Open-Source Report

Proof of knowing your stuff in CSE312

Guidelines

Provided below is a template you must use to write your reports for your project.

Here are some things to note when working on your report, specifically about the **General Information & Licensing** section for each technology.

- **Code Repository**: Please link the code and not the documentation. If you'd like to refer to the documentation in the **Magic** section, you're more than welcome to, but we need to see the code you're referring to as well.
- **License Type**: Three letter acronym is fine.
- **License Description**: No need for the entire license here, just what separates it from the rest.
- **License Restrictions**: What can you *not* do as a result of using this technology in your project? Some licenses prevent you from using the project for commercial use, for example.

Also, feel free to extend the cell of any section if you feel you need more room.

If there's anything we can clarify, please don't hesitate to reach out! You can reach us using the methods outlined on the course website or see us during our office hours.

Koa.js

General Information & Licensing

Code Repository	https://github.com/koajs/koa/
License Type	MIT license
License Description	● Commercial use ● Modification ● Distribution ● Private use
License Restrictions	● Liability ● Warranty

Magic ⋆★˚｡⋆˚☽˚⁀➷｡˚★彡⋆༄

The `http_parser` is from a library called `llhttp`, it detects the first few characters of the data from the socket, if the data starts with `HTTP/`, then the parser knows this is a HTTP response, otherwise, it's a HTTP request. Then, it treats the next line as the beginning of the headers(by looking for `\r\n` or `\n`). And it keeps looking for `:` character to split the header field and value. Then the content following is the body.

```
n('start')
  .match([ '\r', '\n' ], n('start'))
  .otherwise(
   this.load('initial_message_completed', {
     1:     this.invokePausable('on_reset',     ERROR.CB_RESET,
n('after_start')),
  }, n('after_start')),
);
```

```
n('req_or_res_method')
    .select(H_METHOD_MAP, this.store('method',
       this.update('type', TYPE.REQUEST, this.span.method.end(
                        this.invokePausable('on_method_complete',
ERROR.CB_METHOD_COMPLETE, n('req_first_space_before_url')),
      )),
    ))
       .match('HTTP/',  this.span.method.end(this.update('type',
TYPE.RESPONSE,
       this.span.version.start(n('res_http_major')))))
       .otherwise(p.error(ERROR.INVALID_CONSTANT,  'Invalid  word
encountered'));
```

```
n('header_field_colon')
     // https://datatracker.ietf.org/doc/html/rfc7230#section-3.2.4
       //  Whitespace  character  is  not  allowed  between  the  header
field-name
      //  and  colon.  If  the  next  token  matches  whitespace  then  throw
an  error.
     //
      //  Add  a  check  for  the  lenient  flag.  If  the  lenient  flag  is
set,  the
       //  whitespace  token  is  allowed  to  support  legacy  code  not
following
     // http specs.
    .peek(' ', checkLenientFlagsOnColon)
    .peek(':', span.headerField.end().skipTo(onHeaderFieldComplete))
     // Fallback to general header, there're additional characters:
     // `Connection-Duration` instead of `Connection` and so on.
    .otherwise(this.resetHeaderState('header_field_general'));
```

Then, with the HTTP message parsed by the built-in Http Server, the built-in Http Server will fire the callback function passed from the Koa framework, and Koa will take over and call the `handleRequest` function to build a Koa Context object, and this context object will be passed to several Koa Middlewares to add miscellaneous information of this request (such as handle cookie, multi-part form data and so on).