

## Client-side Dynamic Web Service

제출일	2022.11.23	전공	컴퓨터공학과
과목	컴퓨터네트워크(02)	학번	201602037
담당교수	이영석 교수님	이름	이규정

## ① 저번 과제에서 수정한 부분

### 1. Nginx.conf

```
server {  
    listen 8080;  
  
    location /pastebin/api/ {  
        proxy_pass http://host.docker.internal:8889/;  
        proxy_set_header X-Real-IP $remote_addr;  
        proxy_set_header Host $host;  
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
    }  
  
    location /pastebin/ {  
        proxy_pass http://host.docker.internal:8890/pastebin/;  
        proxy_set_header X-Real-IP $remote_addr;  
        proxy_set_header Host $host;  
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
    }  
}
```

Nginx.conf에서 변경한 부분입니다. 과제 요구사항에 맞춰 8890 포트와 연결시켜줍니다.

### 2. App.py

```
app = Flask(__name__)  
  
bp = Blueprint('mybp', __name__,  
               static_folder='static',  
               static_url_path='/pastebin/static',  
               template_folder='templates',  
               url_prefix='/pastebin')  
  
@bp.route(f'/', methods=['GET'])  
@bp.route('/index.html', methods=['GET'])  
  
  
app.register_blueprint(bp)  
  
if __name__ == "__main__":  
    app.run(host="localhost", port="8890", debug=True)
```

지난주 과제에서 bp가 제대로 실행되지 않았는데 실습 수업시간에 말해주신 register\_blueprint를 사용하니 해결되었습니다. App -> bp로 변화를 주었으므로 나머지 html의 인자로 받는 부분도 모두 mybp로 설정해주었습니다.

## ② Client-side Dynamic Web Service

## 과제 구현 조건에서

1) 같은 글이 계속 반복되지 않게 출력

2) 표시할 글을 최대 10개만 출력

부분만 성공적으로 구현하였고, 내림차순으로 상단에 출력하는 부분은 제대로 구현하지 못하였습니다. ##

### 1. Myscript.js

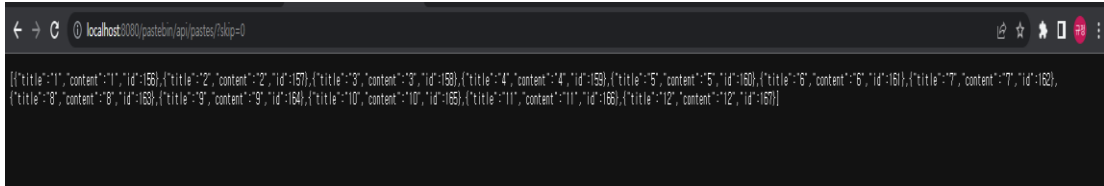
```
var skip_index = 0;
var remove_index = 0;
async function fetchRequestWithError() {
  try {
    skip_index = skip_index;
    const url = 'http://localhost:8080/pastebin/api/pastes/?skip='+skip_index;
    const response = await fetch(url);
    var key_arr = new Array();
    if (response.status >= 200 && response.status < 400) {
      const data = await response.json();
      for (var key in data){
        ndiv = document.createElement('div');
        ndiv.innerHTML = `<h2> ${data[key]['title']}</h2><p> ${data[key]['content']}</p><hr>`;
        pdiv = document.getElementById('newsfeed');
        pdiv.appendChild(ndiv);
        skip_index += 1;
        remove_index += 1;
        while(remove_index > 10){
          pdiv.removeChild(pdiv.firstChild);
          remove_index--;
        }
      }
    }
  } catch (error) {
    console.log(error);
  }
}

fetchint = setInterval(fetchRequestWithError, 10 * 100);
```

우선 URL을 단순하게 pastes들을 보여주는 URL로 접근을 하고 data를 돌면서 출력을 시키면 이미 출력된 paste들이 계속해서 아래에 추가되는 것을 확인했습니다.

이를 막기위해서 실습 6~7주차에 구현했던 RESTful API, pastebin/api/docs에 접속하여 get pastes을 실행하고 URL을 살펴보니 localhost:8080/Pastebin/api/pastes 뒤에 skip과 limit이라는 변수들로 접근할 수 있었습니다.

Skip=0으로 접속하면 모든 paste들이 나오고 skip의 값을 늘려갈수록 앞의 paste부터 나타나지 않는 것을 보았습니다. 이를 이용하여 새로운 글만 업데이트 하도록 구현했습니다.



```
[[{"title": "1", "content": "1", "id": 156}, {"title": "2", "content": "2", "id": 157}, {"title": "3", "content": "3", "id": 158}, {"title": "4", "content": "4", "id": 159}, {"title": "5", "content": "5", "id": 160}, {"title": "6", "content": "6", "id": 161}, {"title": "7", "content": "7", "id": 162}, {"title": "8", "content": "8", "id": 163}, {"title": "9", "content": "9", "id": 164}, {"title": "10", "content": "10", "id": 165}, {"title": "11", "content": "11", "id": 166}, {"title": "12", "content": "12", "id": 167}]
```

위 사진은 1~12까지의 번호를 붙인 paste를 skip=0의 URL로 접속한 결과입니다. 모든 paste들이 나오는 것을 확인할 수 있습니다.

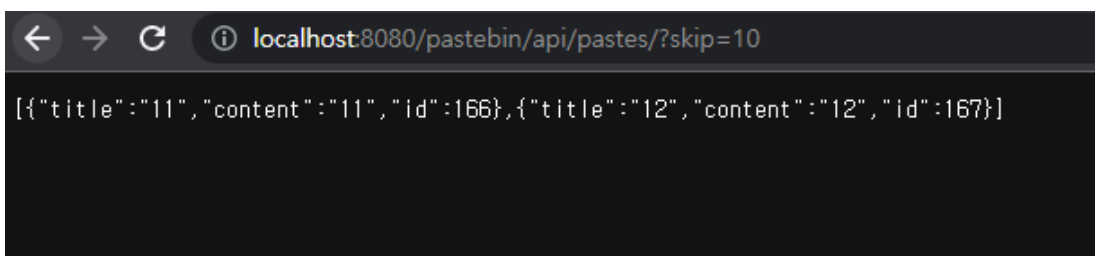
올바른 skip에 접근할 수 있도록 skip\_index라는 변수와 3번 조건인 최대 표시 개수제한을 위한 remove\_index를 0으로 초기화 시켜줍니다.

처음에 url에 접근할 때는 모든 paste들을 받아와야 하므로 skip=0으로 설정하고 req를 받아옵니다. 받아온 req(=response)가 정상적인 경우, 즉 응답이 200~400 사이에 있을 경우 받아온 response.json을 data 변수에 저장합니다.

반복문의 key값 (0~data.length)로 paste에 접근하게 되고 title과 content를 출력합니다.

하나의 paste를 출력할때마다 skip\_index를 1씩 증가시켜주어서 같은 paste를 다시한번 접근하는 일이 없도록 합니다. Remove\_index도 같이 증가시켜주는데, remove\_index가 10 이상의 값이 되면 removeChild()를 사용하여 frist노드를 삭제시키고, remove\_index를 1 감소시킵니다. 이 과정을 통해서 출력되는 paste의 개수를 제한할 수 있습니다.

Skip=0인 url의 과정이 끝나면 설정해준 타이머 후에 다시한번 위 함수를 실행시킵니다. 그때는 paste개수만큼 skip의 값이 설정 되어있어서, 이미 출력한 paste들에 대해서 읽어 오지 못하는 url로 데이터를 받아올 수 있습니다.



```
[{"title": "11", "content": "11", "id": 166}, {"title": "12", "content": "12", "id": 167}]
```

Skip=10으로 설정 시, 그 다음 paste부터 데이터가 나오는 사진입니다.

## 2. 구현하지 못한 부분(내림차순 정렬)

```
//      for (var key in data){ key_arr[key] = key; }  
//      for (var i = key_arr.length - 1; i >= 0 ; i--) {  
//          ndiv = document.createElement('div');  
//          ndiv.innerHTML = `<h2> ${data[i]['title']} </h2><p> ${data[i]['content']}</p><hr>`;  
//          pdiv = document.getElementById('newsfeed');  
//          pdiv.appendChild(ndiv);  
//          skip_index += 1;  
//          remove_index += 1;  
//          while(remove_index > 10){  
//              pdiv.removeChild(pdiv.lastChild);  
//              remove_index--;  
//          }  
//      }  
//  }
```

위와 같이 코드를 작성하면 paste를 입력 후, 새로고침 버튼을 누르면 내림차순으로 올바르게 출력이 됩니다. 하지만 paste 개수가 10개가 넘어가면 실시간으로 출력되는 것이 확인이 안됩니다. 이는 페이지의 윗부분에 출력되는 것이 아니라 아래부분에 계속해서 출력되기 때문인 것 같습니다.

skip으로 받아온 1개의 paste를 위쪽에 삽입하는 방법을 찾기 못해서 해결하지 못했습니다.

아래의 사진은 위 코드로 했을 때 새로 고침 한 결과입니다.



### ③ 실행 결과

영상으로 첨부하겠습니다.