↳ javascript & html
→ python

↳ tracking the stocks
↳ interest rates
↳ expenses budgeting website.

↳ API → purchases

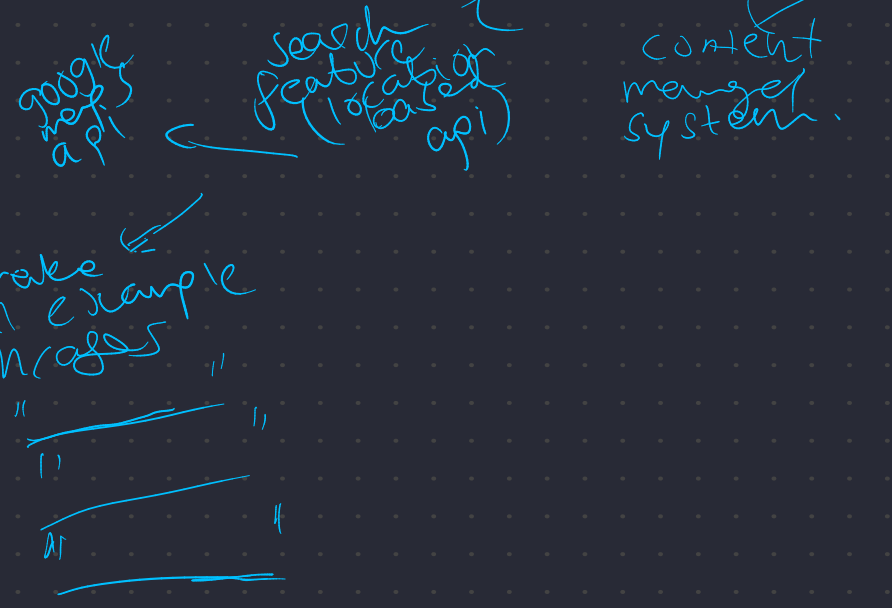
↳ map → latitude & longitude
↳ ~~to~~ tapping into a spacing area and shows weather information.

↳ flip → couponing website
↳ track down sales

↳ image map → tracks down the x, y, z of an image

↳ combines couponing w/ the map.

↳ ~~Sam~~ somebody plugs these integers and see what type of



Idea for the week — coupon website

1. figure out basic function
   - ↳ functionality
   - ↳ database (source)
   - ↳ platform → web scraper.

Web scraping in Javascript

↳ Javascript has only used a single thread and performed blocking operations

↳ use Node.js

↳ we don't need multithreading in JS

```
const http = require('http');
const PORT = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, () => {
  console.log(`Server running at PORT:${port}/`);
});
```

→ not a blocking call (returns immediately)

↳ need to use a HTTP client

Node.js has an HTTP library by default.

1. Built-in HTTP client

```
const http = require('http');

const req = http.request('http://example.com', res => {
    const data = [];

    res.on('data', _ => data.push(_))
    res.on('end', () => console.log(data.join()))
});

req.end();
```

push wanted data?

2. Fetch API
↳ still considered experimental
↳ also opt for polyfill/wrapper library

```
async function fetch_demo()
{
    const resp = await fetch('https://www.reddit.com/r/programming.json');

    console.log(await resp.json());
}

fetch_demo();
```

↳ json object for response object

↳ accepts additional options argument

3. Axios
↳ similar to fetch
↳ Promised based HTTP client
must install

```
npm install axios
```

```
const axios = require('axios')

axios
    .get('https://www.reddit.com/r/programming.json')
    .then((response) => {
        console.log(response)
    })
    .catch((error) => {
        console.error(error)
    });
```

```
async function getForum() {
    try {
        const response = await axios.get(
            'https://www.reddit.com/r/programming.json'
        )
        console.log(response)
    } catch (error) {
        console.error(error)
    }
}
```

call this function

Up superagent!
Has a fairly straightforward API like Axios but has more dependencies

```
const superagent = require("superagent")
const forumURL = "https://www.reddit.com/r/programming.json"

// callbacks
superagent
    .get(forumURL)
    .end((error, response) => {
        console.log(response)
    })

// promises
superagent
    .get(forumURL)
    .then((response) => {
        console.log(response)
    })
    .catch((error) => {
        console.error(error)
    })

// promises with async/await
async function getForum() {
    try {
        const response = await superagent.get(forumURL)
        console.log(response)
    } catch (error) {
        console.error(error)
    }
}
```

Benefits: extensibility -> list of plugins
allows for tweaking of a request or response
ex. superagent-throttle plugin would allow you to define throttling rules for your request

5a Request

```
const request = require('request')
request('https://www.reddit.com/r/programming.json', function (
  error,
  response,
  body
) {
  console.error('error:', error)
  console.log('body:', body)
})
```

con: not actively maintained anymore

Google Map API ideas
- Add your own map markers
- Convert coordinates to an address
- Add click events to the map
- show place details
- add data layers

Web scraping
- Use axios call the js file "scraperapi.js"

```
const axios = require('axios');
const url = 'https://www.turmerry.com/collections/organic-cotton-sheet-sets/proc
    axios(url)
        .then(response => {
            const html = response.data;
            console.log(html);
        })
        .catch(console.error);
```

After running scraper using "node scraperapi.js" in the terminal, it will pull a long unreadable string of HTML

nexf step is to identify the elements we want to extract from the HTML

--use browser dev tools
-open the inspector tool oof the website where you will see a corresponding line of code where you want get the element class

```
<div class="tt-price" itemprop="price" content="99.0">
  <span class="sale-price">$99.00</span> == $0
  <span class="old-price">$119.00</span>
  <p class="price-fs hide" style="display: inline-block
  ortant;/*font-size: 16px;color: #191919;*/"> + Free
  Shipping </p>
</div>
```

Parse the HTML using cherio

```
const axios = require('axios');
const cheerio = require('cheerio')
const url = 'https://www.turmerry.com/collections/organic-cotton-sheet-sets/pro
    axios(url)
        .then(response => {
            const html = response.data;
            const $ = cheerio.load(html)
            const salePrice = $('.sale-price').text()
            console.log(salePrice);
        })
        .catch(console.error);
```

After running your program, it will print the content tagged as .sale-price to the console:

```
}
PS C:\Users\D N L\Desktop\firstscraper> node scraperapi.js
$99.00
PS C:\Users\D N L\Desktop\firstscraper>
```

google api

1. look into data sourcing and how we are going to integrate it

2. algorithm → javascript source code → html
   ↳ set up our algorithm.
   ↳ integrate the API from google maps to js.

3. UI parts what we want to see
   ↳ website URL
   ↳ coupon
   ↳ store into different category
   ↳ groceries → dictionary
   ↳ coordinate value
   ↳ string.

text file