Markdown & Pygments Lexers Cheat Sheet

Esta es una guía que me sirve para recordar todas las posibilidades que ofrecen markdown y Pygments para editar y formatear texto y que empleo para crear los artículos de este blog. Está redactada de forma que no solo me sirva de guía a mí, si no a cualquiera que se acerque por primera vez a markdown o Pygments.

§

Lo que sigue a continuación es una lista detallada de todas las características que se pueden emplear en Markdown y Markdown Extra (empleando *Python Markdown*) y los lexers más comunes de Pygments para resaltar el código fuente.

- Markdown
  - ¿Que es Markdown?
- Sintaxis Markdown
  - Cabeceras
  - Enlaces
  - Parrafos
  - Formato
  - Citas
  - Listas
  - Listas de definiciones
  - Imágenes
  - Tablas
  - Código
  - Lineas Horizontales
  - Escapar caracteres
  - Notas a pie de página
  - Abreviaturas
  - Indentificadores de cabecera
- Pygments
  - Lexers de Pygments más comunes para resaltado de sintaxis

Markdown

Este es el lenguaje de marcado empleado para crear este sitio, que permite formatear el texto fácilmente sin la necesidad de emplear el más engorroso HTML o emplear un editor visual.

¿Que es Markdown?

Markdown es un lenguaje de marcado ligero parecido al que se emplea en muchas wikis y basado originalmente en convenciones existentes en el marcado de los los correos electronicos. Emplea texto plano, procurando que sea legible pero consiguiendo que se convierta en XHTML correctamente formateado. Los artículos de este sitio están elaborados empleando markdown, sin utilizar ningún tipo de editor visual WYSIWYG, lo que facilita el crear documentos XHTML limpios y fácilmente editables en el futuro. Son un buen ejemplo de las capacidades de Markdown. Aunque no es muy conocido, empieza a ser muy popular y utilizado entre los programadores.

Para conocer más sobre markdown, se pueden leer los artículos en los que explico porque es el más adecuado para crear un blog y porque lo he elegido para este sitio, **artículos markdown**

§

Sintaxis Markdown

Cabeceras

Los encabezamientos HTML se producen colocando un número determinado de almohadillas # antes del texto correspondiente al nivel de encabezamiento deseado (HTML ofrece hasta seis niveles). Los encabezamientos posibles se pueden ver en la siguiente tabla:

Tecleas	Obtienes
`# Esto es un H1`	# Esto es un H1
`## Esto es un H2`	## Esto es un H2
`### Esto es un H3`	### Esto es un H3
`#### Esto es un H4`	#### Esto es un H4

```
##### Esto es un H5
```

Esto es un H5

```
###### Esto es un H6
```

Esto es un H6

Se puede encerrar cada encabezado entre almohadillas, por motivos puramente estéticos, porque no es necesario en absoluto, es decir, se puede hacer esto:

Tecleas	Obtienes
`### Esto es un H3 ###`	### Esto es un H3

Para los encabezamientos de los dos primeros niveles existe también otra manera de hacer lo mismo, que sería la siguiente:

Tecleas	Obtienes
`Esto es un H1` `==========`	# Esto es un H1
`Esto es un H2` `----------`	## Esto es un H2

Es decir para los encabezamientos principales se subraya el texto con el signo igual. Para los encabezamientos de segundo nivel se utilizan guiones para subrayar. Es indiferente el número de signos iguales o guiones que se empleen, con uno es suficiente.

§

Enlaces

Existen también dos maneras de crear enlaces, se pueden ver en la siguiente tabla:

Tecleas	Obtienes
`[Con titulo](http://joedicastro.com "titulo")`	**Con titulo**
`[Sin titulo](http://joedicastro.com)`	**Sin titulo**

```
[Enlace 1][1], [Enlace 2][2], [Enlace 3][3]

 [1]: http://joedicastro.com/consejos
 [2]: http://joedicastro.com/consejos "Consejos"
 [3]: http://joedicastro.com/
```

Enlace 1, **Enlace 2**, **Enlace 3**

Existe una manera adicional de crear enlaces automáticos para direcciones URL, simplemente encerrarla entre los caracteres menor < que y mayor que >:

Tecleas	Obtienes
`<http://joedicastro.com>`	**http://joedicastro.com**

§

Párrafos

Para crear párrafos se deja una línea en blanco. De este modo.

Tecleas	Obtienes

`Este es el primer párrafo.`
 	
`Este es el segundo párrafo.`
	Este es el primer párrafo.
	
	Este es el segundo párrafo

Para crear un salto de línea dentro de un párrafo, simplemente se dejan dos espacios al final de la última palabra de esa línea, de este modo:

Tecleas	Obtienes
`Esta es la primera línea` `y este es el salto de línea.`	Esta es la primera línea y este es el salto de línea.

§

Formato

El formato básico del texto, es decir negritas y cursiva, se pueden realizar de varias maneras:

Tecleas	Obtienes
`**Esto es negrita**`	**Esto es negrita**

`__Esto también es negrita__`	**Esto también es negrita**
`*Esto es cursiva*`	*Esto es cursiva*
`_Esto también es cursiva_`	*Esto también es cursiva*
`***Esto es negrita y cursiva***`	***Esto es negrita y cursiva***
`___Esto también es negrita y cursiva___`	***Esto también es negrita y cursiva***

Se pueden emplear indistintamente tanto el asterisco * como el guión bajo _ siempre y cuando no se mezclen y lo que determina el formato es el número de ellos antes y después del bloque de texto a formatear. Uno es cursiva, dos es negrita, y tres ambas a la vez, así de sencillo.

§

Citas

Para crear bloques de cita, se emplea el carácter mayor que > antes del bloque de texto. En la siguiente tabla se pueden ver las opciones para crearlos.

Tecleas	Obtienes

`Esto es una línea normal`
 	
`> Esto es parte de un bloque de cita.`
`> Esto es parte del mismo bloque de cita.`
	Esto es una línea normal
	
	> *Esto es parte de un bloque de cita. Esto es parte del mismo bloque de cita.*

`> Esto es parte de un bloque de cita.`
`Esto continúa el bloque incluso aunque no hay símbolo 'mayor que'.`
 	
`La línea en blanco finaliza el bloque.`
	> *Esto es parte de un bloque de cita. Esto continúa el bloque incluso aunque no hay símbolo 'mayor que'.*
	
	La línea en blanco finaliza el bloque.

`Esto es una línea normal`
 	
`> Esto es parte de un bloque de cita.`
	Esto es una línea normal
	
	> *Esto es parte de un bloque de cita. Esto es parte*

```
> Esto es parte del mismo bloque de cita.
>
> > Esto es otro bloque de cita anidado.
> > Esto es parte del bloque anidado.
>
> Esto es parte del bloque de cita de
primer nivel.
```

> *Esto es parte del mismo bloque de cita.*
>
> > *Esto es otro bloque de cita anidado. Esto es parte del bloque anidado.*
>
> *Esto es parte del bloque de cita de primer nivel.*

§

Listas

Markdown permite crear dos tipos de listas, ordenadas y desordenadas, es decir numeradas o listas de puntos. Para distinguir los tipos y como se crean, nada mejor que verlo con ejemplos:

Tecleas

```
Lista numerada (ordenada)

1. Este es el primer elemento
2. Este es el segundo elemento
3. Este es el tercer elemento
```

Obtienes

Lista numerada (ordenada)

1. Este es el primer elemento
2. Este es el segundo elemento
3. Este es el tercer elemento

```
Lista de puntos (desordenada)

* Un elemento de la lista
* Otro elemento de la lista
* El tercer elemento de la lista
```

Lista de puntos (desordenada)

- Un elemento de la lista
- Otro elemento de la lista
- El tercer elemento de la lista

```
Se pueden emplear también + y - en vez
de *

* Un elemento de la lista
+ Otro elemento de la lista
- El tercer elemento de la lista
```

Se pueden emplear también + y - en vez de *

- Un elemento de la lista
- Otro elemento de la lista
- El tercer elemento de la lista

```
Se pueden mezclar distintos tipos de
listas y anidar unas dentro de otras.

1. Esto es una lista ordenada
2. Segundo elemento de la lista
ordenada
```

Se pueden mezclar distintos tipos de listas y anidar unas dentro de otras.

1. Esto es una lista ordenada
2. Segundo elemento de la lista ordenada

```
    1. Esta es una lista ordenada
anidada dentro de otra
        * Lista desordenada anidada a
tercer nivel
        * Segundo elemento de esta
lista
    2. Este es el segundo elemento de
la lista ordenada anidada
```

1. Esta es una lista ordenada anidada dentro de otra
    - Lista desordenada anidada a tercer nivel
    - Segundo elemento de esta lista
2. Este es el segundo elemento de la lista ordenada anidada

§

Listas de definiciones

Se pueden crear lista de definiciones, que están compuestas de términos y las definiciones de los mismos, como si fuera un diccionario. Su creación es muy sencilla:

Tecleas

```
Primer termino
 : Primera definición

Segundo termino
 : Segunda definición
```

Obtienes

Primer término
: Primera definición
Segundo término
: Segunda definición

```
Se pueden aplicar más de una definición
a un termino

Primer termino
 : Primera definición
 : Segunda definición

Segundo termino
 : Segunda definición
```

Se pueden aplicar más de una definición a un termino

Primer término
: Primera definición
: Segunda definición
Segundo término
: Segunda definición

```
Se pueden aplicar más de un termino a
una definición

Primer termino
Segundo termino
 : Primera definición

Tercer termino
 : Segunda definición
```

Se pueden aplicar más de una definición a un termino

Primer término
Segundo término
: Primera definición
Tercer término
: Segunda definición

```
Si dejamos una línea en blanco entre el
termino y la definición, se creara un
párrafo para la definición.

Primer termino

: Primera definición

Segundo termino
 : Segunda definición
```

Si dejamos una línea en blanco entre el termino y la definición, se creara un párrafo para la definición.

Primer termino
: Primera definición

Segundo termino
: Segunda definición

```
Una definición puede constar de varios
párrafos.

Primer termino
 : Primera definición

Segundo párrafo de la primera
definición

Segundo termino
 : Segunda definición
```

Una definición puede constar de varios párrafos.

Primer término
: Primera definición

: Segundo párrafo de la primera definición

Segundo término
: Segunda definición

§

Imágenes

La manera de enlazar imágenes es básicamente la misma de crear enlaces, con un única diferencia, se añade el carácter exclamación **!** al principio de la pareja de corchetes que definen el nombre del enlace. Ejemplos:

Tecleas	Obtienes
`![Con titulo](pictures/avatar.png "titulo")`	
`![Sin titulo](pictures/avatar.png)`	

```
![Imagen 1][1]  ![Imagen 2][2]
```

```
[1]: pictures/avatar.png
[2]: pictures/scaphandre.png "scaphandre"
```

§

Tablas

Crear tablas es sumamente sencillo, simplemente debemos indicar cuales son los elementos de la cabecera y separar los campos con el símbolo |

Tecleas

```
Cabecera A | Cabecera B
-- | --
Campo A0 | Campo B0
Campo A1 | Campo B1
```

Obtienes

Cabecera A	Cabecera B
Campo A0	Campo B0
Campo A1	Campo B1

Si se desea, por estética, se pueden alinear las columnas e incluso comenzar y finalizar las filas con el símbolo |, pero no es en absoluto necesario.

Tecleas

```
| Cabecera A | Cabecera B |
| ---------- | ---------- |
| Campo A0   | Campo B0   |
| Campo A1   | Campo B1   |
```

Obtienes

Cabecera A	Cabecera B
Campo A0	Campo B0
Campo A1	Campo B1

Se puede especificar la alineación de cada columna mediante la adición de dos puntos a las líneas de separación. Dos puntos a la izquierda de la línea de separación hará que la columna esté alineada a la izquierda, dos puntos a la derecha de la línea hará que la columna esté alineada a la derecha, dos puntos en ambos lados significa que la columna se alinea al centro.

Tecleas

```
| Elemento | Cantidad | Precio |
| :------- | :------: | -----: |
| Item 1   | 15       | 150€   |
| Item 2   | 3250     | 23,65€ |
```

Obtienes

Elemento	Cantidad	Precio
Item 1	15	150€
Item 2	3250	23,65€

§

§

Código

Se pueden crear bloques de código para albergar extractos de código fuente de un lenguaje de programación o para reproducir literalmente cualquier tipo de texto sin que sea interpretado por markdown. Lo único necesario es que cada línea de este bloque empiece por al menos 4 espacios o 1 tabulado.

De todos modos, es mucho más recomendable para estas tareas emplear el resaltado de código que se especifica en esta sección.

Tecleas

```
Esto es un párrafo normal.

    Esto es un párrafo de código.
```

Obtienes

Esto es un párrafo normal.

```
Esto es un párrafo de código.
```

Existe otro modo de crear un bloque de código, encerrándolo entre dos líneas formadas por tres o más caracteres tilde ~

Tecleas

```
Esto es un párrafo normal

~~~
Esto es un párrafo de código.
~~~
```

Obtienes

Esto es un párrafo normal.

```
Esto es un párrafo de código.
```

Por último existe una opción para resaltar pequeños trozos de código dentro de párrafos de texto normal. Para lograr esto debemos encerrar el código entre dos acentos graves `

Tecleas

```
Esto es un párrafo normal, con un trozo de código, `import this` insertado en el medio del mismo.
```

Obtienes

Esto es un párrafo normal, con un trozo de código, `import this` insertado en el medio del mismo.

§

Líneas Horizontales

Para crear líneas horizontales se debe crear una línea rodeada de líneas en blanco y compuesta por 3 o más símbolos, que pueden ser guiones, asteriscos o guiones bajos. Pueden crearse espacios entre estos caracteres si así se desea por estética.

Tecleas	Obtienes
`***`	
`- - -`	
`___`	

Escapar carácteres

¿Que ocurre cuando queremos mostrar un carácter que markdown emplea para el marcado? Es posible que dependiendo de donde y como se emplee este símbolo, sea interpretado por markdown y nos estropee el formato del texto. En este caso lo que se necesita es *escapar* el carácter con el símbolo backslash \ En esta tabla se muestran los símbolos que pueden ser escapados por markdown.

Tecleas	Obtienes
`\\ \` * _ \{\} \[\] \(\) \# \+ \- \. \! \: |`	\ ` * _ {} [] () # + - . ! : |

Notas a pie de página

Las notas de página se crean de una manera muy sencilla en Markdown. Cada nota de pie de página se compone de dos elementos: un marcador al lado del texto que se convierte en un superíndice y de una definición que se puede colocar en una lista de notas al pie al final de documento. Ejemplo:

Tecleas	Obtienes

```
Esto es un texto con nota al pie [^1]

[^1]: Esto es una nota al pie de
página.
```

Esto es un texto con nota al pie [1]

§

1. Esto es una nota al pie de página. ↩

Las definiciones de la nota al pie se pueden encontrar en cualquier parte del documento, pero las notas siempre se mostrarán en el orden en que están vinculados en el texto. Hay que tener en cuenta que no puede hacer dos enlaces a la misma nota al pie: si se intenta, la referencia de la nota segunda quedará como texto sin formato.

Cada marcador de nota debe tener un nombre distinto. Ese nombre se utiliza para vincular la nota a la que hace referencia a las definiciones de la nota, pero no tiene ningún efecto sobre la numeración de las notas al pie. Los nombres pueden contener cualquier carácter válido que sirva para la una Identificación de un atributo HTML (es decir, que cumpla con la expresión regular `[A-Za-z][-A-Za-z0-9_:.]*`), no tienen porque ser necesariamente números. Ejemplo:

Tecleas

```
Esto es un texto con nota al pie [^nota1] y
esta es otra nota [^nota2]

[^nota1]: Esto es una nota al pie de
página.
[^nota2]: Esto es la segunda nota al pie.
```

Obtienes

Esto es un texto con nota al pie [1] y esta es otra nota [2]

§

1. Esto es una nota al pie de página. ↩
2. Esto es la segunda nota al pie. ↩

§

Abreviaturas

Para crear abreviaturas HTML lo único necesario es crear una lista de ellas (normalmente al final del texto) y en cualquier lugar del texto que aparezca la abreviatura se aplicará automáticamente. Las listas de abreviaturas se crean como las listas de enlaces, pero precedidas por un asterisco.

Tecleas

```
La especificación HTML es mantenida por el W3C.

*[HTML]: Hyper Text Markup Language
 *[W3C]:  World Wide Web Consortium
```

Obtienes

La especificación HTML es mantenida por el W3C.

Las abreviaturas son sensibles a mayúsculas, por lo que hay que tenerlo en cuenta. Se pueden crear abreviaturas de más de una palabra.

§

Identificadores de Cabecera

Los identificadores de cabecera nos permiten establecer un Identificador a las cabeceras para luego poder enlazarlas en cualquier otro lugar del texto. Es lo que empleo para crear el índice de esta página. Funcionaría como un *anchor* HTML (ancla) pero que solo se puede aplicar en las cabeceras.

Tecleas

```
### Esto es una cabecera con un Id {#cabecera1}

[Enlace a esa cabecera](#cabecera1)
```

Obtienes

Esto es una cabecera con un Id

Enlace a esa cabecera

En Markdown Python todas las cabeceras llevan por defecto asociado un Id que depende del texto de la misma, aunque siempre prevalece la que nosotros establezcamos.

§

Pygments: Resaltado de Sintaxis para Código Fuente

Para introducir ejemplos de código fuente en el sitio, habilitar el resaltado (o coloreado) de sintaxis mejora la presentación y legibilidad de los mismos. Existen diversos motores que nos permiten realizar esta función y Pygments es uno de los mejores. Está realizado en Python, por lo que se integra perfectamente con el software que genera este sitio y con python markdown.

Resaltar código con markdown y Pygments es realmente sencillo, solamente hay que hacer los mismo que haríamos com markdown, pero añadiendo un **lexer** de Pygments en la primera línea. Un **lexer** es un identificador del lenguaje que queremos resaltar para que el coloreado se haga correctamente. Los lexer se construyen empleando 2 caracteres : seguidos del nombre del lexer, por ejemplo, `:::python` sería el lexer empleado para identificar un fragmento de código en lenguaje Python.

Lo podemos ver mejor con un ejemplo:

Tecleas

Obtienes

```
:::python
```

```
    import lifetime

    for each_day in lifetime.days():
        carpe_diem()
```

```
import lifetime

for each_day in lifetime.days():
    carpe_diem()
```

Lexers de Pygments más comunes para resaltado de sintaxis

A continuación muestro una relación de los lexers más comunes empleados para el resaltado de código fuente.

- `apache` - configuración Apache

```
<VirtualHost *:80>
DocumentRoot /www/example1
ServerName www.example1.com

# Other directives here

</VirtualHost>
```

- `bash` y `console` - Bash y Shell

```
 #!/bin/bash
 echo "Hola mundo"
```

- `bat` - Fichero Batch DOS/Windows

```
@echo ¡Hola, Mundo!
```

- `boo` - Boo

```
print "Hello, world!"
```

- `c` - C

```
#include <stdio.h>

int main()
{
    printf("¡Hola, mundo!\n");
    return 0;
}
```

- cpp - C++

```
#include <iostream.h>
using namespace std;

int main() {
  cout << "¡Hola, mundo!" << endl;
  return 0;
}
```

- csharp - C

```
using System;

class MainClass
{
    public static void Main()
    {
        System.Console.WriteLine("¡Hola, mundo!");
    }
}
```

- css - Cascade Style Sheet (CSS)

```
</pre>
   </td>
   <td class="get">
<css>
body {
    font: 75% georgia, sans-serif;
    color: #555753;
    background: #fff;
```

```
    margin: 0;
    padding: 5px;
}
```

- `diff` ó `udiff` - Diff

```
--- /path/to/original ''timestamp''
+++ /path/to/new      ''timestamp''
@@ -1,3 +1,9 @@
+This is an important
+notice! It should
+therefore be located at
+the beginning of this
+document!
+
 This part of the
 document has stayed the
 same from version to
```

- `erlang` - Erlang

```
-module (hola).
-export([hola_mundo/0]).

hola_mundo() -> io:fwrite("Hola mundo!\n").
```

- `go` - Go

```
package main

import "fmt"

func main() {
   fmt.Println("Hello World!")
}
```

- `haskell` - Haskell

```
holaMundo :: IO ()
```

```
holaMundo = putStrLn "Hola mundo!"
```

- `html` - HTML

```
<html>
  <head>
    <title>Hola Mundo</title>
  </head>
  <body>

¡Hola Mundo!
   </body>
</html>
```

- `java` - Java

```
public class HolaMundo {
       public static void main(String[] args) {
          System.out.println("¡Hola, mundo!");
       }
}
```

- `js` - javascript

```
<script type="text/javascript">
  document.write("¡Hola, mundo!");
</script>
```

- `latex` - LaTeX

```
\documentclass[12pt]{article}
\usepackage{lingmacros}
\usepackage{tree-dvips}
\begin{document}

\section*{Notes for My Paper}
```

- `cl` - Common Lisp

```
(format t "¡Hola, mundo!")
```

- `lua` - Lua

```
print("¡Hola, Mundo!\n")
```

- `mysql` - MySQL

```
SELECT 'HOLA MUNDO';
```

- `pascal` y `delphi` - Pascal y Delphi

```
Program HolaMundo;
Begin
    Write('¡Hola, Mundo!');
End.
```

- `perl` - Perl

```
print "Hola, mundo\n"
```

- `php` - PHP

```
<?php print "Hola Mundo!"; ?>
```

- `python` ó `py` ó `pycon` ó `pytb` ó `python3` ó `cython` - Python

```
print "¡Hola Mundo!"
```

- `ruby` - Ruby

```
puts "Hola Mundo"
```

- `scala` - Scala

```scala
object HelloWorld extends Application {
  println("Hello world!")
}
```

- `scheme` - Scheme

```scheme
(display "Hello World")
```

- `smalltalk` - Smalltalk

```smalltalk
Transcript show: '¡Hola, mundo!'
```

- `sql` - SQL

```sql
SELECT 'HOLA MUNDO'
FROM DUAL;
```

- `sqlite3` - sqlite3

```
sqlite> CREATE TABLE tbl2 (
   ...>   f1 varchar(30) primary key,
   ...>   f2 text,
   ...>   f3 real
   ...> );
sqlite>
```

- `text` - Texto simple monoespaciado

```
Hola Mundo
```

- `vala` - Vala

```vala
class Demo.HelloWorld : GLib.Object {
```

```
    public static int main(string[] args) {
        stdout.printf("Hello, World\n");
        return 0;
    }
}
```

- `vbnet` - Visual Basic .NET

```
Private Sub Form_Load()
   Msgbox "Hola Mundo"
 End Sub
```

- `vim` - Vim Script

```
function! ToggleSyntax()
   if exists("g:syntax_on")
      syntax off
   else
      syntax enable
   endif
endfunction

nmap <silent>  ;s  :call ToggleSyntax()<CR>
```

- `xml` - XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 - <note>
      <to>Tove</to>
      <from>Jani</from>
      <heading>Reminder</heading>
      <body>Don't forget me this weekend!</body>
   </note>
```