



PIVOT TABLES & UNICODE

PRISMA
HEALTHSM

by

Terence M. Lepczyk

8/17/2020

PRISMA HEALTH

Contents

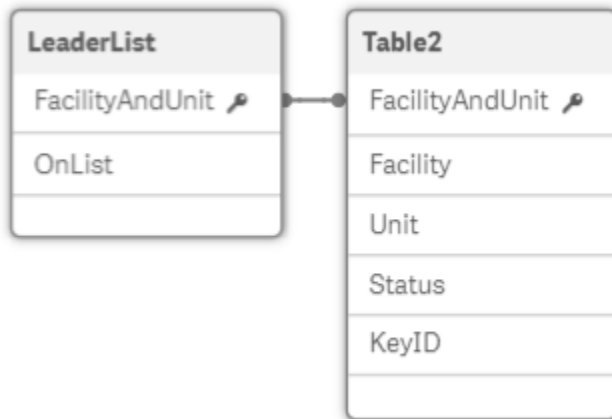
Overview:	2
Emphasizing a Dimension	2
Unicode	4
Chr - script and chart function	6
Use Unicode in your script as an alternative to 'in object'	7
Pivot Table Measure with Symbols	9
VizLib version	10
Standard Qlik version	10
Most Common arrows and symbols	11
More example from the Unicode table:	12
Links to Unicode Resources	13
Hex to Decimal converter	13
RGB Converter	13
Generate your own Unicode table	13
Qlik Sense color functions	14
The following color functions can be used in expressions when coloring by expression.	14
Let's talk Arrows	15
History of Codes and Standards	17
A bit of unrelated history	18
Definitions	19
ASCII	19
BCD	19
EBCDIC	20
Emoji	20
Emoticon	21
Hexadecimal	21
Unicode	22

Overview:

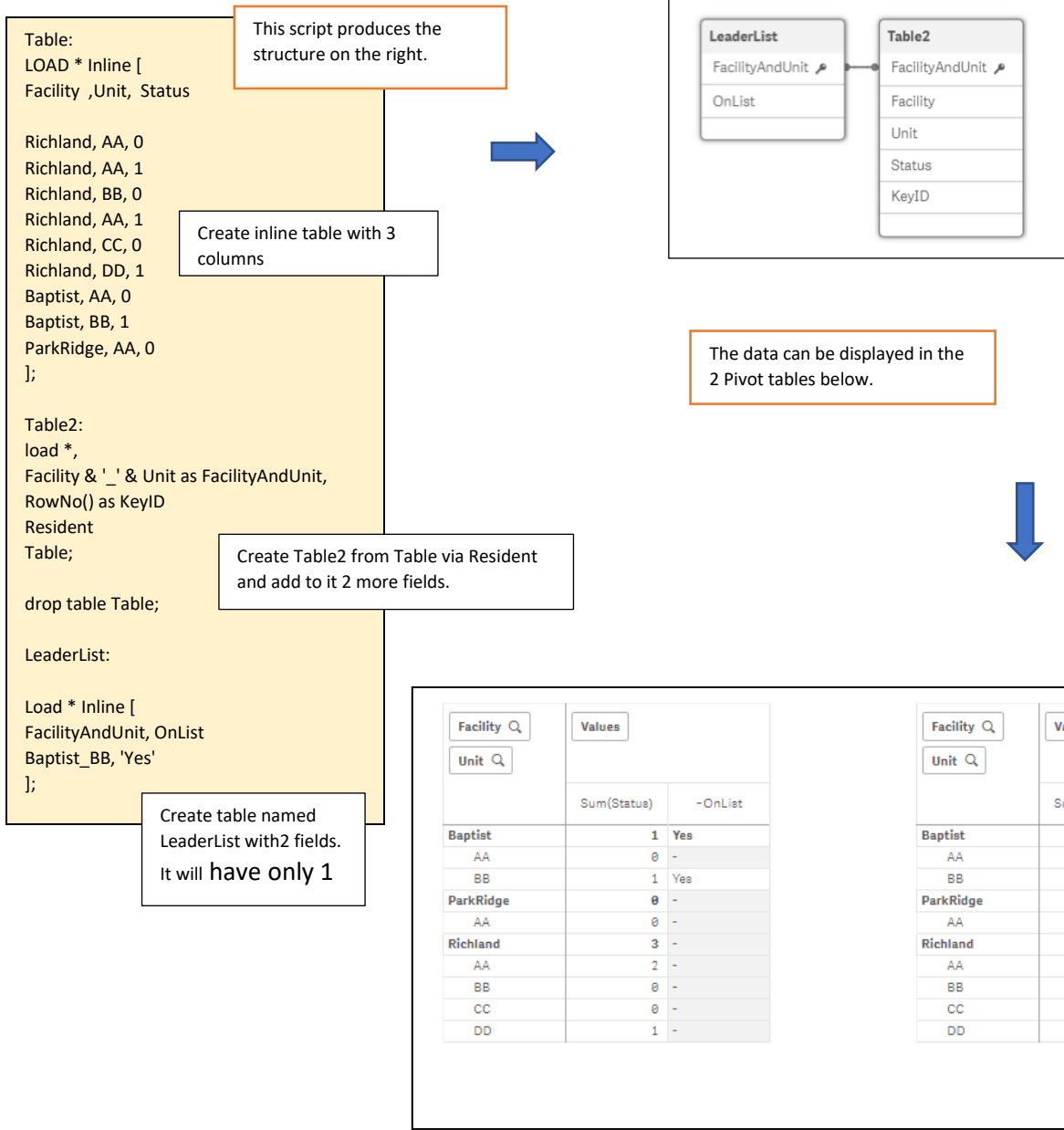
This paper describes how you can use Unicode to emphasize a dimension within a pivot table. Additionally, I will describe how you can utilize Unicode in pivot table measures as symbolic indicators. The described techniques can be easily retrofitted to meet your specific need. Please be aware that Emphasizing Dimensions is entirely different than using symbols in measures.

Emphasizing a Dimension

In this issue we are going to discuss how to visually emphasize Pivot table dimensions which meet specific criteria. In our example we will use two inline tables (LeaderList and Table2). We want to display the data in a Pivot table and emphasize which Units on Table2 ALSO appear on table LeaderList.



Below is the script used to create the schematic above.



The Standard Qlik Pivot table will allow you to conditionally colorize a measure. You can accomplish this by including `'=if(OnList = 'Yes', Yellow(), white())'` in the Background Color of the measure. The table on the left is without colorization, and the table on the right uses colorization.

Using the data above, we create a Pivot table as below, with Facility and Unit as our Dimensions. Our measures are **Sum(Status)** and **OnList**. We can see that *Baptist BB* is indicated as 'on the *LeaderList* table'. The Standard Qlik Pivot table will allow you to conditionally colorize a measure. You can accomplish this by including `'=if(OnList = 'Yes', Yellow(), white())'` in the Background Color of the measure. The Pivot table without colorization is on the left and the Colorized version is on the right.

Facility		Values	
Unit		Without color	
		Sum(Status)	-OnList
Baptist		1	Yes
AA		0	-
BB		1	Yes
ParkRidge		0	-
AA		0	-
Richland		3	-
AA		2	-
BB		0	-
CC		0	-
DD		1	-

Facility		Values	
Unit		With color	
		Sum(Status)	-OnList
Baptist		1	Yes
AA		0	-
BB		1	Yes
ParkRidge		0	-
AA		0	-
Richland		3	-
AA		2	-
BB		0	-
CC		0	-
DD		1	-

But can we emphasize the Dimension as well as the measure? The standard Qlik Pivot table does not provide for this option. If you use a more robust Pivot table such as one provided from VizLib or perhaps another Pivot table downloaded as an extension they may or may not provide the ability to colorize the dimension. My experience is that even when the feature is provided, it sometime will not work as expected. **So, what now?** Here is where Unicode comes in.

Unicode

Unicode is an [information technology](#) (IT) [standard](#) for the consistent [encoding](#), representation, and handling of [text](#) expressed in most of the world's [writing systems](#). The standard is maintained by the [Unicode Consortium](#), and as of March 2020, there is a repertoire of 143,859 characters, with Unicode 13.0 (these [characters](#) consist of 143,696 graphic characters and 163 format characters) covering 154 modern and historic [scripts](#), as well as multiple symbol sets and [emoji](#). The character repertoire of the Unicode Standard is synchronized with [ISO/IEC 10646](#), and both are code-for-code identical.

Simply put, we can integrate symbols from the Unicode character set into the dimension name. Below are 10 typical Unicode characters you could use to provide emphasis. All you need do is include the chr function in your model to display the corresponding Unicode symbol. To display the Heart you would use `chr(10084)`. Keep in mind there are **thousands** of Unicode characters you can display.

-chr(10084)	Q	-chr(9658)	Q	-chr(10062)	Q	-chr(9989)	Q	-chr(10060)
♥		▶		✘		✔		✘
-chr(9940)	Q	-chr(10067)	Q	-chr(10071)	Q	-chr(11088)	Q	-chr(11093)
⊖		?		!		★		⊙

A full list of Unicode characters can be easily found on the Web. This is a link to one such source: <https://www.tamasoft.co.jp/en/general-info/unicode-decimal.html>

When using Unicode be sure to use the Decimal representation.
Unicode symbols which are already colored will not react to color changes.
Generally, symbols which are BLACK will react to color changes as TEXT.

So how does this help?

If you change the Dimension of Unit to
=if(OnList= 'Yes', chr(11088) & ' ' & Unit, Unit)

If the value of OnList is Yes, we will change the Unit name by appending the Unicode of 11088 to the Unit.
11088 is interpreted and rendered as a small star.

Your table will look as below. Notice that the Dimension of Unit under Facility Baptist is now emphasized with a yellow star. This will work in almost any version of the pivot table.

Also note that in most pivot table visuals you have an option to always **fully expand**, below screenshot is fully expanded.

<input type="text" value="Facility"/>		☰ ☰
<input type="text" value="Unit"/>		Sum of Status
⊖ Baptist		1
★ BB		1
AA		0
⊖ ParkRidge		0
AA		0
⊖ Richland		3
AA		2
BB		0
CC		0
DD		1

Chr - script and chart function

Chr() returns the Unicode character corresponding to the input integer.

Syntax:

Chr(int)

Return data type: string

Examples and results:

Example	Result
---------	--------

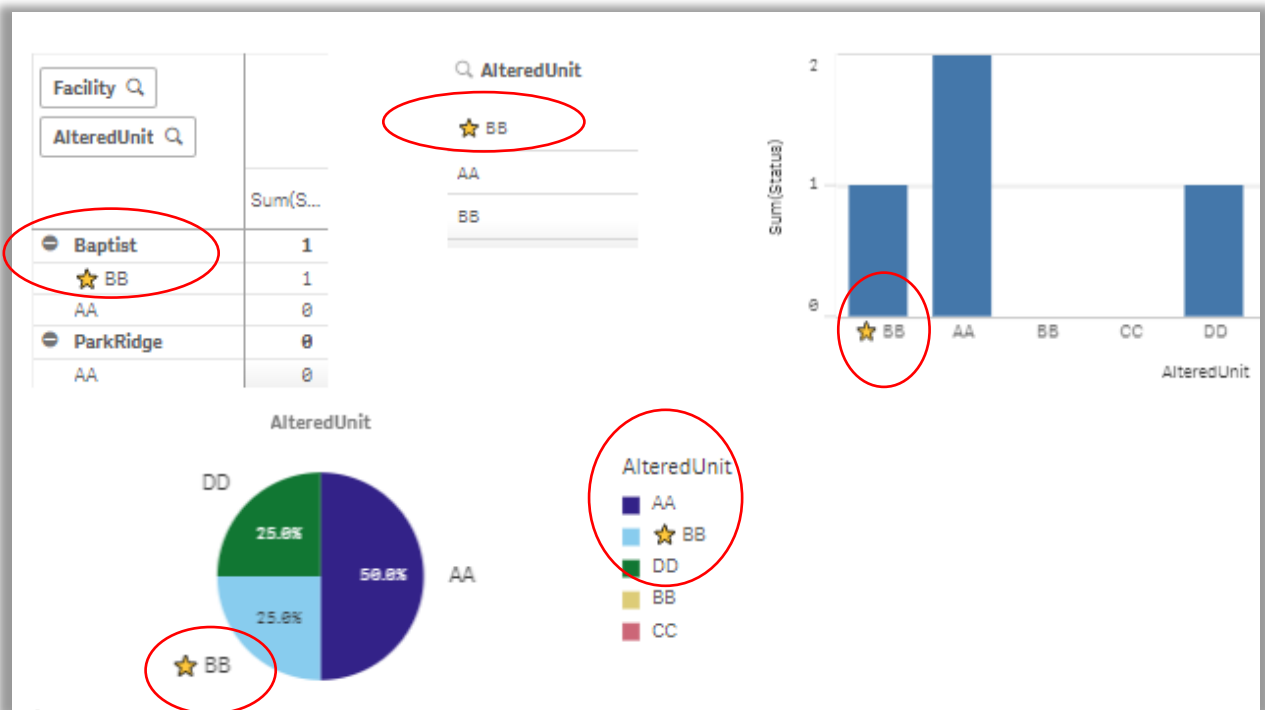
Chr(65)	Returns the string 'A'
---------	------------------------

Use Unicode in your script as an alternative to 'in object'.

In our example, we could have used Unicode within the script to permanently alter the Unit.

You could use ★BB as the Unit for Baptist BB everywhere within your model.

If you use the technique in script, you will not have to include the code in any visuals. Additionally, the Emphasized Unit will inherently display appropriately in any object without need for adjustment. In the screen shot below you will see that the Emphasized Unit displays in filters, Pivot, Bar Charts, Pie Charts, etc. *You can of course change the label name from AlteredUnit to just Unit.*



The code to include the Emphasized Unit is displayed below.


```
Table:
LOAD * Inline [
Facility ,Unit, Status
```

```
Richland, AA, 0
Richland, AA, 1
Richland, BB, 0
Richland, AA, 1
Richland, CC, 0
Richland, DD, 1
Baptist, AA, 0
Baptist, BB, 1
ParkRidge, AA, 0
];
```

```
Table2:
load *,
Facility & '_' & Unit as FacilityAndUnit,
RowNo() as KeyID
Resident
Table;
```

```
drop table Table;
```

```
LeaderList:
```

```
Load * Inline [
FacilityAndUnit, OnList
Baptist_BB, 'Yes'
];
```

```
map:
Mapping
load
FacilityAndUnit, OnList
```

```
Resident LeaderList;
```

```
Table3:
load *,
ApplyMap( 'map', FacilityAndUnit, 'No' ) as OnList
Resident
Table2;
```

```
drop table Table2;
drop table LeaderList;
```

```
Table4:
load *,
if(OnList= 'Yes', chr(11088) & ' ' & Unit, Unit) as AlteredUnit
Resident
```

```
Table3;
```

```
drop table Table3;
```

This script produces the structure on the right.

The script is the same as the previous example, but the script code below the BLUE line is now added.

Table4
Facility
Unit
Status
FacilityAndUnit
KeyID
OnList
AlteredUnit

The code above the blue line is the same as we have used previous. Below the blue line is what we are adding to make the Unit name work in script.

Below is a preview of the data. Notice the new filed named AlteredUnit and the appended Yellow Star.

Table4		Preview of data						
Rows	9	Facility	Unit	Status	FacilityAndUnit	KeyID	OnList	AlteredUnit
Fields	7	Richland	BB	0	Richland_BB	3	No	BB
Keys	0	Richland	AA	1	Richland_AA	4	No	AA
Tags	\$asciil \$text \$numeric \$integer	Richland	CC	0	Richland_CC	5	No	CC
		Richland	DD	1	Richland_DD	6	No	DD
		Baptist	AA	0	Baptist_AA	7	No	AA
		Baptist	BB	1	Baptist_BB	8	Yes	★ BB
		ParkRidge	AA	0	ParkRidge_AA	9	No	AA

Create a mapping table from table LeaderList

Apply the map to Table2 and create Table3.

You can then produce a Pivot table as below.

Facility	Measures	
	Sum(Status)	Sum(Status)
Baptist	1 →	1 ●
★ BB	1 →	1 ●
AA	0 ↓	0 ●
ParkRidge	0 ↓	0 ●
AA	0 ↓	0 ●
Richland	3 ↑	3 ●
AA	2 ↑	2 ●
BB	0 ↓	0 ●
CC	0 ↓	0 ●
DD	1 →	1 ●

The Emphasized Unit name will be named AlteredUnit.

It Should be noted that the technique of using Unicode symbols to emphasize dimensions will work equally as well in straight tables also as depicted below.

Facility	AlteredUnit	Sum(Status)
Totals		4
Baptist	★ BB	1
Baptist	AA	0
ParkRidge	AA	0
Richland	AA	2
Richland	BB	0
Richland	CC	0
Richland	DD	1

Pivot Table Measure with Symbols

The discussion about emphasizing Dimensions with Unicode should not be confused with the ability to use symbols and colors as part of measures. Most straight tables and pivot tables have built in provisions to easily display various indicators as measures. Below is an example of our data displaying the measure *sum(status)* with indicators. In one column I am using arrows, and the next column I am using circles. The example below is constructed using the Vizlib Pivot table. The object allows us to set a threshold, in this case the threshold is set to 1. We can then define the Symbol and Color for Above, Equal, and Below Threshold. Below is the VizLib version of the pivot table object.

VizLib version

Facility		Measures	
AlteredUnit		Sum(Status)	Sum(Status)
⊖ Baptist		1 →	1 ●
★ BB		1 →	1 ●
AA		0 ↓	0 ●
⊖ ParkRidge		0 ↓	0 ●
AA		0 ↓	0 ●
⊖ Richland		3 ↑	3 ●
AA		2 ↑	2 ●
BB		0 ↓	0 ●
CC		0 ↓	0 ●
DD		1 →	1 ●

Standard Qlik version

You can accomplish the same effect as described with the VizLib pivot in the standard Qlik pivot table.

Facility		Values		
AlteredUnit		Sum(Status)	if (Sum(Status) > 1, chr(9650), chr(11044))	if (Sum(Status) > 1, chr(11044), chr(9650))
⊖ Baptist		1	▶	→
★ BB		1	▶	→
AA		0	▼	↓
⊖ ParkRidge		0	▼	↓
AA		0	▼	↓
⊖ Richland		3	▲	↑
AA		2	▲	↑
BB		0	▼	↓
CC		0	▼	↓
DD		1	▶	→

If you are using Circles, the same Unicode symbol is used regardless of Up, Down or Equal, only the color changes.

Your Measure would be **chr(11044)**.

You would use the following as an expression as the text color expression.

**if (Sum(Status) > 1, green(), if (Sum(Status) < 1, rgb(255,0,0) ,
Lightgray()))**

Note: I prefer to use rgb(255,0,0) as opposed to red(), simply a matter of preference. You may need to consider color palettes which are adjusted for color blindness. Colors are discussed within this document in section [Qlik Sense Color Options](#).

If you are using Arrows, you would use the same expression for text color expression, but your Unicode symbols would change. The measure would be as follows.

**if (Sum(Status) > 1, chr(9650), if (Sum(Status) < 1, chr(9660) ,
chr(9658)))**

Most Common arrows and symbols

arrows			
=chr(129092) Q	=chr(129093) Q	=chr(129094) Q	=chr(129095) Q
←	↑	→	↓
=chr(129136) Q	=chr(129137) Q	=chr(129138) Q	=chr(129139) Q
←	↑	→	↓
=chr(129052) Q	=chr(129053) Q	=chr(129054) Q	=chr(129055) Q
←	↑	→	↓
=chr(9664) Q	=chr(9650) Q	=chr(9658) Q	=chr(9660) Q
◀	▲	▶	▼
=chr(128678) Q	=chr(11044) Q	=chr(8987) Q	
🚦	●	⌚	

In some instances, leadership demand the ‘traffic light’ solution, you could always give them this 😊.



And YES, smileys are Unicode also.

More example from the Unicode table:

If you wanted the Hourglass, the code would be 8987

If you are looking for codes on the Webb, be sure to use the DECIMAL equivalent. *This is discussed a bit later*

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
8500	◊	␣	␣	␣	␣	␣	␣	FAX	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8520	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8540	½	¾	¾	¾	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII	L	C	D	M
8560	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8580	◊	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8600	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8620	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8640	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8660	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8680	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8700	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8720	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8740	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8760	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8780	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8800	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8820	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8840	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8860	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8880	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8900	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8920	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8940	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8960	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
8980	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
9000	␣	()	⊗	⊘	⊙	⊚	⊛	⊜	⊝	⊞	⊟	⊠	⊡	⊢	⊣	⊤	⊥	⊦	⊧	
9020	⊨	⊩	⊪	⊫	⊬	⊭	⊮	⊯	⊰	⊱	⊲	⊳	⊴	⊵	⊶	⊷	⊸	⊹	⊺	⊻	
9040	⊼	⊽	⊾	⊿	Ⓚ	Ⓛ	Ⓜ	Ⓝ	Ⓞ	Ⓟ	Ⓠ	Ⓡ	Ⓢ	Ⓣ	Ⓤ	Ⓥ	Ⓦ	Ⓧ	Ⓨ	Ⓩ	
9060	ⓐ	ⓑ	ⓓ	ⓔ	ⓖ	ⓗ	ⓘ	ⓙ	ⓚ	ⓛ	ⓜ	ⓝ	ⓞ	ⓟ	ⓠ	ⓡ	ⓢ	ⓣ	ⓤ	ⓥ	
9080	ⓦ	ⓧ	ⓨ	ⓩ	⓪	⓫	⓬	⓭	⓮	⓯	⓰	⓱	⓲	⓳	⓴	⓵	⓶	⓷	⓸	⓹	
9100	⓺	⓻	⓼	⓽	⓾	⓿	Ⓚ	Ⓛ	Ⓜ	Ⓝ	Ⓞ	Ⓟ	Ⓠ	Ⓡ	Ⓢ	Ⓣ	Ⓤ	Ⓥ	Ⓦ	Ⓧ	
9120	Ⓩ	ⓐ	ⓑ	ⓓ	ⓔ	ⓖ	ⓗ	ⓘ	ⓙ	ⓚ	ⓛ	ⓜ	ⓝ	ⓞ	ⓟ	ⓠ	ⓡ	ⓢ	ⓣ	ⓤ	
9140	⓶	⓷	⓸	⓹	⓺	⓻	⓼	⓽	⓾	⓿	Ⓚ	Ⓛ	Ⓜ	Ⓝ	Ⓞ	Ⓟ	Ⓠ	Ⓡ	Ⓢ	Ⓣ	
9160	Ⓤ	Ⓥ	Ⓦ	Ⓧ	Ⓨ	Ⓩ	ⓐ	ⓑ	ⓓ	ⓔ	ⓖ	ⓗ	ⓘ	ⓙ	ⓚ	ⓛ	ⓜ	ⓝ	ⓞ	ⓟ	
9180	ⓠ	ⓡ	ⓢ	ⓣ	ⓤ	ⓥ	ⓦ	ⓧ	ⓨ	ⓩ	⓪	⓫	⓬	⓭	⓮	⓯	⓰	⓱	⓲	⓳	
9200	⓴	⓶	⓷	⓸	⓹	⓺	⓻	⓼	⓽	⓾	⓿	Ⓚ	Ⓛ	Ⓜ	Ⓝ	Ⓞ	Ⓟ	Ⓠ	Ⓡ	Ⓢ	
9220	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI	DLE	DCC	DC3	DCA	NAK	SYN	ETB		
9240	CAN	EM	SUB	ESC	FS	CS	RS	US	SP	DEL	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	
9260	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	
9280	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	
9300	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	
9320	⓫	⓬	⓭	⓮	⓯	⓰	⓱	⓲	⓳	⓴	⓵	⓶	⓷	⓸	⓹	⓺	⓻	⓼	⓽	⓾	
9340	⓿	Ⓚ	Ⓛ	Ⓜ	Ⓝ	Ⓞ	Ⓟ	Ⓠ	Ⓡ	Ⓢ	Ⓣ	Ⓤ	Ⓥ	Ⓦ	Ⓧ	Ⓨ	Ⓩ	ⓐ	ⓑ	ⓓ	
9360	9.	10.	11.	12.	13.	14.	15.	16.	17.	18.	19.	20.	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	
9380	(i)	(j)	(k)	(l)	(m)	(n)	(o)	(p)	(q)	(r)	(s)	(t)	(u)	(v)	(w)	(x)	(y)	(z)	Ⓐ	Ⓑ	
9400	©	®	™	Ⓢ	Ⓣ	Ⓤ	Ⓥ	Ⓦ	Ⓧ	Ⓨ	Ⓩ	ⓐ	ⓑ	ⓓ	ⓔ	ⓖ	ⓗ	ⓘ	ⓙ	ⓚ	
9420	ⓞ	ⓟ	ⓠ	ⓡ	ⓢ	ⓣ	ⓤ	ⓥ	ⓦ	ⓧ	ⓨ	ⓩ	⓪	⓫	⓬	⓭	⓮	⓯	⓰	⓱	
9440	⓴	⓶	⓷	⓸	⓹	⓺	⓻	⓼	⓽	⓾	⓿	Ⓚ	Ⓛ	Ⓜ	Ⓝ	Ⓞ	Ⓟ	Ⓠ	Ⓡ	Ⓢ	
9460	Ⓤ	Ⓥ	Ⓦ	Ⓧ	Ⓨ	Ⓩ	ⓐ	ⓑ	ⓓ	ⓔ	ⓖ	ⓗ	ⓘ	ⓙ	ⓚ	ⓛ	ⓜ	ⓝ	ⓞ	ⓟ	
9480	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

	00	01	02	03	04	05	07	08	09	10	11	12	13	14	15	16	17	18	19	
9500	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌
9520	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
9540	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
9560	⓪	Ⓛ	Ⓜ	Ⓝ	Ⓞ	Ⓟ	Ⓠ	Ⓡ	Ⓢ	Ⓣ	Ⓤ	Ⓥ	Ⓦ	Ⓧ	Ⓨ	Ⓩ	ⓐ	ⓑ	ⓓ	
9580	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌
9600	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
9620	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬	▬
9640	▯	▯	▯	▯	▯	▯	▯	▯	▯	▯	▯	▯	▯	▯	▯	▯	▯	▯	▯	▯
9660	▰	▰	▰	▰	▰	▰	▰	▰	▰	▰	▰	▰	▰	▰	▰	▰	▰	▰	▰	▰
9680	◔	◕	◖	◗	◘	◙	◚	◛	◜	◝	◞	◟	◠	◡	◢	◣	◤	◥	◦	◧
9700	◨	◩	◪	◫	◬	◭	◮	◯	◰	◱	◲	◳	◴	◵	◶	◷	◸	◹	◺	◻
9720	◼	◽	◾	◿	◿	◿	◿	◿	◿	◿	◿	◿	◿	◿	◿	◿	◿	◿	◿	◿
9740	Ⓚ	Ⓛ	Ⓜ	Ⓝ	Ⓞ	Ⓟ	Ⓠ	Ⓡ	Ⓢ	Ⓣ	Ⓤ	Ⓥ	Ⓦ	Ⓧ	Ⓨ	Ⓩ	ⓐ	ⓑ	ⓓ	
9760	ⓞ	ⓟ	ⓠ	ⓡ	ⓢ	ⓣ	ⓤ	ⓥ	ⓦ	ⓧ	ⓨ	ⓩ	⓪	⓫	⓬	⓭	⓮	⓯	⓰	⓱
9780	⓴	⓶	⓷	⓸	⓹	⓺	⓻	⓼	⓽	⓾	⓿	Ⓚ	Ⓛ	Ⓜ	Ⓝ	Ⓞ	Ⓟ	Ⓠ	Ⓡ	Ⓢ
9800	Ⓤ	Ⓥ	Ⓦ	Ⓧ	Ⓨ	Ⓩ	ⓐ	ⓑ	ⓓ	ⓔ	ⓖ	ⓗ	ⓘ	ⓙ	ⓚ	ⓛ	ⓜ	ⓝ	ⓞ	ⓟ
9820	ⓠ	ⓡ	ⓢ	ⓣ	ⓤ	ⓥ	ⓦ	ⓧ	ⓨ	ⓩ	⓪	⓫	⓬	⓭	⓮	⓯	⓰	⓱	⓲	⓳
9840	⓴	⓶	⓷	⓸	⓹	⓺	⓻	⓼	⓽	⓾	⓿	Ⓚ	Ⓛ	Ⓜ	Ⓝ	Ⓞ	Ⓟ	Ⓠ	Ⓡ	Ⓢ
9860	Ⓤ	Ⓥ	Ⓦ	Ⓧ	Ⓨ	Ⓩ	ⓐ	ⓑ	ⓓ	ⓔ	ⓖ	ⓗ	ⓘ	ⓙ	ⓚ	ⓛ	ⓜ	ⓝ	ⓞ	ⓟ
9880	ⓠ	ⓡ	ⓢ	ⓣ	ⓤ	ⓥ	ⓦ	ⓧ	ⓨ	ⓩ	⓪	⓫	⓬	⓭	⓮	⓯	⓰	⓱	⓲	⓳
9900	⓴	⓶	⓷	⓸	⓹	⓺	⓻	⓼	⓽	⓾	⓿	Ⓚ	Ⓛ	Ⓜ	Ⓝ	Ⓞ				

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
11000	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11020	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11040	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11060	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11080	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11100	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11120	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11140	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11160	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11180	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11200	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11220	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11240	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11260	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11280	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11300	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11320	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11340	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11360	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11380	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11400	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11420	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11440	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11460	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡
11480	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡	⚡

Links to Unicode Resources

https://www.w3schools.com/charsets/ref_utf_arrows.asp

<https://www.youtube.com/watch?v=zl965tnRG7U>

<https://www.youtube.com/watch?v=9CyvIoX39Ow>

[https://en.wikipedia.org/wiki/Arrow_\(symbol\)](https://en.wikipedia.org/wiki/Arrow_(symbol))

Hex to Decimal converter

<https://www.rapidtables.com/convert/number/hex-to-decimal.html?x=1f870>

RGB Converter

<https://www.rapidtables.com/convert/color/rgb-to-hex.html>

Generate your own Unicode table

But sometimes you need a special symbol, but you don't know the number. You can find it in the Internet, but I've found a Qlik way how to do it.

- Just create a straight table.
- Add Calculated Dimension =ValueLoop(1,100) or even =ValueLoop(1,10000)
- Add Expression chr(RowNo())
- And find a symbol that you need.

Qlik Sense color functions

The following color functions can be used in expressions when coloring by expression.

- black()
- darkgray()
- lightgray()
- white()
- blue()
- lightblue()
- green()
- lightgreen()
- cyan()
- lightcyan()
- red()
- lightred()
- magenta()
- lightmagenta()
- brown()
- yellow()


RGB() is used in expressions to set or evaluate the color properties of a chart object, where the color is defined by a red component **r**, a green component **g**, and a blue component **b** with values between 0 and 255.









RGB (r, g, b)

You can find an online RGB to hex converter at:

<https://www.rapidtables.com/convert/color/rgb-to-hex.html>

You can use Hex Color code in expression by writing the code in between single quotations.

In this case you can write like this '#FF0000' is RGB (255,0,0) 

ColorChart							
black	Q	darkgray	Q	lightgray	Q	white	
							
blue	Q	lightblue	Q	green	Q	lightgreen	
							
cyan	Q	lightcyan	Q	red	Q	lightred	
							
magenta	Q	lightmagenta	Q	brown	Q	yellow	
							

Let's talk Arrows



Below is an excerpt from Wikipedia regarding Arrows in Unicode. Pay attention to the fact that the charts below display the Hexadecimal code and not the Decimal code. For Qlik, you will need the Decimal code.

How to use the below charts in a Qlik model. Let's look at one Arrow from the chart, U+2190. The U+ indicates it is a Unicode designation and the 2190 is the Hexadecimal value to generate the actual symbol. To use the Hexadecimal value, you must first convert it to its Decimal equivalent. To convert you could do this by hand, but there are several converters available on the Web, here is one:

<https://www.rapidtables.com/convert/number/hex-to-decimal.html>

2190 converts to 8592

←	Leftwards Arrow	U
U+2190		
←		

Hexadecimal to Decimal converter

From: To:

Enter hex number:
 16

Decimal number:
 10

Decimal from signed 2's complement:
 10

Binary number:
 2

Decimal calculation steps:
 $(2190)_{16} = (2 \times 16^3) + (1 \times 16^2) + (9 \times 16^1) + (0 \times 16^0) = (8592)_{10}$

If you add a table chart with the dimension **=chr(8592)** you will get the below. If you add **=Green()** to the Text Color Expression it will turn to green.



Not all Unicode symbols will react to color change. Generally, if the Symbol is NOT COLORIZED, 'displayed as black', you can probably colorize it yourself. Example of a symbols which do not react to colorization would be the Yellow Star, Red circle, Alarm clock, and the Hourglass.



Below is an excerpt from Wikipedia regarding arrows. I have included just a couple pages of arrows. **Last count there were approximately 611 arrows available in Unicode.**

Arrows are universally recognized for indicating directions.^[1] They are widely used on [signage](#) and for [wayfinding](#),^[1] and are often used in [road surface markings](#).

Upwards pointing arrows are often used to indicate an increase in a numerical value, and downwards pointing arrows indicate a decrease.

In [Unicode](#), the block Arrows occupies the [hexadecimal](#) range U+2190–U+21FF, as described below.

Symbol	Name
Hex	
Picture of this symbol	

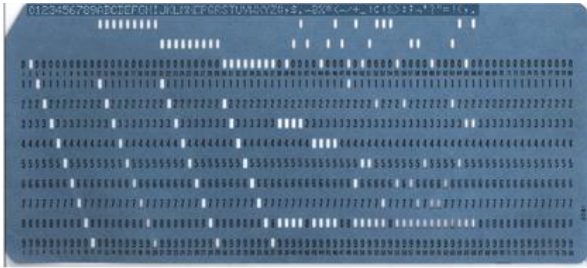
This is one of the basic formats you will see when looking for Unicode symbols. Again, be aware if the code is Hexadecimal or Decimal.

← U+2190 Leftwards Arrow	↩ U+21ac Rightwards Arrow With Loop	⇕ U+21e8 Upwards Paired Arrows	⇐ U+21e4 Leftwards Arrow To Bar
↑ U+2191 Upwards Arrow	↔ U+21ad Left Right Wave Arrow	⇒ U+21e9 Rightwards Paired Arrows	→ U+21e5 Rightwards Arrow To Bar
→ U+2192 Rightwards Arrow	↔ U+21ae Left Right Arrow With Stroke	⇓ U+21ea Downwards Paired Arrows	⇐ U+21e6 Leftwards Thick Arrow
↓ U+2193 Downwards Arrow	↻ U+21af Downwards Zigzag Arrow	⇐ U+21eb Leftwards Harpoons Over Rightwards Harpoon	⇑ U+21e7 Upwards Thick Arrow

History of Codes and Standards

BCD Binary Coded Decimal designed to support the usage of the IBM computer punch card (Herman Hollerith Card). **Contained 6 bits** and was never standardized. Standardization was manufacturer specific. **Invented in 1928.**

Punched card with the Hollerith encoding of the 1964 EBCDIC character set. Contrast at top enhanced to show the printed characters.



Hollerith invented and used a punched card device to help analyze the 1890 U.S. census data. The previous census of **1880 had taken 8 years to process and resulted in a population of 50,189,209, while the 1890 census was completed in 6 years with a resulting population of**

62,979,766.... **Hollerith's** punch cards and **tabulating machines** were a step toward automated computation. His device could automatically read information which **had** been punched onto a card.

Herman Hollerith is widely regarded as the father of modern automatic computation. He chose the **punched card** as the basis for storing and processing information and he built the first **punched**



card tabulating and sorting machines as well as the first key **punch**, and he founded the company that was to become IBM.

EBCDIC Extended Binary Coded Decimal Interchange Code. Developed by IBM and introduced in their 360 line of computers in **1963. Contained 8 bits.**

ASCII American Standard for Code for Information Interchange. Developed from Telegraph code. Creation of ASCII began on October 6, 1968 with the first meeting of ANSI (American National Standards Institute). The first publication of ASCII was released in **1963** and followed by a Major Enhanced revision in **1967. Contained 7 bits.**

UNICODE Universal Standard CODE released in **1991** and as of March 2020 contains 143,859 characters. **Contains 16-bit** structure (2 Bytes), and capable of 1,112,064 characters.

Current list of all Unicode's can be found at:

<https://unicode.org/charts/>

The home page for Unicode information can be found at:

<https://home.unicode.org/>

EMOJI first appeared on Microsoft platforms as the **WingDing font** in 1990. It was further popularized by Japanese iPhones in 1997 and became **universally accepted** in 2010.



SoftBank, known as J-Phone Believed to be one of the first phones to use Emoji.

Originally meaning pictograph, the word **emoji** comes from Japanese e (絵, "picture") + moji (文字, "character"); the resemblance to the English words emotion and emoticon is purely coincidental. ... In 2015, Oxford Dictionaries named the Face with Tears of Joy **emoji** (😄) the Word of the Year.

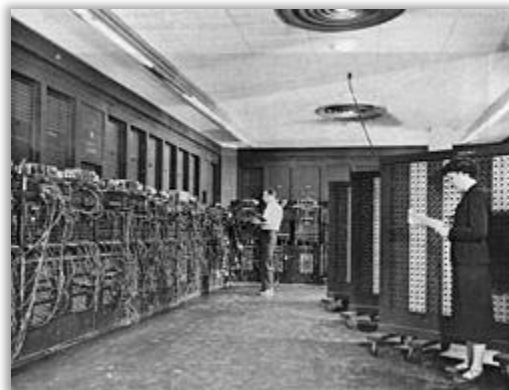
Some emoji are specific to Japanese culture, such as a **bowing** businessman (U+1F647 🙇), the **shoshinsha mark** used to indicate a beginner driver (U+1F530 🚏), a white flower (U+1F4AE 🌸) used to denote "brilliant homework",^[60] or a group of emoji representing popular foods: **ramen** noodles (U+1F35C 🍜), **dango** (U+1F361 🍡), **onigiri** (U+1F359 🍙), **Japanese curry** (U+1F35B 🍛), and **sushi** (U+1F363 🍣). **Unicode Consortium** founder **Mark Davis** compared the use of emoji to a developing language, particularly mentioning the American use of eggplant (U+1F346 🍆) to represent a **phallus**.^[61] Some linguists have classified emoji and **emoticons** as **discourse markers**.^[62]

Current list of Emojis can be found at <https://unicode.org/emoji/charts/full-emoji-list.html>

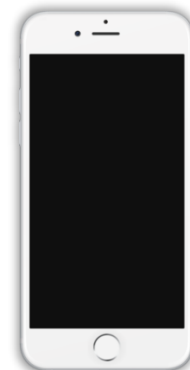
A timeline of Emoji introduction can be found at <https://emojitime.com/>

A bit of unrelated history

ENIAC was one of the first computers used for commercial use, it used 17,468 Vacuum tubes, took up 1,800 square feet and weighed 25 tons. It was capable of 5,000 instructions per second. ENIAC was completed in 1945 and first put to work for practical purposes on December 10, 1945.



The iPhone 6 weighs 4.55 ounces and performs 25 Billion instructions per second.



Definitions

ASCII

ASCII (/ˈæskiː/ [ⓘ] [Ⓘ] ^{listen}) *ASS-kee*),^{[3]:6} abbreviated from **American Standard Code for Information Interchange**, is a [character encoding](#) standard for electronic communication. ASCII codes represent text in computers, [telecommunications equipment](#), and other devices. Most modern character-encoding schemes are based on ASCII, although they support many additional characters.

The [Internet Assigned Numbers Authority](#) (IANA) prefers the name **US-ASCII** for this character encoding.^[2]

BCD

In [computing](#) and [electronic](#) systems, **binary-coded decimal (BCD)** is a class of [binary](#) encodings of [decimal](#) numbers where each [digit](#) is represented by a fixed number of [bits](#), usually four or eight. Sometimes, special bit patterns are used for a [sign](#) or other indications (e.g. error or overflow).

In [byte](#)-oriented systems (i.e. most modern computers), the term *unpacked* BCD^[4] usually implies a full byte for each digit (often including a sign), whereas *packed* BCD typically encodes two digits within a single byte by taking advantage of the fact that four bits are enough to represent the range 0 to 9. The precise 4-bit encoding, however, may vary for technical reasons (e.g. [Excess-3](#)).

The ten states representing a BCD digit are sometimes called [tetrades](#)^{[2][3]} (for the [nibble](#) typically needed to hold them is also known as a tetrad) while the unused, *don't care*-states are named [pseudo-tetrad\(e\)s](#) [de],^{[4][5][6][7][8]} *pseudo-decimals*^[3] or *pseudo-decimal digits*.^{[9][10][nb 1]}

BCD's main virtue, in comparison to binary [positional systems](#), is its more accurate representation and rounding of decimal quantities, as well as its ease of conversion into human-readable representations. Its principal drawbacks are a slight increase in the complexity of the circuits needed to implement basic arithmetic as well as slightly less dense storage.

BCD was used in many early [decimal computers](#), and is implemented in the instruction set of machines such as the [IBM System/360](#) series and its descendants, [Digital Equipment Corporation](#)'s [VAX](#), the [Burroughs B1700](#), and the Motorola [68000](#)-series processors. BCD *per se* is not as widely used as in the past and it is no longer implemented in newer computers' instruction sets (e.g. [ARM](#)); [x86](#) does not support [its BCD instructions](#) in [long mode](#) any more. However, decimal [fixed-point](#) and [floating-point](#) formats are still important and continue to be used in financial, commercial, and industrial computing, where the subtle conversion and fractional [rounding errors](#) that are inherent in floating point binary representations cannot be tolerated.^[11]

EBCDIC

Extended Binary Coded Decimal Interchange Code^[1] (EBCDIC;^[1] /ˈɛbsɪdɪk/) is an eight-bit character encoding used mainly on IBM mainframe and IBM midrange computer operating systems. It descended from the code used with punched cards and the corresponding six-bit binary-coded decimal code used with most of IBM's computer peripherals of the late 1950s and early 1960s.^[2] It is supported by various non-IBM platforms, such as Fujitsu-Siemens' BS2000/OSD, OS-IV, MSP, and MSP-EX, the SDS Sigma series, Unisys VS/9, Burroughs MCP and ICL VME.

EBCDIC was devised in 1963 and 1964 by [IBM](#) and was announced with the release of the [IBM System/360](#) line of mainframe [computers](#). It is an eight-bit character encoding, developed separately from the seven-bit [ASCII](#) encoding scheme. It was created to extend the existing [Binary-Coded Decimal](#) (BCD) Interchange Code, or [BCDIC](#), which itself was devised as an efficient means of encoding the two *zone* and *number* punches on [punched cards](#) into six bits. The distinct encoding of 's' and 'S' (using position 2 instead of 1) was maintained from [punched cards](#) where it was desirable not to have hole punches too close to each other to ensure the integrity of the physical card^{[[citation needed](#)]}.

While IBM was a chief proponent of the ASCII standardization committee,^[3] the company did not have time to prepare ASCII peripherals (such as card punch machines) to ship with its System/360 computers, so the company settled on EBCDIC.^[2] The System/360 became wildly successful, together with clones such as [RCA Spectra 70](#), [ICL System 4](#), and Fujitsu FACOM, thus so did EBCDIC.

All IBM mainframe and [midrange peripherals](#) and [operating systems](#) use EBCDIC as their inherent encoding^[4] (with toleration for ASCII, for example, [ISPF](#) in [z/OS](#) can browse and edit both EBCDIC and ASCII encoded files). Software and many hardware peripherals can translate to and from encodings, and modern mainframes (such as [IBM Z](#)) include processor instructions, at the hardware level, to accelerate translation between character sets.

There is an EBCDIC-oriented [Unicode Transformation Format](#) called [UTF-EBCDIC](#) proposed by the Unicode consortium, designed to allow easy updating of EBCDIC software to handle Unicode, but not intended to be used in open interchange environments. Even on systems with extensive EBCDIC support, it has not been popular. For example, z/OS supports Unicode (preferring [UTF-16](#) specifically), but z/OS only has limited support for UTF-EBCDIC.

[IBM AIX](#) running on the [RS/6000](#) and its descendants including the [IBM Power Systems](#), [Linux running on IBM Z](#), and operating systems running on the [IBM PC](#) and its descendants use ASCII, as did [AIX/370](#) and [AIX/390](#) running on [System/370](#) and [System/390](#) mainframes.

Emoji

Emoji ([Japanese](#): 絵文字^{えもじ}, English: /ɪˈmoʊdʒiː/; [Japanese](#): [emodʒi]; singular *emoji*, plural *emoji* or *emojis*^[1]) are [ideograms](#) and [smileys](#) used in electronic messages and [web pages](#). Some examples of emoji are 😊, 🍌, and 🐱. Emoji exist in various genres, including facial expressions, common objects, places and types of weather, and animals. They are much like [emoticons](#), but emoji are pictures rather than [typographic approximations](#); the term "emoji" in the strict sense refers to such pictures which can be represented as [encoded characters](#), but it is sometimes applied to [messaging](#)

[stickers](#) by extension.^[2] Originally meaning [pictograph](#), the word *emoji* comes from Japanese *e* (絵, "picture") + *moji* (文字, "character"); the resemblance to the English words *emotion* and *emoticon* is [purely coincidental](#).^[3] The [ISO 15924](#) script code for emoji is `Zsye`.

Originating on [Japanese mobile phones](#) in 1997, emoji became increasingly popular worldwide in the 2010s after being added to several mobile operating systems.^{[4][5][6]} They are now considered to be a large part of [popular culture](#) in [the West](#).^[7] In 2015, [Oxford Dictionaries](#) named the [Face with Tears of Joy emoji](#) (😄) the [Word of the Year](#).^{[8][9]}

Emoticon

An **emoticon** (/ɪˈmoʊtɪkɒn/, *i-MOHT-i-kon*, rarely pronounced /ɪˈmɒtɪkɒn/),^{[1][2][3][4]} short for "emotion icon",^[5] also known simply as an **emote**, is a pictorial representation of a [facial expression](#) using [characters](#)—usually [punctuation marks](#), numbers, and letters—to express a person's feelings or mood, or as a time-saving method. The first ASCII emoticons, `: -)` and `: - (`, were written by [Scott Fahlman](#) in 1982, but emoticons actually originated on the [PLATO IV](#) computer system in 1972.^[6]

In Western countries, emoticons are usually written at a right angle to the direction of the text. Users from [Japan](#) popularized a kind of emoticon called **kaomoji** (顔文字; lit. 顔(kao)=face, 文字(moji)=character(s)), utilizing the [Katakana](#) character set, that can be understood without tilting one's head to the left. This style arose on ASCII NET of Japan in 1986.^{[7][8]}

As [SMS](#) and the [internet](#) became widespread in the late 1990s, emoticons became increasingly popular and were commonly used on text messages, [internet forums](#) and [e-mails](#). Emoticons have played a significant role in communication through technology, and some devices and applications have provided stylized pictures that do not use text punctuation. They offer another range of "tone" and feeling through texting that portrays specific emotions through facial gestures while in the midst of text-based cyber communication.^[9]

Hexadecimal

The **hexadecimal numeral system**, often shortened to "**hex**", is a [numeral system](#) made up of 16 symbols ([base](#) 16). The standard numeral system is called [decimal](#) (base 10) and uses ten symbols: 0,1,2,3,4,5,6,7,8,9. Hexadecimal uses the decimal numbers and six extra symbols. There are no numerical symbols that represent values greater than nine, so letters taken from the [English alphabet](#) are used, specifically A, B, C, D, E and F. Hexadecimal A = decimal 10, and hexadecimal F = decimal 15.

Humans mostly use the decimal system. This is probably because humans have ten fingers on their hands. Computers however, only have on and off, called a binary digit (or bit, for short). A binary number is just a string of zeros and ones: 11011011, for example. For convenience, engineers working with computers tend to group bits together. In earlier days, such as the 1960's, they would group 3 bits at a time (much like large decimal numbers are grouped in threes, like the number 123,456,789). Three bits, each being on or off, can represent the eight numbers from 0 to 7: 000 = 0; 001 = 1; 010 = 2; 011 = 3; 100 = 4; 101 = 5; 110 = 6 and 111 = 7. This is called [octal](#).

As computers got bigger, it was more convenient to group bits by four instead of three. This doubles the numbers that the symbol would represent; it can have 16 values instead of eight. *Hex* = 6 and *Decimal* = 10, so it is called hexadecimal. In [computer jargon](#) four bits make a *nibble* (sometimes spelled *nybble*). A nibble is one hexadecimal digit, written using a symbol 0-9 or A-F. Two nibbles make a [byte](#) (8 bits). Most computer operations use the byte, or a multiple of the byte (16 bits, 24, 32, 64, etc.). Hexadecimal makes it easier to write these large binary numbers.

To avoid confusion with decimal, octal or other numbering systems, hexadecimal numbers are sometimes written with a "h" after or "0x" before the number. For example, 63h and 0x63 mean 63 hexadecimal

Unicode

Unicode is an [information technology](#) (IT) [standard](#) for the consistent [encoding](#), representation, and handling of [text](#) expressed in most of the world's [writing systems](#). The standard is maintained by the [Unicode Consortium](#), and as of March 2020, there is a repertoire of 143,859 characters, with Unicode 13.0 (these [characters](#) consist of 143,696 graphic characters and 163 format characters) covering 154 modern and historic [scripts](#), as well as multiple symbol sets and [emoji](#). The character repertoire of the Unicode Standard is synchronized with [ISO/IEC 10646](#), and both are code-for-code identical.