JSON

Data Interchange

- The key idea in Ajax.
- An alternative to page replacement.
- Applications delivered as pages.
- How should the data be delivered?

History of Data Formats

- Ad Hoc
- Database Model
- Document Model
- Programming Language Model

JSON

- JavaScript Object Notation
- Minimal
- Textual
- Subset of JavaScript

JSON

- A Subset of ECMA-262 Third Edition.
- Language Independent.
- Text-based.
- Light-weight.
- Easy to parse.

JSON Is Not...

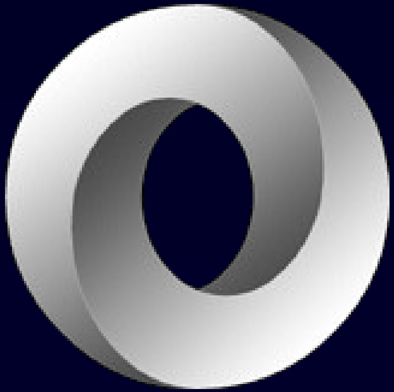
- JSON is not a document format.
- JSON is not a markup language.
- JSON is not a general serialization format.
    - No cyclical/recurring structures.
    - No invisible structures.
    - No functions.

History

- 1999 ECMAScript Third Edition
- 2001 State Software, Inc.
- 2002 JSON.org
- 2005 Ajax
- 2006 **RFC** 4627

Languages

- Chinese
- English
- French
- German
- Italian
- Japanese
- Korean



Languages

- ActionScript
- C / C++
- C#
- Cold Fusion
- Delphi
- E
- Erlang
- Java
- Lisp
- Perl
- Objective-C
- Objective CAML
- PHP
- Python
- Rebol
- Ruby
- Scheme
- Squeak

Object Quasi-Literals

- JavaScript
- Python
- NewtonScript

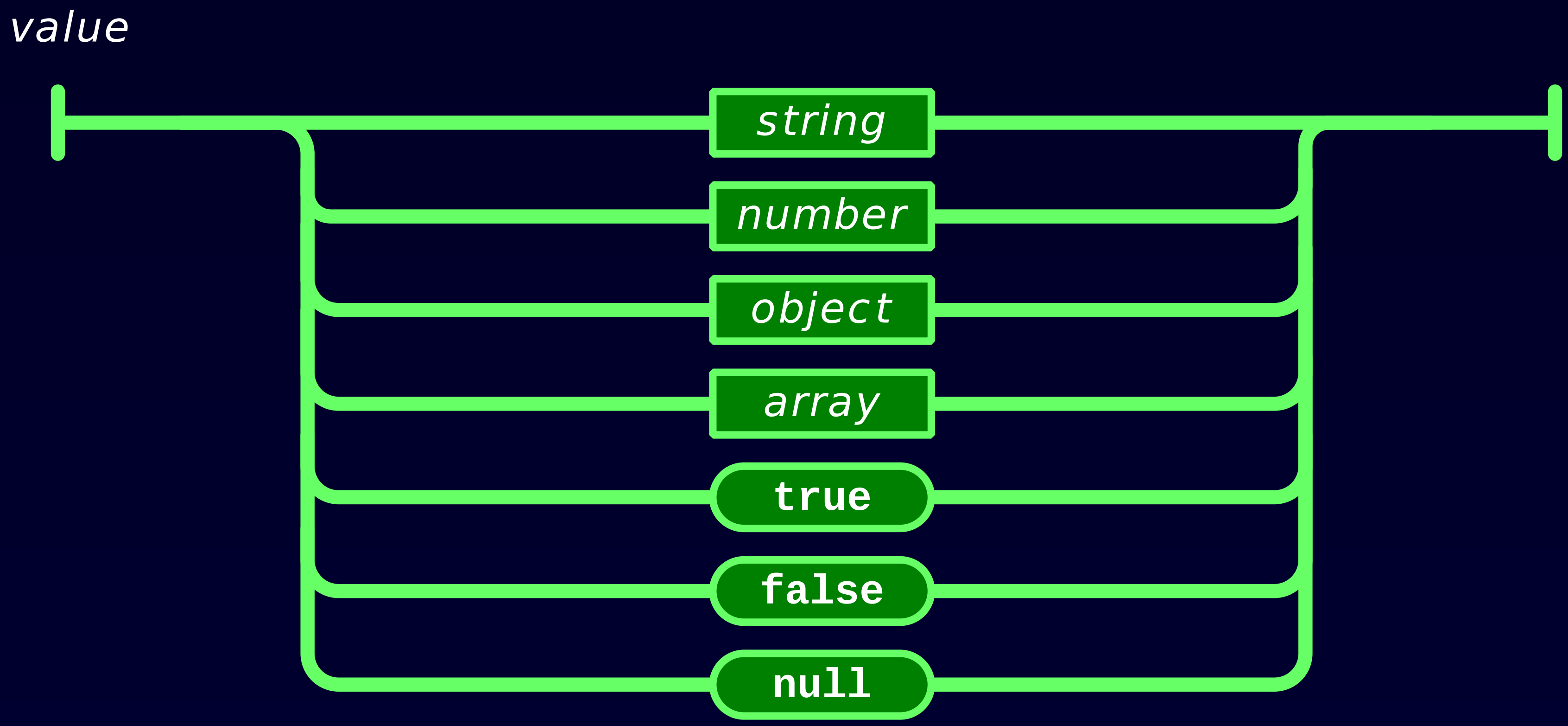
Values

- Strings
- Numbers
- Booleans

- Objects
- Arrays

- **`null`**

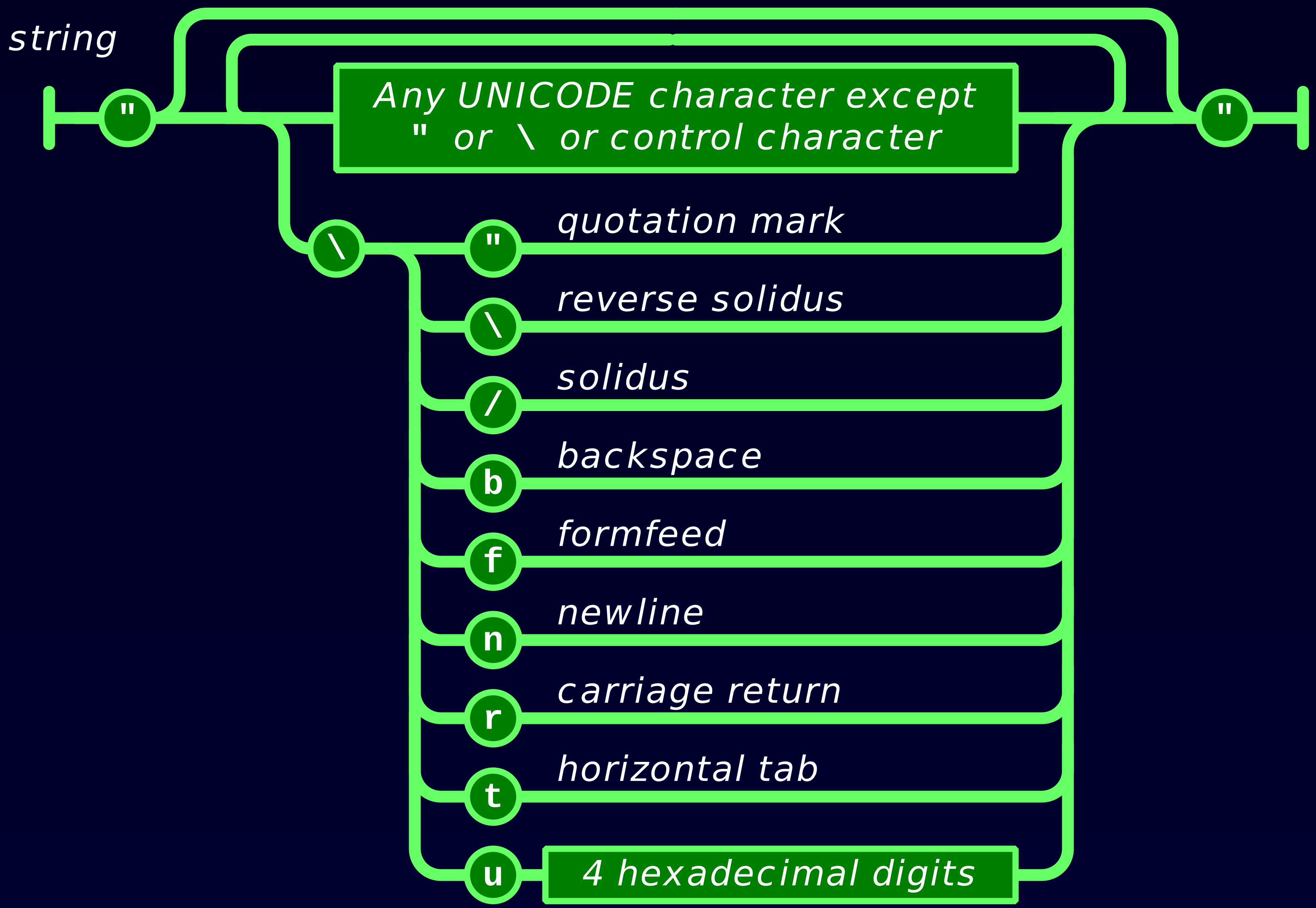
Value



Strings

- Sequence of 0 or more Unicode characters
- No separate character type
  - A character is represented as a string with a length of 1
- Wrapped in "double quotes"
- Backslash escapement

String

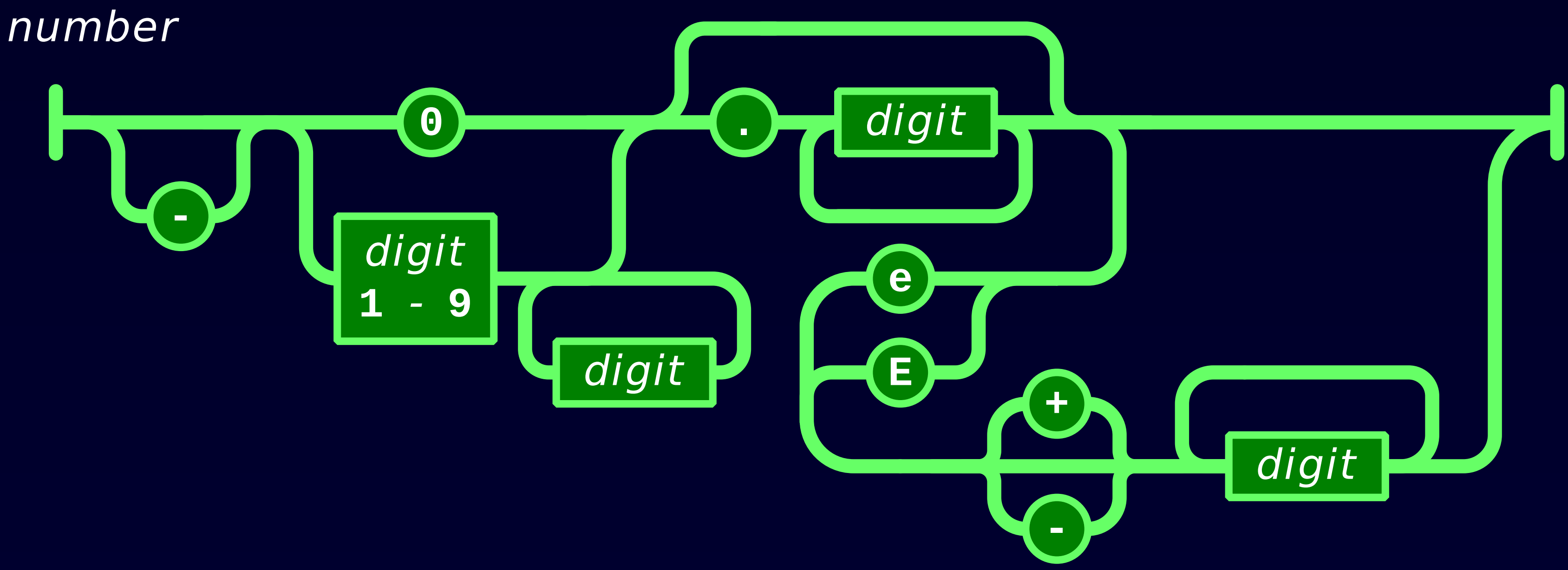

string
"
Any UNICODE character except " or \ or control character
\
"
quotation mark
\
reverse solidus
/
solidus
b
backspace
f
formfeed
n
newline
r
carriage return
t
horizontal tab
u
4 hexadecimal digits
"

Numbers

- Integer
- Real
- Scientific

- No octal or hex
- No **`NaN`** or **`Infinity`**
  - Use **`null`** instead

Number



Booleans

- `true`
- `false`

`null`

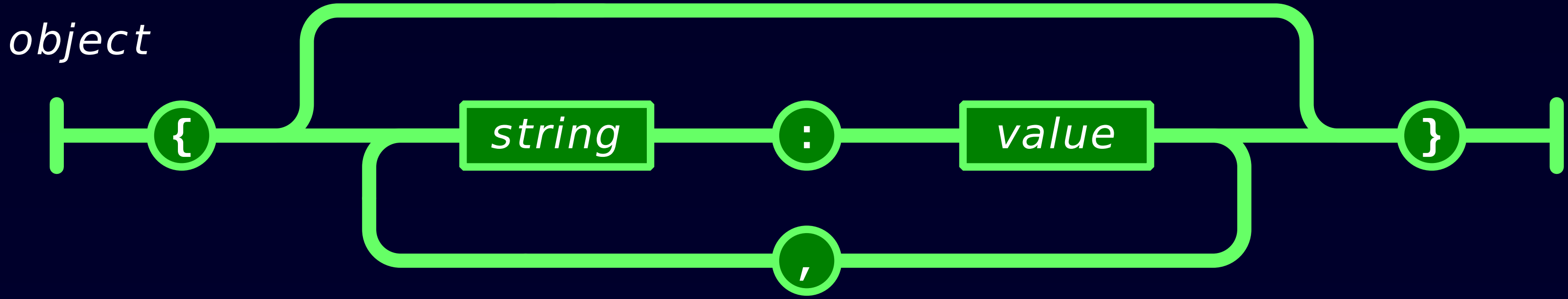
- A value that isn't anything

Object

- Objects are unordered containers of key/value pairs
- Objects are wrapped in `{ }`
- `,` separates key/value pairs
- `:` separates keys and values
- Keys are strings
- Values are JSON values

struct, record, hashtable, object

Object



Object

```
{"name":"Jack B. Nimble","at large":
true,"grade":"A","level":3, "format":
{"type":"rect","width":1920,
"height":1080,"interlace":false,
"framerate":24}}
```

Object

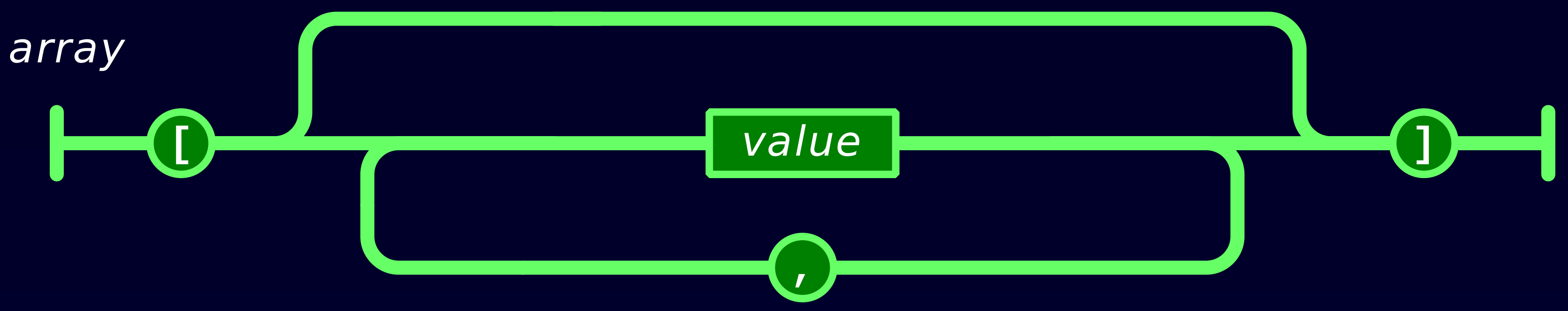
```
{
    "name":      "Jack B. Nimble",
    "at large": true,
    "grade":     "A",
    "format": {
        "type":       "rect",
        "width":      1920,
        "height":     1080,
        "interlace": false,
        "framerate": 24
    }
}
```

Array

- Arrays are ordered sequences of values
- Arrays are wrapped in `[]`
- `,` separates values
- JSON does not talk about indexing.

    An implementation can start array indexing at 0 or 1.

Array



Array

```
["Sunday", "Monday", "Tuesday",
 "Wednesday", "Thursday",
 "Friday", "Saturday"]


[
    [0, -1, 0],
    [1, 0, 0],
    [0, 0, 1]
]
```

Arrays vs Objects

- Use objects when the key names are arbitrary strings.
- Use arrays when the key names are sequential integers.
- Don't get confused by the term Associative Array.

MIME Media Type

`application/json`

Character Encoding

- Strictly UNICODE.
- Default: UTF-8.
- UTF-16 and UTF-32 are allowed.

Versionless

- JSON has no version number.
- No revisions to the JSON grammar are anticipated.
- JSON is very stable.

Rules

- A JSON decoder must accept all well-formed JSON text.
- A JSON decoder may also accept non-JSON text.
- A JSON encoder must only produce well-formed JSON text.
- *Be conservative in what you do, be liberal in what you accept from others.*

Supersets

- YAML is a superset of JSON.
  - A YAML decoder is a JSON decoder.
- JavaScript is a superset of JSON.
  - A JavaScript compiler is a JSON decoder.
- New programming languages based on JSON.

JSON Looks Like Data

- JSON's simple values are the same as used in programming languages.
- No restructuring is required: JSON's structures look like conventional programming language structures.
- JSON's object is record, struct, object, dictionary, hash, associate array...
- JSON's array is array, vector, sequence, list...

Arguments against JSON

- JSON Doesn't Have Namespaces.
- JSON Has No Validator.
- JSON Is Not Extensible.
- JSON Is Not XML.

JSON Doesn't Have Namespaces

- Every object is a namespace. Its set of keys is independent of all other objects, even exclusive of nesting.

- JSON uses context to avoid ambiguity, just as programming languages do.

Namespace

- **http://www.w3c.org/TR/REC-xml-names/**

- In this example, there are three occurrences of the name title within the markup, and the name alone clearly provides insufficient information to allow correct processing by a software module.

```
<section>
    <title>Book-Signing Event</title>
    <signing>
        <author title="Mr" name="Vikram Seth" />
        <book title="A Suitable Boy" price="$22.95" />
    </signing>
    <signing>
        <author title="Dr" name="Oliver Sacks" />
        <book title="The Island of the Color-Blind"
              price="$12.95" />
    </signing>
</section>
```

Namespace

```
{"section":
    "title": "Book-Signing Event",
    "signing": [
        {
            "author": { "title": "Mr", "name": "Vikram Seth" },
            "book": { "title": "A Suitable Boy",
                      "price": "$22.95" }
        }, {
            "author": { "title": "Dr", "name": "Oliver Sacks" },
            "book": { "title": "The Island of the Color-Blind",
                      "price": "$12.95" }
        }
    ]
}}
```

- `section.title`
- `section.signing[0].author.title`
- `section.signing[1].book.title`

JSON Has No Validator

- Being well-formed and valid is not the same as being correct and relevant.

- Ultimately, every application is responsible for validating its inputs. This cannot be delegated.

- A YAML validator can be used.

JSON is Not Extensible

- It does not need to be.
- It can represent any non-recurrent data structure as is.
- JSON is flexible. New fields can be added to existing structures without obsoleting existing programs.

JSON Is Not XML

- objects
- arrays
- strings
- numbers
- booleans
- **`null`**

- element
- attribute
- attribute string
- content
- **`<![CDATA[ ]]>`**
- entities
- declarations
- schema
- stylesheets
- comments
- version
- namespace

Data Interchange

- JSON is a simple, common representation of data.
- Communication between servers and browser clients.
- Communication between peers.
- Language independent data interchange.

Why the Name?

- XML is not a good data interchange format, but it is a document standard.
- Having a standard to refer to eliminates a lot of squabbling.

www.JSON.org