Format number to always show 2 decimal places

651

100

I would like to format my numbers to always display 2 decimal places, rounding where applicable.

Examples:

```
number     display
------     -------
1          1.00
1.341      1.34
1.345      1.35
```

I have been using this:

```
parseFloat(num).toFixed(2);
```

But it's displaying `1` as `1`, rather than `1.00`.

javascript floating-point number-formatting

edited May 26 '11 at 17:13
drudge
25.8k 5 29 41

asked May 26 '11 at 5:22
Varada
5,675 12 43 66

2 I mean if I enter 1 it will not show the number as 1.00, But if I enter 1.345 then it will show 1.35 – Varada May 26 '11 at 5:26

1 I've reworded your question to what I believe you were looking for. Please check to make sure I've understood you correctly. – drudge May 26 '11 at 17:14

precise rounding with ie support . gist.github.com/ArminVieweg/28647e735aa6efaba401 – TarranJones Sep 18 '15 at 13:01

Possible duplicate of In jQuery, what's the best way of formatting a number to 2 decimal places? – zero8 Dec 16 '16 at 7:11

Possible duplicate of Formatting a number with exactly two decimals in JavaScript – VLAZ Feb 20 at 14:05

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

939

This works fine in FF4:

```
parseFloat(Math.round(num3 * 100) / 100).toFixed(2);
```

Live Demo

Show code snippet

Note that it will **round** to 2 decimal places, so the input `1.346` will return `1.35` .

edited Feb 18 '18 at 11:25
Donald Duck
4,197 13 42 65

answered May 26 '11 at 5:27
drudge
25.8k 5 29 41

6 @Kooilnc: OP wants `1` to display as `1.00` , and `1.341` to display as `1.34` . – drudge May 26 '11 at 16:59

37 No need to use round() since toFixed() rounds it. – Milan Babuškov Dec 14 '13 at 21:24

18 toFixed() does round it but don't forget it's returning a string...So this method is really only useful for display. If you want to perform further mathematical computations on the rounded value do not use toFixed(). – TWright Oct 15 '15 at 7:19

7 according to MDN, Math.round will not always give accurate results due to rounding errors. I tested this with 1.005, which should round to 1.01, but it gives 1.00. Use my answer for consistent accuracy: stackoverflow.com/a/34796988/3549440. – Nate Jan 14 '16 at 18:39

9 This entire answer could be reduced to `(+num).toFixed(2)` . It even retains the rounding bug in the original, see *Nate's answer*. – RobG Aug 15 '17 at 4:01

1

Just run into this one of longest thread, below is my solution:

parseFloat(Math.round((parseFloat(num * 100)).toFixed(2)) / 100).toFixed(2)

Let me know if anyone can poke a hole

answered May 28 at 20:29

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

0

function numberWithCommas(number) {

```
var newval = parseFloat(Math.round(number * 100) / 100).toFixed(2);

return newval.toString().replace(/\B(?=(\d{3})+(?!\d))/g, ",");
```

}

answered Apr 30 at 8:45

Tsuna Sawada
96 1 1 13

78

This answer will fail if `value = 1.005` .

As a better solution, the rounding problem can be avoided by using numbers represented in exponential notation:

```
Number(Math.round(1.005+'e2')+'e-2'); // 1.01
```

Cleaner code as suggested by @Kon, and the original author:

```
Number(Math.round(parseFloat(value + 'e' + decimalPlaces)) + 'e-' + decimalPlaces)
```

Credit: Rounding Decimals in JavaScript

edited Apr 26 at 17:18

answered Aug 24 '15 at 9:30

razu
1,042 9 14

13 Amazingly, everyone else in here failed to see that `toFixed` has that rounding problem. Your answer should be the accepted one. – Armfoot Sep 25 '15 at 10:46

1 Just tested to see if this was still the case. you are still correct for Chrome Version 59.0.3071.115 Your solution will round up for 1.005 (1.01) and round down for 1.00499999 (1.00) – Artistan Jul 19 '17 at 13:26

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

surround Number in parseFloat is its returning string – user889030 Sep 26 '17 at 13:55

This is a solid answer. A couple of improvements I can think of: (1) VS Code (TypeScript file) doesn't like Math.round() passing in a string. (2) Make number of decimal places dynamic (not hard-coded to 2). (3) toFixed() seems unnecessary. So what I came up with is `Number(Math.round(parseFloat(value + 'e' + decimalPlaces)) + 'e-' + decimalPlaces)` – Kon Apr 25 at 19:12

0

here is another solution to round only using floor, meaning, making sure calculated amount won't be bigger than the original amount (sometimes needed for transactions):

```
Math.floor(num* 100 )/100;
```

edited Apr 22 at 7:57
RopAli Munshi
884 2 7 23

answered Dec 9 '18 at 12:51
Naty
437 4 6

1

For modern browsers, use `toLocaleString` :

```
var num = 1.345;
num.toLocaleString(undefined, { maximumFractionDigits: 2, minimumFractionDigits: 2 });
```

Specify a locale tag as first parameter to control the decimal separator. For a dot, use for example English U.S. locale:

```
num.toLocaleString("en-US", { maximumFractionDigits: 2, minimumFractionDigits: 2 });
```

which gives:

> 1.35

Most countries in Europe use a comma as decimal separator, so if you for example use Swedish/Sweden locale:

```
num.toLocaleString("sv-SE", { maximumFractionDigits: 2, minimumFractionDigits: 2 });
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

1,35

answered Nov 19 '18 at 15:43

holmis83
10.3k 3 45 61

Not sure why this is downvoted, it seemed to work well for me. Please add a comment if you intend to downvote if something is wrong with this! – John Culviner Jun 10 at 16:02

266

```
Number(1).toFixed(2);         // 1.00
Number(1.341).toFixed(2);     // 1.34
Number(1.345).toFixed(2);     // 1.34 NOTE: See andy's comment below.
Number(1.3450001).toFixed(2); // 1.35
```

Show code snippet

edited Oct 16 '18 at 6:54

Florian
819 10 28

answered Nov 8 '12 at 16:06

Abel ANEIROS
3,455 1 17 17

1 Last line is false. I tried this in Chrome console and ``Number(1.365).toFixed(2) returns "1.37" – Andre Jul 16 '13 at 14:37

1 @Andre, Chrome 29.0.1547.57 gives me `1.34` for expression `Number(1.345).toFixed(2)` . – Drew Noakes Aug 22 '13 at 22:29

1 toFixed *does* do rounding, which you can see on almost every test number. – andy Nov 8 '13 at 18:55

33 Accidentally submitted that last comment before finishing.. `1.345` is an example of a number that can't be stored exactly in floating point, so I think the reason that it doesn't round as you expect, is that it's actually stored as a number slightly less than 1.345 and it rounds down. If you test instead with `(1.34500001).toFixed(2)` then you see it correctly rounds up to `1.35` – andy Nov 8 '13 at 19:02

I had to decide between the parseFloat() and Number() conversions before I could make toFixed() call. Here's an example of a number formatting post-capturing user input.

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```
<input type="number" class="dec-number" min="0" step="0.01" />
```

Event handler:

```
$('.dec-number').on('change', function () {
     const value = $(this).val();
     $(this).val(value.toFixed(2));
});
```

The above code will result in TypeError exception. Note that although the html input type is "number", the user input is actually a "string" data type. However, toFixed() function may only be invoked on an object that is a Number.

My final code would look as follows:

```
$('.dec-number').on('change', function () {
     const value = Number($(this).val());
     $(this).val(value.toFixed(2));
});
```

The reason I favor to cast with Number() vs. parseFloat() is because I don't have to perform an extra validation neither for an empty input string, nor NaN value. The Number() function would automatically handle an empty string and covert it to zero.

answered Oct 12 '18 at 19:18

vitek
11 2

0

`parseInt(number * 100) / 100;` worked for me.

answered Sep 17 '18 at 9:38

Be Wake Pandey
150 2 9

That shouldn't sole the issue with 1.00 right – Mathijs Segers Nov 26 '18 at 13:15

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

-3

This is how I solve my problem:

```
parseFloat(parseFloat(floatString).toFixed(2));
```

edited Aug 6 '18 at 15:28
Liam
17k 16 80 133

answered Jul 15 '17 at 13:54
Alexei
1 1

2

```
function number_format(string,decimals=2,decimal=',',thousands='.',pre='R$ ',pos='
Reais'){
  var numbers = string.toString().match(/\d+/g).join([]);
  numbers = numbers.padStart(decimals+1, "0");
  var splitNumbers = numbers.split("").reverse();
  var mask = '';
  splitNumbers.forEach(function(d,i){
    if (i == decimals) { mask = decimal + mask; }
    if (i>(decimals+1) && ((i-2)%(decimals+1))==0) { mask = thousands + mask; }
    mask = d + mask;
  });
  return pre + mask + pos;
}
var element = document.getElementById("format");
var money= number_format("10987654321",2,',','.');
element.innerHTML = money;


#format{
display:inline-block;
padding:10px;
border:1px solid #ddd;
background:#f5f5f5;
}


<div id='format'>Test 123456789</div>
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

answered Oct 20 '17 at 18:26

Leonardo Filipe
311 1 3

12

A much more generic solution for rounding to N places

```
function roundN(num,n){
  return parseFloat(Math.round(num * Math.pow(10, n)) /Math.pow(10,n)).toFixed(n);
}


console.log(roundN(1,2))
console.log(roundN(1.34,2))
console.log(roundN(1.35,2))
console.log(roundN(1.344,2))
console.log(roundN(1.345,2))
console.log(roundN(1.344,3))
console.log(roundN(1.345,3))
console.log(roundN(1.3444,3))
console.log(roundN(1.3455,3))
```

Output

```
1.00
1.34
1.35
1.34
1.35
1.344
1.345
1.344
1.346
```

answered Sep 6 '17 at 12:31

PirateApp
2,133 2 21 33

1 This alswer fails with certain numbers close to 0, `roundN(-3.4028230607370965e+38,2)` returns `"-3.4028230607370965e+38"` instead of the expected 0.00 Ferrybig Apr 17 at 9:38

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

1 Note that the before mentioned number should not show zero, but a huge number instead, this was just an error in my brain. But it still doesn't work, since it still doesn't show 2 decimals – Ferrybig Apr 17 at 11:51

3

Where specific formatting is required, you should write your own routine or use a library function that does what you need. The basic ECMAScript functionality is usually insufficient for displaying formatted numbers.

~~A thorough explanation of rounding and formatting is here: http://www.merlyn.demon.co.uk/js-round.htm#RiJ~~

As a general rule, rounding and formatting should only be peformed as a last step before output. Doing so earlier may introduce unexpectedly large errors and destroy the formatting.

edited Aug 15 '17 at 3:55

answered May 26 '11 at 5:58

RobG
102k 19 115 152

There are times when I shake my head at my own choice to get so deeply involved of late in this whole JS/Ecma platform. :("Where specific formatting is required, you should write your own routine or use a library function that does what you need. The basic ECMAScript functionality is usually insufficient for displaying formatted numbers." What an asinine statement - not that you made it, but because that fact exists. Just sad. Get on the ball, Javascript! :) – ChrisH Dec 21 '17 at 1:41

0

Extend **Math** object with **precision** method

```
Object.defineProperty(Math, 'precision',{
   value: function (value,precision,type){
             var v = parseFloat(value),
                 p = Math.max(precision,0)||0,
                 t = type||'round';
              return (Math[t](v*Math.pow(10,p))/Math.pow(10,p)).toFixed(p);
          }
    });

console.log(
    Math.precision(3.1,3), // round 3 digits
    Math.precision(0.12345,2,'ceil'), // ceil 2 digits
    Math.precision(1.1) // integer part
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

Run code snippet Copy snippet to answer Expand snippet

edited Apr 9 '17 at 9:56

answered Feb 1 '17 at 7:37

bortunac
2,698 18 17

7

Is this what you mean?

```
function showAsFloat(num, n){
      return !isNaN(+num) ? (+num).toFixed(n || 2) : num;
}

document.querySelector('#result').textContent =
    [
     'command                      | result',
     '---------------------------------------------',
     'showAsFloat(1);              | ' + showAsFloat(1),
     'showAsFloat(1.314);          | ' + showAsFloat(1.314),
     'showAsFloat(\'notanumber\')    | ' + showAsFloat('notanumber'),
     'showAsFloat(\'23.44567\', 3)   | ' + showAsFloat('23.44567', 3),
     'showAsFloat(2456198, 5)      | ' + showAsFloat('2456198', 5),
     'showAsFloat(0);              | ' + showAsFloat(0)
    ].join('\n');


<pre id="result"></pre>
```

Run code snippet Copy snippet to answer Expand snippet

edited Oct 17 '16 at 7:55

answered May 26 '11 at 6:17

KooiInc
83.6k 23 111 146

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

0

I do like:

```
var num = 12.749;
parseFloat((Math.round(num * 100) / 100).toFixed(2)); // 123.75
```

Round the number with 2 decimal points, then make sure to parse it with `parseFloat()` to return Number, not String unless you don't care if it is String or Number.

answered Oct 16 '16 at 3:38
Yuichi
347 6 19

I guess if the purpose was to display with 2 decimals precision, parsing the float will mess with that. E.g. `parseFloat("1.00") // 1` – Stijn de Witt Aug 1 '17 at 20:21

1

```
var quantity = 12;

var import1 = 12.55;

var total = quantity * import1;

var answer = parseFloat(total).toFixed(2);

document.write(answer);
```

edited Jul 15 '16 at 19:36
Allan Pereira
2,427 4 16 26

answered Sep 18 '14 at 11:46
Darshak Shekhda
595 4 7

Convert a number into a string, keeping only two decimals:

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```
var num = 5.56789;
var n = num.toFixed(2);
```

The result of n will be:

```
5.57
```

answered Jun 21 '16 at 5:58

Behnam Mohammadi
7,843 1 33 31

11

For the most accurate rounding, create this function:

```
function round(value, decimals) {
    return Number(Math.round(value +'e'+ decimals) +'e-'+ decimals).toFixed(decimals);
}
```

and use it to round to 2 decimal places:

```
console.log("seeked to " + round(1.005, 2));
> 1.01
```

Thanks to Razu, this article, and MDN's Math.round reference.

edited May 23 '17 at 12:18

Community ♦
1 1

answered Jan 14 '16 at 18:24

Nate
1,636 2 14 27

1 what about 1.0049999999999998934 ? – 4esn0k Oct 2 '16 at 6:40

1 It does not go beyond 2 d.p – Stephen Adelakun Oct 4 '16 at 13:27

1 jsfiddle.net/Artistan/qq895bnp tested, this is the only consistent method I have seen. Thanks. – Artistan Jul 19 '17 at 17:20

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

9

If you're already using jQuery, you could look at using the jQuery Number Format plugin.

The plugin can return formatted numbers as a string, you can set decimal, and thousands separators, and you can choose the number of decimals to show.

```
$.number( 123, 2 ); // Returns '123.00'
```

You can also get jQuery Number Format from GitHub.

edited Jan 11 '16 at 1:32

answered Nov 8 '12 at 23:47

Sam Sehnert
2,585 1 16 23

9 It is overkill to use a plugin "just to have fixed length decimal part". – Lashae Sep 4 '13 at 14:04

9 @Lashae, sure, if thats all you want to do. I posted this in case the OP or anyone else wanted the extra functionality that the plugin provides as well. – Sam Sehnert Sep 9 '13 at 1:31

if the poster of the question had added the jQuery tag of course ;) – fullstacklife Mar 30 '15 at 0:56

Dead link!!!!!! – P i Dec 31 '15 at 22:51

@Pi, thanks, fixed. – Sam Sehnert Jan 11 '16 at 1:32

9

```
var number = 123456.789;

console.log(new Intl.NumberFormat('en-IN', { maximumFractionDigits: 2
}).format(number));
```

https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_Objects/NumberFormat

answered Nov 28 '15 at 9:09

zloctb
5,377 3 47 59

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

3

Here's also a generic function that can format to any number of decimal places:

```
function numberFormat(val, decimalPlaces) {

    var multiplier = Math.pow(10, decimalPlaces);
    return (Math.round(val * multiplier) / multiplier).toFixed(decimalPlaces);
}
```

answered Sep 18 '15 at 12:50

Minas Mina
1,270 1 13 24

2

```
function currencyFormat (num) {
    return "$" + num.toFixed(2).replace(/(\d)(?=(\d{3})+(?!\d))/g, "$1,")
}

console.info(currencyFormat(2665));   // $2,665.00
console.info(currencyFormat(102665)); // $102,665.00
```

answered Aug 19 '15 at 4:49

Ar No
691 9 16

-2

```
(num + "").replace(/^([0-9]*)(\.[0-9]{1,2})?.*$/,"$1$2")
```

answered Jul 8 '15 at 10:01

Crow Soup
7 2

2 could you explain – depperm Jul 8 '15 at 13:44

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

12

Simplest answer:

```
var num = 1.2353453;
num.toFixed(2); // 1.24
```

Example: http://jsfiddle.net/E2XU7/

answered Apr 8 '13 at 18:37

macio.Jun
7,338 1 39 35

7 Well, `toFixed` was already suggested in stackoverflow.com/a/13292833/218196. What additional information does your question provide? – Felix Kling Apr 8 '13 at 18:42

that answer does not include round functionality, my answer includes tho. – macio.Jun Apr 9 '13 at 20:00

7 Uh? It's exactly the same answer. Calling `toFixed` on a number. – Felix Kling Apr 9 '13 at 20:19

1 Correct, same function, but the result of that answer is misleading, I just rectified it by expressing the round functionality. – macio.Jun Apr 10 '13 at 2:28

The question states : "...rounding where applicable". Your answer does not involve rounding. – Chris Jun 20 '13 at 8:40

18

```
var num = new Number(14.12);
console.log(num.toPrecision(2));//outputs 14
console.log(num.toPrecision(3));//outputs 14.1
console.log(num.toPrecision(4));//outputs 14.12
console.log(num.toPrecision(5));//outputs 14.120
```

answered Sep 7 '12 at 13:16

Tiberiu Petcu
628 5 9

That gives unexpected results, if your number can be 1.4, and 23.654, and 0, what precision would you take? – shinzou Oct 15 '16 at 15:36

2 Note the OP is asking for **"rounding where applicable"**. `toPrecision` only formats the number to a specific number of decimal places, simply leaving out redundant places, but **not rounding them**. This could be very useful too of course, but it's important to understand the difference. – Boaz Nov 9 '16

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

1 If you want to strip trailing zeros, cast it to a Number or Float after using `toFixed` : `const formattedVal = Number(val.toFixed(2));` Do not use `toPrecision` , as it counts the non-decimal numbers when using the precision param. – James L. Nov 27 '17 at 17:26

1

You are not giving us the whole picture.

`javascript:alert(parseFloat(1).toFixed(2))` shows 1.00 in my browsers when I paste it int0 the location bar. However if you do something to it afterwards, it will revert.

```
var num = 2
document.getElementById('spanId').innerHTML=(parseFloat(num).toFixed(2)-1)


shows 1 and not 1.00
```

edited May 26 '11 at 5:50

answered May 26 '11 at 5:39

mplungjan
93.4k 22 132 190

Works in FF 50, Chrome 49 and IE 8 – Florian Straub Jan 7 '17 at 11:21

4

Are you looking for floor?

```
var num = 1.42482;
var num2 = 1;
var fnum = Math.floor(num).toFixed(2);
var fnum2 = Math.floor(num2).toFixed(2);
alert(fnum + " and " + fnum2); //both values will be 1.00
```

answered May 26 '11 at 5:26

samwise
276 3 14

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.