ReadMe

‘Article Reading Service’

By

Ben Rochlin *@BenjaminRochlin*

Shelby El-rassi *@theartydev*

What is ReadMe?

- ReadMe is an 'article reading service' in which a user inputs a link to an article and ReadMe website converts the text from the article to audio and outputs a file that can either be downloaded or streamed directly from the site.
- Purpose? It is convenient service for article reading, whether on the go, driving, public transport, or simply prefer listening over reading. It is also is useful as filters out any unnecessary text and ads.

Initial Idea/Plans

- Utilising a web scraper for any news article/blog/post
- Amazon Polly for text to speech
- Amazon s3 for file storage of mp3
- Service for emailing the file to user
- File be available for download
- Instant streaming

Implementation

- Amazon Identity Pool houses the AWS creds
- NPM modules used, with the help of Browserify Module which enabled these modules to be used client-side for this project
- Heroku Cors Anywhere proxy
- Instantiating a new AWS polly

```
const axios = require('axios')
const cheerio = require('cheerio')
const AWS = require('aws-sdk');


// Initialize the Amazon Cognito credentials provider
AWS.config.region = 'ap-southeast-2'; // Region
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
IdentityPoolId: 'ap-southeast-2:668ff36c-bac5-46f3-84af-d36e3fb590ec',
});



//enables no issues with CORS
const proxyurl = "https://cors-anywhere.herokuapp.com/";
```

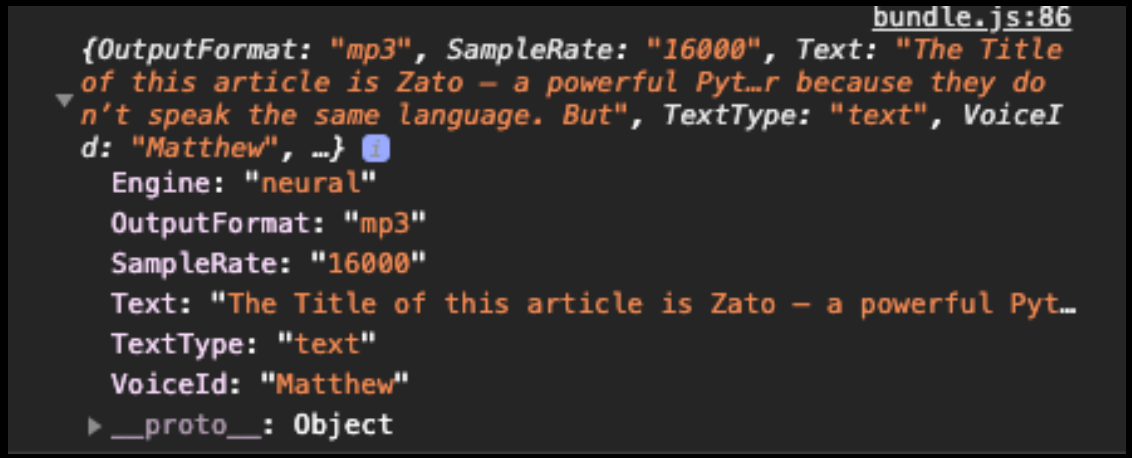
```
//getting certain elements
const myButton = document.getElementById("submit-button");
const main = document.getElementById("main");
const articleURL = document.getElementById("articleURL");
const spinnerLoad = document.getElementById("spinny-loader");


// Create the Polly service object and presigner object
var polly = new AWS.Polly({apiVersion: '2016-06-10'});
```

- User can input a URL into the search
- Starts an Async function
- When loading a loading bar will appear
- Axios will then get the URL
- Cheerio will scrape the article headings and text from the page and input them into a variable (initially looked at Puppeteer, but that can only be used server side)
- Trimmed the story length using substring due to character limits on free AWS
- Loading stops once that article title has appended to the page and link alert appended

```
//Get request the request URL
async function makeGetRequest() {
  //event.preventDefault();
  //adding the spinner class from bootstrap on for a loading bar
  spinnerLoad.classList.add("spinner-border")
  let url = document.getElementById("searched-url");

// Using the URL parameters to get the data from the page
  let res = await axios.get(proxyurl + url.value);
  let data = res.data;
  const $ = cheerio.load(data);

  // Print some specific article content
  let story = [];
  let title = $("body h1").first().text().trim();
  let storyArticle = ['The Title of this article is ' + title ].join(' ');
  story.push(storyArticle);
  $("p").map((_, element) => {
      let text = $(element).text();
      if (text.trim() !== 'Written by') {
          story.push($(element).text());
      }
  });
   console.log(story.join(''))


  //trim story due to free amazon limits to 2999 characters
  const trimmedStory = story.join('').substring(0, 2999)
  //console.log(trimmedStory.length)
```

```
  //removing the spinner class from bootstrap on for a loading bar$
  spinnerLoad.classList.remove("spinner-border")
   //appending the title to the webpage
  const textArticle = document.createElement("div");
  const textnode = document.createTextNode(storyArticle);
  main.appendChild(textArticle);         // Create a text node
  textArticle.appendChild(textnode);

  //audio link alert
  articleURL.innerHTML = "Here is your Audio Link!"

  return trimmedStory;
}
```

- On the user clicking search the get request is made then..
- Using parameters given (including the text) Amazon Polly will then Synthesize the text into a URL
- Amazon Polly will also process the URL into a clickable link and feed into audio element
- (we preferred to use a Speech Task with Polly to allow for more character limits, due to time frame we kept it simple with just Speech URL method.
- The Article Title Will Output for reference and play notification



Future Plans

- User being able to input their email address for the file to be sent to them
- Fine tuning of the text scraping so most articles would be compatible(Node Module Puppeteer would allow for this, however can only be used with a server)
- The output file being a downloadable audio file
- Settings Page: Choose Voice, speaking speed and other optional parameters
- Fine tuning validation- Url validator, errors for invalid etc

Thank you!