Building API functions in JS

Building API functions with declarative I/O contracts & the ARCcore.filter library

https://encapsule.io/docs/ARCcore/filter

SeattleJS Meetup Lightning Talk May 11, 2017

Chris Russell // cdr@encapsule.io
https://github.com/Encasule
https://twitter.com/Encapsule

How would you write...

… a JavaScript function that accepts:

- Only a string?
- An array of strings from a fixed set of words?
- An object with required properties `x` and `y` that are both numerical values?
- A dictionary of arrays of numbers between 0 and 100?
- An object with optional properties that get set to default values if not specified?
- Some larger combination of all of the above?

How would you ensure...

- Your algorithm doesn't process bad input?
- Your algorithm doesn't produce an unexpected result?
- That function I/O errors are reported consistently?
- Other developers are able to use your API function with high confidence?
- The documentation is up-to-date?
- Requirements are easily shared with your team/partners?
- Your team can respond to changing requirements quickly?

Typical answers...

- **We don't have time to worry about it. Make it work!**

Or, if everyone agrees it's mission critical:

- Write lots of nasty **prologue code** in your API functions to “filter” out bad input & set default values.
- Write & maintain lots of **external unit tests** to ensure against bad output result(s).
- **Write docs** by hand. Or, use machine-readable comments in source code to automate documentation.

~~Typical~~ BAD answers...

- ~~**We don't have time to worry about it. Make it work!**~~

~~**Or, if everyone agrees it's mission critical:**~~

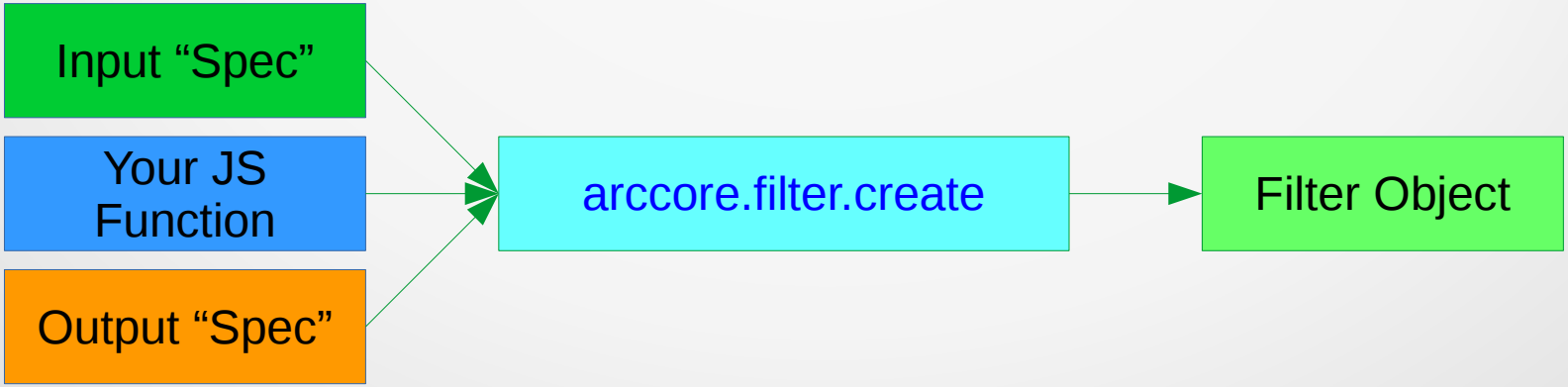
API's are mission critical. Period.

- ~~Write lots of nasty preamble code in your API functions to “filter” out bad input & set default values.~~
- ~~Write & maintain lots of external unit tests to ensure against bad output result(s).~~
- ~~Write docs by hand. Or, use machine-readable comments in source code to automate documentation.~~

We need to accomplish these tasks more cheaply!

ARCcore.filter factory

- ARCcore is an npm package
- **filter** is an exported library
- Export function arccore.filter.create is a factory function:
  - Input and output “specifications” (optional)
  - Developer-defined JS **transform** function (optional)



Build a filter object (live)

- Simple demo in Node.js console...

- **In-depth**:
  - https://encapsule.io/docs/ARCcore/filter/api
  - https://encapsule.io/docs/ARCcore/filter/architecture

Filter specifications

In depth: https://encapsule.io/docs/ARCcore/filter/specs

- Simple declarative objects
- Recursive structure that's simple to read
- Uses reserved **quanderscore** directives (four underscore prefix)
- Property name validation / pruning
- JSON elements types + functions + opaque (any)
- Small number of value checks
  - In set
  - In range
- Default values

Filter demos (live)

- Use the online interactive example to learn & explore
- Online: https://encapsule.io/docs/ARCcore/filter/examples

Summary

- Filter is backed by over 500 tests
- Use in Node.js and browser (via webpack)
- Time to learn filter < time to implement one non-trivial API function
- Simplify design process with filter specifications
- Makes large impact on code quality
- Filter reduces the size of your app bundles
- Filter works at scale:
  - e.g. Holism is a filter-derived replacement for Express

Thanks to SeattleJS!