

# JSON

WORKING WITH WEB DATA IN R



**Charlotte Wickham**  
Instructor

# JSON (JavaScript Object Notation)

<http://www.json.org/>

- Plain text format
- Two structures:
  - objects: `{"title" : "A New Hope", "year" : "1977"}`
  - arrays: `[1977, 1980]`
- Values: `"string"` , `3` , `true` , `false` , `null` , or another object or array

# An example JSON data set

```
[  
  {  
    "title" : "A New Hope",  
    "year" : 1977  
  },  
  {  
    "title" : "The Empire Strikes Back",  
    "year" : 1980  
  }  
]
```

# Identifying a JSON response

```
library(httr)
url <- "http://httpbin.org/get"
r <- GET(url)
http_type(r)
```

```
"application/json"
```

```
writeln(content(r, as = "text"))
```

```
No encoding supplied: defaulting to UTF-8.
```

```
{  
  "args": {},  
  "headers": {  
    "Accept": "application/json, text/xml, application/xml, */*",  
    "Accept-Encoding": "gzip, deflate",  
    "Connection": "close",  
    "Host": "httpbin.org",  
    "User-Agent": "libcurl/7.54.0 r-curl/2.8.1 httr/1.2.1"  
  },  
  "origin": "98.232.182.170",  
  "url": "http://httpbin.org/get"  
}
```

# Let's practice!

WORKING WITH WEB DATA IN R

# Manipulating JSON

WORKING WITH WEB DATA IN R



**Oliver Keyes**  
Instructor

# Movies example

```
[  
  {  
    "title" : "A New Hope",  
    "year" : 1977  
  },  
  {  
    "title" : "The Empire Strikes Back",  
    "year" : 1980  
  }  
]
```



# Movies example

```
movies_json <- '  
  [  
    {  
      "title" : "A New Hope",  
      "year" : 1977  
    },  
    {  
      "title" : "The Empire Strikes Back",  
      "year" : 1980  
    }  
  ]'
```

```
fromJSON(movies_json, simplifyVector = FALSE)
```

```
[[1]]  
[[1]]$title  
[1] "A New Hope"  
  
[[1]]$year  
[1] 1977  
  
[[2]]  
[[2]]$title  
[1] "The Empire Strikes Back"  
  
[[2]]$year  
[1] 1980
```

# Simplifying the output (I)

`simplifyVector = TRUE` (arrays of primitives become vectors)

# Simplifying the output (II)

```
simplifyDataFrame = TRUE
```

 (arrays of objects become data frames)

```
fromJSON(movies_json, simplifyDataFrame = TRUE)
```

```
      title year
1  A New Hope 1977
2 The Empire Strikes Back 1980
```

# Extracting data from JSON (I)

```
fromJSON(movies_json, simplifyDataFrame = TRUE)$title
```

```
[1] "A New Hope"           "The Empire Strikes Back"
```

# Extracting data from JSON (II)

Iterate over list

- `rlist`
- `base`
- `tidyverse`

# Let's practice!

WORKING WITH WEB DATA IN R

# XML structure

WORKING WITH WEB DATA IN R



**Charlotte Wickham**  
Instructor



# Movies in XML

```
<?xml version="1.0" encoding="UTF-8"?>
<movies>
  <movie>
    <title>A New Hope</title>
    <year>1977</year>
  </movie>
  <movie>
    <title>The Empire Strikes Back</title>
    <year>1980</year>
  </movie>
</movies>
```

- Tags: `<tagname>` ... `</tagname>` .
- E.g. `<movies>` , `<movie>` , `<title>` , `<year>`

# Tags can have attributes

```
<?xml version="1.0" encoding="UTF-8"?>
<movies>
  <movie>
    <title year = "1977">A New Hope</title>
  </movie>
  <movie>
    <title year = "1980">The Empire Strikes Back</title>
  </movie>
</movies>
```

# The hierarchy of XML elements

```
<?xml version="1.0" encoding="UTF-8"?>
<movies>
  <movie>
    <title>A New Hope</title>
    <year>1977</year>
  </movie>
  <movie>
    <title>The Empire Strikes Back</title>
    <year>1980</year>
  </movie>
</movies>
```

# The hierarchy of XML elements

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<movies>
```

```
  <movie>
```

```
    <title>A New Hope</title>
```

```
    <year>1977</year>
```

```
  </movie>
```

**movie element**

```
  <movie>
```

```
    <title>The Empire Strikes Back</title>
```

```
    <year>1980</year>
```

```
  </movie>
```

```
</movies>
```

# The hierarchy of XML elements

```
<?xml version="1.0" encoding="UTF-8"?>
<movies>
  <movie>
    <title>A New Hope</title>
    <year>1977</year>
  </movie>
  <movie>
    <title>The Empire Strikes Back</title>
    <year>1980</year>
  </movie>
</movies>
```

**title element**

**year element**

# The hierarchy of XML elements

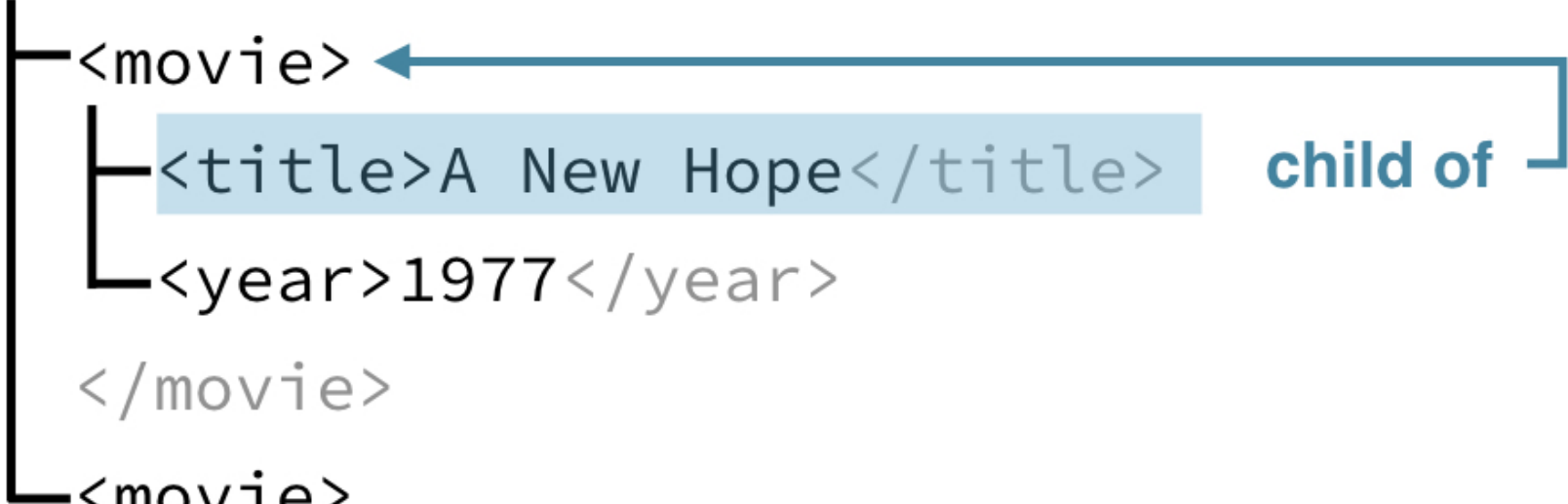
```
<?xml version="1.0" encoding="UTF-8"?>
<movies>
  <movie>
    <title>A New Hope</title>  text
    <year>1977</year>
  </movie>
  <movie>
    <title>The Empire Strikes Back</title>
    <year>1980</year>
  </movie>
</movies>
```

# Understanding XML as a tree

```
<?xml version="1.0" encoding="UTF-8"?>
<movies>
├─<movie>
│   ├─<title>A New Hope</title>
│   └─<year>1977</year>
└─</movie>
├─<movie>
│   ├─<title>The Empire Strikes Back</title>
│   └─<year>1980</year>
└─</movie>
</movies>
```

# Understanding XML as a tree

```
<?xml version="1.0" encoding="UTF-8"?>
<movies>
  <movie>
    <title>A New Hope</title>
    <year>1977</year>
  </movie>
  <movie>
    <title>The Empire Strikes Back</title>
    <year>1980</year>
  </movie>
</movies>
```



child of



# Understanding XML as a tree

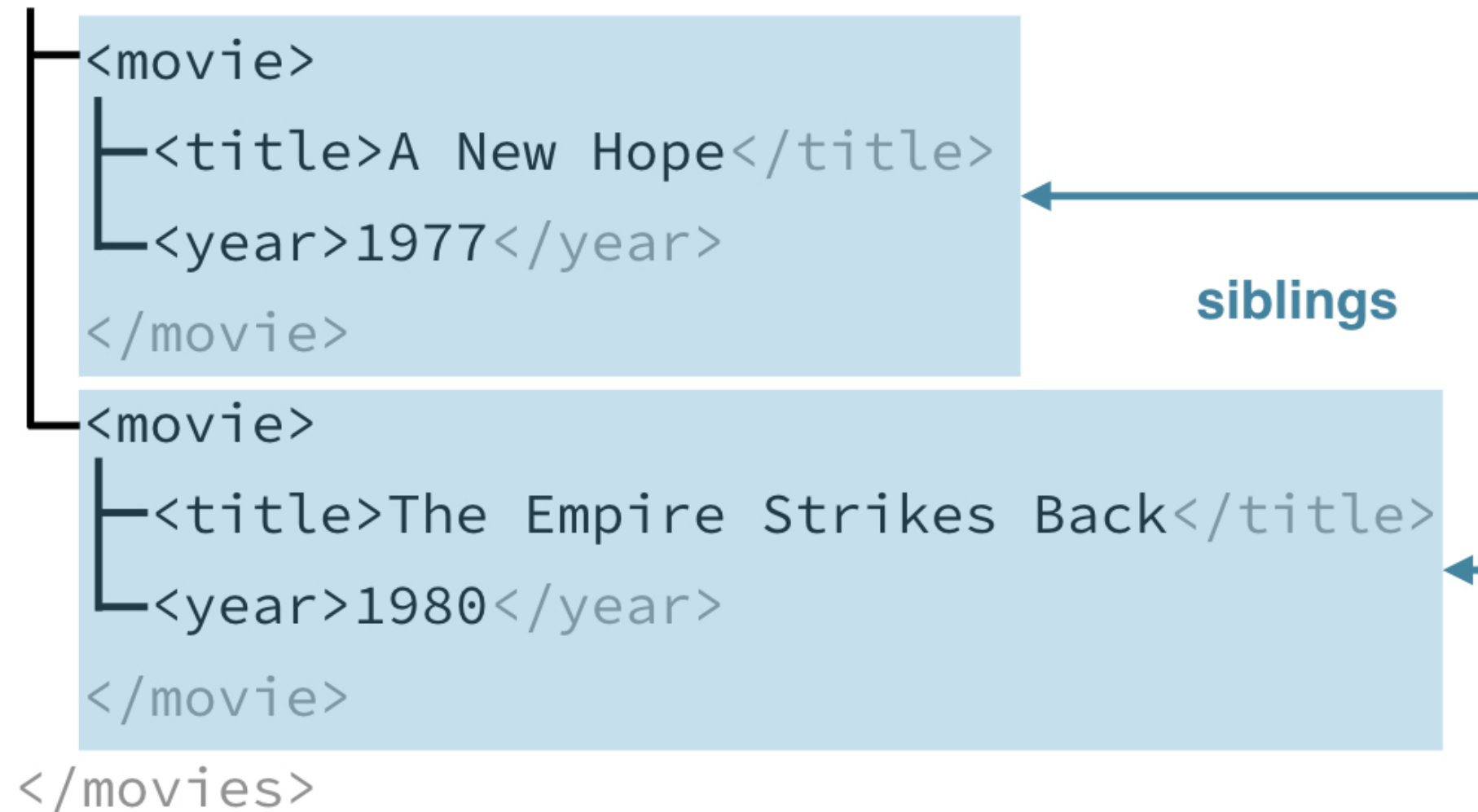
```
<?xml version="1.0" encoding="UTF-8"?>
<movies>
  <movie>
    <title>A New Hope</title>
    <year>1977</year>
  </movie>
  <movie>
    <title>The Empire Strikes Back</title>
    <year>1980</year>
  </movie>
</movies>
```

siblings

# Understanding XML as a tree

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<movies>
```



# Let's practice!

WORKING WITH WEB DATA IN R

# XPATHS

WORKING WITH WEB DATA IN R



**Oliver Keyes**  
Instructor

# Movies example

```
<?xml version="1.0" encoding="UTF-8"?>
<movies>
  <title>"Star Wars"</title>
  <movie episode = "IV">
    <title>A New Hope</title>
    <year>1977</year>
  </movie>
  <movie episode = "V">
    <title>The Empire Strikes Back</title>
    <year>1980</year>
  </movie>
</movies>
```

# Movies example

```
movies_xml <- read_xml('
<?xml version="1.0" encoding="UTF-8"?>
<movies>
  <title>"Star Wars"</title>
  <movie episode = "IV">
    <title>A New Hope</title>
    <year>1977</year>
  </movie>
  <movie episode = "V">
    <title>The Empire Strikes Back</title>
    <year>1980</year>
  </movie>
</movies>')
```

# XPATHS

- Specify locations of nodes, a bit like file paths:

```
/movies/movie/title
```

- `xml_find_all(x = ____, xpath = __)`

# XPATHS

```
xml_find_all(movies_xml, xpath = "/movies/movie/title")
```

```
{xml_nodeset (2)}  
[1] <title>A New Hope</title>  
[2] <title>The Empire Strikes Back</title>
```



# XPATHS

```
xml_find_all(movies_xml, xpath = "/movies/movie/title")
```

```
{xml_nodeset (2)}  
[1] <title>A New Hope</title>  
[2] <title>The Empire Strikes Back</title>
```

```
# Store the title nodeset  
title_nodes <- xml_find_all(movies_xml,  
                             xpath = "/movies/movie/title")  
  
# Extract contents with xml_text()  
xml_text(title_nodes)
```

```
[1] "A New Hope"           "The Empire Strikes Back"
```

# Other XPATH Syntax

- `//` - a node at any level below
- `//title`

```
xml_find_all(movies_xml, "//title")
```

```
{xml_nodeset (3)}  
[1] <title>"Star Wars"</title>  
[2] <title>A New Hope</title>  
[3] <title>The Empire Strikes Back</title>
```

# Other XPATH Syntax

- `@` - to extract attributes
- `//movie/@episode`

```
xml_find_all(movies_xml, "//movie/@episode")
```

```
{xml_nodeset (2)}  
[1] episode="IV"  
[2] episode="V"
```

# Or...

- `xml_attr()`
- `xml_attrs()`

# Wrap Up

XPATH	Meaning
<code>/node</code>	Elements with tag <code>node</code> at this level
<code>//node</code>	Elements with tag <code>node</code> anywhere at or below this level
<code>@attr</code>	Attribute with name <code>attr</code>

- Get nodes with `xml_find_all()`
- Extract contents with `xml_double()` , `xml_integer()` or `as_list()`

# Let's practice!

WORKING WITH WEB DATA IN R