POC FOR MORE INFO (mor1arty): https://evil.com

Node.js API list that can utilize AbortController

Authors: Benjamin Gruenbaum (**Please add yourself in as you make edits and please feel free to make edits)**
- Philip Chimento

Ongoing WIP going over the module list and figuring out what there can utilize AbortController based on the action being actually abortable and discussions of what the API could look like:

Node.js module list (from the docs):

- ~~**Assertion Testing - no integration points**~~
  - ~~None of these APIs take callbacks or return promises. So probably none relevant. The only thing here that has the word "promise" in it is `assert.reject/doesNotReject(asyncFn[, error][, message])`.~~
    - ~~Looking at prior art from C#, it looks like passing a controller or signal to it doesn't make sense.~~
- ~~**Async Hooks**~~
  - ~~Note: async hooks are still marked experimental.~~
  - ~~It is super interesting to investigate how async hooks are impacted by cancellation but none of the async hooks APIs (for understandable reasons) use promises. So none appropriate.~~
  - ~~Opened issue at diagnostics repo to make sure there are no unforeseen consequences to async_hooks~~
- ~~**Buffer**~~
  - ~~None of the buffer APIs are asynchronous and thus none of them support cancellation.~~
  - ~~I think cancellation could be interesting (aborting an expensive copy from a controller on another worker_thread) but I don't think we should investigate it initially.~~
- **C++ Addons**
  - We would have to expose cancellation to native addons somehow. Chromium's AbortController is in C++ and can be seen as prior art. In general cancellation should be usable from c++ but I recommend we sanction a way (at least for n-api).

- **C/C++ Addons with N-API**
    - In general n-api has decent support for promises and we probably want to consider a sanctioned way to cancel or react to cancellation. We probably want to expose a specific way to cancel in the future but for the first iteration people can use the regular n-api APIs for working with the AbortController/Signal like regular JS values.
- **C++ Embedder API**
    - Benjamin: Not relevant as far as I know but we probably want someone who is more familiar with the API to discuss.
- ~~**Child Processes**~~
    - ~~Do we want cancellation to `kill(SIGTERM)` the child process or to send a "softer" signal?~~
        - ~~We probably want to use whatever is passed as `killSignal` since that's what a timeout sends and cancellation should probably behave like the closest thing we have now (timeout).~~
    - ~~API list:~~
        - ~~child_process.exec(command[, options])~~
            - ~~Options can get a new `signal` argument like `fetch` has. This can work for both the callback and promisifed version. When invoking it it can send a killSignal to the child.~~
        - ~~child_process.execFile(file[, args][, options])~~
            - ~~Options can get a new `signal` argument like `fetch` has. This can work for both the callback and promisifed version. When invoking it it can send a killSignal to the child.~~
        - ~~child_process.fork(modulePath[, args][, options])~~
            - ~~Apparently fork doesn't have a killSignal argument (why? No timeout?) but we can apply the same logic as the two above cases.~~
        - ~~child_process.spawn(command[, args][, options])~~
            - ~~Apparently fork doesn't have a killSignal argument (why? No timeout?) but we can apply the same logic as the three above cases.~~
- **Cluster**
    - Random note: I had no idea cluster is stable and not deprecated :D
        - Cluster workers are currently aborted like `child_process`es with a `kill` method.
            - I recommend creating a controller on the master side and passing its signal to `.fork`, then the child gets the controller exposed to it.

        - Prior art of a similar design in Piscina
- **Command Line Options**
    - Not relevant (these are just CLI options and not APIs)
- **Console**
    - We would ideally only support these APIs here when WHATWG does too. None of these APIs are async.
- **Crypto**
    - Our current API doesn't expose easy ways to cancel, probably because it's not usually desirable.
    - Async crypto calls (like scrypt) typically schedule work on the threadpool.
        - It looks like a lot of these APIs don't support cancellation in OpenSSL (examples: RAND_bytes, EVP_PBE_scrypt).
        - We *could* theoretically kill the thread but we probably *shouldn't* as the overhead would likely not be worth it except in pathological cases.
- **Debugger**
    - This is just a doc about using the debugger. Once tokens land it could be interesting to explore a tighter integration with the V8 inspector.
- **Deprecated APIs**
    - This is just a list of deprecated APIs :]
- **DNS**
    - DNS currently supports cancellation with the callback API with `resolver.cancel()`.
    - DNS currently does not support cancellation with the promises API.
        - Allow passing a signal to the resolver constructor: `new Resolver({signal})`
        - When that resolver aborts - perform the same logic the callback API performs when you call cancel. Ideally also support passing a `signal` to the callback API.
- **Domain**
    - I do not think we should need any integration for this.
- **ECMAScript Modules**
    - Plenty of things here like `import` probably - but I don't think we should make a choice here without JavaScript language support as dynamic import is part of the language.
    - Opened an issue in the modules team repo
- **Errors**
    - No relevant APIs (this module just lists the list of errors).
- **Events**

    - We probably don't want to bake cancellation into our EventEmitters. Dom EventTarget actually does have a form of cancellation (of future handlers) but not with AbortController.
- File System
    - This is a big one, but basically:
        - Everything streams based gets cancellation from the streams. For example `fs.createReadStream`.
        - Everything else needs ad-hoc cancellation where it makes sense (supported by libuv or internally in Node.js)
        - Cancellation is more important for typically slower actions (fs.readFile) and not relatively fast ones (like fs.exists or fs.access)
            - TODO(benjamin): check with Deno and browser fs API
        - Libuv does support cancellation (uv_cancel) of fs requests (uv_fs_t) which looks like it mostly just remove the work from the queue rather than OS level cancellation. (see uv__work_cancel)
        - **TODO(Benjamin) - keep working on this one with the branch.**
- Globals
    - queueMicrotask? There isn't currently a way to cancel a microtask
- HTTP
    - We probably only want to wait for `fetch` with this if we want our client to do it.
    - http.get, http.ClientRequest
    -
- HTTP/2
- HTTPS
- Inspector
    - inspector.Session.post?
    - Other APIs are blocking, not asynchronous
- Internationalization
    - No relevant APIs
- Modules
    - No relevant APIs
- Net
    - net.Server options
    - net.Socket options
    - net.connect options, net.createConnection options, net.createServer options
- OS
    - No relevant APIs

- Path
    - No relevant APIs
- Performance Hooks
- Policies
- Process
- Punycode
- Query Strings
- Readline
    - Question API waits for user input, could support cancellation
- REPL
- Report
- Stream
- String Decoder
- Timers
    - Probably dependant on web supporting signal option
- TLS/SSL
- Trace Events
- TTY
- UDP/Datagram
- URL
- Utilities
- V8
    - Some heap stat APIs return streams, they may be cancellable
- VM
    - vm.measureMemory returns a promise, may be cancellable
- WASI
- Worker Threads
- Zlib