# fx JSON viewer and processor

**START HERE**

`fx <file>`
Browse JSON interactively

`fx <file> .items '?.active'`
Filter an array

`curl <url> | fx`
Browse piped JSON

`fx --yaml <file>`
Read YAML

`fx <file> .<key>`
Pick one field

`fx <file> '.items.map(x => x.id)'`
Transform with JS

`fx <file> .`
Pretty-print

`fx <file> .items @.name`
Map over an array

**Input** file or stdin → parse → **Arg 1** .users → x → **Arg 2** @.name → x → **Output** JSON or text

Each argument gets the previous result as x

## CONCEPTS

Every argument is a JavaScript function; the result of one is the input x of the next. Start with . to skip x ⇒ x.

fx draws the interactive UI on stderr, so stdout only carries what you print with P or pipe out.

## RUN AND INPUT

`fx <file>`
Open a file interactively

`<cmd> | fx`
Open piped JSON interactively

`fx <file> <expr>`
Apply an expression, print result

`<cmd> | fx .`
Pretty-print stdin

`fx <file> <expr1> <expr2>`
Chain expressions

`curl -i <url> | fx`
Show response headers too

`fx --slurp <expr>`
Read a stream as one array

`fx --strict .`
Reject invalid JSON

`fx --version`
Print version

## RUN AND INPUT (cont.)

`fx --comp bash`
Print shell completion script
*Also zsh and fish: source <(fx --comp zsh)*

## EXPRESSIONS

`.<key>`
Select a field

`.<key>[0]`
Select an array element

`.<key>[]`
Flatten array items

`x ⇒ x.<key>`
Explicit arrow function

`'.map(x ⇒ x.<key>)'`
Map over an array in JS

`'.filter(x ⇒ x.<key> > 1)'`
Filter an array in JS

`@.<key>`
Shorthand for map

`?.<key>`
Shorthand for filter

`'({...x, <key>: 1})'`
Copy an object with changes

`'Object.keys'`
Call any JS function by name

`` '`Hi ${x}`' ``
Template string output

`'x.<key> = 1, x'`
Edit, then return the input

## BUILT-IN FUNCTIONS

`len`
Length of array, string or object

`keys / values`
Object keys or values

`sort / sortBy(fn)`
Sort an array

`uniq`
Unique values

`groupBy(fn)`
Group items by a result

`map(fn) / filter(fn)`
Curried map and filter

`flatten`
Flatten nested arrays

`reverse`
Reverse an array

`chunk(<n>)`
Split into groups of n

`walk(fn)`
Apply fn to every value, bottom-up

`list`
Print array as lines for xargs

`save`
Write edits back to the input file

`toBase64 / fromBase64`
Encode or decode base64

## BUILT-IN FUNCTIONS (cont.)

`YAML.stringify`
Convert JSON to YAML

`YAML.parse`
Convert YAML text to JSON

## FORMATS

`--yaml`
Read YAML

`--toml`
Read TOML

`--xml`
Read XML (attributes as "@name")

`--html`
Read HTML

`--raw / -r`
Process input line by line as text

`-rs`
Raw lines as one string array

File extensions pick the format automatically (.yaml, .toml, .xml). stdin has no name, so it needs the flag.

## INTERACTIVE: MOVE

**Arrows**
Move; h j k l also work

## INTERACTIVE: MOVE (cont.)

**g** / **G** Top / bottom

**Space** / **b** Page down / up

**d** / **u** Half page down / up

**J** / **K** Next / previous sibling

**[** / **]**
Back / forward in history

**@**
Fuzzy-search a path and jump

**q** Quit

## INTERACTIVE: FOLD AND VIEW

**Enter** / **l** Expand node

**Backspace** / **h**
Collapse node

**L** / **H**
Expand / collapse recursively

**e** / **E** Expand / collapse all

**1-9** Collapse to nth level

**?** Show help

## INTERACTIVE: SEARCH AND OUTPUT

**.** Open query input

## INTERACTIVE: SEARCH AND OUTPUT (cont.)

**/** Regex search

**n** / **N** Next / previous match

**Tab** / **Shift** + **Tab**
Complete or cycle in query input

**Esc** Cancel the query

**y** Yank (copy) value

**p** Preview value

**P**
Print value to stdout and exit

`:w <file>`
Write on-screen JSON to a file

## RECIPES

**Pull one column as lines**

```
curl -s <url> | fx .items @.name list
```

**Convert YAML config to JSON**

```
fx config.yaml . > config.json
kubectl get pod -o yaml | fx --yaml .spec
```

**Rename a field in place**

```
fx <file> 'x.name = x.name.toUpperCase(), x' save
```

**Explore, pick, save**

```
curl -s <url> | fx > part.json
# move to a value, press P
```