Reset/remove CSS styles for element only

Asked 6 years, 5 months ago Active 1 month ago Viewed 747k times

434

170

I'm sure this must have been mentioned/asked before but have been searching for an age with no luck, my terminology must be wrong!

I vaguely remember a tweet I saw a while ago that suggested that there was a css rule available that would remove any styles previously set in the stylesheet for a particular element.

A good use example might be in a mobile-first RWD site where much of the styling used for a particular element in the small-screen views needs 'resetting' or removing for the same element in the desktop view.

A css rule that could achieve something like:

```
.element {
  all: none;
}
```

Example usage:

```
/* mobile first */
.element {
   margin: 0 10;
   transform: translate3d(0, 0, 0);
   z-index: 50;
   display: block;
   etc..
   etc..
}

@media only screen and (min-width: 980px) {
  .element {
    all: none;
  }
}
```

So we could quickly remove or re-set styling without having to declare every property.

css

edited Jul 17 '18 at 8:22
A J A Y
173 1 2 15

asked Apr 9 '13 at 11:49
matt_50
2,405 4 13 12

5 No, such a thing does not exist. Once an element has received a certain CSS style via a rule, it can not be just "taken back" – the only way is to overwrite every CSS property with the desired value explicitly. – CBroe Apr 9 '13 at 11:53

The way to do it is to restrict it in the first place with media queries – Kevin Lynch Apr 9 '13 at 11:58

related : stackoverflow.com/questions/7398279/… – Milche Patern Apr 9 '13 at 13:06

12 There *is* a property called `all` that is being proposed for resetting *all* CSS properties for a given element to certain CSS-wide values - the value you want to use would be `unset`, which resets a property to either its inherited value if it inherits by default, or otherwise, its initial value. No word on implementation, but it's nice to know somebody has thought of it. – BoltClock ♦ Apr 30 '14 at 4:58

2 `all: revert` will do. See my answer. @CBroe Yes such a thing is exists now. – Asim K T Mar 28 '16 at 6:06

14 Answers

542

✓

The CSS3 keyword `initial` sets the CSS3 property to the initial value as defined in the spec. The `initial` keyword has broad browser support except for the IE and Opera Mini families.

Since IE's lack of support may cause issue here are some of the ways you can reset some CSS properties to their initial values:

```
.reset-this {
    animation : none;
    animation-delay : 0;
    animation-direction : normal;
    animation-duration : 0;
    animation-fill-mode : none;
    animation-iteration-count : 1;
    animation-name : none;
    animation-play-state : running;
    animation-timing-function : ease;
```

```
background-color : transparent;
background-image : none;
background-origin : padding-box;
background-position : 0 0;
background-position-x : 0;
background-position-y : 0;
background-repeat : repeat;
background-size : auto auto;
border : 0;
border-style : none;
border-width : medium;
border-color : inherit;
border-bottom : 0;
border-bottom-color : inherit;
border-bottom-left-radius : 0;
border-bottom-right-radius : 0;
border-bottom-style : none;
border-bottom-width : medium;
border-collapse : separate;
border-image : none;
border-left : 0;
border-left-color : inherit;
border-left-style : none;
border-left-width : medium;
border-radius : 0;
border-right : 0;
border-right-color : inherit;
border-right-style : none;
border-right-width : medium;
border-spacing : 0;
border-top : 0;
border-top-color : inherit;
border-top-left-radius : 0;
border-top-right-radius : 0;
border-top-style : none;
border-top-width : medium;
bottom : auto;
box-shadow : none;
box-sizing : content-box;
caption-side : top;
clear : none;
clip : auto;
color : inherit;
columns : auto;
column-count : auto;
column-fill : balance;
```

```
    column-rule-style : none;
    column-rule-width : none;
    column-span : 1;
    column-width : auto;
    content : normal;
    counter-increment : none;
    counter-reset : none;
    cursor : auto;
    direction : ltr;
    display : inline;
    empty-cells : show;
    float : none;
    font : normal;
    font-family : inherit;
    font-size : medium;
    font-style : normal;
    font-variant : normal;
    font-weight : normal;
    height : auto;
    hyphens : none;
    left : auto;
    letter-spacing : normal;
    line-height : normal;
    list-style : none;
    list-style-image : none;
    list-style-position : outside;
    list-style-type : disc;
    margin : 0;
    margin-bottom : 0;
    margin-left : 0;
    margin-right : 0;
    margin-top : 0;
    max-height : none;
    max-width : none;
    min-height : 0;
    min-width : 0;
    opacity : 1;
    orphans : 0;
    outline : 0;
    outline-color : invert;
    outline-style : none;
    outline-width : medium;
    overflow : visible;
    overflow-x : visible;
    overflow-y : visible;
    padding : 0;
    padding-top : 0;
    page-break-after : auto;
    page-break-before : auto;
    page-break-inside : auto;
    perspective : none;
    perspective-origin : 50% 50%;
    position : static;
    /* May need to alter quotes for different locales (e.g fr) */
    quotes : '\201C' '\201D' '\2018' '\2019';
    right : auto;
    tab-size : 8;
    table-layout : auto;
    text-align : inherit;
    text-align-last : auto;
    text-decoration : none;
    text-decoration-color : inherit;
    text-decoration-line : none;
    text-decoration-style : solid;
    text-indent : 0;
    text-shadow : none;
    text-transform : none;
    top : auto;
    transform : none;
    transform-style : flat;
    transition : none;
    transition-delay : 0s;
    transition-duration : 0s;
    transition-property : none;
    transition-timing-function : ease;
    unicode-bidi : normal;
    vertical-align : baseline;
    visibility : visible;
    white-space : normal;
    widows : 0;
    width : auto;
    word-spacing : normal;
    z-index : auto;
    /* basic modern patch */
    all: initial;
    all: unset;
}

/* basic modern patch */

#reset-this-root {
    all: initial;
    }
  }
```

- Relevent github repo with a *december 2017* more exaustive list
- Related
- Related from MDN
- Related W3C specs

As mentioned in a comment by @user566245 :

> this is correct in principle, but individual mileage may vary. For example certain elements like textarea by default have a border, applying this reset will render those textarea's border less.

[POST EDIT FEB 4, '17] Upvoted for becoming a modern norm, user Joost

```
#reset-this-parent {
  all: initial;
  * {
    all: unset;
  }
}
```

EXAMPLE FROM W3

> For example, if an author specifies all: initial on an element it will block all inheritance and reset all properties, as if no rules appeared in the author, user, or user-agent levels of the cascade.
>
> This can be useful for the root element of a "widget" included in a page, which does not wish to inherit the styles of the outer page. Note, however, that any "default" style applied to that element (such as, e.g. display: block from the UA style sheet on block elements such as) will also be blown away.

JAVASCRIPT ?

Nobody thought about other than css to reset css? Yes?

> getElementsByTagName("*") will return all elements from DOM. Then you may set styles for each element in the collection:

answered Feb 9 '13 at 20:15 by VisioN

```
var allElements = document.getElementsByTagName("*");
for (var i = 0, len = allElements.length; i < len; i++) {
    var element = allElements[i];
    // element.style.border = ...
}
```

With all this said; i don't think a css reset is something feasable unless we end up with only one web browser .. if the 'default' is set by browser in the end.

For comparison, here is Firefox 40.0 values list for a `<blockquote style="all: unset;font-style: oblique">` where `font-style: oblique` triggers DOM operation.

```
align-content: unset;
align-items: unset;
align-self: unset;
animation: unset;
appearance: unset;
backface-visibility: unset;
background-blend-mode: unset;
background: unset;
binding: unset;
block-size: unset;
border-block-end: unset;
border-block-start: unset;
border-collapse: unset;
border-inline-end: unset;
border-inline-start: unset;
border-radius: unset;
border-spacing: unset;
border: unset;
bottom: unset;
box-align: unset;
box-decoration-break: unset;
box-direction: unset;
box-flex: unset;
box-ordinal-group: unset;
```

```
    box-sizing: unset;
    caption-side: unset;
    clear: unset;
    clip-path: unset;
    clip-rule: unset;
    clip: unset;
    color-adjust: unset;
    color-interpolation-filters: unset;
    color-interpolation: unset;
    color: unset;
    column-fill: unset;
    column-gap: unset;
    column-rule: unset;
    columns: unset;
    content: unset;
    control-character-visibility: unset;
    counter-increment: unset;
    counter-reset: unset;
    cursor: unset;
    display: unset;
    dominant-baseline: unset;
    empty-cells: unset;
    fill-opacity: unset;
    fill-rule: unset;
    fill: unset;
    filter: unset;
    flex-flow: unset;
    flex: unset;
    float-edge: unset;
    float: unset;
    flood-color: unset;
    flood-opacity: unset;
    font-family: unset;
    font-feature-settings: unset;
    font-kerning: unset;
    font-language-override: unset;
    font-size-adjust: unset;
    font-size: unset;
    font-stretch: unset;
    font-style: oblique;
    font-synthesis: unset;
    font-variant: unset;
    font-weight: unset;
    font: ;
    force-broken-image-icon: unset;
    height: unset;
    image-rendering: unset;
    ime-mode: unset;
    inline-size: unset;
    isolation: unset;
    justify-content: unset;
    justify-items: unset;
    justify-self: unset;
    left: unset;
    letter-spacing: unset;
    lighting-color: unset;
    line-height: unset;
    list-style: unset;
    margin-block-end: unset;
    margin-block-start: unset;
    margin-inline-end: unset;
    margin-inline-start: unset;
    margin: unset;
    marker-offset: unset;
    marker: unset;
    mask-type: unset;
    mask: unset;
    max-block-size: unset;
    max-height: unset;
    max-inline-size: unset;
    max-width: unset;
    min-block-size: unset;
    min-height: unset;
    min-inline-size: unset;
    min-width: unset;
    mix-blend-mode: unset;
    object-fit: unset;
    object-position: unset;
    offset-block-end: unset;
    offset-block-start: unset;
    offset-inline-end: unset;
    offset-inline-start: unset;
    opacity: unset;
    order: unset;
    orient: unset;
    outline-offset: unset;
    outline-radius: unset;
    outline: unset;
    overflow: unset;
    padding-block-end: unset;
    padding-block-start: unset;
    padding-inline-end: unset;
    page-break-before: unset;
    page-break-inside: unset;
    paint-order: unset;
    perspective-origin: unset;
    perspective: unset;
    pointer-events: unset;
    position: unset;
    quotes: unset;
    resize: unset;
    right: unset;
    ruby-align: unset;
    ruby-position: unset;
    scroll-behavior: unset;
    scroll-snap-coordinate: unset;
    scroll-snap-destination: unset;
    scroll-snap-points-x: unset;
    scroll-snap-points-y: unset;
    scroll-snap-type: unset;
    shape-rendering: unset;
    stack-sizing: unset;
    stop-color: unset;
    stop-opacity: unset;
    stroke-dasharray: unset;
    stroke-dashoffset: unset;
    stroke-linecap: unset;
    stroke-linejoin: unset;
    stroke-miterlimit: unset;
    stroke-opacity: unset;
    stroke-width: unset;
    stroke: unset;
    tab-size: unset;
    table-layout: unset;
    text-align-last: unset;
    text-align: unset;
    text-anchor: unset;
    text-combine-upright: unset;
    text-decoration: unset;
    text-emphasis-position: unset;
    text-emphasis: unset;
    text-indent: unset;
    text-orientation: unset;
    text-overflow: unset;
    text-rendering: unset;
    text-shadow: unset;
    text-size-adjust: unset;
    text-transform: unset;
    transform: unset;
    transition: unset;
    user-focus: unset;
    user-input: unset;
    user-modify: unset;
    user-select: unset;
    vector-effect: unset;
    vertical-align: unset;
    visibility: unset;
    white-space: unset;
    width: unset;
    will-change: unset;
    window-dragging: unset;
    word-break: unset;
    word-spacing: unset;
    word-wrap: unset;
    writing-mode: unset;
    z-index: unset;
```

edited Dec 7 '18 at 1:21

answered Apr 9 '13 at 13:23
Milche Patern
15.1k 5 29 51

8 i think this is correct in principle, but individual mileage may vary. For example certain elements like textarea by default have a border, applying this reset will render those textarea's border less. So it not a true reset. I ended up using it for only certain properties which I cared about. You can also combine it with the * selector to reset all elements or reset all elements inside a specific element. – user566245 Feb 18 '14 at 11:13

8 @user566245 Applying this with a * selector will kill your browser AND a kitten. This is NOT a true reset. True reset just don't exist. – Milche Patern Feb 18 '14 at 16:48

@Milkywayspatterns lol, you are probably right. For me I only took the attributes which i wanted to reset and applied to "div#theid *". Hopefully this doesn't kill anyones kitten :) – user566245 Feb 19 '14 at 2:00

1 @Jeremy: You're thinking of browser defaults, which vary for different elements. The initial value of display is always inline regardless of which element it's being applied to. – BoltClock ♦ Apr 30 '14 at 4:52

1 @mmmshuddup Thanks for the tip. If you take a look at the original answer, i reformed it CSS like. For the compression, well, this is an answer, not a copy-paste patch. Isn't it? – Milche Patern Jul 7 '14 at 13:29

For future readers. I think this is what was meant but currently isn't really wide supported (see below):

```
#someselector {
  all: initial;
  * {
    all: unset;
  }
}
```

- Supported in (source): Chrome 37, Firefox 27, IE 11, Opera 24
- Not supported: Safari

answered Jul 9 '15 at 12:47
joost
4,667 2 24 33

11 The source claims that Internet Explorer doesn't support `all` . – Dan Dascalescu Sep 6 '15 at 7:33

1 At long last. This should be the new accepted answer. – JS_Riddler Sep 17 '15 at 20:17

2 Microsoft lists `all` as under consideration. A future version of Edge may well support it. – Kevin Oct 14 '15 at 1:06

1 I just read "On inherited properties, the initial value may be surprising and you should consider using the inherit, unset, or revert keywords instead. Also, does this comes back to browser-specific? Where this initial initial is .. set by the ... ? DDT? – Milche Patern Dec 13 '15 at 20:17

15 The only place I've seen nested CSS like `#someselector { ... * { all: unset; } ... }` is in Sass. You haven't mentioned Sass here - is this some new CSS3 thing? Searching for "nested CSS" just gets me entry-level tutorials and Sass info. Adding the `... * { ... } ...` nested part to my CSS (in HTML5) breaks my document (my child elements individually take the style I just wanted to apply to the parent). – i336_ Sep 16 '16 at 1:48

24

Let me answer this question thoroughly, because it's been a source of pain for me for several years and very few people really understand the problem and why it's important for it to be solved. If I were at all responsible for the CSS spec I'd be embarrassed, frankly, for having not addressed this in the last decade.

The Problem

You need to insert markup into an HTML document, and it needs to look a specific way. Furthermore, you do not own this document, so you cannot change existing style rules. You have no idea what the style sheets *could* be, or what they may change to.

Use cases for this are when you are providing a displayable component for unknown 3rd party websites to use. Examples of this would

1. An ad tag
2. Building a browser extension that inserts content
3. Any type of widget

Simplest Fix

Put everything in an iframe. This has it's own set of limitations:

1. Cross Domain limitations: Your content will not have access to the original document at all. You cannot overlay content, modify the DOM, etc.
2. Display Limitations: Your content is locked inside of a rectangle.

If your content *can* fit into a box, you can get around problem #1 by having your content write an iframe and explicitly set the content, thus skirting around the issue, since the iframe and document will share the same domain.

CSS Solution

I've search far and wide for the solution to this, but there are unfortunately none. The best you can do is explicitly override all possible properties that can be overridden, and override them to what you *think* their default value should be.

Even when you override, *there is no way to ensure a more targeted CSS rule won't override yours*. The best you can do here is to have your override rules target as specifically as possible and hope the parent document doesn't accidentally best it: use an obscure or random ID on your content's parent element, and use !important on all property value definitions.

answered Feb 6 '15 at 18:51
JS_Riddler
550 1 8 14

2 You can use the all property, which is supported by all modern browsers. – Dan Dascalescu Sep 6 '15 at 7:32

1 Interesting, maybe someone should tell caniuse caniuse.com/#feat=css-all – Shanimal Nov 5 '15 at 12:33

1 The proper solution to this general problem is to use Web Components – GetFree Jan 7 '16 at 6:56

1 This is a very real issue, but only ever exists as a result of poorly developed CSS in the first place. If you are creating markup and CSS, if you have done it properly, none of your styles should bleed into a third party app. If I were responsible for the CSS spec, I would not be embarrassed, but upset people are brutally misusing what I had created. – ESR Apr 4 '17 at 5:34

21

There's a brand new solution found to this problem.

> You need **"A css rule available that would remove any styles previously set in the stylesheet for a particular element."**

So, if the element have a class name like `remove-all-styles` :

Eg:

HTML:

```
<div class="remove-all-styles other-classe another-class">
   <!-- content -->
   <p class="text-red other-some-styles"> My text </p>
</div>
```

With CSS:

```
  .remove-all-styles {
    all: revert;
  }
```

Will reset all styles applied by `other-class` , `another-class` and all other inherited and applied styles to that `div` .

Or in your case:

```
/* mobile first */
.element {
   margin: 0 10;
   transform: translate3d(0, 0, 0);
   z-index: 50;
   display: block;
   etc..
   etc..
}

@media only screen and (min-width: 980px) {
    }
  }
```

Will do.

> Here we used one cool CSS property with another cool CSS value.
>
> 1. `revert`
>
> Actually `revert` , as the name says, reverts that property to its user or user-agent style.
>
> 2. `all`
>
> And when we use `revert` with the `all` property, all CSS properties applied to that element will be reverted to user/user-agent styles.

Click here to know difference between author, user, user-agent styles.

For ex: if we want to **isolate embedded widgets/components from the styles of the page that contains them**, we could write:

```
.isolated-component {
 all: revert;
}
```

Which will reverts all `author styles` (ie developer CSS) to `user styles` (styles which a user of our website set - less likely scenario) or to `user-agent` styles itself if no user styles set.

More details here: https://developer.mozilla.org/en-US/docs/Web/CSS/revert

And only issue is the support: only Safari 9.1 and iOS Safari 9.3 have support for `revert` value at the time of writing.

So I'll say use this style and fallback to any other answers.

edited Dec 7 '18 at 9:13
SC1000
235 4 13

answered Mar 28 '16 at 6:05
Asim K T
7,627 3 51 71

1 Would be cool, but unfortunately the browser support still has holes: caniuse.com/#feat=css-all (though smaller than caniuse shows, for example `all:`

10

another ways:

1) include the css code(file) of Yahoo CSS reset and then put everything inside this DIV:

```
<div class="yui3-cssreset">
    <!-- Anything here would be reset-->
</div>
```

2) or use

- https://cssreset.com/scripts/vanilla-css-un-reset/
- https://cssreset.com/scripts/html5-doctor-css-reset-stylesheet/
- https://cssreset.com/scripts/eric-meyer-reset-css/
- https://cssreset.com/scripts/tripoli-css-reset-david-hellsing/
- https://cssreset.com/scripts/normalize-css/

edited Aug 2 at 7:40

answered Feb 28 '14 at 13:04
T.Todua
34.3k 12 151 145

3

I do not recommend using the answer that has been marked as correct here. It is a huge blob of CSS which tries to cover everything.

I would suggest that you evaluate how to remove the style from an element on a per case basis.

Lets say for SEO purposes you need to include an H1 on a page which has no actual heading in the design. You might want to make the nav link of that page an H1 but ofcourse you do not want that navigation link to display as a giant H1 on the page.

What you should do is wrap that element in an h1 tag and inspect it. See what CSS styles are being applied specifically to the h1 element.

Lets say I see the following styles applied to the element.

```
//bootstrap.min.css:1
```

```
    font-weight: normal;
    font-style: normal;
    text-transform: uppercase;
}

//bootstrap.min.css:1
h1, .h1 {
    font-size: 36px;
}

//bootstrap.min.css:1
h1, .h1, h2, .h2, h3, .h3 {
    margin-top: 20px;
    margin-bottom: 10px;
}

//bootstrap.min.css:1
h1, h2, h3, h4, h5, h6, .h1, .h2, .h3, .h4, .h5, .h6 {
    font-family: inherit;
    font-weight: 500;
    line-height: 1.1;
    color: inherit;
}

//bootstrap.min.css:1
h1 {
    margin: .67em 0;
    font-size: 2em;
}

//user agent stylesheet
h1 {
    display: block;
    font-size: 2em;
    -webkit-margin-before: 0.67em;
    -webkit-margin-after: 0.67em;
    -webkit-margin-start: 0px;
    -webkit-margin-end: 0px;
    font-weight: bold;
}
```

Now you need to pin point the exact style which are applied to the H1 and unset them in a css class. This would look something like the following:

```
    font-weight: unset !important;
    font-style: unset !important;
    text-transform: unset !important;
    margin-top: unset !important;
    margin-bottom: unset !important;
    font-family: unset !important;
    line-height: unset !important;
    color: unset !important;
    margin: unset !important;
    display: unset !important;
    -webkit-margin-before: unset !important;
    -webkit-margin-after: unset !important;
    -webkit-margin-start: unset !important;
    -webkit-margin-end: unset !important;
}
```

This is much cleaner and does not just dump a random blob of code into your css which you don't know what it's actually doing.

Now you can add this class to your h1

```
<h1 class="no-style-h1">
     Title
</h1>
```

answered Oct 26 '17 at 14:53

Harry
2,197 2 6 22

3

If you happen to be using sass in a build system, one way to do this that will work in all the major browsers is to wrap all your style imports with a :not() selector like so...

```
:not(.disable-all-styles) {
  @import 'my-sass-file';
  @import 'my-other-sass-file';
}
```

Then you can use the disable class on a container and the sub-content won't have any of your styles.

```
</div>
```

Of course all your styles will now be prepended with the :not() selector, so it's a little fugly, but works well.

answered Aug 9 '18 at 18:43

BrandonReid
571 1 8 15

1

You mentioned mobile-first sites... For a responsive design, it's certainly possible to override small-screen styles with large-screen styles. But you might not need to.

Try this:

```
.thisClass {
    /* Rules for all window sizes. */
}

@media all and (max-width: 480px) {
    .thisClass {
        /* Rules for only small browser windows. */
    }
}

@media all and (min-width: 481px) and (max-width: 960px) {
    .thisClass {
        /* Rules for only medium browser windows. */
    }
}

@media all and (min-width: 961px) {
    .thisClass {
        /* Rules for only large browser windows. */
    }
}
```

Those media queries don't overlap, so their rules don't override each other. This makes it easier to maintain each set of styles separately.

answered Nov 11 '15 at 11:10

1

In my specific scenario i wanted to skip applying common styles to a specific part of the page, better illustrated like this:

```
<body class='common-styles'>
    <div id='header'>Wants common styles</div>
    <div id='container'>Does NOT want common styles</div>
    <div id='footer'>Wants common styles</div>
</body>
```

After messing with CSS reset which didn't bring much success (mainly because of rules precedence and complex stylesheet hierarchy), brought up ubiquitous jQuery to the rescue, which did the job very quickly and reasonably dirty:

```
$(function() {
    $('body').removeClass('common-styles');
    $('#header,#footer').addClass('common-styles');
});
```

(Now tell how evil it is to use JS to deal with CSS :-))

answered Dec 26 '17 at 12:27
esteewhy
995 9 22

0

For those of you trying to figure out how to actually remove the styling from the element only, without removing the css from the files, this solution works with jquery:

```
$('.selector').removeAttr('style');
```

edited Jan 27 '16 at 1:14
Tristan
3,193 8 17 25

answered Jan 27 '16 at 0:48
Jeff Davenport
1,110 2 9 17

developer.mozilla.org/en-US/docs/Web/API/Element/ – csuh Feb 23 '16 at 21:13

0

BETTER SOLUTION

Download "copy/paste" stylesheet to reset css properties to default (UA style):
https://github.com/monmomo04/resetCss.git

Thanks@Milche Patern!
I was really looking for reset/default style properties value. My first try was to copy the computed value from the browser Dev tool of the root(html) element. But as it computed, it would have looked/worked different on every system.
For those who encounter a browser crash when trying to use the asterisk * to reset the style of the children elements, and as I knew it didn't work for you, I have replaced the asterisk "*" with all the HTML tags name instead. The browser didn't crash; I am on Chrome Version 46.0.2490.71 m.
At last, it's good to mention that those properties will reset the style to the default style of topest root element but not to the initial value for each HTML element. So to correct this, I have taken the "user-agent" styles of webkit based browser and implemented it under the "reset-this" class.

Useful link:

Download "copy/paste" stylesheet to reset css properties to default (UA style):
https://github.com/monmomo04/resetCss.git

User-agent style:
Browsers' default CSS for HTML elements
http://trac.webkit.org/browser/trunk/Source/WebCore/css/html.css

Css specifity (pay attention to specifity) :
https://css-tricks.com/specifics-on-css-specificity/

https://github.com/monmomo04/resetCss.git

edited May 23 '17 at 12:02
Community ♦
1 1

answered Oct 27 '15 at 0:57
Monero Jeanniton
160 1 12

---

if you set your CSS within classes, you can easly remove them using jQuery removeClass() Method. The code below removes .element class:

```
    <div class="element">source</div>
    <div class="destination">destination</div>
      <script>
        $(".element").removeClass();
      </script>
```

If no parameter is specified, this method will remove ALL class names from the selected elements.

edited Oct 24 '17 at 20:31 answered Oct 24 '17 at 20:16
A. K.
44 5

-1

Any chance you're looking for the !important rule? It doesn't undo all declarations but it provides a way to override them.

"When an !important rule is used on a style declaration, this declaration overrides any other declaration made in the CSS, wherever it is in the declaration list. Although, !important has nothing to do with specificity."

https://developer.mozilla.org/en-US/docs/CSS/Specificity#The_!important_exception

answered Apr 9 '13 at 13:00
Erik Nijland
651 1 6 21

!Importants is not the way to go. It's to bold to use and you should only use it as a last resort (for instance, when a plugin used an !important too) – Maarten Wolfsen Jan 20 '17 at 13:39

-2

No, this is just a matter of managing your css structure better.

In your case i would order my css something like this:

```
.element, .element1, .element2 p{z-index: 50; display: block}
.element, .element1{margin: 0 10}
.element2 p{transform: translate3d(0, 0, 0)}
```

```
    .element, .element1, .element2 p{display: none}
    }
```

Just experiment.

answered Apr 9 '13 at 12:54

WIWIWWIISpitFire
764 1 8 17