How to convert decimal to hex in JavaScript?

How do you convert decimal values to their hex equivalent in JavaScript?

javascript hex tostring base number-formatting

edited Mar 23 '12 at 16:45
rzr
1,296 ● 10 ● 19

asked Sep 11 '08 at 22:26
Luke Smith
5,368 ● 5 ● 20 ● 24

2 Just a warning here that is you are starting from a string representation its very easy to lose precision when you turn it into Number as part of converting it to hex. See danvk.org/wp/2012-01-20/…. – studgeek Jun 2 '13 at 5:38

18 Answers

Convert a number to a hexadecimal string with:

```
hexString = yourNumber.toString(16);
```

and reverse the process with:

```
yourNumber = parseInt(hexString, 16);
```

edited Jun 14 '12 at 18:52

answered Sep 11 '08 at 22:28
Prestaul
40.3k ● 8 ● 53 ● 70

11 yourNum is a hex string in this case. E.g. (255).toString(16) == 'ff' && parseInt('ff', 16) == 255 – Prestaul Jul 8 '11 at 19:14

2 You were looking at `Object.toString`. You need `Number.toString()` ... developer.mozilla.org/en/JavaScript/Reference/Global_Objects/… – Prestaul Feb 1 '12 at 21:50

5 @forste, you will not "loose precision" if you convert a javascript Number (that is a Number object, which in ECMA script is a Double) to hex and back using this technique. The question you linked is specifically referencing numbers too large to fit into a Double (hence the string representation in the question). If you've got a Number then this will work. If you have something too big to be a javascript Number object (a Double) then you'll have to find something else. – Prestaul Mar 30 '12 at 5:36

3 @Derek, I have a psychological issue that won't allow me to tolerate unnecessary parentheses... @everyone-else, `yourNumber` is a variable. If you want to use a numeric literal then you'll have to do something like `(45).toString(16)`, but if you are hard-coding a number then please just write it as a hex string yourself... `(45).toString(16)` will always equal `'2d'`, so don't waste cpu cycles to figure that out. – Prestaul Jun 14 '12 at 19:01

2 @Derek, please reread my previous comment where I clarify 1) that `yourNumber` is a variable, not a numeric literal, and 2) why you should never need to type `(16).toString(16)` – Prestaul Jun 14 '12 at 20:41

If you need to handle things like bit fields or 32-bit colors, then you need to deal with signed numbers. The javascript function toString(16) will return a negative hex number which is usually not what you want. This function does some crazy addition to make it a positive number.

```
function decimalToHexString(number)
{
    if (number < 0)
    {
        number = 0xFFFFFFFF + number + 1;
    }

    return number.toString(16).toUpperCase();
}
```

answered Mar 30 '09 at 16:05
Tod
2,085 ● 1 ● 19 ● 21

3 is that two's complement? – Julian Jul 12 '13 at 7:19

This conversion is normally not needed as JavaScript can represent all 32-bit bit fields as unsigned numbers (See Number.MAX_SAFE_INTEGER). For the same reason the conversion to unsigned can be written as: `number = 0x100000000 + number;` – Johannes Matokic Oct 22 '14 at 9:35

A short note on my previous comment: While hex representation should work for numbers up to Number.MAX_SAFE_INTEGER this doesn't hold for bitwise operations (which are often used to create 32-bit colors). The result of bitwise operations is always a signed 32-bit integer. Therefore bitwise results >= 2^31 are negative and 0x100000000 | 0 === 0. – Johannes Matokic Feb 18 at 15:47

The code below will convert the decimal value d to hex. It also allows you to add padding to the hex result. so 0 will become 00 by default.

```
function decimalToHex(d, padding) {
    var hex = Number(d).toString(16);
    padding = typeof (padding) === "undefined" || padding === null ? padding = 2 :
padding;

    while (hex.length < padding) {
        hex = "0" + hex;
    }

    return hex;
}
```

edited Oct 15 '09 at 13:58

answered Sep 11 '08 at 22:29
Luke Smith
5,368 ● 5 ● 20 ● 24

2 This won't properly handle negative values. decimalToHex(-6, 4) would return 00-6. – JonMR Jun 17 '10 at 18:09

It also has problems with floats, but putting in Math.round() fixed that. (+1ed) – Michael Mar 30 '12 at 18:13

With padding:

```
function dec2hex(i) {
   return (i+0x10000).toString(16).substr(-4).toUpperCase();
}
```

answered Jul 13 '11 at 14:29
quartzo
601 ● 6 ● 4

Without the loop :

```
function decimalToHex(d) {
  var hex = Number(d).toString(16);
  hex = "000000".substr(0, 6 - hex.length) + hex;
  return hex;
}

//or "#000000".substr(0, 7 - hex.length) + hex;
//or whatever
//*Thanks to MSDN
```

Also isn't it better not to use loop tests that have to be evaluated eg instead of:

```
for (var i = 0; i < hex.length; i++){}
```

have

```
for (var i = 0, var j = hex.length; i < j; i++){}
```

edited Sep 11 '10 at 3:56

answered Sep 11 '10 at 3:05

mystifeid
196 ● 1 ● 4

```
function toHex(d) {
    return  ("0"+(Number(d).toString(16))).slice(-2).toUpperCase()
}
```

edited Nov 5 '12 at 21:26
andrewsi
9,607 ● 11 ● 21 ● 40

answered Nov 5 '12 at 21:08
Baznr
121 ● 1 ● 2

1 This, or something like it should be the accepted answer. In 99% of the cases, you do not want variable length hex strings! – user239558 Apr 1 '14 at 6:45

For completion, if you want the two's-complement hexadecimal representation of a negative number, you can use the zero-fill-right shift `>>>` operator. For instance:

```
> (-1).toString(16)
"-1"

> ((-2)>>>0).toString(16)
"fffffffe"
```

There is however one limitation: javascript bitwise operators treat their operands as a sequence of 32 bits, that is, you get the 32-bits two's-complement.

edited Dec 16 '13 at 23:43
rlemon
10.7k ● 5 ● 50 ● 83

answered Jun 14 '13 at 10:59
Alberto
1,454 ● 1 ● 17 ● 34

1 Brilliant! but `(-2>>>0).toString(16)` should be `fffffffe` – hiroshi Dec 16 '13 at 13:33

@hiroshi you're absolutely right. Thanks for the heads-up! – Alberto Dec 16 '13 at 23:46

Combining some of these good ideas for an rgb to hex function (add the # elsewhere for html/css):

```
function rgb2hex(r,g,b) {
    if (g !== undefined)
        return Number(0x1000000 + r*0x10000 + g*0x100 + b).toString(16).substring(1);
    else
        return Number(0x1000000 + r[0]*0x10000 + r[1]*0x100 +
r[2]).toString(16).substring(1);
}
```

edited Nov 15 '12 at 13:11
WEFX
3,609 ● 4 ● 38 ● 66

answered Nov 15 '12 at 12:44
Keith Mashinter
79 ● 1 ● 1

Thanks for this! I ment to drop a comment a long time ago. It was the key for my answer. stackoverflow.com/questions/5560248/… – Pimp Trizkit Feb 4 '14 at 0:51

```
var number = 3200;
var hexString = number.toString(16);
```

The 16 is the radix and there are 16 values in a hexadecimal number :-)

edited Aug 28 '13 at 1:46
Peter Hansen
9,489 ● 29 ● 51

answered Sep 11 '08 at 22:32
Danny Wilson
301 ● 3 ● 7

If you want to convert a number to a hex representation of an RGBA color value, I've found this to be the most useful combination of several tips from here:

```
        function toHexString(n) {
            if(n < 0) {
                n = 0xFFFFFFFF + n + 1;
            }

            return "0x" + ("00000000" + n.toString(16).toUpperCase()).substr(-8);
        }
```

answered Jun 13 '12 at 9:48

korona
1,277 ● 1 ● 9 ● 24

AFAIK comment 57807 is wrong and should be something like: **var hex = Number(d).toString(16);** instead of **var hex = parseInt(d, 16);**

```
function decimalToHex(d, padding) {
    var hex = Number(d).toString(16);
    padding = typeof (padding) === "undefined" || padding === null ? padding = 2 :
padding;

    while (hex.length < padding) {
        hex = "0" + hex;
    }

    return hex;
}
```

answered Sep 18 '09 at 20:07 community wiki

I've updated my reply with this fix :) – Luke Smith Oct 15 '09 at 13:58

```
function dec2hex(i)
{
  var result = "0000";
  if      (i >= 0    && i <= 15)    { result = "000" + i.toString(16); }
  else if (i >= 16   && i <= 255)   { result = "00"  + i.toString(16); }
  else if (i >= 256  && i <= 4095)  { result = "0"   + i.toString(16); }
  else if (i >= 4096 && i <= 65535) { result =         i.toString(16); }
  return result
}
```

answered May 30 '09 at 18:47

Fred

+1 thanks! When working with CSS, the toString(16) is important so you get results like FF0000 – Tyler Egeto Feb 21 '11 at 6:54

6 when working with css (or svg, which accepts css-style color specifications) you can sidestep the entire issue by writing `color: rgb(r,g,b)` where r g and b are decimal numbers. – telent Feb 27 '11 at 11:09

Should be: `function decimalToHexString(i) { var result = "00"; if (i >= 0 && i <= 15) { result += "000" + i.toString(16); } else if (i >= 16 && i <= 255) { result += "00" + i.toString(16); } else if (i >= 256 && i <= 4095) { result += "0" + i.toString(16); } else if (i >= 4096 && i <= 65535) { result += i.toString(16); } return result }` – Aykut Çevik May 20 '11 at 9:04

And if the number is negative?

Here is my version.

```
function hexdec (hex_string) {
    hex_string=((hex_string.charAt(1)!='X' && hex_string.charAt(1)!='x')?
hex_string='0X'+hex_string : hex_string);
    hex_string=(hex_string.charAt(2)<8 ? hex_string =hex_string-0x00000000 :
hex_string=hex_string-0xFFFFFFFF-1);
    return parseInt(hex_string, 10);
}
```

answered Oct 21 '12 at 7:32

Eliarh
31 ● 1

Constrained/Padded to a set number of characters:

```
function decimalToHex(decimal, chars) {
    return (decimal + Math.pow(16, chars)).toString(16).slice(-chars).toUpperCase();
}
```

answered Jan 27 '12 at 13:35
Adamarla
95 ● 10

As the accepted answer states, the easiest way to convert from dec to hex is `var hex = dec.toString(16)`. However, you may prefer to add a string conversion, as it ensures that string representations like `"12".toString(16)` work correctly.

```
// avoids a hard to track down bug by returning `c` instead of `12`
(+"12").toString(16);
```

To reverse the process you may also use the solution below, as it is even shorter.

```
var dec = +("0x" + hex);
```

It seems to be slower in Google Chrome and Firefox, but is significantly faster in Opera. See http://jsperf.com/hex-to-dec.

answered Jun 10 '13 at 11:57
R D
21 ● 3

I'm doing conversion to hex string in a pretty large loop, so I tried several techniques in order to find the fastest one. My requirements were to have a fixed-length string as a result, and encode negative values properly (-1 => ff..f).

Simple `.toString(16)` didn't work for me since I needed negative values to be properly encoded. The following code is the quickest I've tested so far on 1-2 byte values (note that `symbols` defines the number of output symbols you want to get, that is for 4-byte integer it should be equal to 8):

```
var hex = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e',
'f'];
function getHexRepresentation(num, symbols) {
    var result = '';
    while (symbols--) {
        result = hex[num & 0xF] + result;
        num >>= 4;
    }
    return result;
}
```

It performs faster than `.toString(16)` on 1-2 byte numbers and slower on larger numbers (when `symbols` >= 6), but still should outperform methods that encode negative values properly.

answered Dec 13 '12 at 17:35
almaz
130 ● 6

The accepted answer did not take into account single digit returned hex codes. This is easily adjusted by:

```
   function numHex(s)
   {
      var a = s.toString(16);
      if( (a.length % 2) > 0 ){ a = "0" + a; }
      return a;
   }
```

and

```
   function strHex(s)
   {
      var a = "";
      for( var i=0; i<s.length; i++ ){
         a = a + numHex( s.charCodeAt(i) );
         }

      return a;
   }
```

I believe the above answers have been posted numerous times by others in one form or another. I wrap these in a toHex() function like so:

```
   function toHex(s)
   {
    var re = new RegExp( /^\s*(\+|-)?((\d+(\.\d+)?)|(\.\d+))\s*$/ );

    if( re.test(s) ){ return '#' + strHex( s.toString() ); }
        else { return 'A' + strHex( s ); }
}
```

Note that the numeric regular expression came from 10+ Useful JavaScript Regular Expression Functions to improve your web applications efficiency.

Update: After testing this thing several times I found an error (double quotes in the RegExp) so I fixed that. HOWEVER! After quite a bit of testing and having read the post by almaz - I realized I could not get negative numbers to work. Further - I did some reading up on this and since all Javascript numbers are stored as 64 bit words no matter what - I tried modifying the numHex code to get the 64 bit word. But it turns out you can not do that. If you put "3.14159265" AS A NUMBER into a variable - all you will be able to get is the "3" because the fractional portion is only accessible by multiplying the number by ten(IE:10.0) repeatedly. Or to put that another way - the HEX value of 0xf causes the FLOATING POINT value to be translated into an INTEGER before it is ANDed which removes everything behind the period. Rather than taking the value as a whole (ie: 3.14159265) and ANDing the FLOATING POINT value against the 0xf value. So the best thing to do, in this case, is to convert the 3.14159265 into a STRING and then just convert the string. Because of the above, it also makes it easy to convert negative numbers because the minus sign just becomes 0x26 on the front of the value. So what I did was on determining that the variable contains a number - just convert it to a string and convert the string. What this means to everyone is that on the server side you will need to unhex the incoming string and then to determine the incoming information is numeric. You can do that easily by just adding a "#" to the front of numbers and "A" to the front of a character string coming back. See the toHex() function.

Have fun!

edited Nov 6 '14 at 20:03

answered Nov 6 '14 at 16:21
Mark Manning
59 ● 5

To sum it all up;

```
function toHex(i, pad) {

  if (typeof(pad) === 'undefined' || pad === null) {
    pad = 2;
  }

  var strToParse = i.toString(16);

  while (strToParse.length < pad) {
    strToParse = "0" + strToParse;
  }

  var finalVal =  parseInt(strToParse, 16);

  if ( finalVal < 0 ) {
    finalVal = 0xFFFFFFFF + finalVal + 1;
  }

  return finalVal;
}
```

However, if you don't need to convert it back to an integer at the end (i.e. for colors), then just making sure the values aren't negative should suffice.

answered Dec 10 '13 at 0:14
strawn_04
83 ● 1 ● 5
