JavaScript Object Notation (JSON)

JSON is a very popular format for storing data, particularly data that is served or stored on a web server. JavaScript is the de facto language of the internet, so it is not surprising we use a data format devised for this language online.

The thing that makes JSON data tricky for data science is that it does not need to be (and rarely is) tabular. It often has a nested structure that needs to be manually and carefully turned into a rectangular data format.

JSON Elements

JSON has six data types:

- **Numbers**: just what you think; all numbers are doubles (no seperate integers)
- **Strings:** a string (UTF-8 encoding)
- **Boolean:** either true or false; what R classes lgl/logical
- **Array:** similar to an unnamed R list; can contain any other object type
- **Object:** similar to a named R list with unique names; can contain any other object
- **null:** a special value used to representing missing data

The abilty for Arrays and Objects to contain other arrays and objects is what allows JSON data to be incredibly flexible, useful, and (for us) difficult.

JSON Example

Here is an example of a small data set shown in JSON format (taken from the Wikipedia page).

Note that names in Objects are quoted and followed by a colon.

Newlines and spaces are ignored when parsing and added here for readability.

```
{
  "firstName": "John",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": ["Alice", "Jill"],
  "spouse": null
}
```

JSON in R

Getting a JSON file into R is relatively easy. We just use the **read_json()** function just as with a CSV file. The difference is that the object read into R will be, rather than a tibble, a list object. Here are the rules for how R creates a list from JSON data:

- **Scalar** values (Numbers, Strings, Booleans, null) become length-1 vectors
- **Arrays** are turned into unnamed lists
- **Objects** are turned into named lists

In the code that follows, let's assume that we have read the previous JSON file into R using this code:

```
obj <- read_json("file.json")
```

Parsing JSON in R

```
{
  "firstName": "John",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": ["Alice", "Jill"],
  "spouse": null
}
```

```
> obj$firstName
[1] "John"


> obj$address
$streetAddress
[1] "21 2nd Street"

$city
[1] "New York"

$state
[1] "NY"

$postalCode
[1] "10021-3100"
```

Parsing JSON in R

```
{
  "firstName": "John",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": ["Alice", "Jill"],
  "spouse": null
}
```

```
tibble(
  fname = obj$firstName,
  age = obj$age
)
```

```
# A tibble: 1 × 2
  fname   age
  <chr> <int>
1 John     27
```

Parsing JSON in R

```
{
  "firstName": "John",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": ["Alice", "Jill"],
  "spouse": null
}
```

```
tibble(
  part = names(obj$address),
  value = flatten_chr(obj$address)
)
```

```
# A tibble: 4 × 2
  part          value
  <chr>         <chr>
1 streetAddress 21 2nd Street
2 city          New York
3 state         NY
4 postalCode    10021-3100
```

Parsing JSON in R

```
{
  "firstName": "John",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": ["Alice", "Jill"],
  "spouse": null
}
```

```
map_chr(obj$phoneNumbers, ~ .x$type)
```

```
[1] "home"   "office"
```

Parsing JSON in R

```
{
  "firstName": "John",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": ["Alice", "Jill"],
  "spouse": null
}
```

```
tibble(
  fname = obj$firstName,
  type = map_chr(obj$phoneNumbers, ~ .x$type),
  num = map_chr(obj$phoneNumbers, ~ .y$number)
)
```

```
# A tibble: 2 × 3
  fname type   num
  <chr> <chr>  <chr>
1 John  home   212 555-1234
2 John  office 646 555-4567
```