$('selector').autoNumeric({options});

meta="{aSep: '.', aDec: ',', aSign: '€ ', vMax: '99999.987', mRound: 'D'}"

€ 1 234 567,89
1,234,567.89 USD
¥ 1,234,567.89

autoNumeric() is a jQuery plugin that formats input fields with numeric strings, supporting most International numeric formats and currency symbols. It automatically places the thousand separator and currency symbol as you type. We have provided two public functions to format and or remove a formats.

Support for International currency formats has never been this easy, flexible and powerful.

Download: autoNumeric-1.7.4-B.js - 'heavy on comments' Donations are appreciated Donate

Authors:

- Bob Knothe - bob{@}decorplanit.com (remove brackets)
- Sokolov Yura - aka funny_falcon https://github.com/funny-falcon/auto_numeric_js

I would like to take this opportunity to personally thank Sokolov Yura. Without his expertise and unselfish commitment this update would not have been possible. If anyone is looking for a talented coder I would gladly recommend him - *Bob*

Key features:

- Compatible with jQuery-1.5.0 and higher.
- Supports multiple international numeric formats including those used in most of Europe, North and South America, Asia and India (lakhs). The varied formats are controlled by metadata that is placed in the input tag. Multiple formats can reside on the same page. Please see the metadata code generator below.
- Most currency symbols are supported and can be placed as a prefix or suffix in the input field.
- Supports nine different rounding methods (Round-Half-Up Symmetric & Asymmetric, Round-Half-Down Symmetrical & Asymmetric, Round-Half-Even "Bankers Rounding", Round Up, Round Down, Round to Ceiling & Round to Floor).
- NEW - Precision control of minimum & maximum values to the n^{th} decimal places!
- NEW - The number of decimal places and negative sign are now set via the min/max values. Also positive and negative values are easily toggled from any caret position within the input field.
- NEW - Public functions $('selector').autoNumericGet({options}) & $('selector').autoNumericSet(value, {options}) to strip and format values that use jQuery's selector format.
- NEW - Callback function that is very powerful.
- NEW - Options for input behavior including, padding, cell to be empty or always display the currency sign or zero.
- Pasting is supported as long as the values being pasted conform to the format set for that target field.
- There may be some compatibility issues with versions 1.5.0 to 1.6.2. This version is **NOT** compatible with versions prior to 1.5.0

Implementing:

First - include the jQuery.js, autoNumeric-1.7.0.js javascript files in the header. If you are going to use the options also add the metadata plugin.

```
<SCRIPT src="jquery-1.5.min.js" type=text/javascript></SCRIPT>

<SCRIPT src="jquery.metadata.js" type=text/javascript></SCRIPT> <!--when changing defaults-->

<SCRIPT src="autoNumeric-1.7.4.js" type=text/javascript></SCRIPT>highlight: html
```

Selector - is any valid jQuery selector

options - can be passed here but are not required

```
// example uses the selector "input" with the class "auto" & no options passed
jQuery(function($) {
  $('input.auto').autoNumeric();
});

// example uses the selector "input" with the class "auto" & with options passed
// See details below on allowed options
jQuery(function($) {
  $('input.auto').autoNumeric({aSep: '.', aDec: ','});
});

// input tag for the above
<input id="some ID" value="" type="text" class="auto" size="25" />highlight: javascript
```

The example below demonstrates the defaults in the first column. The second and third columns show the Get 'strip' and Set 'format' functions respectively - more on this later.

$('selector').autoNumeric({options}) $('selector').autoNumericGet() $('selector').autoNumericSet(value)

Demo Reset form!

Behavior notes: when entering numbers to the left of the decimal separator, the characters entered fills to the left of the insertion point. When the maximum/minimum values are reached additional characters can not be entered. When typing to the right of the decimal separator the caharters also fill to the left with the following exception: if the maximum number of decimal places has been reached and there are numeric characters to the right of the caret postion, any number entered will over write the value to the right so long as the maximum value has not been exceeded.

Defaults and options: There are 11 options that can be used in the metadata code if the defaults do not meet your requirements. I will go through each option (key) and the values that can be changed to obtain the desired format.

Below is an easy to use metadata code generator that should handle most numeric / currency formats. Checking on the options below will generate the metadata code string for the desired format. This can be copied and pasted into your targeted input field.

Note: these are case sensitive and deviating from the suggested options may produce unexpected results.

aSep - controls the thousand separator (note - the thousand & decimal separators can not be the same)

- aSep: ',' - comma (**default**)
- aSep: '\'' - apostrophe (note: the apostrophe is escaped)
- aSep: '.' - period
- aSep: ' ' - space
- aSep: '' - none

dGroup - controls the digital grouping - the placement of the thousand separator

- dGroup: '3' - produces 333,333,333 (**default**)
- dGroup: '2' - produces 22,22,22,333 (India's lakhs format)
- dGroup: '4' - produces 4,4444,4444 used in some Asian country's

aDec - controls the decimal (note - the thousand & decimal separators can not be the same)

- aDec: '.' - period (**default**)
- aDec: ',' - comma

altDec - This was developed to accommodate different keyboard layouts. altDec allows you to declare an alternative key to enter the decimal separator assigned in aDec. Word of caution - use with discretion because it has the potential of being very confusing to your users.

Pl t th t th f ll t th i d ill t th d i l t h th

- ( ) [ ] - limited to one non-numeric character

aSign - desired currency symbol (examples: € or EUR). Note: other symbols can be used, such as %, °C, °F, km/h & MPH the possibilties are endless.

Note: the currency symbol **can not** contain an apostrophe, comma or numeric character. NEW - currency symbols with a full stop "period" are now supported. Example "Rs."

Note: Arabic characters are problematic because of the right to left entry - help on modifying the plugin to handle these characters would be appreciated.

Note: There are several currency symbols that are not supported by common fonts (Arial & Tahoma) - I have put together a table you may find useful (Click here to view) of the symbols that are supported by the aforementioned fonts.

Note: Quote marks are automatically added. Spacing between the currency symbol and numeric characters are set using the aSign option. Example no spacing '$' and with spacing '$ ' or ' $' depending on the location (prefix or suffix) of the symbol.

- (•) aSign: '' - none (**default**)
- ( ) [ ] - select option then enter the desired symbol. Click here to view common symbols

pSign - controls the placement of the currency symbol.

- (•) pSign: 'p' - prefix to the left (**default**)
- ( ) pSign: 's' - suffix to the right

Notes on the minimum (vMin) / maximum (vMax) values:

- The maximum value should always be greater than the minimum value - sorry I had to list this rule!
- If the minimum / maximum values are both positive or negative you MUST have a default value assigned to the field that is greater than or equal to the minimum value and less than or equal to the maximum value.
- The number of decimal places is determined from the minimum / maximum values - whichever has value has the greater number of decimal places.
- The values can contain only numeric characters and one decimal (full stop) character. This metadata code generator automatically places quote marks on the value. If you manually write the metadata code these quote marks are required.
- The limits of the code generator are 20 significant digits and 10 decimal places - if your requirements exceed these limits you will need to manually adjust the metadata values.

vMin - Enter the minimum value allowed. Allowed values can be whole numbers, floating point, positive, zero or negative.

- (•) vMin: '0.00' (**default**)
- ( ) [ ] select the option then enter the desired value to the left.

vMax - Enter the maximum value allowed. Allowed values can be whole numbers, floating point, positive, zero or negative.

- (•) vMax: '999999999.99' (**default**)
- ( ) [ ] select the option then enter the desired value to the left.

mDec - Only needed if you want to override the number of decimal places that are determined from the vMin & vMax values.

- (•) mDec: null decimal places set via vMin & vMax values (**default method**)
- ( ) [ ] enter the number of decimal places - this will over ides values set by vMin & vMax

mRound - controls the rounding method. To test the various rounding methods please see below. For in depth details on rounding methods visit the DIY Calculator site.

- (•) mRound: 'S' - Round-Half-Up Symmetric (**default**)
- ( ) mRound: 'A' - Round-Half-Up Asymmetric

- mRound: 'U' - Round Up "Round-Away-From-Zero"
- mRound: 'D' - Round Down "Round-Toward-Zero" - same as truncate
- mRound: 'C' - Round to Ceiling "Toward Positive Infinity"
- mRound: 'F' - Round to Floor "Toward Negative Infinity"

aPad - controls padding of the decimal places.

- aPad: true - always pads the decimal with zeros (**default**)
- aPad: false - no padding - rounding occurs when the decimal length exceeds the decimal places

wEmpty - controls input display behavior.

- wEmpty: 'empty' - allows input to be empty (no value) (**default**)
- wEmpty: 'zero' - input field will have at least a zero value
- wEmpty: 'sign' - the currency symbol is always present

Metadata code if needed will appear in Red - copy and paste in the input field tag as described above.

Reset defaults

Various samples: The table below demonstrates the flexibility of the autoNumeric plugin and it's ability to format to different international standards using the above defaults and options.

The 1st column allows the user to enter the values. Multiple formats are demonstrated.

The 2nd column shows the non-formatted numeric values from the 1st column via the Get function.

The 3rd column shows the value from the second column formatted to another standard via the set function.

$('selector').autoNumeric({options})	$('selector').autoNumericGet()	$('selector').autoNumericSet(value)
{aPad: false}	Get results	no metadata
{aSep: ' ', aDec: ',', aSign: '€ '}		{aSign: ' USD', pSign: 's'}
{aSign: ' ¥ ', vMin: '-19.29'}		{aSign: ' 元', pSign: 's'}
{aSign: 'Rs.', dGroup: 2}		{aSign: '£'}
{aSep: '\'', aSign: ' CHF', pSign: 's'}		{aSign: 'R$'}
{vMin: '-9999.999', vMax: '9999.999'}		{aSign: 'U$A '}
{vMin: '9.99'}"		{aSign: 'AU$ '}
9.99		
{aSign: 'руб. ', wEmpty: 'sign'}		{aSign: 'U$A '}
руб.		
{aSign: ' %', pSign: 's', vMax: '100.00'}		no metadata

Reset form!

Public Functions: the need to format and remove formats for mathematical operations is a must. So two public functions were created that can be called from another script that easily handle these duties. For users of version 1.6.2 and eariler these are much easier to use then in previous versions. Please note: $.autoNumeric.Strip(id, {options}); & $.autoNumeric.Format(id, value, {options}); still function but it is easy to see that the new versions are superior and they conform to jQuery's convention. The use of setting options via the "autoNumericGet()" & "autoNumericSet()" functions has been depreciated as of version 1.7.2.

- Selector - any valid jQuery selector can be used but it must point to only one input that has been previously called by autoNumeric().
    - $('input#my').autoNumericGet(); " the defaults or the meta data assigned to the input will be used"

$(selector).autoNumericSet(value); - This function will format a string\value to the format assigned to the target field. Please note: the argument can be a numeric value or text (9999.99 or '9999.99') - demonstrated in the third column of the above table.

- Selector - any valid jQuery selector can be used but it must point to only one input that is recognized by autoNumeric().
- Value - can be literal or variable that contains no formatting other than a decimal point "full stop".
    - '1234' or 1234.99' note: the value/string will be rounded based on the settings you have for the input.
- Note: when the value exceeds the vMin \ vMax settings the result result is either an empty field or 0.00 depending on the input settings. If the decimal length exceeds the mDec value the result will be rounded and the remaining digits will be truncated (dropped). If the original unrounded value is needed you will need to store for later use.

Default values

Guides to follow for default values:

- The default value should be of the same format that you have assigned to the input via autoNumeric().
- If a default value is assigned it needs to be equal to or greater than the vMin value and equal to or less than the vMax value.

There is one undocumented default "aForm: false". When set to true the default value in the following format "#######.##" will be automatically formatted to the format you have assigned to this field. Please note this feature is a little buggy on page refresh so use at your own risk until a fix is found.

Dynamically loaded values

Assigning values once the page is ready can easily be accomplished with the autoNumericSet() function. Just be sure the string/value does not have any format (99999.99). Here is an example using JSON data on the ready event.

```
jQuery(function($) {
  $.getJSON("test_JSON.php", function(data) {
    var valueFormatted = '';
    $.each(data, function(key, value) { // loops through JSON keys and returns value
      $('#' + key).autoNumericSet(value);
    });
  });
// JSON values can be sent as text or numeric values (Note: this has mixed values)
{
  "idA": "1355.527",
  "idB": "9712547",
  "idC": 0.524,
  "idD": 135544.5244
}highlight: javascript
```

Below: a live example of populating the input fields with JSON data on page ready with the above script.

```
<input type="text" size="35" id="idA" class="auto"/>
```

1,355.53

```
<input type="text" size="35" id="idB" class="auto" meta="{aSep: ' ', aDec: ',', aSign: '€ '}"/>"
```

€ 9 712 547,00

```
<input type="text" size="35" id="idC" class="auto" meta="{aSign: '¢', pSign: 's', mRound: 'C'}"/>
```

0.53¢

```
<input type="text" size="35" id="idD" class="auto" meta="{aSign: ' C$', pSign: 's'}"/>
```

135 544 52 C$

Note: re-load page to retrieve JSON values

Callback function - This may be the most powerful new feature added in this version allowing the return value to be taken as parameter to the default/option key.

Note: Changing the decimal & thousand separator via the callback method can create unexpected results - please use with caution.

There are three methods to use this feature:

1) Pass a "function" when calling autoNumeric via options.

```
<script type="text/javascript">
  var functionName = function(){
    val = 11 * Math.random(); /* example only */
  return val; /* return sets the vMax value */
}

  /* Calls autoNumeric() for the input with the id="someID" and passes a function
  ** note the the function name does NOT use "parenthesis" */
  $('#someID').autoNumeric({vMax: functionName})

</script>

/* The above calls autoNumeric for this input and sets parameter (value) for "vMax:"
** no metadata used in the input tag */
<input id="someID" value="" >highlight: javascript
```

2) Pass a "function name" via options\metadata. Note that the function has been attached to autoNumeric previously. The return is taken as a parameter value.

```
<script type="text/javascript">

 /* Note function name attached to autoNumeric */
 $.autoNumeric.functionName = function($this, opts, name) {
   val = 11 * Math.random();    /* example only */
   return val;  /* return sets the vMax value */
 }

 /* Calls autoNumeric for the input with the id="someID" */
 $('#someID').autoNumeric()

</script>

 /* You can also pass function name via metadata as shown below
** the function name is proceeded by "fun:" all is enclosed with "parenthesis"
** the return value from the function is taken as a parameter of "vMax" */
<input id="someID" value="" meta="{vMax:'fun:FunctionName'}">highlight: javascript
```

alternative method:

```
<script type="text/javascript">

/* Note function name attached to autoNumeric */
$.autoNumeric.functionName = function($this, opts, name) {
  val = 11 * Math.random();  /*example only */
  return val;  /* return sets the vMax value */
}

/* The following method passes the function name via options.
** Note the function name is proceeded by "fun:" all is enclosed with "parenthesis"
** In this example the parameter (value) of "vMax" is set */
$('#someID').autoNumeric({vMax: 'fun:FunctionName'})

</script>

/* the above calls autoNumeric for this ID and passes the function name to autoNumeric()
** the return value from the function is taken as a parameter of "vMax" */
<input id="someID" value="">highlight: javascript
```

3) Pass a "css selector" recognized by jQuery via options\metadata - value of input is taken as a parameter value.

```
 /* Calls autoNumeric for the input with the id="someID" */
 $('#someID').autoNumeric()

</script>

/* You can also pass the css selector via metadata as shown below
** Note the selector is proceeded by "css:" all is enclosed with "parenthesis"
** This example uses the ID selector that requires the pound sign #
** In this example the value of the Selector element sets "aSign" parameter */
<input id="someID" value="" meta="{aSign:'css:#Selector'}">

/* The value of the element used as the parameter of "aSign:" */
<select id="selector" >
  <option value="km">Kilometers</option>
  <option value="m">Meters</option>
</select>highlight: javascript
```

alternative method:

```
<script type="text/javascript">

  /* The following method passes the CSS selector via options.
  ** Note the selector is proceeded by "css:" all is enclosed with "parenthesis"
  ** This example uses the ID selector that requires the pound sign #
  ** In this example the value of the Selector element sets "aSign" parameter  */
  $('#someID').autoNumeric({aSign: 'css:#Selector'})

</script>

/* The above calls autoNumeric for this ID
** no metadata used in the input tag */
<input id="someID" value="" >

/* The value of the element used as the parameter of "aSign:" */
<select id="selector" >
  <option value="km">Kilometers</option>
  <option value="m">Meters</option>
</select>highlight: javascript
```

The following callback demo uses all three callback methods shown above. This simple example allows the user to enter a length in either kilometers (n) or the equivalent amount of meters (n x 1000) and be able to switch back and forth. You will need to change the vMax (maximum) value, the mDEc (decimal places) and the aSign (length abbreviation) depending on which UOM is selected.

Select the preferred unit of measure, then enter the length. You can change the UOM back and forth, each switch changes the values for vMax, aSign & mDec options.

Select the desired metric unit:

kilometers

Please enter a maximum of 99999.999 kilometers or 99999999 meters.

km

HTML and script for the callback demo:

```
<form id="demo_callback">

 <label for="get_metricUnit">Metric:</label>
 <select id="get_metricUnit">
  <option value=" km" selected="selected">kilometers</option>
  <option value=" m">meters</option>
 </select>

 <label for="length">Length:</label>
 <input id="length" meta="{
 pSign:'s',
 aSign:'css:#get_metricUnit',
 mDec:'fun:get_mDec',
 aPad: false
 }" value=" km" type="text">
</form>highlight: html

<script type="text/javascript">
$.autoNumeric.get_mDec = function() { /* get_mDec function attached to autoNumeric() */
  var set_mDec = $('#get_metricUnit').val();
  if (   t    D        ' k   '){
```

```
 }
 return set_mDec; /* set mDec decimal places */
}

var get_vMax = function(){ /* set the maximum value allowed based on the metric unit */
 var set_vMax = $('#get_metricUnit').val();
 if(set_vMax == ' km'){
  set_vMax = '99999.999';
 } else {
   set_vMax = '99999999';
 }
 return set_vMax;
}

$('#length').autoNumeric({vMax: get_vMax});/* calls autoNumeric & passes function get_vMax */

$('#get_metricUnit').change(function(){
 var set_value = $('#length').autoNumericGet();
 if (this.value == ' km'){
  set_value = set_value / 1000;
 } else {
  set_value = set_value * 1000;
 }
 $('#length').autoNumericSet(set_value);
});
</script>highlight: javascript
```

Rounding options - The table below is for testing the rounding methods that are supported. Experiment by entering various values in the upper left input field. The right column will display the 9 different rounding methods. Decimal places can range from 0 to 6.

Note: If the decimal length exceeds the mDec value the result will be rounded and the remaining digits will be truncated (dropped). If the original unrounded value is required you will need to store/save the value.

For further details on rounding, the DIY Calculator site is an excellent resource. Additional information can be found on Wikipedia.

Entered value to be rounded		Rounded results
{vMin: '-999999.999999', vMax: '999999.999999'}"		mRound: 'S' - default
	Round-Half-Up "Symmetric"	
Set decimal places for rounding test(s) id="rDec"		mRound: 'A'
2	Round-Half-Up "Asymmetric"	
		mRound: 's'
Round Reset form!	Round-Half-Down "Symmetric"	
		mRound: 'a'
	Round-Half-Down "Asymmetric"	
		mRound: 'B'
	Round-Half-Even "Bankers"	
		mRound: 'U'
	Round Up "Away From Zero"	
		mRound: 'D'
	Round-Down "Toward Zero"	
		mRound: 'C'
	Round to Ceiling "Toward Positive Infinity"	
		mRound: 'F'
	Round to Floor "Toward Negative Infinity"	

Pasting support: the value being pasted should be of the same format assigned to the target field and fall within the min\max range. Please note: if the value being pasted is not formatted it will be formatted on the keydown or blur event. Also if the value being pasted is greater or less than the min\max range the field will be cleared.

Closing comments: We hope that you find this plugin easy and useful and that it shortens your

Known issues (Help here is appreciated):

- HTML5 when input type=number
- Overrides "readonly" attribute in somes browsers - can be controlled by using onfocus='this.blur()'
- Some mobile browsers including "Android stock browser 2.3.3" - this appears to be a browser bug and finding the correct caret position. Works great on Mobile FireFox and iPhone.

Future improvements:

- Create a new plugin "mobileNumeric" that addresses of the known caret issues in some of the new smart phone browsers.
- HTML5 compatible.
- Support for jQuery Data - depreciate use of metadata.
- Support for jQuery Live.
- Clean the code and reduce the footprint.

Limited testing conducted on the following browsers:

- Chrome 12.0.742.122
- Firefox 5.0
- IE 7.0.5730.13, IE 8.0.76.16385, IE9.0.8112.16421
- iPad2
- iPhone 3 - back space may not function
- iPhone 4
- Opera 11.50
- Safari 5.0.3 Mac
- Safari 5.0.5 WIN

History

Version 1.7.4-B

- Fixed caret routine for IE (oops mistake).

Version 1.7.4

- Depreciated the use of "options" in the autoNumericGet() & autoNumericSet() public functions.
- Bug fix - prevent multiple instances of autoNumeric from being loaded.
- Bug fix - corrected loop through a array using "for(i in left_ar)" - thanks Peter Kovari.
- Eliminated the use of ++ & -- to iterate and the "continue" statement.
- General code maintenance.

Version 1.7.2 & 1.7.3

- Bug fix - limited releases.

Version 1.7.1 click here to view

- Bug fix - change event is corrected.

Version 1.7.0

- Precision control of minimum & maximum values to the n^{th} decimal places!
- The number of decimal places and negative sign are now set via the min/max values. Also positive and negative values are easily toggled from any caret position within the input field.
- Public functions $('selector').autoNumericGet({options}) & $('selector').autoNumericSet(value, {options}) to strip and format values that use jQuery's selector format.

Version 1.6.2 click here to view

- Bug fix - on setting the caret position.
- General code clean up.

Version 1.6.1

- Bug fix - that prevented a decimal character from being entered when only the currency symbol is present.
- Bug fix - when entering a decimal character when the thousand separator is a space ' ' and the currency symbol has a space and in placed as a suffix.

Version 1.6.0

- Eliminated the hard coded space between the currency symbol and numeric characters.
- Spacing between the currency symbol and numeric characters is now determined by the developer (you).
- Re-wrote the routine that handles the decimal character key events.
- Re-wrote the routine that handles the numeric character(s) key events.
- Re-wrote the set caret position routine.
- Added focus out event
- Fixed leading zero when no numeric characters were entered to the left of the decimal point
- Currency symbol and padding are only displayed when a numeric and/or decimal character have been entered - special thanks to B. Cull for his contribution.

Version 1.5.4

- Added padding option - see aPad above for greater details - special thanks Jonas Johansson for his contribution
- Fixed select all (ctrl + a)
- Fixed caret position when a negative sign is added when a currency symbol is present

Version 1.5.3

- Corrected issues with id's that used special characters - Thanks Anthony & Evan C.
- MAC command key is now recognized (version 1.5.2) - Thanks Bart B.
- Changed the blur event to change - this corrected the onchange not being thrown in IE7/IE8 - Thanks Javier P.

Version 1.5.1

- Corrected return key support (thanks Bart B.)

Version 1.5.0

- Added currency symbol support
- Added meta data support and eliminated alt code that was used in previous versions.

Version 1.4.5 (Last version using the alt code) click here to view

- Correct caret position when one digit was present (thanks Bart V.)

Version 1.4.3

- Added set caret routine to the keyup event. Keeps the caret in the proper place after the format routine
- Added focus event handler - corrected issues with pages that have multiple formats
- Corrected the focus control when client tabs in to the field on the keyup event (reverted to earlier version)
- Corrected the Caps Lock key issue on the keyup event.
- Change the leading zero routine to the above

  Added o tine th t h ndle l e th t e eed the m im m

Versions 1.3.3 to 1.4.2: (requested custom versions)

Version 1.3.2:

- Changed keyup event and replaced e.keyCode
- Corrected a "bad assignment" - JSLint Gotch Yah

Version 1.3.1:

- Allow input to remain blank no value stays no value

Version 1.3.0:

- Added keyDown keyCode support
- Corrected "tab in" focus IE issue - you can now over write or delete the hi-lighted portion of the input value even when field has reached max length. Thanks Bart
- Corrected the "tab out" issue in Opera. Version 1.3.2:
- Eliminated the Opera specific call that was meant to handle a paste event
- jQuery 1.4 compatible

Version 1.2.0 - 1.2.2: (please do not use and upgrade to 1.3.0 or greater)

- Removed the Dom Calls and replaced with jQuery
- Added Home and End (e.keyCode 35 & 36) support to the keyup event (thanks JPM USA)
- Paste values are accepted in IE, FF, Chrome & Safari. (Thanks to Josh at Digitalbush)

Version 1.1.0 - abandoned

Original version 1.0.0 - 1.0.2

Please contact me if you have questions, comments and or suggestions at:
bob{@}decorplanit.com (remove brackets)