Globalizing Modern Apps

by Rafael Xavier / @rxaviers / gh:rxaviers

What is it about?

ICU

International Components for Unicode

"format numbers javascript"

How to format numbers using javascript?

16

6

I want to format numbers using javascript.

For example:

```
10      => 10.00
100     => 100.00
1000    => 1,000.00
10000   => 10,000.00
100000 => 100,000.00
```

javascript

share edit flag

edited Apr 23 at 15:24

Nicolae Surdu

2,774 • 3 • 33 • 61


```
function formatNumber(number)
{
    var number = number.toFixed(2) + '';
    var x = number.split('.');
    var x1 = x[0];
    var x2 = x.length > 1 ? '.' + x[1] : '';
    var rgx = /(\d+)(\d{3})/;
    while (rgx.test(x1)) {
        x1 = x1.replace(rgx, '$1' + ',' + '$2');
    }
    return x1 + x2;
}
```

share edit flag

edited Jul 24 '14 at 22:10

Vizath

290 ● 1 ● 7

answe

"format currencies javascript"

How can I format numbers as money in JavaScript?

673

I would like to format a price in JavaScript.
I'd like a function which takes a `float` as an argument and returns a `string` for

```
"$ 2,500.00"
```

What's the best way to do this?

256

javascript formatting currency

share edit flag

edited May 7 at 22:53

community
10 revs, 6

926

You can use:

```
  var profits=2489.8237
  profits.toFixed(3) //returns 2489.824 (round up)
  profits.toFixed(2) //returns 2489.82
  profits.toFixed(7) //returns 2489.8237000 (padding)
```

Then you can add the sign of '$'.

If you require ',' for thousand you can use:

```
Number.prototype.formatMoney = function(c, d, t){
var n = this,
    c = isNaN(c = Math.abs(c)) ? 2 : c,
    d = d == undefined ? "." : d,
    t = t == undefined ? "," : t,
    s = n < 0 ? "-" : "",
    i = parseInt(n = Math.abs(+n || 0).toFixed(c)) + "",
    j = (j = i.length) > 3 ? j % 3 : 0;
   return s + (j ? i.substr(0, j) + t : "") + i.substr(j).replace(/(\d{3})(?=\d)/g, "$1
 };
```

And use it with:

```
(123456789.12345).formatMoney(2, '.', ',');
```

If you're always going to use '.' and ',', you can leave them off your method call, and the method will default them for you.

```
(123456789.12345).formatMoney(2);
```

If your culture has the two symbols flipped (i.e. Europeans), just paste over the following two lines in the `formatMoney` method:

```
    d = d == undefined ? "," : d,
```

41

The new ECMAScript Internationalization API offers a numberformat function.

- http://www.ecma-international.org/ecma-402/1.0/
- https://developer.mozilla.org/en-US/docs/JavaScript/Reference/Global_Objects/Num

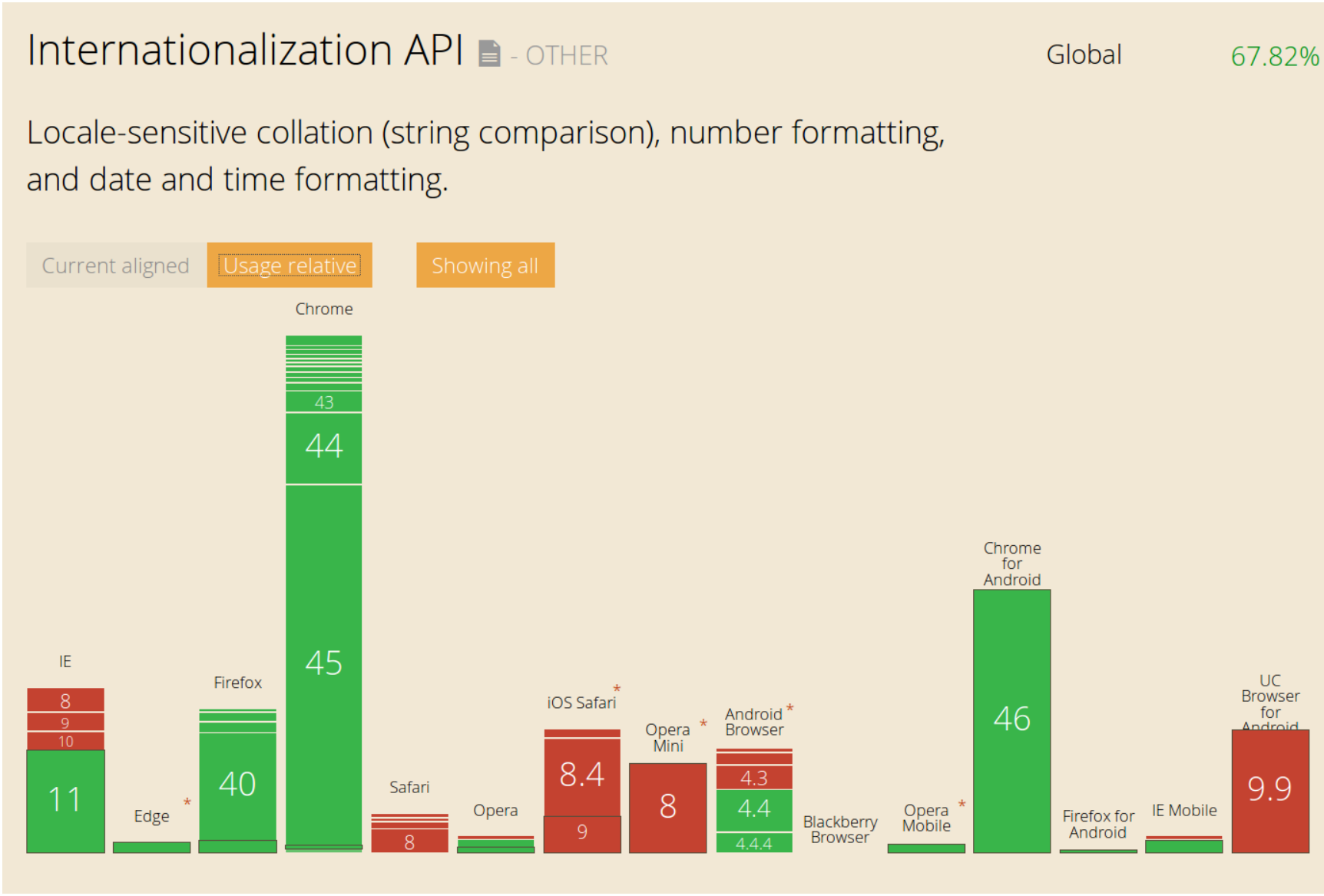
Using that API you do this:

```
// Create our number formatter.
var formatter = new Intl.NumberFormat('en-US', {
  style: 'currency',
  currency: 'USD',
  minimumFractionDigits: 2,
});

alert(formatter.format(349)); /* $349.00 */
```

Intl.NumberFormat()	123,456
Intl.NumberFormat(, {style: 'currency'})	$69,900.00
Intl.DateTimeFormat()	27/10/2015
Intl.Collator()	<localized comparison>

						iOS	Android	Opera Mobile
Intl.NumberFormat	✘	✔	✔	✔	11+	✘	4.4+	✘
Intl.DateTimeFormat	✘	✔	✔	✔	11+	✘	4.4+	✘
Intl.Collator	✘	✔	✔	✔	11+	✘	4.4+	✘



caniuse.com/#search=intl





					iOS	Android	Opera Mobile
Intl.NumberFormat	✘	✔	✔	✔	11+	✘	4.4+	✘
Intl.DateTimeFormat	✘	✔	✔	✔	11+	✘	4.4+	✘
Intl.Collator	✘	✔	✔	✔	11+	✘	4.4+	✘
Pluralization	✘	✘	✘	✘	✘	✘	✘	✘
Message formatting	✘	✘	✘	✘	✘	✘	✘	✘
Relative Time fmt	✘	✘	✘	✘	✘	✘	✘	✘
Unit formatting	✘	✘	✘	✘	✘	✘	✘	✘

								Opera	
					iOS	Android	Mobile	Node.js
Intl.NumberFormat	✘	✔	✔	✔	11+	✘	4.4+	✘	✔ (en)
Intl.DateTimeFormat	✘	✔	✔	✔	11+	✘	4.4+	✘	✔ (en)
Intl.Collator	✘	✔	✔	✔	11+	✘	4.4+	✘	✔ (en)
Pluralization	✘	✘	✘	✘	✘	✘	✘	✘	✘
Message formatting	✘	✘	✘	✘	✘	✘	✘	✘	✘
Relative Time fmt	✘	✘	✘	✘	✘	✘	✘	✘	✘
Unit formatting	✘	✘	✘	✘	✘	✘	✘	✘	✘
Abbreviation	✘	✘	✘	✘	✘	✘	✘	✘	✘
Text segmentation	✘	✘	✘	✘	✘	✘	✘	✘	✘

JavaScript Libraries

JavaScript Libraries

Comparison Grid

github.com/rxaviers/javascript-globalization/

- Always up-to-date (CLDR as a peer dependency)
- Quick development (declarative code)
- Smallest and fastest bundles (optimization for production)

Library Foo

```
.
└── dist
    ├── ar.js
    ├── de.js
    ├── en-GB.js
    ├── en.js
    ├── es.js
    ├── zh.js
    └── zh-TW.js
```

Library Foo

```
.  // library foo at version X
└── dist
    ├── ar.js     // contains CLDR at version Y
    ├── de.js     // contains CLDR at version Y
    ├── en-GB.js  // contains CLDR at version Y
    ├── en.js     // contains CLDR at version Y
    ├── es.js     // contains CLDR at version Y
    ├── zh.js     // contains CLDR at version Y
    └── zh-TW.js  // contains CLDR at version Y
```

My application

```
package.json:
{
    "name": "my-application",
    "dependencies": {
        "globalize": "1.x"
    },
    "peerDependencies": {
        "cldr-data": "28"
    }
}
```

- Always up-to-date (CLDR as a peer dependency)
- Quick development (declarative code)
- Smallest and fastest bundles (optimization for production)

My application

```
// Before we can use Globalize, we need to feed it
// on the appropriate I18n content (Unicode CLDR).
Globalize.load(
        require( "cldr-data/main/en/ca-gregorian" ),
        require( "cldr-data/main/en/dateFields" ),
        require( "cldr-data/main/en/numbers" ),
        require( "cldr-data/supplemental/likelySubtags" ),
        require( "cldr-data/supplemental/timeData" ),
        require( "cldr-data/supplemental/weekData" )
);

// Set the default locale.
Globalize.locale( "en" );

// Use Globalize to format dates
```

My application

```
// Use Globalize to format dates.
Globalize.formatDate( new Date(), { datetime: "medium" } );
```

- Always up-to-date (CLDR as a peer dependency)
- Quick development (declarative code)
- Smallest and fastest bundles (optimization for production)

Currency codes

ADP, AED, AFA, AFN, ALK, ALL, AMD, ANG, AOA, AOK, AON, AOR, ARA, ARL, ARM, ARP, ARS
AZM, AZN, BAD, BAM, BAN, BBD, BDT, BEC, BEF, BEL, BGL, BGM, BGN, BGO, BHD, BIF, BMD
BOP, BOV, BRB, BRC, BRE, BRL, BRN, BRR, BRZ, BSD, BTN, BUK, BWP, BYB, BYR, BZD, CAD
CHW, CLE, CLF, CLP, CNX, CNY, COP, COU, CRC, CSD, CSK, CUC, CUP, CVE, CYP, CZK, DDM
DOP, DZD, ECS, ECV, EEK, EGP, ERN, ESA, ESB, ESP, ETB, EUR, FIM, FJD, FKP, FRF, GBP
GHS, GIP, GMD, GNF, GNS, GQE, GRD, GTQ, GWE, GWP, GYD, HKD, HNL, HRD, HRK, HTG, HUF
ILR, ILS, INR, IQD, IRR, ISJ, ISK, ITL, JMD, JOD, JPY, KES, KGS, KHR, KMF, KPW, KRH
KYD, KZT, LAK, LBP, LKR, LRD, **LSL**, LTL, LTT, LUC, LUF, LUL, LVL, LVR, LYD, MAD, MAF
MGA, MGF, MKD, MKN, MLF, MMK, MNT, MOP, MRO, MTL, MTP, MUR, MVP, MVR, MWK, MXN, MXP
MZM, MZN, NAD, NGN, NIC, NIO, NLG, NOK, NPR, NZD, OMR, PAB, PEI, PEN, PES, PGK, PHP
PTE, PYG, QAR, RHD, **ROL**, RON, RSD, RUB, RUR, RWF, SAR, SBD, SCR, SDD, SDG, SDP, SEK
SKK, SLL, SOS, SRD, SRG, SSP, **STD**, SUR, SVC, SYP, SZL, THB, TJR, TJS, TMM, TMT, TND
TRY, TTD, TWD, TZS, UAH, UAK, UGS, UGX, USD, USN, USS, UYI, UYP, UYU, UZS, VEB, VEF
WST, XAF, XAG, XAU, XBA, XBB, XBC, XBD, XCD, XDR, XEU, XFO, XFU, XOF, XPD, XPF, XPT
XUA, XXX, YDD, YER, YUD, YUM, YUN, YUR, ZAL, ZAR, ZMK, ZMW, ZRN, ZRZ, ZWD, ZWL, ZWR

Currency data for `en`

```
{
    ...
    "UGX": {
      "displayName": "Ugandan Shilling",
      "displayName-count-one": "Ugandan shilling",
      "displayName-count-other": "Ugandan shillings",
      "symbol": "UGX"
    },
    "USD": {
      "displayName": "US Dollar",
      "displayName-count-one": "US dollar",
      "displayName-count-other": "US dollars",
      "symbol": "$",
      "symbol-alt-narrow": "$"
    }
```

Gregorian calendar data

```
"calendars": {
          "gregorian": {
            "months": {
              "format": {
                "abbreviated": {
                  "1": "Jan",
                  "2": "Feb",
                  "3": "Mar",
                  "4": "Apr",
                  "5": "May",
                  "6": "Jun",
                  "7": "Jul",
                  "8": "Aug",
                  "9": "Sep",
                  "10": "Oct"
```

"July 4"

"07/04"

Gregorian calendar translations


```
"Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov" "Dec" "J" "F" "M"
"A" "S" "O" "N" "D" "January" "February" "March" "April" "May" "June" "July" "Augus
"October" "November" "December" "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Se
"Dec" "J" "F" "M" "A" "M" "J" "J" "A" "S" "O" "N" "D" "January" "February" "March"
"June" "July" "August" "September" "October" "November" "December" "Sun" "Mon" "Tue
"Fri" "Sat" "S" "M" "T" "W" "T" "F" "S" "Su" "Mo" "Tu" "We" "Th" "Fr" "Sa" "Sunday"
"Tuesday" "Wednesday" "Thursday" "Friday" "Saturday" "Sun" "Mon" "Tue" "Wed" "Thu"
"M" "T" "W" "T" "F" "S" "Su" "Mo" "Tu" "We" "Th" "Fr" "Sa" "Sunday" "Monday" "Tuesd
"Thursday" "Friday" "Saturday" "Q1" "Q2" "Q3" "Q4" "1" "2" "3" "4" "1st quarter" "2
quarter" "4th quarter" "Q1" "Q2" "Q3" "Q4" "1" "2" "3" "4" "1st quarter" "2nd quart
"4th quarter" "midnight" "AM" "am" "noon" "PM" "pm" "in the morning" "in the aftern
evening" "at night" "mi" "a" "am" "n" "p" "pm" "in the morning" "in the afternoon"
"at night" "midnight" "AM" "am" "noon" "PM" "pm" "in the morning" "in the afternoon
evening" "at night" "midnight" "AM" "am" "noon" "PM" "pm" "in the morning" "in the
the evening" "at night" "midnight" "AM" "am" "noon" "PM" "pm" "in the morning" "in
"in the evening" "at night" "midnight" "AM" "am" "noon" "PM" "pm" "morning" "aftern
"night" "Before Christ" "Before Common Era" "Anno Domini" "Common Era" "BC" "BCE" "
```

```
formatter = Globalize.numberFormatter()
```

10 secs

```
formatter( 3.141592 )
```

1 sec