Documentation

Article

Mustache Syntax

Overview of Mustache Syntax

Overview

Mustache is a “logic-less” templating engine. The core language has no flow control statements. Instead it has tags that can be replaced with a value, nothing or a series of values. Below we document all the standard tags

Context

Mustache renders a template with a context stack. A context is a list of key/value pairs. These can be represented by either a `Dictionary` or the reflection information from `Mirror`. For example the following two objects will render in the same way

```
let object = ["name": "John Smith", "age": 68]
```

```
struct Person {
    let name: String
    let age: Int
}
let object = Person(name: "John Smith", age: 68)
```

Initially the stack will consist of the root context object you want to render. When we enter a section tag we push the associated value onto the context stack and when we leave the section we pop that value back off the stack.

Tags

All tags are surrounded by a double curly bracket `{{}}`. When a tag has a reference to a key, the key will be searched for from the context at the top of the context stack and the associated value will be output. If the key cannot be found then the next context down will be searched and so on until either a key is found or we have reached the bottom of the stack. If no key is found the output for that value is `nil`.

A tag can be used to reference a child value from the associated value of a key by using dot notation in a similar manner to Swift. eg in `{{main.sub}}` the first context is searched for the `main` key. If a value is found, that value is used as a context and the key `sub` is used to search within that context and so on.

If you want to only search for values in the context at the top of the stack then prefix the variable name with a “.” eg `{{.key}}`

Tag types

- `{{key}}`: Render value associated with `key` as text. By default this is HTML escaped. A `nil` value is rendered as an empty string.
- `{{{name}}}`: Acts the same as `{{name}}` except the resultant text is not HTML escaped. You can also use `{{&name}}` to avoid HTML escaping.
- `{{#section}}`: Section render blocks either render text once or multiple times depending on the value of the key in the current context. A section begins with `{{#section}}` and end with `{{/section}}`. If the key represents a `Bool` value it will only render if it is true. If the key represents an `Optional` it will only render if the object is non-nil. If the key represents an `Array` it will then render the internals of the section multiple times, once for each element of the `Array`. Otherwise it will render with the selected value pushed onto the top of the context stack.
- `{{^section}}`: An inverted section does the opposite of a section. If the key represents a `Bool` value it will render if it is false. If the key represents an `Optional` it will render if it is `nil`. If the key represents a `Array` it will render if the `Array` is empty.
- `{{! comment }}`: This is a comment tag and is ignored.
- `{{>partial}}`: A partial tag renders another mustache file, with the current context stack. In Swift Mustache partial tags only work for templates that are a part of a library and the tag is the name of the referenced file without the “.mustache” extension.

- `{{*>dynamic}}`: Is a partial that can be dynamically loaded.
- `{{<parent}}`: A parent is similar to a partial but allows for the user to override sections of the include file. A parent tag is a section tag so needs to end with a `{{/parent}}` tag.
- `{{$}}`: A block is a section of a parent that can be overriden. If this is found inside a parent section then it is the text that will replace the overriden block.
- `{{=<% %>=}}`: The set delimiter tag allows you to change from using the double curly brackets as tag delimiters. In the example the delimiters have been changed to `<% %>` but you can change them to whatever you like.

You can find out more about the standard Mustache tags in the Mustache Manual.

See Also

Mustache

- Mustache Features

  An overview of the features of swift-mustache.