CSS PROGRAMMING COOKBOOK

HOT RECIPES FOR CSS DEVELOPMENT

FABIO CIMO

WEB CODE GEEKS
WEB DEVELOPERS RESOURCE CENTER

CSS Programming Cookbook

Contents

Preface

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. Although most often used to change the style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any kind of XML document, including plain XML, SVG and XUL. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications.

CSS is designed primarily to enable the separation of document content from document presentation, including elements such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content, such as semantically insignificant tables that were widely used to format pages before consistent CSS rendering was available in all major browsers. CSS makes it possible to separate presentation instructions from the HTML content in a separate file or style section of the HTML file. (Source: https://en.wikipedia.org/wiki/Cascading_Style_Sheets)

In this ebook, we provide a compilation of CSS based examples that will help you kick-start your own web projects. We cover a wide range of topics, from text styling and table design, to class inheritance and hover effects. With our straightforward tutorials, you will be able to get your own projects up and running in minimum time.

About the Author

Fabio is a passionate student in web tehnologies including front-end (HTML/CSS) and web design. He likes exploring as much as possible about the world wide web and how it can be more productive for us all. Currently he studies Computer Engineering, at the same time he works as a freelancer on both web programming and graphic design.

Chapter 1

CSS Inheritance

In this example, we'll focus on a css property value that I guess many of you ignore (do not use) while styling elements on websites.

It is `inherit`. Before we go straight into that, know the following:

Each element in HTML is part of a *tree*, and every element (except the initial `html` element) has a parent element that encloses it.

Whatever styles applied to the parent element can be applied to the elements inside it if the properties are inherited.

Below, we're going to have a look at several cases of inheritance property.

1.1 Prerequisites

Create a `html` document with the basic syntax inside like this:

```
<!DOCTYPE html>
<html>
<head>
        <title>CSS Inheritance Property Value</title>
</head>
<body>

        <!-- STYLE SECTION -->
        <style type="text/css">

        </style>

        <!-- HTML SECTION -->

</body>
</html>
```

In the html section, we'll add some elements to show their relation to other elements and understand when they automatically inherit property values from their parents and when not.

Lets first see a basic application of the `inherit` property value.

1.2 Basic Application of Inherit

The inherit property value is at some cases automatically applied to child elements, and at other cases needs to be applied manually.

1.2.1 Automatic Inheritance

I always tend to give the `body` element a custom font-family so that all elements have the same font applied.

In other words, all elements `inherit` the font-family property from the body parent.

Look at the code below:

```
<!-- HTML SECTION -->

 This is a <span>parent</span> element.

<div class="child">This is a <span>child</span> element.

This is a <a href="#">link</a> inside a paragraph

<!-- end child element -->
</div><!-- end parent element -->
```

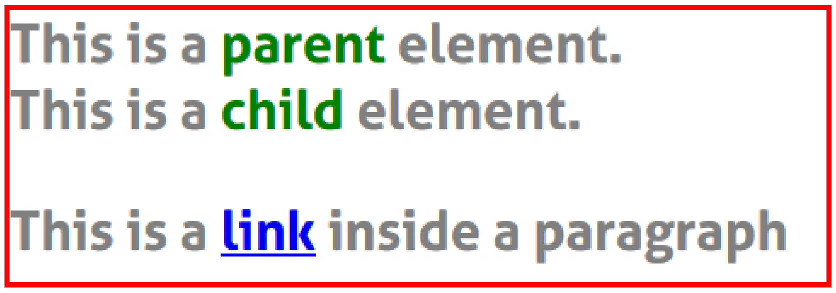
I have added a parent division and inside that two children, another div and a paragraph.

The classes given are so that we can give attributes to parent and see the results to children.

Now below look at the some initial properties I've given to these elements:

```
<!-- STYLE SECTION -->

<style type="text/css">

body {
        font-family: "Aller","sans-serif";
}

.parent {
        width: 15em;
        height: 5em;
        color: gray;
        font-weight: bold;
        font-size: 2em;
        border: 0.1em solid red;
}

.parent span {
        color: green;
}

</style>
```

Now to see what what properties have been inherited look at the browser view:



This is a parent element.
This is a child element.

This is a link inside a paragraph

Figure 1.1: Initial Test on Inheritance

Seen this result, we can state that the following elements get inherited properties automatically:

1. All elements `inherit` the `font-family` property from a higher level parent, which is body. Notice all three lines have the same font applied.
2. All child elements `inherit` the `color` (font-color) from the parent element. Notice all the lines have the same gray color applied (except span and link).
3. The parent `span` element is set to have a green color, but also the child span gets a green font-color. So the child inherits from parent automatically span elements too.

Think everything is inherited?

The `a` anchor element has not been styled, but it shows in blue and underlined. That's because these two properties are set to be default ones for all anchor tags over the page.

The link did not inherit the color from its parent element, because it has a different color (blue). Look at the border. It has only been applied to the parent element, and it shows only there. That's why we sometimes need to apply inheritance manually to make these element fit the desired view.

1.2.2 Forced/Applied Inheritance

Simple as that, refer to the `.child` class and inherit the `border` for the child and `color` property from parent for the link:

```
.parent a {                                     /* you can refer even to the child class */
        color: inherit;         /* inherited color from the parent class */
}

.child {
        border: inherit;        /* inherited the border attributes to child and elements  ↩
            inside */
}
```

The browser view:

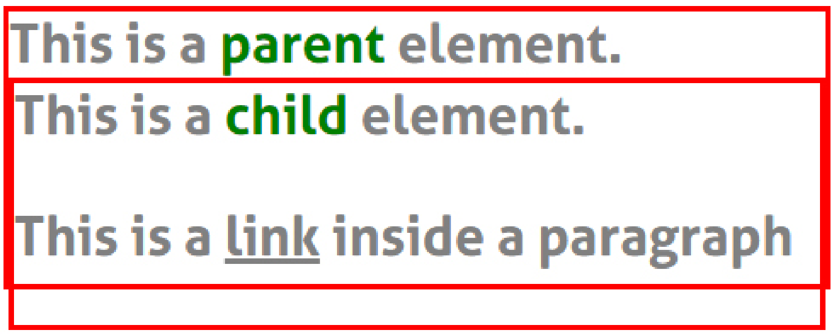


Figure 1.2: Inherit Value Applied to Color and Border

Notice that:

1. The link became the **same color as the parent** (its parent can be considered both the parent class element and the child class element, becuase the paragraph is nested inside the child element).
2. The border attributes got applied to the child element which contains another element inside, that's why you see the second smaller red border wrapping both the second and third line (child and paragraph elements).

That was a basic application of the inherit property value. Now let's see other cases and examples.

1.3 Cases and Examples

Here we look at some essential cases and examples where we can use the `inherit` property value.

1.3.1 Size and Background Inheritance

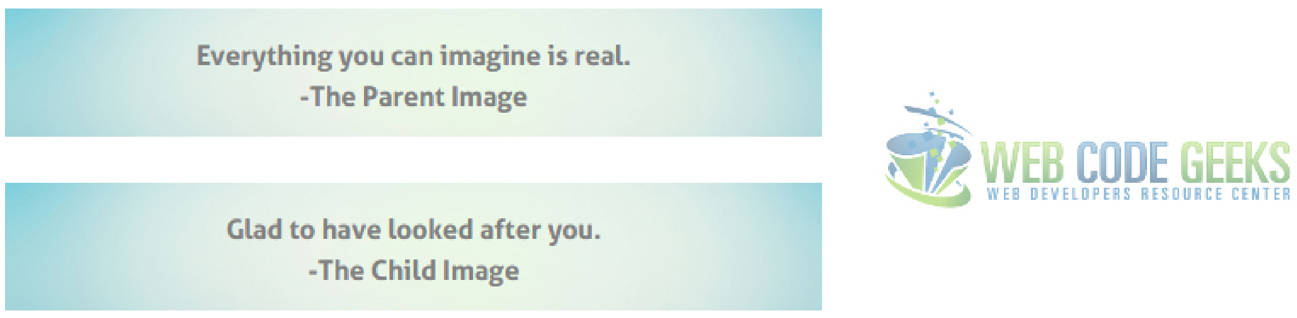
Change the html code to accommodate the following image and text elements:

```
<h4>Everything you can imagine is real.<br>-The Parent Image</h4>

<div class="child-image">
<h4>Glad to have looked after you.<br>-The Child Image</h4>

<!-- end child image -->
</div><!-- end parent image -->
```

and the css code accordingly:

```
.parent-image  {
        width: 30em;
        height: 30em;
        background-image: url("images/img1.jpg");
        background-repeat: no-repeat;
}

.parent-image h4 {
        color: gray;
        text-align: center;
        line-height: 1.5em;             /* vertically align text in the center */
        padding: 1em;                   /* just to differentiate the two elements */
}

.child-image {
        width: inherit;                                 /* inherited width */
        height: inherit;                                /* inherited height */
        background-image: inherit;              /* inherited background */
        background-repeat: inherit;
}
```

Now notice the child element inherits the background and size attributes from the parent:


Everything you can imagine is real.
-The Parent Image

Glad to have looked after you.
-The Child Image

Figure 1.3: Size and Background Values Inherited

1.3.2 Inheritance Manipulations

You can also use the inherit property value to avoid inheritance from one level higher parent.

Look at the codes below:

HTML

```
<!-- HTML SECTION -->



<div class="child1">
<h3>A title here</h3>
A paragraph text here
<button>A button here</button>
<!-- end child1 element -->


<h3>A title here</h3>
A paragraph text here
<button>A button here</button>
<!-- end child2 element -->

</div><!-- end parent element -->
```

CSS:

```
.parent h3 {
        font-variant: small-caps;
        color: red;
}
```

```
.parent p {
        text-indent: 1em;
        color: blue;
}

.parent button {
        padding: 1em 2em;
        background-color: gray;
        color: white;
        border-radius: 0.5em;
        border: 0.1em solid gray;
}

.child2 p {                                              /* declined inherited attributes ←
    from parent */
        text-indent: inherit;
        color: inherit;
}

.child2 h3 {                                     /* declined inherited attributes from ←
    parent */
        font-variant: inherit;
        color: inherit;
}

.child2 button {                                 /* declined inherited attributes from ←
    parent */
        background-color: inherit;
        color: inherit;
}
```

Lets look at the browser view and then comment it:

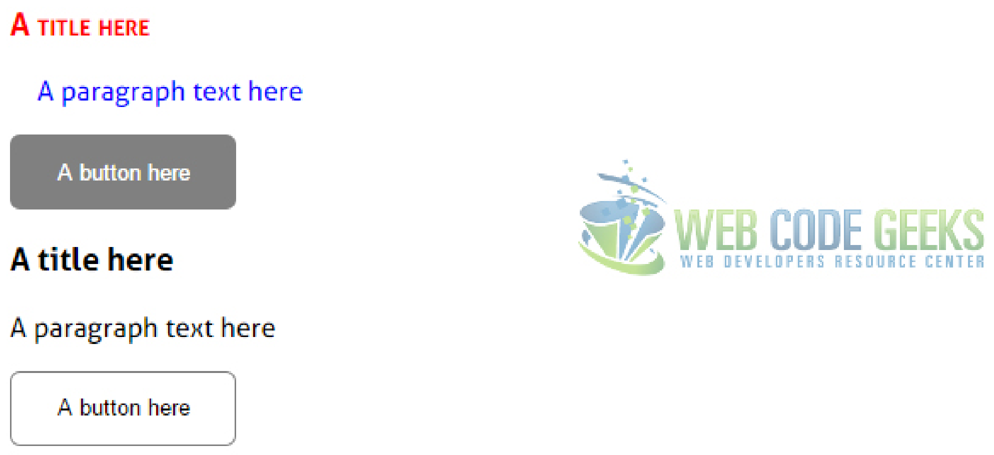


Figure 1.4: Vice-Versa Inheritance

Notice that the parent attributes are applied to the first child but not to the second one.

That is because we chose to inherit attributes when they were already automatically inherited.

This case would result in an inheritance of the default attributes of a higher level parent (i.e body).

1.3.3 Tag and Location Inheritance

- **Tag Inheritance**

To explain this, take for example the `button` tag of html:

```
<button>Button</button>
```

With no styling at all, this element has already a view (attributes):

Button
WEB CODE GEEKS
WEB DEVELOPERS RESOURCE CENTER

Figure 1.5: Tag Inheritance - Button Example

You can see it has a `border`, a gradient `background`, a `hover` state ect.

This is called tag inheritance, it means the element inherits default properties/attributes pre set by the browser.

Whenever you style a button, you just override the browser default properties for that element.

- **Location Inheritance**

Basically, location inheritance means using the same attributes for a set of elements under the same tag/class.

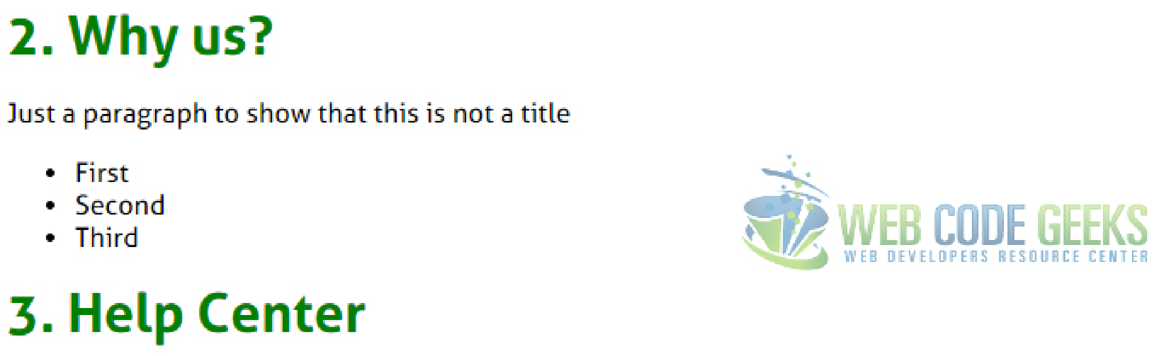
For example, if you want to have all titles styled with a green color and bold you could just refer to the following:

```
.title {
        color: green;
        font-weight: bold;
        font-size: 2em;
}
```

And apply this class to all elements wrapping a title like so:

```
2. Why us?

Just a paragraph to show that this is not a title.
<ul>
        <li>First</li>
        <li>Second</li>
        <li>Third</li>
</ul>

3. Help Center
```

Notice that elements given the `title` class now have a title look and feel.


2. Why us?

Just a paragraph to show that this is not a title

- First
- Second
- Third

3. Help Center

Figure 1.6: Location Inheritance - Title Example

1.3.4 Class Inheritance

Class inheritance is more and more being used in nowadays websites.

It means to apply certain styles from a predefined class, and other styles from another predefined class.

Take, for example, the multiple classes application in css, like below:

HTML

```
Just a paragraph here. It could be lorem ipsum.
<ul class="text-style-1">
        <li>This will be just a text line</li>
        <li class="link bold">Download Now</li>
        <li>Third line goes here</li>
</ul>
```

CSS

```
.text-style-1 {
        color: #50838e;
        font-weight: italic;
        font-size: 1.2em;
}

.link {
        color: blue;
        text-decoration: underline;
}

.bold {
        font-weight: bolder;
}
```

So above I have declared three classes and given them attributes.

Then, I have used them to style the elements by giving them these classes.

- This will be just a text line
- Download Now
- Third line goes here

WEB CODE GEEKS
WEB DEVELOPERS RESOURCE CENTER

Figure 1.7: Class Inheritance - Text Example

Now all three lines will have the attributes of `text-style-1` but in addition to that, the second `li` is going to have added the attributes of the `link` class and the `bold` class.

We can say that the second line **inherits** all ul attributes and just keeps adding new (or overriding existing) attributes.

1.4 Conclusion

On a more professional approach, we can state that using inheritance is not of much usage as most elements will need specific styling and independence from parent elements they are in.

However, when using inheritance property value to give elements styling attributes, keep in mind that `inherit` can be applied just as a single value (i.e you can't have something like: border: inherit 1em #ccc;) so you don't get individual values inherited.

If you want to see the browser default styles (from which your element attributes are inherited), you can inspect element and check "Show Inherited Properties" in Chrome.

1.5 Download

Download You can download the full source code of this example here: **CSS Inheritance Example**