

Package `ascii-emoticon` v. 1.0 Implementation

Conrad Kosowsky

August 2026

`kosowsky.latex@gmail.com`

Overview

The `ascii-emoticon` package provides control sequences for 222 ASCII emoticons (not emojis) and an interface to define even more. Emoticons work in both regular text and math mode.

This file documents the code for the `emoticon` package. It is not a user guide! For instructions on adding emoticons to your documents, see `ascii-emoticon-user-guide.pdf`, which is included in the `emoticon` installation and is available on CTAN. Section 1 includes the package declaration and error messages. Section 2 defines the built-in emoticons, and Section 3 lays out the interface for defining new emoticons. Version history, code index, and emoticon index appear at the end of the document.

1 Setup and Errors

First, the package should declare itself.

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{ascii-emoticon}[2026/08/24 v. 1.0]
```

We define four error messages as laid out in Table 1.

```
3 \def\EmoticonDefinedError#1#2{\PackageError{ascii-emoticon}
4   {Bad emoticon^^Jname "#1."@\gobble}
5   {Your \string\DeclareEmoticon\space command was ignored. I^^J%
6   can't use "#1" as an emoticon name because^^J%
7   the control sequence #2 has already been^^J%
8   defined. To resolve this error, make sure^^J%}
```

Table 1: Error Messages

Error Name	Arguments	Use
DefinedError	#1: Emoticon name #2: <code>\string\<command></code>	If <code>\<command></code> or <code>\tt<command></code> is already defined
MultiTokensError	#1: Multiple (unexpandable) tokens	If should be single token
ExtractMathError	None	If <code>\emoticon</code> appears in math mode
NotCSError	#1: One (unexpandable) token	If the token is not a control sequence

```

9   your emoticon name does not correspond to^^J%
10  a control sequence currently in use.^^J}}
11 \def\EmoticonMultiTokensError#1{\PackageError{ascii-emoticon}
12  {Bad emoticon^^Jname "#1."@\gobble}
13  {Your \string\DeclareEmoticon\space command was ignored. I^^J%
14  can't use "#1" as an emoticon name because^^J%
15  it contains multiple tokens. To resolve this^^J%
16  error, make sure your first argument of^^J%
17  \string\DeclareEmoticon\space is a single control sequence.^^J}}
18 \def\EmoticonExtractMathError{\PackageError{ascii-emoticon}
19  {Command \string\emoticon^^Jignored in math mode}
20  {It looks like you tried to use \string\emoticon\space in^^J%
21  math mode, but the command doesn't work inside^^J%
22  equations. To resolve this error, call \string\emoticon^^J%
23  only in vertical or horizontal mode.^^J}}
24 \def\EmoticonNotCSErrors#1{\PackageError{ascii-emoticon}
25  {Bad emoticon^^Jname "#1."@\gobble}
26  {Your \string\DeclareEmoticon\space command was ignored. I^^J%
27  can't use "#1" as an emoticon name because^^J%
28  it isn't a control sequence. To resolve this^^J%
29  error, make sure to use a control sequence^^J%
30  as your emoticon name.^^J}}

```

A definition macro to create robust macros.

```

31 \ifx\protected@\undefined
32   \def\emoticon@robust{\DeclareRobustCommand}
33 \else
34   \def\emoticon@robust{\protected\def}
35 \fi

```

2 Built-in Emoticons

We need to make sure that when we scan emoticon definitions in what is essentially a verbatim environment. Specifically, we want all the ascii punctuation marks to have catcode 11 or 12. I'm assuming all letters have catcode 11—if not, we'll have bigger problems. This is mostly an issue for the usual special characters, but some packages turn other punctuation marks into active characters. We also avoid the possibility of `emoticon` breaking from a more exotic catcode change. We use `\makeatletter` so that we can use a few internal $\TeX$ macros when we define `\@emoticonlist`.

```

36 \def\emoticon@catcodes{%
37   \@makeother\!
38   \@makeother\"
39   \@makeother\#
40   \@makeother\$
41   \@makeother\%
42   \@makeother\&
43   \@makeother\'

```

```

44 \@makeother\(  

45 \@makeother\  

46 \@makeother\  

47 \@makeother\  

48 \@makeother\  

49 \@makeother\  

50 \@makeother\  

51 \@makeother\  

52 \@makeother\  

53 \@makeother\  

54 \@makeother\  

55 \@makeother\  

56 \@makeother\  

57 \@makeother\  

58 \@makeother\  

59 \@makeother\  

60 \@makeother\  

61 \@makeother\  

62 \@makeother\  

63 \@makeother\  

64 \@makeother\  

65 \makeatletter}

```

We come to the built-in emoticon definitions. We call `\emoticon@catcodes` inside a group and set `{` and `}` to catcode 12 because we have a few emoticons that use braces. We still need begin and end group characters, so we set `Q` and `R` to catcodes 1 and 2 respectively. In honor of `doc`'s use of `^^A` as the comment character, we also set `A` to catcode 14. These settings are okay because none of the emoticons contain `A`, `Q`, or `R`.

```

66 \bgroup
67 \catcode`\Q=\@ne      % new (temporary) left brace
68 \catcode`\R=\tw@      % new (temporary) right brace
69 \catcode`\A=14\relax  % new (temporary) percent char
70 \emoticon@catcodes
71 \@makeother\{
72 \@makeother\}

```

Now scan `\@emoticonlist` with `\@firstofone`. We use `\@backslashchar` with manual expansion because it's an easy way to get a `\` inside `\@emoticonlist`.

```

73 \expandafter\egroup
74 \@firstofoneQ
75 \expandafter\def\expandafter\@emoticonlist\expandafterQ\expandafter

```

The raw emoticon definitions.

```

76 \woot          \@backslashchar(^o^)/,A
77 \rabbit        !_!,A
78 \tictactoe     #,A
79 \payday        $_$,A
80 \confused      %),A
81 \confusedn     %-),A

```

82	\varbouquet	%>-,A
83	\steampunk	%_%,A
84	\unphased	'-',A
85	\bang	(!),A
86	\ppayday	(\$_\$),A
87	\psteampunk	(%_%),A
88	\shrine	(&),A
89	\punphased	('-''),A
90	\salute	('-'')>,A
91	\pufferfish	(*),A
92	\lovely	(*^.^*),A
93	\pcrazy	(*_*),A
94	\psoporific	(--_--),A
95	\pderp	(-_-),A
96	\pawake	(._.),A
97	\calling	(._.)?,A
98	\bandage	(::[:]:),A
99	\ptears	(;_;) ,A
100	\fatcat	(=^.^=)~,A
101	\psleepy	(=_=),A
102	\asleep	(=_=)zzz,A
103	\hug	(>._.)>,A
104	\pugh	(>_<),A
105	\riddle	(?),A
106	\pquestioning	(?_?),A
107	\mollusk	(@),A
108	\pamazed	(@_@),A
109	\punsure	(^_~'),A
110	\phappy	(^_~),A
111	\beercup	(_)3,A
112	\teacup	(_)>,A
113	\peyebags	(i_i),A
114	\eyeball	(O),A
115	\phello	(o_o),A
116	\puhoh	(o_0),A
117	\pmonocle	(o_q),A
118	\pglasses	(p_q),A
119	\pteyes	(T_T),A
120	\pdead	(x_x),A
121	\pworried	(~_~),A
122	\star	*,A
123	\marge	****:-),A
124	\barbell	*-* ,A
125	\troll	*8:,A
126	\rooster	*:>,A
127	\santa	*<]:),A
128	\santan	*<]:-),A

129	\crazy	*_*,A
130	\bola	*~*,A
131	\anchor	+~3,A
132	\twig	~,A
133	\wine	-(,A
134	\flower	~*,A
135	\horseshoecrab	~*D,A
136	\stick	--,A
137	\soporific	--_--,A
138	\trident	--E,A
139	\hammer	-;,A
140	\antibody	-<,A
141	\cottoncandy	-<>,A
142	\luminaire	-<D,A
143	\knife	~=,A
144	\derp	-_-,A
145	\fork	-E,A
146	\spoon	-O,A
147	\lollipop	-o,A
148	\martini	-{,A
149	\rocket	.^.,A
150	\awake	._. ,A
151	\cry	: '(,A
152	\cryn	: '-(,A
153	\sad	: (,A
154	\monkey	: (),A
155	\smiley	:),A
156	\mosquito	: -,A
157	\sadr	: -(,A
158	\monkeyn	: -(),A
159	\smiley	: -),A
160	\mehn	: -/,A
161	\catn	: -3,A
162	\branch	: -: ,A
163	\grinn	: -D,A
164	\tonguen	: -P,A
165	\keepsecret	: -X,A
166	\meh	: /,A
167	\cat	: 3,A
168	\logs	: =:,A
169	\bird	: >,A
170	\beetle	: @:,A
171	\grin	: D,A
172	\bug	: O:,A
173	\tongue	: P,A
174	\keepsecret	: X,A
175	\winky	;),A

```

176  \winkyn      ;-),A
177  \jokeyn      ;-D,A
178  \tears       ;_,A
179  \jokey       ;D,A
180  \duckling    <"=,A
181  \wow          <('O')>,A
182  \amazing     <(^o^)>,A
183  \gah          <(`^^)>,A
184  \sunburst    <*>,A
185  \batling     <+>,A
186  \broken      </3,A
187  \heart       <3,A
188  \mouse       <:3(_)_~,A
189  \penguin     <@>,A
190  \paperhat    <],A
191  \goose       <^>,A
192  \evileye     <O>,A
193  \witch       <|:,A
194  \stingray    <|>,A
195  \satellite   =+=,A
196  \comet       =-*,A
197  \sleepy      =_,A
198  \plug        =D~,A
199  \gnat        >,A
200  \crabbling   >"<,A
201  \bonbon      >*<,A
202  \chalice     >-{,A
203  \mole        >.<,A
204  \mad         >:(,A
205  \evil        >:),A
206  \madn        >:-(,A
207  \eviln       >:-),A
208  \devious     >;),A
209  \deviousn    >;-),A
210  \fish        ><))v)'>,A
211  \gnu         >>,A
212  \python      >>>,A
213  \ugh         >_<,A
214  \dragonfly   >j<,A
215  \questioning ?_?,A
216  \snail       @,A
217  \potato      @-*,A
218  \nettle      @-3,A
219  \weevil      @->,A
220  \lion        @.?,A
221  \frog        @:,A
222  \squid       @:-,A

```

223	\peacock	@:>,A
224	\phage	@=C,A
225	\bouquet	@>-,A
226	\rose	@>-->--,A
227	\owl	@@,A
228	\caterpillar	@@:,A
229	\eggs	@@@,A
230	\lizard	@^>,A
231	\amazed	@_@,A
232	\varrose	@}->->-,A
233	\jellyfish	@~,A
234	\fishbones	@~<,A
235	\demon	@~>,A
236	\office	[&],A
237	\moth	[+],A
238	\vartiefighter	[-o-],A
239	\robot	[:],A
240	\powerup	[?],A
241	\sandwich	[],A
242	\bee	^*^,A
243	\coat1	^?^,A
244	\dragon	^@^,A
245	\happy	^_ ^,A
246	\unsure	^_ ^',A
247	\sunglasses	B),A
248	\sunglassesn	B-),A
249	\mushroom	c-,A
250	\pear	C>-,A
251	\bell	D-,A
252	\plunger	D--,A
253	\sconce	D>-,A
254	\bread	E3,A
255	\toast	E3~,A
256	\eyebags	i_i,A
257	\varkilroy	m('u')m,A
258	\kilroy	m(._.)m,A
259	\bat	n(.m.)n,A
260	\human	o+<,A
261	\angel	O:),A
262	\angeln	O:-),A
263	\hello	o_o,A
264	\uhoh	o_O,A
265	\monocle	o_q,A
266	\skateboard	o{<],A
267	\glasses	p_q,A
268	\teyes	T_T,A
269	\crab	V.v.V,A

```

270 \campfire      X*~,A
271 \laughingn    X-D,A
272 \dead         x_x,A
273 \laughing     XD,A
274 \dragonfruit  {*},A
275 \tiefighter   |-o-|,A
276 \street       |:|,A
277 \bridge       |=|,A
278 \chandelier   }-3,A
279 \worm         ~,A
280 \homer        ~(_:-(|),A
281 \polyp        ~*,A
282 \mouseling    ~*: ,A
283 \rat          ~*>,A
284 \alligator    ~*X,A
285 \fox          ~:>,A
286 \candle       ~=,A
287 \dog          ~=*,A
288 \wolf         ~=>,A
289 \gulper       ~=[,A
290 \hammerhead   ~=I,A
291 \shark        ~^>,A
292 \worried      ~_~,A
293 \tea          ~D,A
294 \tadpole      ~o,A
295 \snake        ~~ ,A
296 \viper        ~~<,A
297 \soup         ~~DRR

```

The `\process@emoticon` command defines emoticons. The #1 argument should be a single control sequence, and the #2 argument should be the emoticon itself. The command defines #1 to be a `\protected` macro containing (1) a `\relax` to stop any scanning; (2) a check for math mode; and (3) an `\hbox` with the #2 argument. In math mode, we use `\mathchoice` to select the appropriate font size, and we need braces around the construction to mark it as a single subformula. For example, T_EX complains if we say $\langle emoticon \rangle$ without the braces around `\mathchoice`. The `\leavevmode` is there in case the user calls the emoticon in vertical mode.

```

298 \def\process@emoticon#1#2\@nil{%
299   \emoticon@robust#1{\relax
300     \ifmmode
301       {\mathchoice{\hbox{\fontsize{\tf@size}{\tf@size}\selectfont #2}}%
302         {\hbox{\fontsize{\tf@size}{\tf@size}\selectfont #2}}%
303         {\hbox{\fontsize{\sf@size}{\sf@size}\selectfont #2}}%
304         {\hbox{\fontsize{\ssf@size}{\ssf@size}\selectfont #2}}}%
305     \else
306       \leavevmode\hbox{#2}%
307     \fi}%

```

Same thing except with a monospaced font. We expand the `\csname` and hence the `\string` when `\escapechar` is `-1` to avoid an extra `\` inside the control sequence name. We don't need a `\selectfont` inside the `\hbox` this time because `\ttfamily` contains one already.

```

308 \begingroup
309   \escapechar\m@ne
310   \expandafter
311 \endgroup
312 \expandafter\emoticon@robust\csname tt\string#1\endcsname{\relax
313   \ifmmode
314     {\mathchoice{\hbox{\fontsize{\tf@size}{\tf@size}\ttfamily #2}}%
315       {\hbox{\fontsize{\tf@size}{\tf@size}\ttfamily #2}}%
316       {\hbox{\fontsize{\sf@size}{\sf@size}\ttfamily #2}}%
317       {\hbox{\fontsize{\ssf@size}{\ssf@size}\ttfamily #2}}}%
318   \else
319     \leavevmode\hbox{\ttfamily #2}%
320   \fi}}

```

And then we actually define the emoticons. We loop through `\@emoticonlist` and, for each entry, call `\process@emoticon`.

```

321 \wlog{Package emoticon info: Defining emoticons.}
322 \@for\@i:=\@emoticonlist\do{\expandafter\process@emoticon\@i\@nil}

```

The `\emoticon` macro expects the following token to be a control sequence defined above or through `\process@emoticon`. It checks if $\TeX$ is currently in math mode and if so raises an error. If not, we play a fun game of expansion and gobbling tokens from `\process@emoticon` to leave us with just the contents of the `\hbox`. In order, `\emoticon` needs to

- Expand the next token/emoticon control sequence
- Gobble the initial `\relax`
- Expand `\ifmmode`
- Gobble the `\leavevmode` and `\hbox`
- Use `\@firstoftwo` to remove the braces from the emoticon and get rid of the final `\fi` (which would expand to nothing on its own but no reason to avoid doing it now)

As mentioned in the user guide, `\emoticon` is not compatible with the `\tt` emoticon control sequences. Using it with them will add an extra `\ttfamily` to whatever you're doing, which is probably bad if you needed to use `\emoticon` in the first place. It would be great to add some error checking here to make sure that `#1` is actually an emoticon control sequence, but I'm not sure how to do that easily and still leave `\emoticon` fully expandable.

```

323 \def\emoticon#1{%
324   \ifmmode
325     \expandafter\EmoticonExtractMathError
326   \else
327     \expandafter\expandafter\expandafter
328     \extract@emoticon\expandafter\gobble#1%
329   \fi}
330 \def\extract@emoticon{\expandafter\expandafter\expandafter
331   \@firstoftwo\expandafter\gobbletwo}

```

3 New Emoticons

Before anything else, we define a helper macro. This command appears in recent versions of the 2_ε kernel, but less recent versions may not have it.

```
332 \long\def\@gobblethree#1#2#3{}
```

The `\DeclareEmoticon` command allows the user to define new emoticons. It takes two arguments:

- **#1**: a single control sequence
- **#2**: a string of (ascii) characters

Technically, the definition only has a **#1** argument because we scan what would be the **#2** argument later with the correct catcodes using `\sc@nem@tic@n` (or gobble it if there's a problem). Because the package is non- ϵ -TeX friendly, the parameter specification for `\DeclareEmoticon` is different depending on the engine. We store [1] or ##1 in `\@tempa` and then expand it as parameter text before defining `\DeclareEmoticon`.

```
333 \edef\@tempa{\ifx\protected\@undefined[1]\else##1\fi}
```

```
334 \expandafter\emoticon@robust\expandafter\DeclareEmoticon\@tempa{%
```

First is error checking, which uses `\count@`. When `\count@` stays 0, that means we can use **#1** for a new emoticon, and when `\count@` ends up positive, that means we encountered an error. We start by checking that **#1** is a single control sequence.

```
335 \count@\z@
```

```
336 \expandafter\ifx\expandafter\@nnil\@gobble#1\@nnil % is single token?
```

```
337 \ifcat\relax\noexpand#1% is a cs?
```

If **#1** is a single control sequence, we can use `\ifin@` to check whether it appears in `\@emoticonlist`. If yes, we are redefining an emoticon that already exists, and that's fine. If no, we check whether **#1** and `\tt#1` are both undefined because the package should not automatically overwrite any definitions, and if both control sequences are free, we are good to go.

```
338 \expandafter\in@\expandafter#1\expandafter{\@emoticonlist}
```

```
339 \ifin@ % is \cs already an emoticon?
```

```
340 \wlog{Package emoticon info: Redefining the \string#1 emoticon.}
```

```
341 \else % \cs is not an emoticon
```

```
342 \ifx#1\@undefined % is \cs undefined?
```

If **#1** is not already an emoticon or a defined control sequence, we need to check that `\tt#1` is undefined to prevent something like `\family` or `\default` from becoming an emoticon. We extract the name of **#1** by expanding `\string#1` inside `\@tempa` when `\escapechar` is -1, and then we redefine `\@tempa` to contain `\tt#1`.

```
343 \begingroup
```

```
344 \escapechar\m@ne
```

```
345 \expandafter
```

```
346 \endgroup
```

```
347 \expandafter\def\expandafter\@tempa\expandafter{\string#1}
```

```
348 \edef\@tempa{\expandafter\noexpand\csname tt\@tempa\endcsname}
```

```
349 \expandafter\ifx\@tempa\@undefined % is \ttcs undefined?
```

```
350 \wlog{Package emoticon info: Defining the
```

```
351 \string#1\space emoticon.}
```

Update \@emoticonlist with the new emoticon control sequence.

```
352      \expandafter\def\expandafter\@emoticonlist\expandafter
353      {\@emoticonlist#1}
```

Now we set \count@ to the correct value depending on how the previous sequence of ifs played out.

```
354      \else % \ttcs already defined
355      \count@\@ne
356      \fi
357      \else % \cs already defined
358      \count@\tw@
359      \fi
360      \fi
361      \else % not a cs
362      \count@\thr@@
363      \fi
364      \else % cs has multiple tokens
365      \count@=4\relax
366      \fi
```

The last part of \DeclareEmoticon sets up code to process the emoticon. Inside a group, we set \emoticon@catcodes and check the value of \count@. If \count@ is 0, we're okay to define the new emoticon. We (temporarily) redefine \, \{, and \} to contain catcode-12 versions of each character, and then we call \sc@nem@tic@n, which scans and defines the new emoticon.

```
367      \bgroup
368      \emoticon@catcodes
369      \ifcase\count@
370      \let\\ \@backslashchar
371      \let\{ \@charlb
372      \let\} \@charrb
373      \def\@tempa{#1}
```

And here's where the magic happens. We store the control sequence that will become the emoticon name in \@tempa. We hit the \or with an \expandafter first, then gobble the last three tokens in the definition of \DeclareEmoticon, so \sc@nem@tic@n can process the emoticon definition.

```
374      \expandafter\expandafter\expandafter
375      \sc@nem@tic@n\expandafter\@gobblethree
```

For all other cases, we raise an error message and then (1) finish expanding the \ifcase; (2) gobble what would be the #2 argument of \DeclareEmoticon; and (3) close the group. Gobbling happens outside the conditional, and it must be before \egroup so that % has the right catcode. Cases 1 and 2 are when \tt#1 or #1 is already defined.

```
376      \or % if \count@ = 1
377      \EmoticonDefinedError{\string#1}
378      {\expandafter\string\@tempa}
379      \or % if \count@ = 2
380      \EmoticonDefinedError{\string#1}
381      {\string#1}
```

Cases 3 and 4 are, respectively, if #1 is not a control sequence or if #1 contains multiple tokens.

```

382   \or % if \count@ = 3
383     \EmoticonNotCSError{\string#1}
384   \or % if \count@ = 4
385     \def\@tempa{#1}
386     \@onelevel@sanitize\@tempa
387     \EmoticonMultiTokensError{\@tempa}
388   \fi
389   \expandafter\egroup\@gobble}

```

The command to scan the new emoticon. Here #1 is what would be the second argument of `\DeclareEmoticon`. The macro expands #1 inside an `\edef` in case it contains `\\`, `\{`, or `\}` (with their temporary new definitions from `\DeclareEmoticon`) and includes an `\egroup` to close the group from `\DeclareEmoticon`. Would it be better to use `\scantokens` in the scanning macro in case someone declares an emoticon inside a macro? Maybe, but I think it's okay to not support that functionality. (Currently `\@tempa` holds the control sequence that will be the emoticon name.)

```

390 \def\sc@nem@tic@n#1{\egroup
391   \edef\@tempb{%
392     \noexpand\process@emoticon
393     \expandafter\noexpand\@tempa
394     #1\noexpand\@nil}
395   \@tempb}

```

Things will probably be fine if we don't restrict `\DeclareEmoticon` to the preamble, but it has the flavor of a preamble-only command.

```

396 \@onlypreamble\DeclareEmoticon
397 \@onlypreamble\process@emoticon
398 \@onlypreamble\sc@nem@tic@n

```

Done!

Version History

New features and changes from each version. Listed in no particular order.

1.0 August 2026
—initial release

Code Index

Entries refer to lines in the code, and bold means a definition.

Symbols	
<code>\@backslashchar</code>	370
<code>\@charlb</code>	371
<code>\@charrb</code>	372
<code>\@emoticonlist</code>	75, 322, 338, 352, 353
<code>\@extract@emoticon</code>	328, 330
<code>\@firstoftwo</code>	331
<code>\@makeother</code>	37–64, 71, 72
<code>\@</code>	370
<code>\{</code>	71, 371
<code>\}</code>	72, 372
A	
<code>\A</code>	69
C	
<code>\catcode</code>	67–69
D	
<code>\DeclareEmoticon</code> ..	5, 13, 17, 26, 334, 396
<code>\DeclareRobustCommand</code>	32
E	
<code>\emoticon</code>	19, 20, 22, 323
<code>\emoticon@catcodes</code>	36 , 70, 368
<code>\emoticon@robust</code>	32 , 34 , 299, 312, 334
<code>\EmoticonDefinedError</code>	3 , 377, 380
<code>\EmoticonExtractMathError</code>	18 , 325
<code>\EmoticonMultiTokensError</code>	11 , 387
<code>\EmoticonNotCSError</code>	24 , 383
F	
<code>\fontsize</code>	301–304, 314–317
M	
<code>\makeatletter</code>	65
<code>\mathchoice</code>	301, 314
P	
<code>\process@emoticon</code>	298 , 322, 392, 397
Q	
<code>\Q</code>	67
R	
<code>\R</code>	68
S	
<code>\sc@nem@tic@n</code>	375, 390 , 398
<code>\sf@size</code>	303, 316
<code>\ssf@size</code>	304, 317
T	
<code>\tf@size</code>	301, 302, 314, 315
<code>\ttfamily</code>	314–317, 319

Emoticon Index

Entries refer to lines in the code where a given emoticon name appears in \@emoticonlist.

A			
\alligator	284	\crab	269
\amazed	231	\crabbling	200
\amazing	182	\crazy	129
\anchor	131	\cry	151
\angel	261	\cryn	152
\angeln	262	D	
\antibody	140	\dead	272
\asleep	102	\demon	235
\awake	150	\derp	144
B		\devious	208
\bandage	98	\deviousn	209
\bang	85	\dog	287
\barbell	124	\dragon	244
\bat	259	\dragonfly	214
\batling	185	\dragonfruit	274
\bee	242	\duckling	180
\beercup	111	E	
\beetle	170	\eggs	229
\bell	251	\evil	205
\bird	169	\evileye	192
\bola	130	\eviln	207
\bonbon	201	\eyebags	256
\bouquet	225	\eyeball	114
\branch	162	F	
\bread	254	\fatcat	100
\bridge	277	\fish	210
\broken	186	\fishbones	234
\bug	172	\flower	134
C		\fork	145
\calling	97	\fox	285
\campfire	270	\frog	221
\candle	286	G	
\caterpillar	228	\gah	183
\catn	161	\glasses	267
\chalice	202	\gnat	199
\chandelier	278	\gnu	211
\coat1	243	\goose	191
\comet	196	\grin	171
\confused	80	\grinn	163
\confusedn	81	\gulper	289
\cottoncandy	141		

H		N	
\hammer	139	\nettle	218
\hammerhead	290		
\happy	245	O	
\heart	187	\office	236
\hello	263	\owl	227
\homer	280		
\horseshoecrab	135	P	
\hug	103	\pamazed	108
\human	260	\paperhat	190
		\pawake	96
J		\payday	79
\jellyfish	233	\pcrazy	93
\jokey	179	\pdead	120
\jokeyn	177	\pderp	95
		\peacock	223
K		\pear	250
\keepsecret	174	\penguin	189
\keepsecretn	165	\peyebags	113
\kilroy	258	\pglasses	118
\knife	143	\phage	224
		\phappy	110
L		\phello	115
\laughing	273	\plug	198
\laughingn	271	\plunger	252
\lion	220	\pmonocle	117
\lizard	230	\polyp	281
\logs	168	\potato	217
\lollipop	147	\powerup	240
\lovely	92	\ppayday	86
\luminaire	142	\pquestioning	106
		\psleepy	101
M		\psoporific	94
\mad	204	\psteampunk	87
\madn	206	\ptears	99
\marge	123	\pteyes	119
\martini	148	\pufferfish	91
\meh	166	\pugh	104
\mehn	160	\puhoh	116
\mole	203	\punphased	89
\mollusk	107	\punsure	109
\monkey	154	\pworried	121
\monkeyn	158	\python	212
\monocle	265		
\mosquito	156	Q	
\moth	237	\questioning	215
\mouse	188		
\mouseling	282	R	
\mushroom	249	\rabbit	77

<code>\rat</code>	283
<code>\riddle</code>	105
<code>\robot</code>	239
<code>\rocket</code>	149
<code>\rooster</code>	126
<code>\rose</code>	226

S

<code>\sad</code>	153
<code>\sadm</code>	157
<code>\salute</code>	90
<code>\sandwich</code>	241
<code>\santa</code>	127
<code>\santan</code>	128
<code>\satellite</code>	195
<code>\sconce</code>	253
<code>\shark</code>	291
<code>\shrine</code>	88
<code>\skateboard</code>	266
<code>\sleepy</code>	197
<code>\smiley</code>	155
<code>\smileyn</code>	159
<code>\snail</code>	216
<code>\snake</code>	295
<code>\soporific</code>	137
<code>\soup</code>	297
<code>\spoon</code>	146
<code>\squid</code>	222
<code>\star</code>	122
<code>\steampunk</code>	83
<code>\stick</code>	136
<code>\stingray</code>	194
<code>\street</code>	276
<code>\sunburst</code>	184
<code>\sunglasses</code>	247
<code>\sunglassesn</code>	248

T

<code>\tadpole</code>	294
-----------------------------	-----

<code>\tea</code>	293
<code>\teacup</code>	112
<code>\tears</code>	178
<code>\teyes</code>	268
<code>\tictactoe</code>	78
<code>\tiefighter</code>	275
<code>\toast</code>	255
<code>\tongue</code>	173
<code>\tonguen</code>	164
<code>\trident</code>	138
<code>\troll</code>	125
<code>\twig</code>	132

U

<code>\ugh</code>	213
<code>\uhoh</code>	264
<code>\unphased</code>	84
<code>\unsure</code>	246

V

<code>\varbouquet</code>	82
<code>\varkilroy</code>	257
<code>\varrose</code>	232
<code>\vartiefighter</code>	238
<code>\viper</code>	296

W

<code>\weevil</code>	219
<code>\wine</code>	133
<code>\winky</code>	175
<code>\winkyn</code>	176
<code>\witch</code>	193
<code>\wolf</code>	288
<code>\woot</code>	76
<code>\worm</code>	279
<code>\worried</code>	292
<code>\wow</code>	181