Object Oriented CSS: Catchy Buzz-phrase or Legitimate Design Approach?

Asked 15 years, 9 months ago Modified 4 years, 9 months ago

Viewed 2k times

16

It seems there is a new catch-phrase emerging in the web development field: object-oriented CSS.

On the face of it, this strikes me as simply being best-practice packaged up in a catchy slogan. I understand and fully respect the intentions behind the movement, but is there any more to it?

Does anyone have any further insight that sets this approach apart as something more credible or should I just take it as a reminder to make sure I inherit and cascade my classes correctly?

css **oop** **methodology** **oocss**

Share

Improve this question

Follow

edited Sep 4, 2011 at 22:45

asked Mar 18, 2009 at 8:45

Phil.Wheeler
16.8k ● 10 ● 102 ● 156

10 Answers

Sorted by: Highest score (default)

10

I would say it's more of a catchy buzz-phrase for something already present in CSS. Of course, before we start talking about what is OO and what is not and how CSS is object-oriented, we would have to define what it actually is - which is something others have struggled with before and is subject to heated debates. But if we assume that the basic principles of OO are:

- Class
- Object
- Instance
- Method
- Message passing
- Inheritance
- Abstraction
- Encapsulation
- Polymorphism
- Decoupling

we can say, that Cascading Style Sheets are somewhat object-oriented, because they allow classes to be defined, instances/objects created (by assigning a class

to an element), inheritance of classes (even multiple inheritance), abstraction (e.g. by defining styles for the plain elements) and polymorphism (by defining the same class name for different elements). Of course methods/message passing is not possible, because of the static nature of CSS.

So in general I would say its a valid approach to develop CSS in an object-oriented manner, but I would not really call it Object Oriented CSS, because at least to me, it is something deeply inherent to CSS. It would be somewhat like saying "I am doing Object Oriented Java..."

Share Improve this answer

Follow

answered Mar 18, 2009 at 9:18

Simon Lehmann

10.9k ● 4 ● 43 ● 54

I like this argument. I think it answers the question well, but presents a new one: are web developers forgetting how to build well-formed stylesheets? Maybe a question for later.
– Phil.Wheeler Mar 18, 2009 at 10:28

6 -1: This entirely misses the point of OOCSS. CSS is *hard* to encapsulate (look at what happens when you nest display types), and "best" practices tend to discourage DRY principles (e.g. don't use presentational class names). Nicole Sullivan's OOCSS outlines principles that allow you to *actually* use CSS in a modular, maintainable manner.
– theazureshadow Aug 25, 2011 at 17:47

@Phil.Wheeler "answers the question well and presents a new one" nope. That new one is what gave rose to the term OOCSS in the first place, to describe a set of principles and patterns that allow CSS to be used in a dry, consistent way. Not having any awareness of how the term OOCSS is used

in the CSS community, when that 'new question' has been answered by it, does not provide a good answer to the first question. – Barney Sep 1, 2013 at 9:44

5

Catchy buzz-phrase AND legitimate design approach.

Though I think some of the ideas there are a bit naive in that they tend to forget the "client changes eveything as it goes" web development paradigm.

Share Improve this answer

Follow

answered Mar 18, 2009 at 8:53

Julian Aubourg

11.4k ● 1 ● 30 ● 29

1 Hmm, I think I shouldn't have said that out loud ;) – Julian Aubourg Mar 18, 2009 at 9:19

4

I think the buzzwordiness of calling it "Object-oriented CSS" is actually detracting from its usefulness and the wider adoption of the concept.

When I read up on it, I think the claim that it was OO actually slowed my understanding of what it really was.

People who know what "object-oriented" means in programming will be skeptical, because it's not *really* OO, is it? It's just an attempt to apply some OO principles to CSS to improve its efficiency. And on the other side, most client-side developers will not understand the concept at

all if they're not programmers, and so they'll just be baffled or embarrassed.

So, great concept, needs rebranding. Maximum CSS! Power CSS! CSS Reborn! CSS Squared! CSS Prime! Something like that.

Share Improve this answer

Follow

answered Mar 19, 2009 at 11:16

AmbroseChapel

12.1k ● 7 ● 49 ● 69

3

I've been saying this for *years*.

CSS selectors are based on instance IDs and classes. It even supports multiple inheritance. How much more obvious could that be?

The beauty of thinking of CSS in object terms is that it becomes very straightforward to start "casting" your mark-up elements. Need that div to suddenly become INotRenderedAnymore? Just let JS extend it's class attribute to match `.removed`, don't mess around with style properties.

Obviously that's a fairly trite example, but the benefits should be clear - especially in the context of JS manipulation. You can decouple the JS from the actual styles it has to modify, which is both a maintenance and abstraction bonus, but also has fringe benefits like not setting very high-specificity styles directly (which CSS cannot override).

Share Improve this answer

Follow

answered Mar 18, 2009 at 9:04

annakata

75.7k • 18 • 115 • 180

So this comes back to the "Progressive Enhancement" ideology. In order to have a distinct separation of concerns (content, presentation, behaviour), you need to ensure that your [css] classes are structured according to best practice? – Phil.Wheeler Mar 18, 2009 at 9:19

yes, exactly so - and if you think of CSS in OO terms, this is an easy way to get there – annakata Mar 18, 2009 at 9:38

IDs are better thought to be about polymorphism IMO. When something regular gets re-used in a different context (anything under an element with an ID umbrella) there might be variations. The closest to "instances" in CSS that we'll find is in the ultimate rendering of HTML + CSS. – Erik Reppen Aug 31, 2013 at 2:03

3

Writing this answer all these years later for posterity. All the answers above have clearly never heard serious talk of object-oriented CSS. When the OP asked if it was a buzzword, that should have been an indication that the expression referred to a subject that merited conversation (more so than "What do we mean by object-oriented?").

Objected-Oriented CSS is a term I believe was coined by Yahoo front-end tech evangelist Nicole Sullivan, after she consulted for Facebook in refactoring their presentational front-end code into something slimmer and more manageable that could yet be extended and modified easily as time went on. The term 'object-oriented' refers

not to the nature of CSS itself (as such), but to an approach to writing CSS and affected markup that lends to conciseness and extensibility.

Some example principles of OO CSS include:

- Avoid using IDs as selectors: the fact that an object is unique does not necessarily mean that its key properties should be defined in such a way that they override all others.
- Avoid long selectors based on nested markup: defining an element's appearance based on its incidental location in document structure is often a logical fallacy, and moving it to a new location will force you to re-write the selector. As with the aforementioned, this bad practice also gives cause for conditional overrides or extensions to need extra strength and specificity in their selectors, which is rarely helpful.
- Use non-semantic markup to create elements with distinct presentational purposes, rather than trying to overload all the specifics of an individual element's appearance into one rule. This leads to less verbose CSS.

Ultimately the goal of OOCSS is DRY. You shouldn't be writing thousands of lines of CSS when you're essentially creating variations of similar things.

Nicole has given some good talks on the subject – she's also written a few articles expanding on some useful

techniques to employ and bad practices to avoid.

'OOCSS' is also the name of the framework she's written that gives some boilerplate code for extensible CSS layouts. How and why this framework is so good (the grids module is something I use virtually everywhere) isn't completely self-evident from the code itself, and reading up on her blog posts on the ideas behind will definitely help make better use of it and CSS in general.

Share Improve this answer

Follow

answered Aug 30, 2012 at 13:03

Barney

16.4k ● 5 ● 65 ● 80

> IDs when used well reduce cruft. Specificity is a tool, not the enemy. Removing an entire tier of the specificity orders of magnitude will ultimately contribute to cruft in selector quantity, which IMO, is the real enemy not IDs. IDs are perfect when a re-used element needs variation in a new context. It's closer to taxonomy than OOP, IMO. DRY is a good smell-test even in CSS but ultimately we're talking about layout and design. It's very easy for two design elements to have near-identical property-sets while contributing to different aspects of a design that are worth separating. – Erik Reppen Aug 31, 2013 at 1:23

▲

3

▼

I have been embarrassing both OOCSS and the B.E.M. naming conventions for quite some time and will NEVER look back. For those who are claiming that it's just a "catchy buzz-word" or "CSS already does this stuff", do not understand the potential of writing css using both of these methodologies.

Let's look at the simplest of objects, a list with links. It comes in many different flavors:

1. Menus
2. Toolbars
3. Tabs
4. Panels (Bootstrap)

In OOCSS, we find the common properties of each of these and create a base object. I usually call it nav.

```
/*  Nav
    ============================================================*

    /*  B
        ------------------------------------------------------*

        .nav
        {
            margin-left:            0;

            padding-left:           0;

            list-style:             none;
        }

    /*  E
        ------------------------------------------------------*

        .nav__item
        {
            float:                  left;
        }

        .nav__link
        {
            display:                block;
```

```
            color:                  inherit;

            text-decoration:        none;
        }

    /*  M
        ------------------------------------------*

        .nav--right
        {
            float:                  right;
        }

        .nav--stack .nav__item
        {
            float:                  none;
        }
```

You will notice a few things:

1. Nav is the base object that gets applied to the block element
2. Child elements are prefixed with nav_
3. Modifiers are prefixed with nav--
4. A modifier is an option that changes behavior. For example --right floats nav right.

Once I have my base object witten, I create skins that will change the appearance of the object. This will turn it into toolbars, tabs, etc. Microsoft has Pivot tabs on their phones. It is an easier skin to create fpr now.

```
/*  Nav
    ============================================*

    /*  E
```

```
        ------------------------------------------*

        .pivot .nav__item
        {
            margin-left:            24px;

            color:                  #aaa;

            font-size:          36px;
        }

        .pivot .nav__item--active, .pivot .nav__item:h
        {
            color:                  #000;
        }
```

To use this object and skin, you would write

```
<ul class="pivot nav">

   <li class="nav__item">

        <a class="nav__link"> Item 1 </a>

    </li>

   <li class="nav__item">

        <a class="nav__link"> Item 2 </a>

    </li>

</ul>
```

Because of its location independence, you can also write it as

```
<nav class="pivot nav">
```

```
    <div class="nav__item">
        <a class="nav__link"> Item 1 </a>
    </li>
    <div class="nav__item">
        <a class="nav__link"> Item 2 </a>
    </div>
</nav>
```

Ultimately, you are separating the container from the skin. I would suggest start smaller with Nicole Sullivans Media Object. Take a look at Twiter Bootstrap and Inuit.css for more inspiration.

Share Improve this answer

Follow

answered Aug 6, 2013 at 16:13

Eric Harms

569 ● 5 ● 17

Okay, so I wish to add a modifier to all my nav elements that live within a certain kind of section that appears in a wide variety of places across my app. For whatever reason, I find it more advantageous for that particular lot to adopt inline-block rather than a float scheme. Why am I hunting down every template that has one of those sections and adding a `nav--<modifier>` in the HTML when I could have just wrote property definitions for the selector `#annoying_section_2chng .nav > <some element>` – Erik Reppen Aug 31, 2013 at 1:38

'OOCSS', 'SMACSS' and 'Atomic design' really are just different buzz words that reference the same design pattern. It's a very useful design pattern, though, that I based my own

CSS framework on (cascade-framework.com)..... I don't favor BEM syntax, though. BEM is too verbose, too restrictive and too plain ugly for my taste. In fact, I'd go as far as saying that BEM removes many of the advantages gained from using OOCSS, SMACSS or however you want to call it.
– John Slegers Jan 7, 2014 at 17:50

2

The term "Object-oriented CSS" is a misnomer.

"Object-oriented CSS" is really just a design pattern for how to get most out of your CSS and is basicly the same approach Jonathan Snooks calls SMACSS.

Whether you call it OOCSS or SMACSS, the key to the approach is that you create generic UI elements like the nav abstraction. These UI elements can then be enhanced with more specific features by adding extra classes to the element and/or a container element. Or, as an alternative, you can add your own custom CSS rules using the element's ID or semantic classes.

Cascade Framework is a brand new CSS framework based on this approach. It gives you optimal performance, optimal flexibility and optimal modularity with just a tiny footprint.

Share Improve this answer Follow

edited Mar 11, 2020 at 16:44

answered Mar 25, 2013 at 21:01


John Slegers
47k • 23 • 203 • 172

CSS is similar to OO languages in many ways: write

```
p { color: red }
p span { color: blue }
```

and you have essentially the inheritance. Here's more complicated example with having terrier extend dog extend animal classes:

```
.animal { font-weight:bold; color: blue; }
.dog:before, .terrier:before { content: "GRRR"; }
.animal, .dog, .terrier { color: brown }
```

Now you can use classes animal, dog and terrier in an OO manner.

It is important to remember that CSS very good in solving the problem it's been made for: specifying the styles for elements in a transparent way. Could it be better with more OO concepts? I'm not sure. Let's say somebody says: the CSS file would be simpler if it looked like:

```
@class dog @derives_from animal /* the syntax i just i
@class terrier @derives_from dog

.animal { font-weight:bold; color: blue; }
.dog:before { content: "GRRR"; }
.terrier { color: brown }
```

This does look simpler, but an even simpler solution is to drop @class thing while adding 'dog' to any 'terrier' and 'animal' to any 'dog' server-side (trivial replace statement) or with javascript.

The best thing about CSS is that it's simple and it falls back easily, meaning browsers don't need to interpret CSS they don't understand and things works out reasonably fine. Since you'll have to break this backward compatibility with major new CSS structures, I think this makes object-oriented CSS more of a buzz phrase.

Share Improve this answer

Follow

answered May 31, 2009 at 22:00

ilya n.

18.8k ● 16 ● 74 ● 89

0

its really one of those debatable things like tables vs divs etc etc.

In my opinion, theres a lot of developers so entrenched in OO that they try to stick it on everything, first Javascript and now CSS. Dont get me wrong JavaScript has elements of OO as well, but i digress.

Since CSS is already a buzzword in itself (all the employers want the web 2.0 CSS approach) a lot of new developers are discovering it. This is not a bad thing, however as developers they did what they do best and tried to improve on CSS. In a developers mind (I'm a developer) organizing CSS according to the OO

principles makes perfect sense - hence the new buzzword.

Ultimately what I am trying to say is that OO CSS is just an approach that certain people take, since it seems more logical. If you are writing CSS that is going to be maintained by developers then this approach will suit well. It really comes down to you how you write your CSS and your own personal style...

Personally I don't care how people write their CSS - if it falls on me to maintain it, Firebug makes the job trivial anyway.

Share Improve this answer

Follow

answered Mar 18, 2009 at 9:03

Darko

38.8k ● 15 ● 83 ● 107

I do care how i write my own CSS though, just so people dont get the wrong idea :) – Darko Mar 18, 2009 at 9:05

tables vs divs is not debatable, there *is* a right and a wrong there, the only discussion is that some people feel "it works, tables are fine" is a pragmatic solution and others feel that's tantamount to giving up. GOTO "works" too, there's a reason why nobody uses it. – annakata Mar 18, 2009 at 9:12

thats why I say its debatable - pragmatism vs style, you dont need to preach to the converted brother. well not converted, i never really used tables for layout... – Darko Mar 18, 2009 at 9:19

false economy vs learning css more like, but duly noted my anti-table colleague :) – annakata Mar 18, 2009 at 9:40

> Just for the record I love OO AND CSS and I hate it when people put OO on CSS. @annakata For about 10 years it was impossible to vertically center an element of unknown height because IE couldn't be bothered to catch up with Netscape Navigator circa 2002ish on the god !@#$ing damn table-display properties established by CSS2. There was in fact a highly specific argument for a very long time but yeah, if you're not supporting IE<=7 there is none currently although I still find those table-display properties kind of brittle. – Erik Reppen Aug 31, 2013 at 1:57

There's not much to it, i believe.

OOCSS *Object Oriented Cascading stylesheet*

It reduces the code to repeat over and over again and you can set a global css for `foocontainer` once and reuse everywhere and have custom style for `header` `body` and `footer`.

```
<div class="foocontainer header"> Your header </div> <
<div class="foocontainer body"> Your body </div>  <!--
<div class="foocontainer footer> Some footer </div> <!
```

BEM *Block--Element__Modifier*

It keeps the code easy to read but ugly to write and you can easily identify the children with their parents.

```
<div class="foocontainer--header"> Your header </div>
<div class="foocontainer--body"> Your body </div> <!--
```

```
<div class="foocontainer--footer> Some footer </div> <
```

Share Improve this answer

Follow

answered Aug 11, 2016 at 4:51

Dexter

9,254 ● 4 ● 46 ● 46