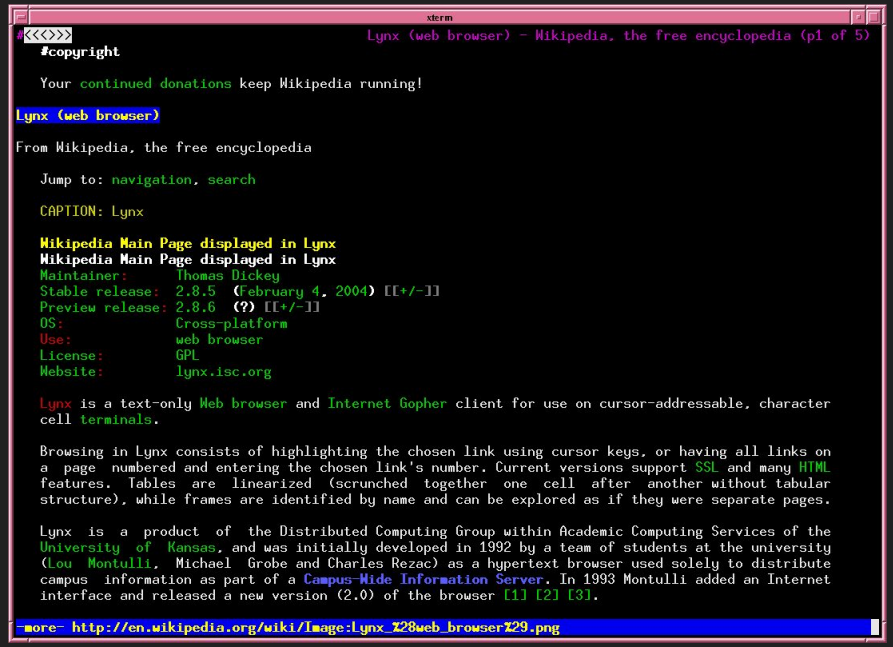
Text-Based Web Browser

Team Good Meowning

By Yichen Cui, Andi Fan, Nicki Nguyen, Myles Thiessen

What did we build?



Lynx Web Browser
https://lynx.invisible-island.net/

Good Meowning's
Text-based Web Browser

Why does our project matter?

- To take a deep dive into learning about and implementing the web browser
  - Understand the basic concept of web browser: fetching data through HTTP requests, processing HTML data, displaying data in the terminal
  - Revealed the difficulty behind implementation of some core features present in modern browsers
- Text-based web browsers are still in-use today! For instance, the Gentoo Linux installation handbook suggest users to use a terminal web browser (Lynx) for initial tasks like downloading stage files for installation

What are our restrictions?

1. HTML only!
    - No CSS, JavaScript, JSON, etc...
    - No JSON
    - No cookies
    - No local storage
2. No media
    - No images, animations, videos, etc...

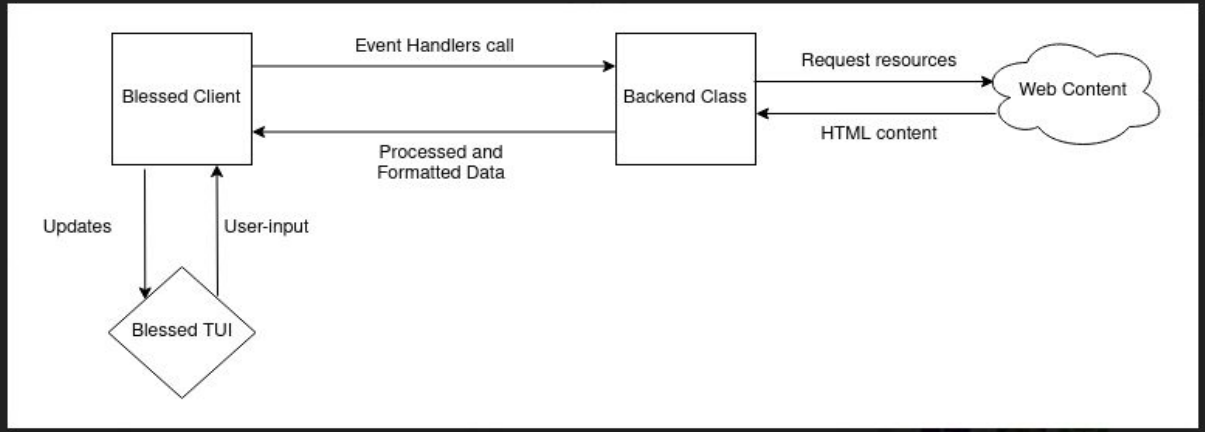
DEMO

Major Features

- Visit web pages through our terminal browser
  - Accept an URL to a website or a path to a local file
  - Display the text from the HTML file at the specified URL or file path
  - Display the current URL visited on the top left of the screen
  - Include an insert-mode to enter URLs or embedded hyperlinks
- Interact with web pages through our terminal browser
  - Cycle through and visit embedded hyperlinks
  - Scroll up/down the screen
- Quality-of-life features
  - Include a help box to display commands (stretch goal)
  - Multiple browsing tabs (stretch goal)
  - Keep a list of browsing history (stretch goal)

Major Components

1. Frontend User Client
   - Using Blessed as our TUI library
   - Initializes blessed components and event handlers
   - Displays parsed web content
2. Backend Class
   - Retrieves HTML content, and stores in a hierarchical structure using Cheerios
   - Sanitizes user input
   - Standardizes respond before passing to blessed-client
   - Parses content tree and assigns blessed styling tags to HTML elements




```
src
├── app.ts
├── backend-class
│   ├── blessed-server.ts
│   ├── data-fetcher.ts
│   ├── data-parser.ts
│   └── index.ts
└── blessed-client
    ├── blessed-client.ts
    ├── box-attributes.ts
    └── index.ts
test
├── cases-data-parser
│   ├── cases-axios.ts
│   ├── cases-fs.ts
│   ├── cases-href.ts
│   └── cases-url.ts
├── cases-data-server
│   ├── cases-axios.ts
│   └── cases-fs.ts
└── test.ts
test.sh
```

Adjustments

1. Remove HTML forms as a stretch goal feature

    - HTML forms without JavaScript are not useful
    - Difficult to implement due to variations

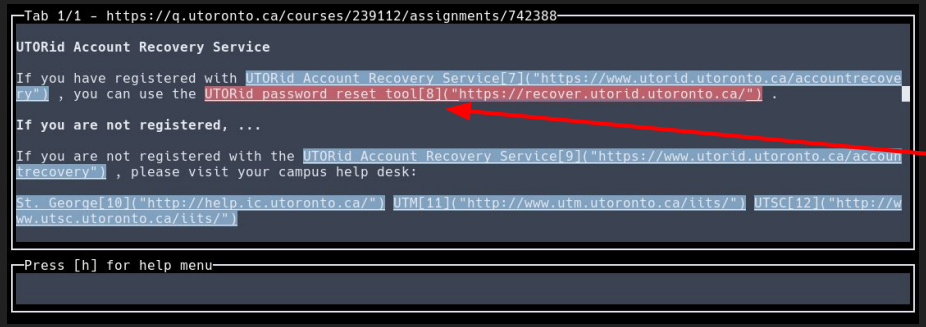
A form with input fields for text:

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname">
</form>
```

Adjustments

2. Interact with hyperlinks through key-presses instead of mouse clicks

- Ideally we wanted the user to be able to click on the hyperlinks, but there is no support for this feature in the frontend library
- Workaround ideas:
  - Use a Layout Element as a container class, involves nesting Blessed elements
  - Calculating the index of the hyperlink using the 2D cursor click position
  - Hashing text around hyperlinks to use text snapshots around click position to search for the corresponding hyperlink



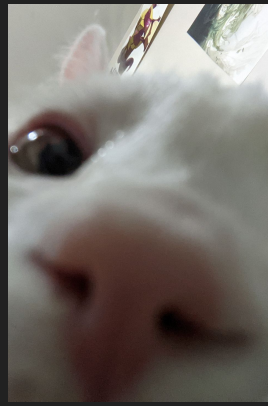
Testing and Validation

- Verified acceptance criteria for features with industry partner
- Branch protection rules
- Manual QA
- Units tests written for backend features
  - Covers all edge-cases that we can think of, and all past bugs we have experienced
  - Overall branch code coverage of 88% according to istanbul.js

```
// Remote files via Axios
describe("Test remote files", () => {
  // Positive cases
  casesDataServerHTTP.forEach(
    ([url, parsedURL, output, outputForward, outputBackward, href]) => {
      it(url, async () => {
        const dataServer = new DataServer();
        await dataServer.visitURL(url);
        chai.assert.deepEqual(dataServer.renderPage(), [output, parsedURL]);
        chai.assert.deepEqual(dataServer.renderPage(6), [
          outputForward,
          parsedURL
        ]);
        chai.assert.deepEqual(dataServer.renderPage(-8), [
          outputBackward,
          parsedURL
        ]);
        chai.assert.equal(dataServer.getHrefURL(), href);
      });
    }
  );
```







❤️ Thank you for listening



❤️ Special thanks to Mike, Greg, and Julia for your help and guidance! ❤️