Fast campus

JSON

(Javascript Object Notation)

표기법

Giftbot

JSON

- Javascript 언어로부터 파생 (JavaScript Object Notation)

- 프로그래밍 언어와 플랫폼 간 독립적이고 가벼워서 XML 방식을 대체하여
  현재 거의 표준으로 사용되고 있는 데이터 교환 형식

- 두 개의 구조를 기본으로 가짐
  - 'Name : Value' 형태의 쌍을 이루는 콜렉션 타입. 각 언어에서 Hash table, Dictionary 등으로 구현
  - 값들의 순서화된 리스트. 대부분의 언어들에서 Array, Vector, List 또는 Sequence 로 구현

- XML 에 비해 기능이 적고 구조가 단순하여 파싱이 쉽고 빠르며 적은 용량으로 저장 가능
  따라서 사람이 읽고 쓰는 것뿐 아니라 기계가 분석하고 생성하는 것에도 (상대적으로) 더 용이

- contents type 은 application/json 이며, 파일 확장자는 .json, 기본 인코딩은 UTF-8 을 사용

JSON Format Example


```
{
    "message": "success",
    "number": 3,
    "people": [
        {
            "craft": "ISS",
            "name": "Anton Shkaplerov"
        },
        {
            "craft": "ISS",
            "name": "Scott Tingle"
        },
        {
            "craft": "ISS",
            "name": "Norishige Kanai"
        }
    ]
}
```


딕셔너리는 []
JSON은 {}
JSON 안에 JSON 포함

JSON vs XML

```
{"widget": {
    "debug": "on",
    "window": {
        "title": "Sample Konfabulator Widget",
        "name": "main_window",
        "width": 500,
        "height": 500
    },
    "text": {
        "data": "Click Here",
        "size": 36,
        "style": "bold",
        "name": "text1",
        "hOffset": 250,
        "vOffset": 100,
        "alignment": "center",
        "onMouseUp": "sun1.opacity = (sun1.opacity / 100) * 90;"
    }
}}
```

JSON vs XML

```
<widget>
    <debug>on</debug>
    <window title="Sample Konfabulator Widget">
        <name>main_window</name>
        <width>500</width>
        <height>500</height>
    </window>
    <text data="Click Here" size="36" style="bold">
        <name>text1</name>
        <hOffset>250</hOffset>
        <vOffset>100</vOffset>
        <alignment>center</alignment>
        <onMouseUp>
            sun1.opacity = (sun1.opacity / 100) * 90;
        </onMouseUp>
    </text>
</widget>
```

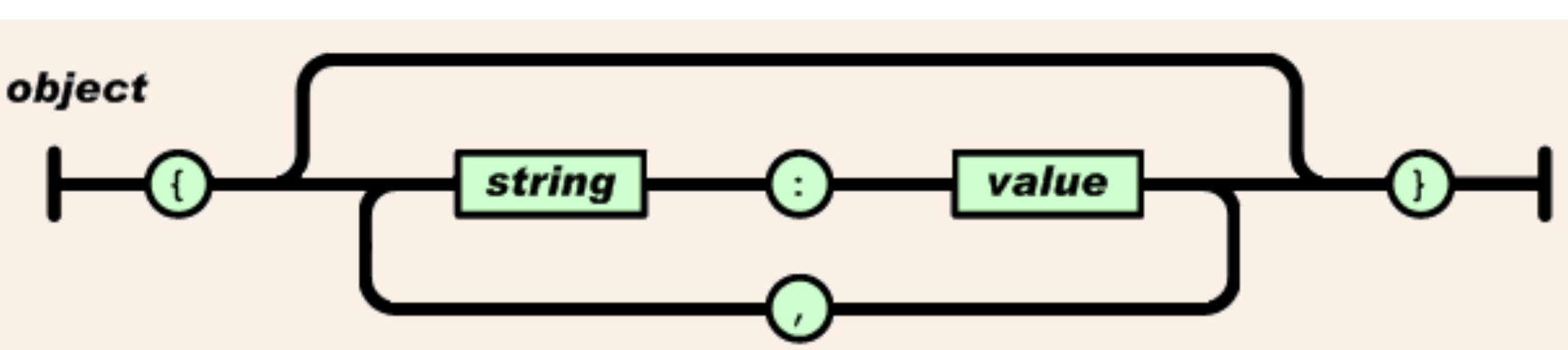
Object

딕셔너리

object 는 name/value 쌍들의 비순서화된 SET|

좌측 중괄호({) 로 시작하고 우측 중괄호(}) 로 끝내어 표현

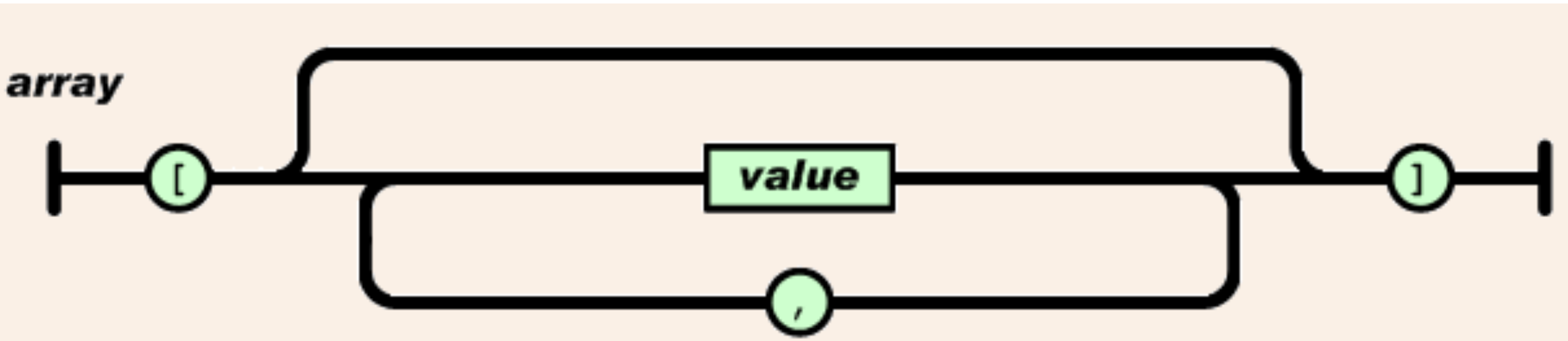
name 뒤 콜론(:) 을 붙이고 콤마(,)로 name/value 쌍들을 구분



Array

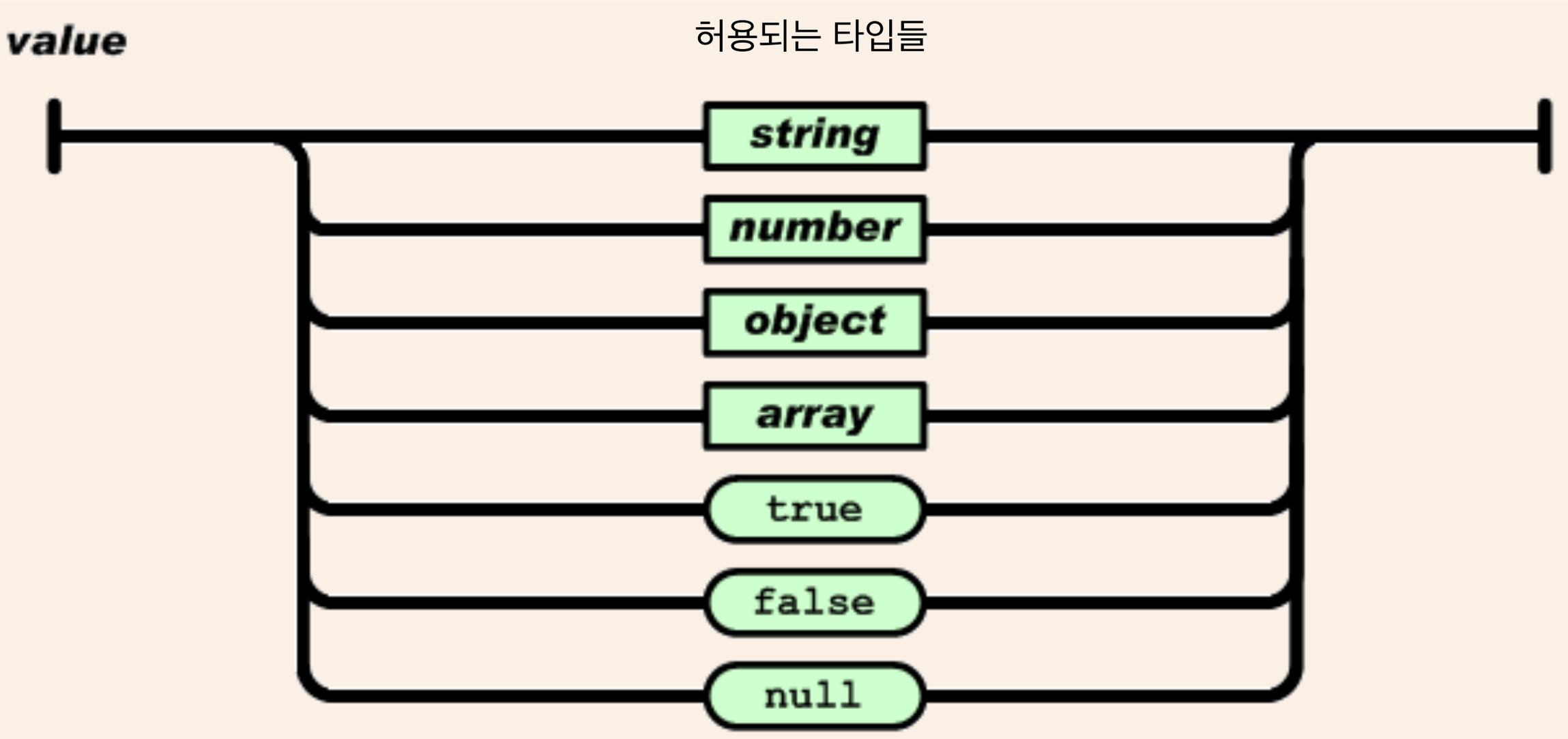
값들의 순서화된 Collection

좌측 대괄호([ )로 시작해서 우측 대괄호 ( ]) 로 끝내어 표현하고 콤마(,)로 각 값들을 구분



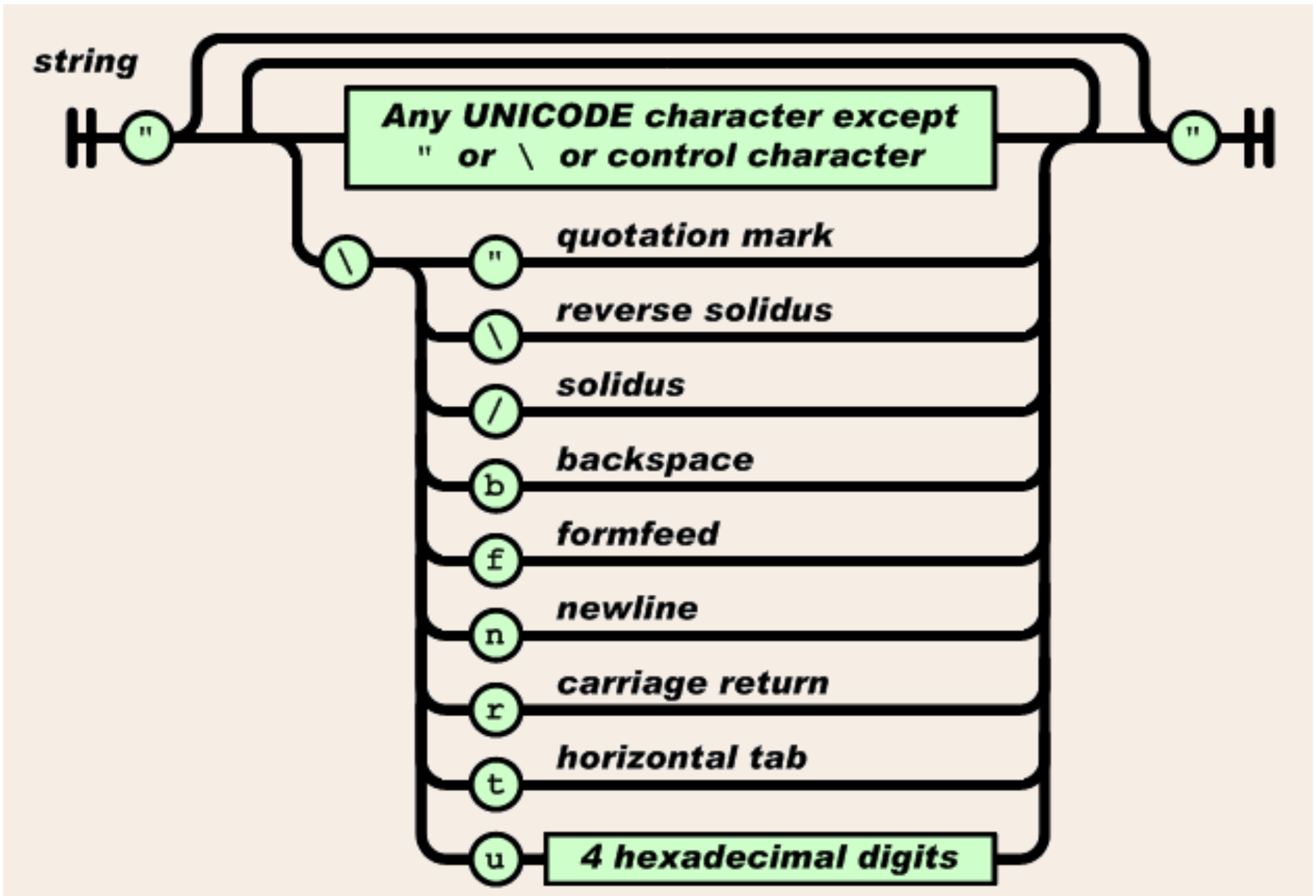
Value

큰 따옴표안에 string, number, true, false, null, object, array 등의 구조가 포함



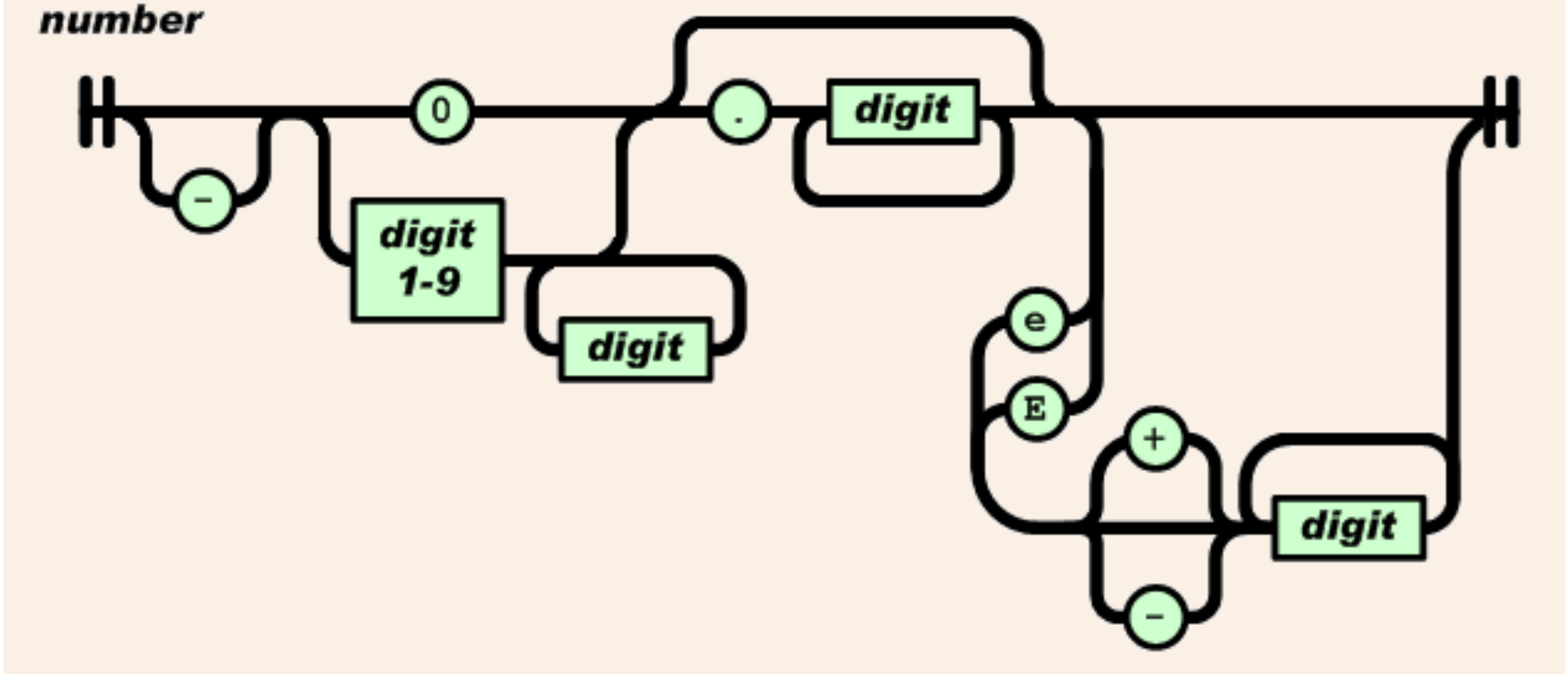
String

큰 따옴표 (“ ”) 안에 둘러 싸인 0 이상의 Unicode 문자들의 조합이며 backslash escape 가 적용



Number

8진수와 16진후 형식을 사용하지 않는 것을 제외하면 C, JAVA 등의 number 와 유사



JSON in Swift

Fast campus

Swift4 - 멀티라인 스트링

```
let jsonString = """
{
  "someNumber" : 1,
  "someString" : "Hello",
  "someArray"  : [1, 2, 3, 4],
  "someDict"   : {
    "someBool" : true
  }
}
"""
let jsonData = jsonString.data(using: .utf8)!
let jsonObject = try! JSONSerialization.jsonObject(with: jsonData)
print(jsonObject)
```

JSONSerialization

- **JSON 과 이에 상응하는 Foundation 객체 간 변환하는 객체이며 iOS 7 이후로 Thread Safety**

- **Data 는 다음의 5가지 인코딩 지원 형식 중 하나여야 하며, 이 중 UTF-8 이 기본값으로 쓰이고 가장 효율적**

  (UTF-8, UTF-16LE, UTF-16BE, UTF-32LE, UTF-32BE)

  LE - Little Endian
  BE - Big Endian
  데이터를 전송하고 받을때 형식

- **JSON 으로 변환되기 위한 Foundation 객체는 다음 속성을 따라야 함**
  - **Top Level Object : NSArray, NSDictionary**
  - **모든 객체는 NSString, NSNumber, NSArray, NSDictionary, NSNull 의 인스턴스**
  - **모든 Dictionary 의 Key 는 NSString 인스턴스**
  - **숫자는 NaN 이나 무한대 값이 아니어야 함**

- **JSON data 로 변환 가능 여부는 isValidJSONObject(_:) 메서드를 통해 확인 가능**

Creating a JSON Object

Fast campus

```
class func jsonObject(with: Data, options: JSONSerialization.ReadingOptions = [])
```

Returns a Foundation object from given JSON data.

```
class func jsonObject(with: InputStream, options: JSONSerialization.ReadingOptions =
[])
```

Returns a Foundation object from JSON data in a given stream.

Creating JSON Data

```
class func data(withJSONObject: Any, options: JSONSerialization.WritingOptions = [])
```

Returns JSON data from a Foundation object.

```
class func writeJSONObject(Any, to: OutputStream, options: JSONSerialization.Writing
Options = [], error: NSErrorPointer)
```

Writes a given JSON object to a stream.

```
class func isValidJSONObject(Any)
```

Returns a Boolean value that indicates whether a given object can be converted to JSON data.

JSONSerialization.ReadingOptions

2 개의 mutable 옵션은 Swift 에서 var 타입 변수 생성으로 대체

.allowFragments - 파싱하려는 JSON Data 가 Array/Dictionary 가 아닌 경우에도 허용

```
@available(iOS 5.0, *)
struct ReadingOptions : OptionSet {
  init(rawValue: UInt)
  static var mutableContainers: JSONSerialization.ReadingOptions
  static var mutableLeaves: JSONSerialization.ReadingOptions
  static var allowFragments: JSONSerialization.ReadingOptions
}
```

JSONSerialization.WritingOptions

.prettyPrinted - 사람이 읽기 편하도록 json 문자열 중간에 whitespace 를 삽입.

이 옵션이 없을 경우 가장 압축된 형태의 JSON 생성

.sortedKeys - Dictionary 키를 systemLocale에 맞춰 정렬 후 출력. 비교는 NSNumericSearch 기준

```
@available(iOS 5.0, *)
struct WritingOptions : OptionSet {
  init(rawValue: UInt)
  static var prettyPrinted: JSONSerialization.WritingOptions

  @available(iOS 11.0, *)
  public static var sortedKeys: JSONSerialization.WritingOptions
}
```

Example

```
private func jsonObjectWithData() {
  let jsonString = """
     { "hello": "world", "foo": "bar", "iOS": "Swift" }
  """
  let jsonData = jsonString.data(using: .utf8)!

  do {
    let jsonObject = try JSONSerialization.jsonObject(with: jsonData)
    if let jsonDict = jsonObject as? [String: Any] {
      print(jsonDict)
    }
  } catch {
    print(error.localizedDescription)
  }
}
```