GO reference build passing

jsonpath

This package extends the json.Decoder to support navigating a stream of JSON tokens. You should be able to use this extended Decoder places where a json.Decoder would have been used.

This Decoder has the following enhancements...

- The Scan method supports scanning a JSON stream while extracting particular values along the way using PathActions.
- The SeekTo method supports seeking forward in a JSON token stream to a particular path.
- The Path method returns the path of the most recently parsed token.
- The Token method has been modified to distinguish between strings that are object keys and strings that are values. Object key strings are returned as the KeyString type rather than a native string.

Installation

```
go get -u github.com/exponent-io/jsonpath
```

Example Usage

SeekTo

```
import "github.com/exponent-io/jsonpath"

var j = []byte(`[
  {"Space": "YCbCr", "Point": {"Y": 255, "Cb": 0, "Cr": -10}},
  {"Space": "RGB",   "Point": {"R": 98, "G": 218, "B": 255}}
]`)

w := json.NewDecoder(bytes.NewReader(j))
var v interface{}

w.SeekTo(1, "Point", "G")
w.Decode(&v) // v is 218
```

Scan with PathActions

```
var j = []byte(`{"colors":[
  {"Space": "YCbCr", "Point": {"Y": 255, "Cb": 0, "Cr": -10, "A": 58}},
  {"Space": "RGB",   "Point": {"R": 98, "G": 218, "B": 255, "A": 231}}
]}`)

var actions PathActions

// Extract the value at Point.A
```

```go
actions.Add(func(d *Decoder) error {
  var alpha int
  err := d.Decode(&alpha)
  fmt.Printf("Alpha: %v\n", alpha)
  return err
}, "Point", "A")

w := NewDecoder(bytes.NewReader(j))
w.SeekTo("colors", 0)

var ok = true
var err error
for ok {
  ok, err = w.Scan(&actions)
  if err != nil && err != io.EOF {
    panic(err)
  }
}
```