Commonly useful Promistream packages

All of the existing Promistream packages can be found in the package list, but they're not very well-explained. The majority of these should be functional and have an `example.js` demonstrating their use.

Here's a selection of the packages you are most likely to need:

Common cases

- `@promistream/pipe`: the core package that pipes streams together into a pipeline. Technically optional but strongly recommended to use.
- `@promistream/debug`: transform stream that simply prints everything that goes through it, optionally with a label. Only for pipeline debugging use.
- `@promistream/simple-source`: low-level source stream abstraction. Implements the specification responsibilities, leaving you to worry only about how to produce values. Suitable for the majority of usecases.
- `@promistream/simple-sink`: low-level sink stream abstraction. Same as above, but on the other end.
- `@promistream/map`: like the array method, but as a Promistream. Also functions as a general-purpose low-level transform stream.
- `@promistream/filter`: like the array method, but as a Promistream.
- `@promistream/collect`: high-level sink stream, that simply read-loops and collects all read values into an array, then resolves with that array. Often what you want. Also a good example of `@promistream/simple-sink` use, internally.
- `@promistream/from-node-stream`: source/sink/transform wrappers for all types of Node.js streams, to integrate them with a Promistream pipeline.
- `@promistream/from-iterable`: creates source stream from a synchronous or asynchronous iterable (including arrays).
- `@promistream/range-numbers`: high-level source stream, generates numbers in a specified range.

Complex cases

- `@promistream/buffer`: reads an array (of 0 or more items) from upstream, and then dispenses the values in that array (if any) one by one on subsequent reads by its

downstream. Often composed with others.
- `@promistream/dynamic`: lets you pass a value through different streams/pipelines depending on the value. Finicky and high-overhead; usually fork-and-merge is a better option.
- `@promistream/parallelize`: lets you run N reads (up to and including `Infinity`) simultaneously.
- `@promistream/sequentialize`: forces inbound reads from downstream to occur sequentially, 'protecting' its upstream. **Mandatory** to use if your stream does not support parallel operation and you intend to publish it, or you use `parallelize` elsewhere in your pipeline.
- `@promistream/rate-limit`: as the name implies, throttles reads going through it but leaves results otherwise unmodified.
- `@promistream/simple-queue`: high-level source stream that functions as a task queue; items can be added externally.

Specific usecases

- `@promistream/read-file`: as the name implies. Produces buffers.
- `@promistream/decode-string`: as the name implies. Takes buffers, produces strings.
- `@promistream/split-lines`: as the name implies. Takes strings.
- `@promistream/parse-xml`: streaming XML parser.

There are other Promistream packages, and there will be many more! These are just some of the ones currently available, that you're likely to need at some point.

Revision #1
Created 10 December 2024 23:48:12 by joepie91
Updated 10 December 2024 23:48:48 by joepie91