THE TECH ACADEMY

COMMON MISTAKES TO AVOID WITH CSS

1. Forgetting to recognize that not all browsers were created equal.

A common blunder beginning developers make is expecting their code to be rendered in same fashion across all browsers. Unfortunately, there is no perfect world and despite The World Wide Web Consortium's efforts to standardize the markup language, competing browser technologies are not at all identical. What may look perfect in a Firefox browser may not necessarily appear as intended by an Internet Explorer Browser. Your CSS should make use of multiple viewport controls and media breaks.

2. Seldom (or never) commenting CSS.

/* Just as it is true with most programming languages, you should always comment your CSS. */

Neglecting to comment frequently will produce counter-efficiency later on when you or someone else needs to revisit the CSS.

3. Not planning a project's design to formulate consistent concepts of standardization.

When done properly and cleverly, a generic class framework that manages styling aspects which are anticipated to be repeated over and over again from one page to the next will save a considerable amount of time in the long run. Implementing standardizations early in a project so styles can be efficiently manipulated in CSS is the key.

4. Publishing markup with syntax mistakes.

Though it may seem a minor and easily overlooked mistake, a browser, which is interpreting the instructions, may not easily discern the intended purpose and often times it would only take one forgotten and unclosed tag or improperly applied quotation mark to break the code.

5. Convoluting or overruling content styling.

Invoking too much in-line CSS styles makes your code errant and difficult to manage. You will often times find yourself on wild-goose chases, searching endlessly for conflicting style references. It is best practice to learn and master how the specificity hierarchy of styling gets handled.

6. Using redundant code.

Apply the DRY principle - Don't Repeat Yourself. Whenever you discover that your CSS is repeating itself, you should refactor it. Take the following code, for example:

```
p.ex1 {
font: italic bold 12px/30px Georgia, serif;
color: #0000ff;
}

p.ex2 {
font: italic bold 12px/30px Georgia, serif;
color: #0000ff;
}
```

Appropriately applying the DIY principle, it would be more efficient to combine redundant code that is producing similar effects. You will accomplish the very same outcome, save space, and make the code easier to manage. The DIY method might appear like:

```
p.ex1, p.ex2 {
font: italic bold 12px/30px Georgia, serif;
color: #0000ff;
}
```

7. Trying to use HTML to perform more than just the structure and content.

It is best to avoid using deprecated styling features such as <b>, <i>, , <u>, , <font>, or <align> - all examples of HTML tags that manipulate the style of document text, rather than the structure and content of the document. It is more efficient to address these effects with the CSS.

8. Forgetting that many properties are inherited from their parent element.

Beginning developers tend to waste enormous amounts of their time debugging code due to this very fact. They will often assign properties to parent elements and then later nest a child element into a particular parent element and later fail to immediately discover why the child element is behaving in a particular fashion.

9. Being too specific with class names or using IDs too liberally.

CSS was developed to help make your design a lot easier to manage. By using less specific and generalized class styling, it will be much easier to manage similar elements by using generic class styling. That is not to discredit specific selectors in any way, they have their merits

but it is best to use them sparingly when it is possible. Each coder has their own methods but for the aspiring web designer, developing good habits early on is paramount.

10. Trying to reinvent the wheel.

New developers sometimes focus too much of their valuable time reinventing their CSS when more often than not, there are already notable CSS frameworks available to swiftly accomplish what they were striving for. If your purpose is to learn CSS intimately and you have plenty of time to spare, than it is an excellent opportunity to write CSS from scratch. But, if time is of the essence, using an established CSS framework can save you from a headache and considerable time.

11. Playing the blame game...and losing!

Many times the novice developer may assume that the fault lies with errant CSS coding when it could just as easily have been HTML's fault. This is why it is especially important to always validate your HTML file first before focusing too much on your CSS. (Learn more about validating using the W3C's Markup Validation Service).

12. Failing to see things from the perspective of your audience.

Most likely, not every visiting patron will be viewing your page using the very same kind of device, with the same display, or even with the same browser technology - and being able to account for this aspect alone will separate the seasoned developer from the ranks of novices. Many designers employ bug testing via a suite of viewports, device emulators, browser brands and versions, and other techniques. Developers will further compose their CSS with media breakpoints to provide a contingent of solutions.

END OF ARTICLE