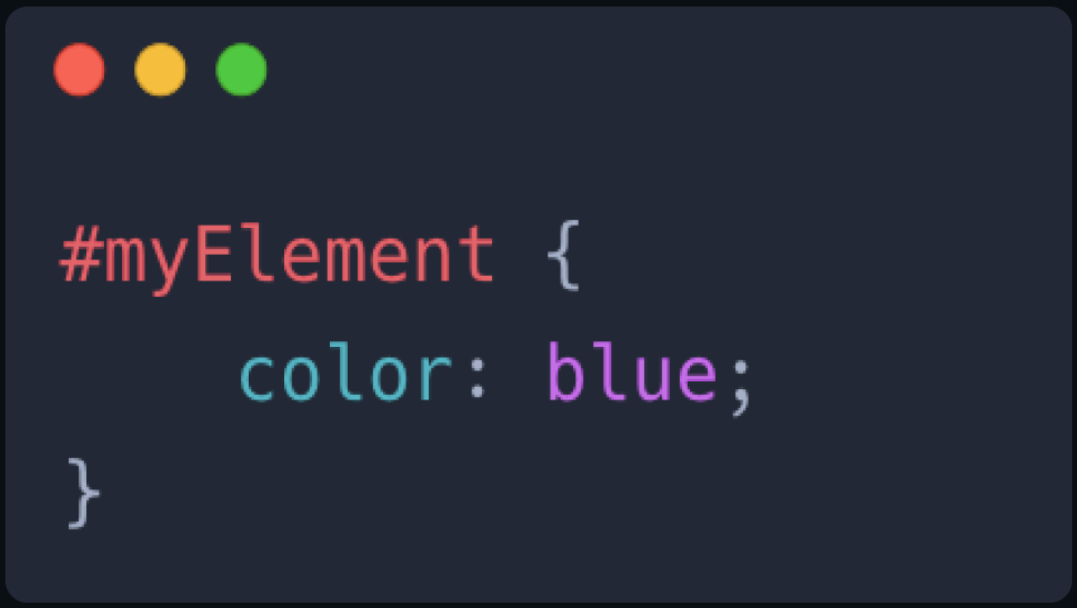
Will hurt your feelings*
(And help you code better)

Your CSS goes here. The more, the better. Linting works best when we see the big picture, so give us everything you've got.

3- Cascading and Specificity

CSS follows a **cascading order** where more specific rules override less specific ones.

Familiarize yourself with how specificity works to troubleshoot why certain styles are not applying as expected. For example, if a **class selector** is being overridden by an **ID selector,** you will need to adjust your selectors accordingly.



```css
#myElement {
    color: blue;
}
```

4 - Isolate the Problem

If you're facing a complex issue, simplify your CSS by **commenting out sections of code or isolating specific elements**.

This method allows you to pinpoint **exactly where the problem lies** without the distraction of other styles.

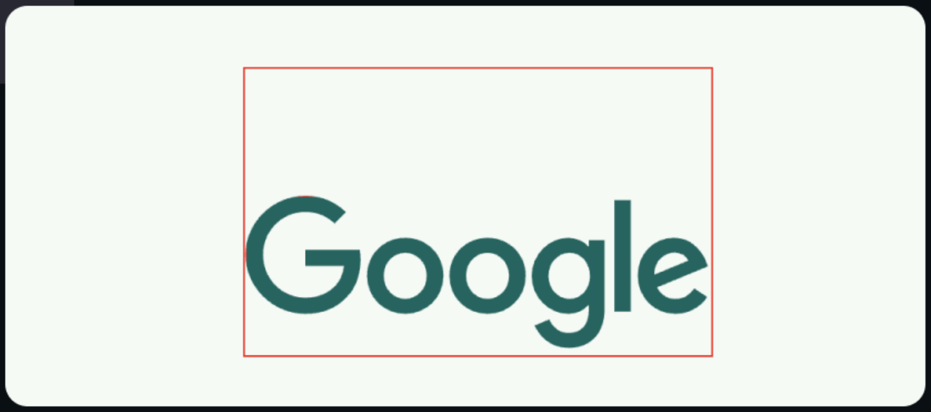
5 - Validate Your CSS

Use validation tools like the **W3C CSS Validation Service** to check your CSS against standards. This will help identify **unsupported properties** or errors that might be causing issues.

6 - Visual Debugging Tricks

- **Outline Trick:** Apply an outline to all elements, like this:

```css
* {
    outline: 1px solid tomato;
}
```



- **Grid Overlay for CSS Grid:** If you're using CSS Grid, enable grid overlays in Chrome DevTools to see how elements are laid out within the grid structure.

7 - Cross-Browser Compatibility

Test your website across **different browsers (Chrome, Firefox, Safari)** as styles may render differently due to browser-specific quirks.

Ensure that you address any **cross-browser compatibility** issues.

Master Next.js!

Take that step to become the developer you know you can be.

FOR DEVELOPERS WHO WANT TO STAND OUT AND BECOME IRREPLACEABLE

Become a top 1% Next.js 14 developer in only one course

jsmastery.pro/next14