



오픈소스 SW활용 개발계획서  
프로젝트 명 : webConverter

이름	고현우 / 소완열
학과	소프트웨어 공학과
학번	32170171 / 32172110
과목명	오픈소스SW기여
분반	1분반
교수님	송인식 교수님
제출일자	2023 / 03 / 22

# 1. 기능 조사

## 사용 예정 기능

### 1. load

http주소나 html, mht, xml와 같이 로컬에 있는 파일을 electron상의 web contents로 로드하는 기능

```
This module cannot be used until the ready event of the app module is emitted.
```

```
// In the main process.  
const { BrowserWindow } = require('electron')  
  
const win = new BrowserWindow({ width: 800, height: 600 })  
  
// Load a remote URL  
win.loadURL('https://github.com')  
  
// Or load a local HTML file  
win.loadFile('index.html')
```

### 2. printToPdf

webContents 상의 내용을 웹 브라우저 내의 pdf로 인쇄와 동일하게 pdf로 인쇄 진행하는 기능

### 3. show / openDevTools

디버깅용 툴로, webContents가 로드 된 후 스크립트로 조작할 때 이슈가 발생한 경우 이를 통하여 트래킹

#### Using the `ready-to-show` event

While loading the page, the `ready-to-show` event will be emitted when the renderer process has rendered the page for the first time if the window has not been shown yet. Showing the window after this event will have no visual flash:

```
const { BrowserWindow } = require('electron')  
const win = new BrowserWindow({ show: false })  
win.once('ready-to-show', () => {  
  win.show()  
})
```

An example of `webContents.printToPDF`:

```
const { BrowserWindow } = require('electron')
const fs = require('fs')
const path = require('path')
const os = require('os')

const win = new BrowserWindow()
win.loadURL('http://github.com')

win.webContents.on('did-finish-load', () => {
  // Use default printing options
  const pdfPath = path.join(os.homedir(), 'Desktop', 'temp.pdf')
  win.webContents.printToPDF({}).then(data => {
    fs.writeFile(pdfPath, data, (error) => {
      if (error) throw error
      console.log(`Wrote PDF successfully to ${pdfPath}`)
    })
  }).catch(error => {
    console.log(`Failed to write PDF to ${pdfPath}: `, error)
  })
})
```

#### 4. `executeJavaScript`

`webContents`에서 파라미터로 전달한 js를 실행하는 함수로, 페이지의 margin이나, footer / header등의 문구를 추가하는 기능

`contents.executeJavaScript(code[, userGesture])`

- `code` string
- `userGesture` boolean (optional) - Default is `false`.

Returns `Promise<any>` - A promise that resolves with the result of the executed code or is rejected if the result of the code is a rejected promise.

Evaluates `code` in page.

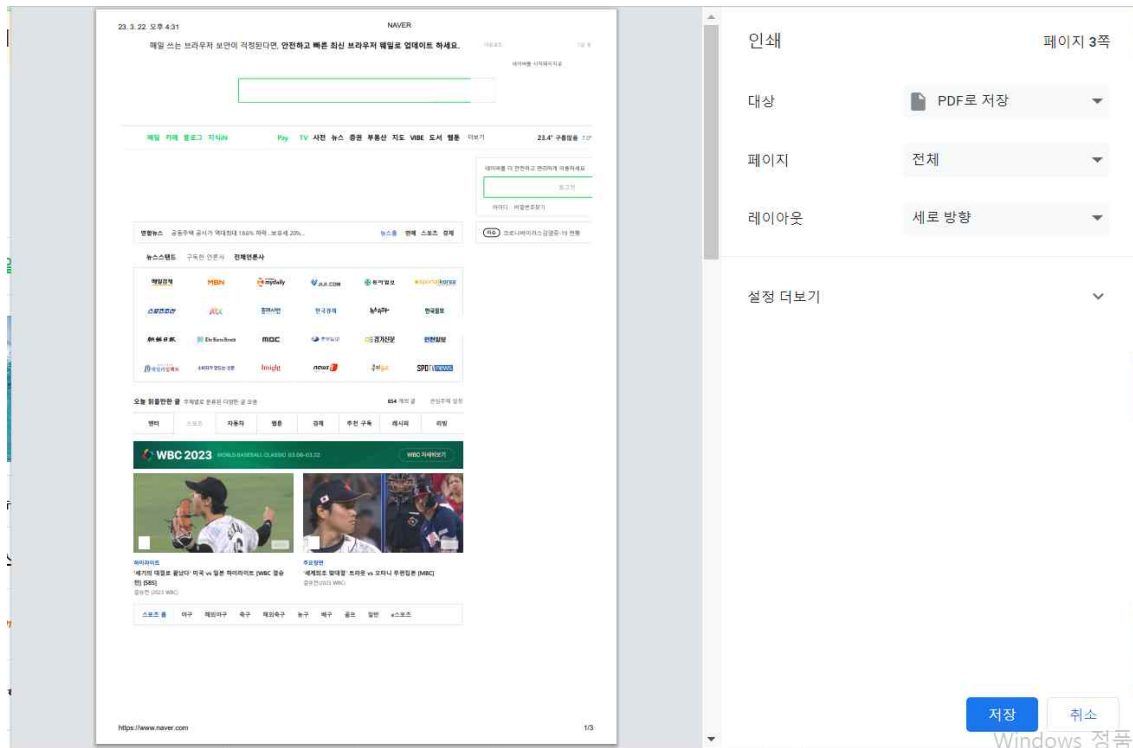
In the browser window some HTML APIs like `requestFullscreen` can only be invoked by a gesture from the user. Setting `userGesture` to `true` will remove this limitation.

Code execution will be suspended until web page stop loading.

```
contents.executeJavaScript('fetch("https://jsonplaceholder.typicode.com/users/1")'.
  .then((result) => {
    console.log(result) // Will be the JSON object from the fetch call
  })
```

## 2. 기존 상황

기존의 PDF나 web page를 pdf로 변환하려면 아래와 같이 페이지를 뷰어로 열람 후 직접 프린트를 했어야 합니다.

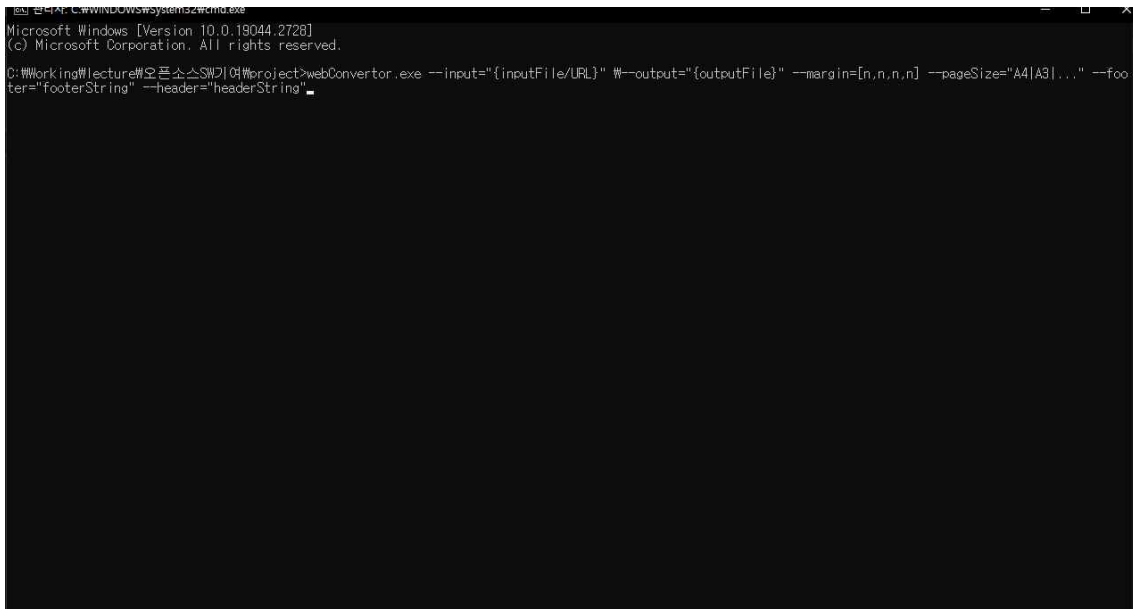


이는 서버단에서 직접 진행하기 어려운 상황이며, 이를 오토매틱하게 제공하는 콘솔 프로그램을 개발하고자 합니다.

또한, url이나 파일을 pdf로 변환해 주는 콘솔 어플리케이션을 제공하는 오픈소스 프로젝트를 확인할 수 없었습니다.



### 3. 예상 결과



```
명령 프롬프트: C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19044.2728]
(c) Microsoft Corporation. All rights reserved.

C:\Working\lecture\오픈소스\기여\project>webConverter.exe --input="{inputFile/URL}" --output="{outputFile}" --margin=[n,n,n,n] --pageSize="A4|A3|..." --footer="{footerString}" --header="{headerString}"
```

위의 콘솔과 같이 input / output을 설정한 후, margin과 pageSize등의 옵션을 받아 커스텀도 함께 진행 가능한 콘솔 어플리케이션을 개발할 예정입니다.

이후, 특정 web page는 cookie나 특정 데이터 없이 접속하기가 어려운 점을 감안하여 cookie값도 함께 세팅할 수 있도록 추가할 예정입니다.