UNB

JSON & GeoJSON

Emmanuel Stefanakis

estef@unb.ca

JSON

- JSON…
  - **J**ava**S**cript **O**bject **N**otation
- What is it ?
  - An **alternative** language to **XML**
- Where is it used ?
  - for **serializing** and **transmitting** structured data over a network connection

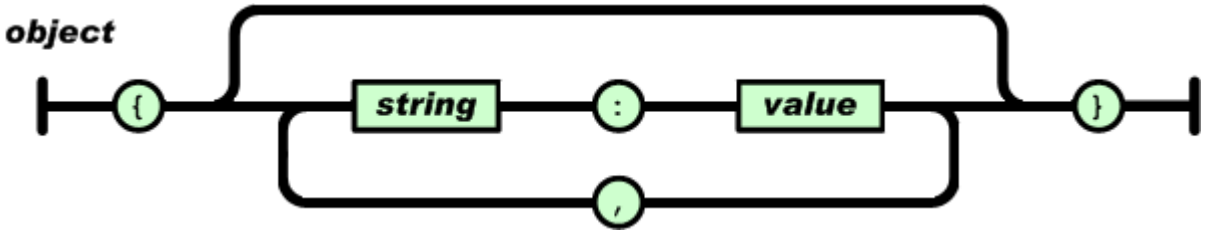
JSON

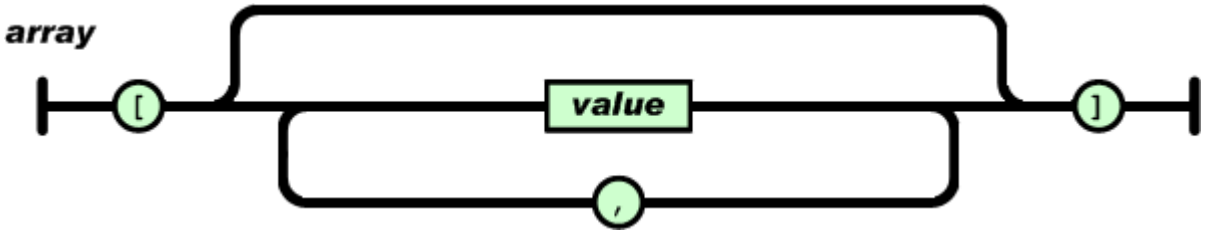
- Properties…
  - a lightweight text-based **open standard** **(since 2002)**
  - **human-readable** format
  - derived from the **JavaScript** scripting language
  - represents **objects**
    - simple data structures and associative arrays
  - **language-independent** with many parsers
    - despite its relationship to JavaScript
  - filename extension is: **.json**

JSON Structure

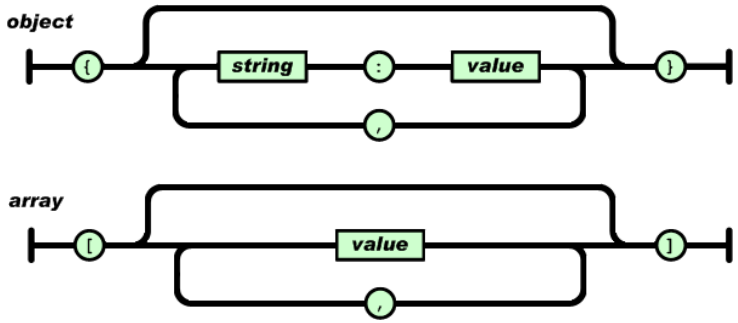
- Two structures…
  - a collection of name/value pairs [**object**]
    - e.g., object, record, struct, dictionary, hash table, keyed list, or associative array.
  - an ordered list of values [**array**]
    - e.g., an array, vector, list, or sequence.

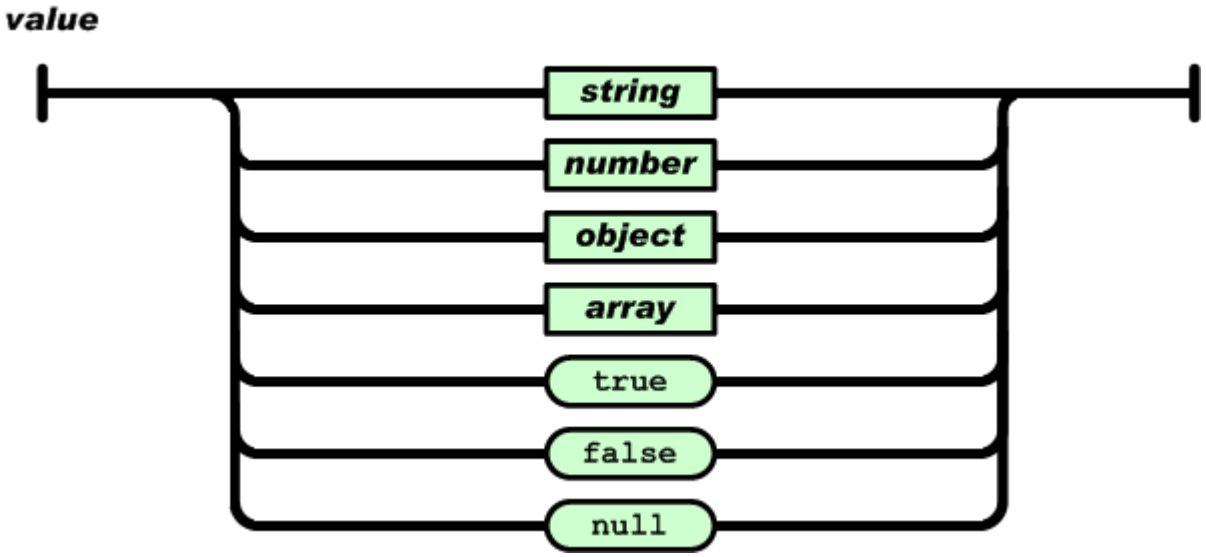
JSON Structure (syntax)



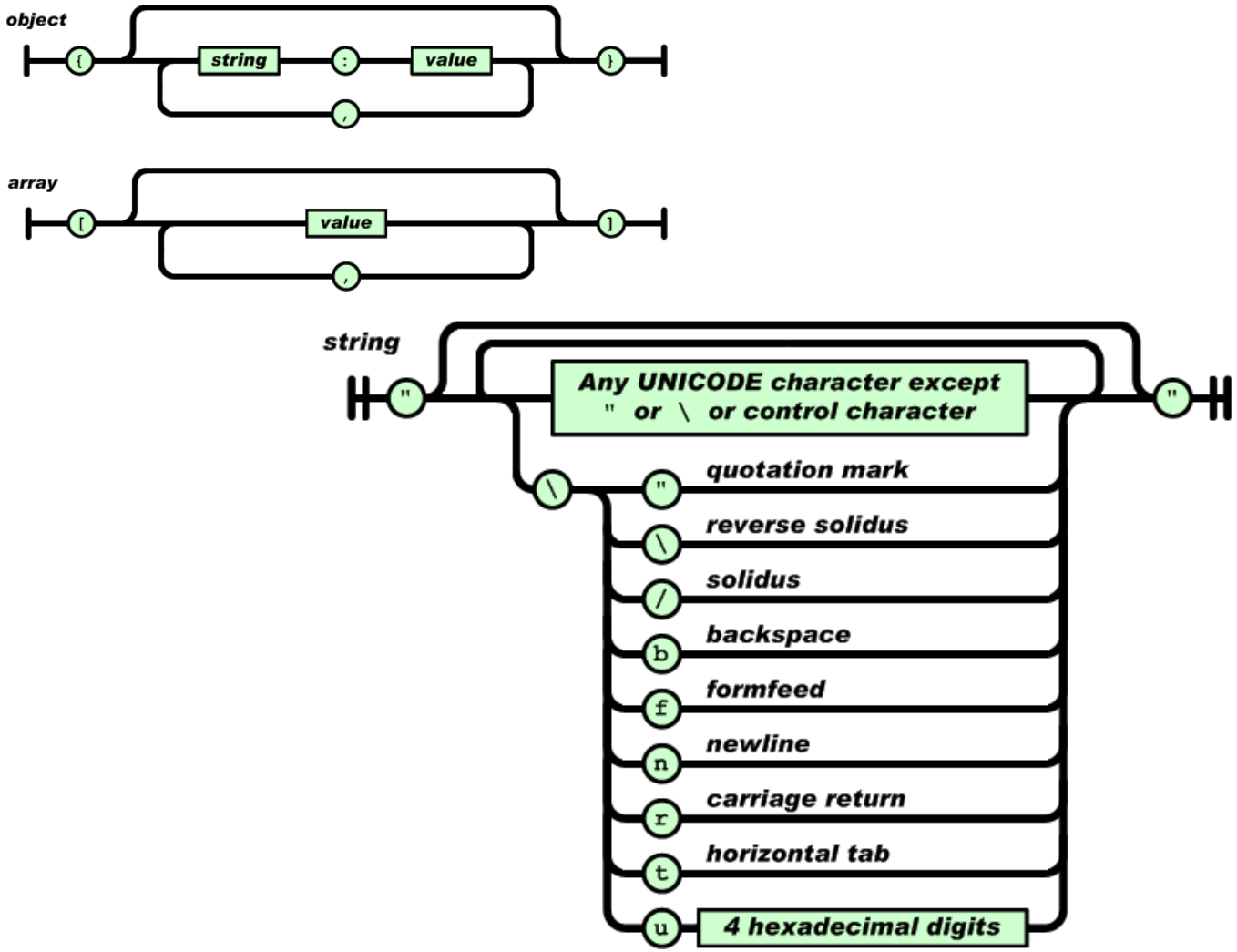


JSON Structure (syntax)

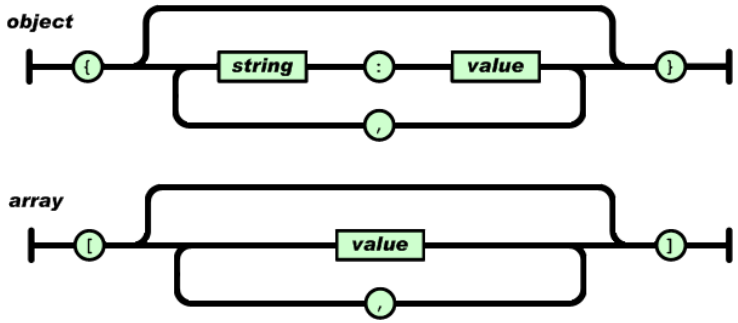


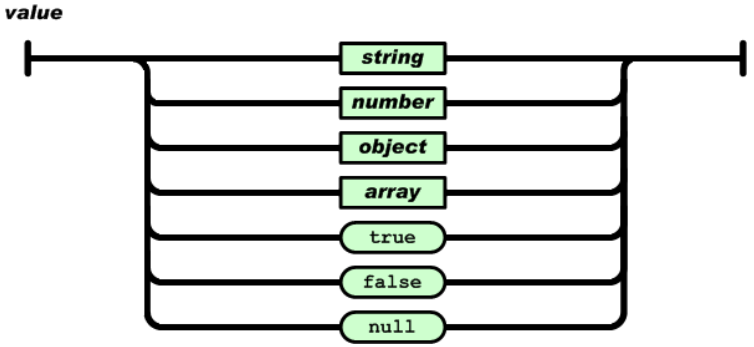


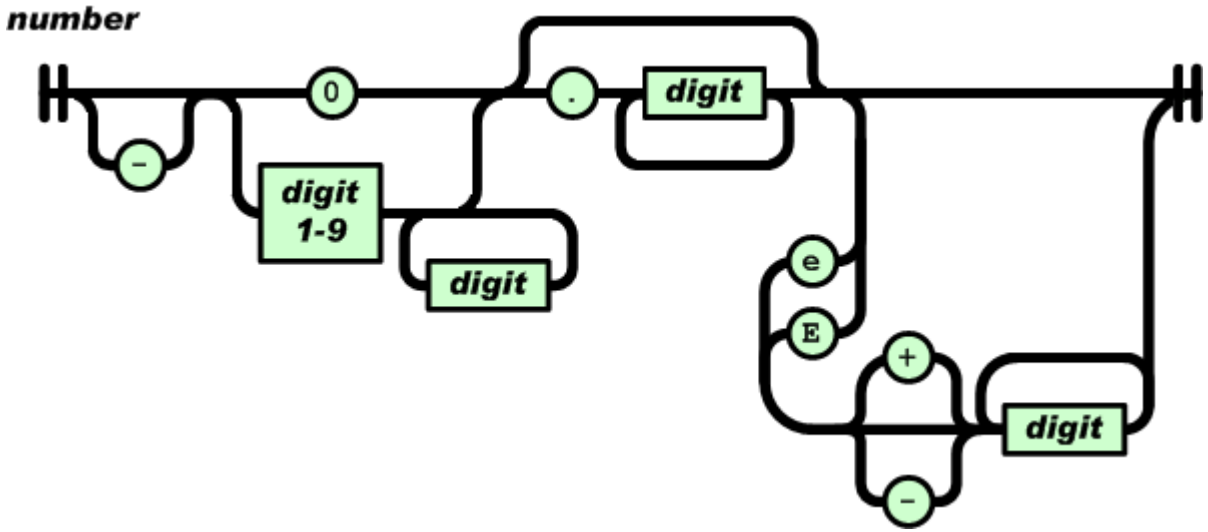
JSON Structure (syntax)



JSON Structure (syntax)







JSON Structure (syntax)

object
- {}
- { *members* }

members
- *pair*
- *pair* , *members*

pair
- *string* : *value*

array
- []
- [*elements*]

elements
- *value*
- *value* , *elements*

value
- *string*
- *number*
- *object*
- *array*
- **true**
- **false**
- **null**

string
- ""
- " *chars* "

chars
- *char*
- *char chars*

char
- *any-Unicode-character-except-"-or-\-or-control-character*
- \"
- \\
- \/
- \b
- \f
- \n
- \r
- \t
- \u *four-hex-digits*

number
- *int*
- *int frac*
- *int exp*
- *int frac exp*

int
- *digit*
- *digit1-9 digits*
- *- digit*
- *- digit1-9 digits*

frac
- *. digits*

exp
- *e digits*

digits
- *digit*
- *digit digits*

e
- e
- e+
- e-
- E
- E+
- E-

JSON Structure (syntax)

Object: person

```
{
    "firstName": "John",
    "lastName" : "Smith",
    "age"      : 25,
    "address"  :
    {
        "streetAddress": "21 2nd Street",
        "city"         : "New York",
        "state"        : "NY",
        "postalCode"   : "10021"
    },
    "phoneNumber":
    [
        {
          "type"  : "home",
          "number": "212 555-1234"
        },
        {
          "type"  : "fax",
          "number": "646 555-4567"
        }
    ]
}
```

fields for first name and last name, a number field for age

nested object representing the person's address

a list (an array) of phone number objects

JSON Structure (syntax)

```
{
        "id": 1,
        "name": "Foo",
        "price": 123,
        "tags": ["Bar","Eek"],
        "stock": { "warehouse":300, "retail":20 }
}
```

How can the validity of a JSON document be tested ?

JSON Schemas

```
{
        "id": 1,
        "name": "Foo",
        "price": 123,
        "tags": ["Bar","Eek"],
        "stock": { "warehouse":300, "retail":20 }
}
```

```
{
        "name":"Product",
        "properties":
        {
                "id":
                {
                        "type":"number",
                        "description":"Product identifier",
                        "required":true
                },
                "name":
                {
                        "type":"string",
                        "description":"Name of the product",
                        "required":true
                },
                "price":
                {
                        "type":"number",
                        "minimum":0,
                        "required":true
                },
```

JSON Schemas

```
{
        "id": 1,
        "name": "Foo",
        "price": 123,
        "tags": ["Bar","Eek"],
        "stock": { "warehouse":300, "retail":20 }
}
```

```
                "tags":
                {
                        "type":"array",
                        "items":
                        {
                                "type":"string"
                        }
                },
                "stock":
                {
                        "type":"object",
                        "properties":
                        {
                                "warehouse":
                                {
                                        "type":"number"
                                },
                                "retail":
                                {
                                        "type":"number"
                                }
                        }
                }
        }
}
```

JSV: JSON Schema Validators

GeoJSON

- What is it?
  - a format for **encoding geographic data** structures
  - a **JSON object**
    - JSON tools can be used for processing GeoJSON data
  - generally **more compact than XML**

GeoJSON

- A GeoJSON object
  - may represent a **geometry**, a **feature**, or a **collection of features**
    - features contain a geometry object and additional properties; a feature collection is a list of features

- GeoJSON supports the **geometry types**:
  - Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, and GeometryCollection.

GeoJSON

```
{ "type": "FeatureCollection",
  "features": [
    { "type": "Feature",
      "geometry": {"type": "Point", "coordinates": [102.0, 0.5]},
      "properties": {"prop0": "value0"}
      },
    { "type": "Feature",
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [102.0, 0.0], [103.0, 1.0], [104.0, 0.0], [105.0, 1.0]
          ]
        },
      "properties": {
        "prop0": "value0",
        "prop1": 0.0
        }
      },
    { "type": "Feature",
       "geometry": {
         "type": "Polygon",
         "coordinates": [
           [ [100.0, 0.0], [101.0, 0.0], [101.0, 1.0],
             [100.0, 1.0], [100.0, 0.0] ]
           ]
       },
       "properties": {
         "prop0": "value0",
         "prop1": {"this": "that"}
         }
       }
     ]
   }
```

a GeoJSON feature collection

GeoJSON

- Geometry examples…

```
{ "type": "Point", "coordinates": [100.0, 0.0] }
```

```
{ "type": "LineString",
  "coordinates": [ [100.0, 0.0], [101.0, 1.0] ]
  }
```

```
{ "type": "Polygon",
  "coordinates": [
    [ [100.0, 0.0], [101.0, 0.0], [101.0, 1.0], [100.0, 1.0], [100.0, 0.0] ]
    ]
 }
```

…with holes

```
{ "type": "Polygon",
  "coordinates": [
    [ [100.0, 0.0], [101.0, 0.0], [101.0, 1.0], [100.0, 1.0], [100.0, 0.0] ],
    [ [100.2, 0.2], [100.8, 0.2], [100.8, 0.8], [100.2, 0.8], [100.2, 0.2] ]
    ]
 }
```

GeoJSON

```
{ "type": "MultiPoint",
  "coordinates": [ [100.0, 0.0], [101.0, 1.0] ]
  }
```

```
{ "type": "MultiLineString",
  "coordinates": [
      [ [100.0, 0.0], [101.0, 1.0] ],
      [ [102.0, 2.0], [103.0, 3.0] ]
    ]
  }
```

```
{ "type": "MultiPolygon",
  "coordinates": [
    [[[102.0, 2.0], [103.0, 2.0], [103.0, 3.0], [102.0, 3.0], [102.0, 2.0]]],
    [[[100.0, 0.0], [101.0, 0.0], [101.0, 1.0], [100.0, 1.0], [100.0, 0.0]],
     [[100.2, 0.2], [100.8, 0.2], [100.8, 0.8], [100.2, 0.8], [100.2, 0.2]]]
    ]
  }
```

```
{ "type": "GeometryCollection",
  "geometries": [
    { "type": "Point",
      "coordinates": [100.0, 0.0]
      },
    { "type": "LineString",
      "coordinates": [ [101.0, 0.0], [102.0, 1.0] ]
      }
  ]
}
```

GeoJSON

- Coordinate Reference System definition…

```
"crs": {
  "type": "name",
  "properties": {
    "name": "urn:ogc:def:crs:OGC:1.3:CRS84"
    }
  }
```

```
"crs": {
  "type": "link",
  "properties": {
    "href": "http://example.com/crs/42",
    "type": "proj4"
    }
  }
```

```
"crs": {
  "type": "link",
  "properties": {
    "href": "data.crs",
    "type": "ogcwkt"
    }
  }
```

OGC crs property (preferred) or with an EPSG code.

If a crs is not defined, GeoJSON will use the WGS84 geoid by default.

```
{
    "type":"Feature",
    "id":"OpenLayers.Feature.Vector_314",
    "properties":{},
    "geometry":{
        "type":"Point",
        "coordinates":[97.03125, 39.7265625]
    },
    "crs":{
        "type":"OGC",
        "properties":{
            "urn":"urn:ogc:def:crs:OGC:1.3:CRS84"
        }
    }
}
```

GeoJSON

- Bounding box definition…
  - over a feature…

```
{ "type": "Feature",
  "bbox": [-180.0, -90.0, 180.0, 90.0],
  "geometry": {
    "type": "Polygon",
    "coordinates": [[
      [-180.0, 10.0], [20.0, 90.0], [180.0, -5.0], [-30.0, -90.0]
      ]]
    }
  ...
  }
```

  - over a feature collection…

```
{ "type": "FeatureCollection",
  "bbox": [100.0, 0.0, 105.0, 1.0],
  "features": [
    ...
    ]
  }
```

References

- Wikipedia
  - http://en.wikipedia.org/wiki/Json
- JSON Specification
  - http://json.org/
- GeoJSON Specification
  - http://geojson.org/

UNB

JSON & GeoJSON

Emmanuel Stefanakis

estef@unb.ca