Aprende X en Y minutos

Comparte esta página

Donde X=yaml

Descarga el código: learnyaml-es.yaml

Tutorial de YAML en español.

YAML es un lenguaje de serialización de datos diseñado para ser leído y escrito por humanos.

Basa su funcionalidad en JSON, con la adición de líneas nuevas e indentación inspirada en Python. A diferencia de Python, YAML no permite tabulaciones literales.

```
# Los comentarios en YAML se ven así.

###################
# TIPOS ESCALARES #
###################

# Nuestro objeto raíz (el cual es el mismo a lo largo de todo el
# documento) será un mapa, equivalente a un diccionario, hash,
# u objeto en otros lenguajes.

llave: valor
```

```yaml
otra_llave: Otro valor
un_valor_numerico: 100
notacion_cientifica: 1e+12
booleano: true
valor_nulo: null
llave con espacios: valor
# Nótese que los strings no deben estar entre comillas, aunqué también es válido.
llave: "Un string, entre comillas."
"Las llaves tambien pueden estar entre comillas.": "valor entre comillas"

# Los strings de líneas múltiples pueden ser escritos
# como un 'bloque literal' (usando pipes |)
# o como un 'bloque doblado' (usando >)

bloque_literal: |
    Este bloque completo de texto será preservado como el valor de la llave
    'bloque_literal', incluyendo los saltos de línea.

    Se continúa guardando la literal hasta que se cese la indentación.
        Cualquier línea que tenga más indentación, mantendrá los espacios dados
        (por ejemplo, estas líneas se guardarán con cuatro espacios)

bloque_doblado: >
    De la misma forma que el valor de 'bloque_literal', todas estas
    líneas se guardarán como una sola literal, pero en esta ocasión todos los
    saltos de línea serán reemplazados por espacio.

    Las líneas en blanco, como la anterior, son convertidas a un salto de línea.
```

```
        Las líneas con mayor indentación guardan sus saltos de línea.
        Esta literal ocuparán dos líneas.

# La indentación se usa para anidar elementos
un_mapa_indentado:
    llave: valor
    otra_llave: otro valor
    otro_mapa_indentado:
        llave_interna: valor_interno

# Las llaves de los mapas no requieren ser strings necesariamente
0.25: una llave numérica

# Las llaves también pueden ser objetos de multiples líneas,
# usando ? para indicar el inicio de una llave
? |
    Esto es una llave
    que tiene múltiples líneas
: y este es su valor

#########################
# TIPOS DE COLECCIONES #
#########################

# Las colecciones en YAML usan la indentación para delimitar el alcance
# y cada elemento de la colección inicia en su propia línea.
# YAML tambien permite colecciones como llaves, pero muchos lenguajes de
```

```yaml
# programación se quejarán.

# Las secuencias (equivalentes a listas o arreglos) se ven así:
- Amarillo
- Verde
- Azul

# Se puede usar una secuencia como valor para una llave.
secuencia:
    - Elemento 1
    - Elemento 2
    - Elemento 3
    - Elemento 4

# Las secuencias pueden contener secuencias como elementos.
- [Uno, Dos, Tres]
- [Domingo, Lunes, Martes]
- [Luna, Marte, Tierra]

# Las secuencias pueden tener distintos tipos en su contenido.
secuencia_combinada:
    - texto
    - 5
    - 0.6
    - llave: valor # se convierte en un json dentro de la secuencia
    -
        - Esta es una secuencia
        - ...dentro de otra secuencia
```

```yaml
# Dado que todo JSON está incluído dentro de YAML, también puedes escribir
# mapas con la sintaxis de JSON y secuencias:
mapa_de_json_1: {"llave": "valor"}
mapa_de_json_2:
  llave: valor

# Las secuencias tambien se pueden escribir como un arreglo al estilo JSON
secuencia_de_json_1: [3, 2, 1, "despegue"]
secuencia_de_json_2:
  - 3
  - 2
  - 1
  - "despegue"

# YAML también soporta conjuntos usando el simbolo ?
# y se ven de la siguiente forma:
set:
    ? item1
    ? item2
    ? item3

# Se puede usar el tag !!set
# Al igual que Python, los conjuntos sólo son mapas con valores nulos.
# El ejemplo de arriba es equivalente a:
set2:
    item1: null
    item2: null
```

```yaml
    item3: null

###################################
# CARACTERÍSTICAS EXTRAS DE YAML #
###################################

# YAML usa tres guiones (---) para diferenciar entre directivas
# y contenido del documento.
# Por otra parte, tres puntos (...) se utilizan para indicar
# el final del documento en casos especiales.

# YAML tiene funciones útiles llamadas 'anchors' (anclas), que te permiten
# duplicar fácilmente contenido a lo largo de tu documento.
# El ampersand indica la declaración del ancla,
declara_ancla: &texto texto de la llave
# el asterisco indica el uso de dicha ancla.
usa_ancla: *texto # tendrá el valor "texto de la llave"

################
# TAGS EN YAML #
################

# En YAML, los nodos que no tienen un tag obtienen su tipo
# según la aplicación que los use, al usar un tag
# se pueden declarar tipos explícitamente.
string_explicito: !!str 0.5 # !!str para declarar un string
integer_explicito: !!int 5 # !!int para declarar un integer
float_explicito: !!float 1.2 # !!float para declarar un float
```

```yaml
conjunto_explicito: !!set # !!set para declarar un conjunto
  ? Uno
  ? Dos
  ? Tres
mapa_ordenado_explicito: !!omap # !!omap para declarar un mapa ordenado
- Primero: 1
- Segundo: 2
- Tercero: 3
- Cuarto: 4

# Tags para los numeros enteros
llave_canonica: 5222
llave_decimal: +5222
llave_octal: 010
llave_hexadecimal: 0xC

#Tags para los numeros flotantes
llave_canonica: 1.215e+3
llave_exponencial: 12.3555e+02
llave_fija: 12.15
llave_negativa_infinita: -.inf
llave_numero_invalido: .NaN

# Tags para las fechas y horas
llave_canonica: 2001-12-15T02:59:43.1Z
llave_iso8601: 2001-12-14t21:59:43.10-05:00
llave_con_espacios: 2001-12-14 21:59:43.10 -5
llave_fecha: 2002-12-14
```

```yaml
# Además existen tags para
null: #valor nulo
booleans: [ true, false ] # Valores booleanos
string: '012345' # Valor en string


# Algunos parseadores implementan tags específicas del lenguaje, como el
# que se muestra a continuación, encargado de manejar números complejos en
# Python:
numero_complejo_python: !!python/complex 1+2j

# El tag !!binary indica que un string es en realidad un blob
# representado en base-64.
archivo_gif: !!binary |
    R0lGODlhDAAMAIQAAP//9/X17unp5WZmZgAAAOfn515eXvPz7Y6OjuDg4J+fn5
    OTk6enp56enmlpaWNjY6Ojo4SEhP/++f/++f/++f/++f/++f/++f/++f/++f/+
    +f/++f/++f/++f/++f/++SH+Dk1hZGUgd2l0aCBHSU1QACwAAAAADAAMAAAFLC
    AgjoEwnuNAFOhpEMTRiggcz4BNJHrv/zCFcLiwMWYNG84BwwEeECcgggoBADs=
```

Recursos adicionales

- Sitio oficial de YAML
- Parser en línea de de YAML
- Validador en línea de YAML

¿Tienes una sugerencia o rectificación? Abre un issue en el repositorio de Github, o haz un pull request tu mismo

Originalmente contribuido por Adam Brenecki, y actualizado por 0 colaborador(es).

Translated by: Daniel Zendejas