AJAX

Pirogramming 14 최지민

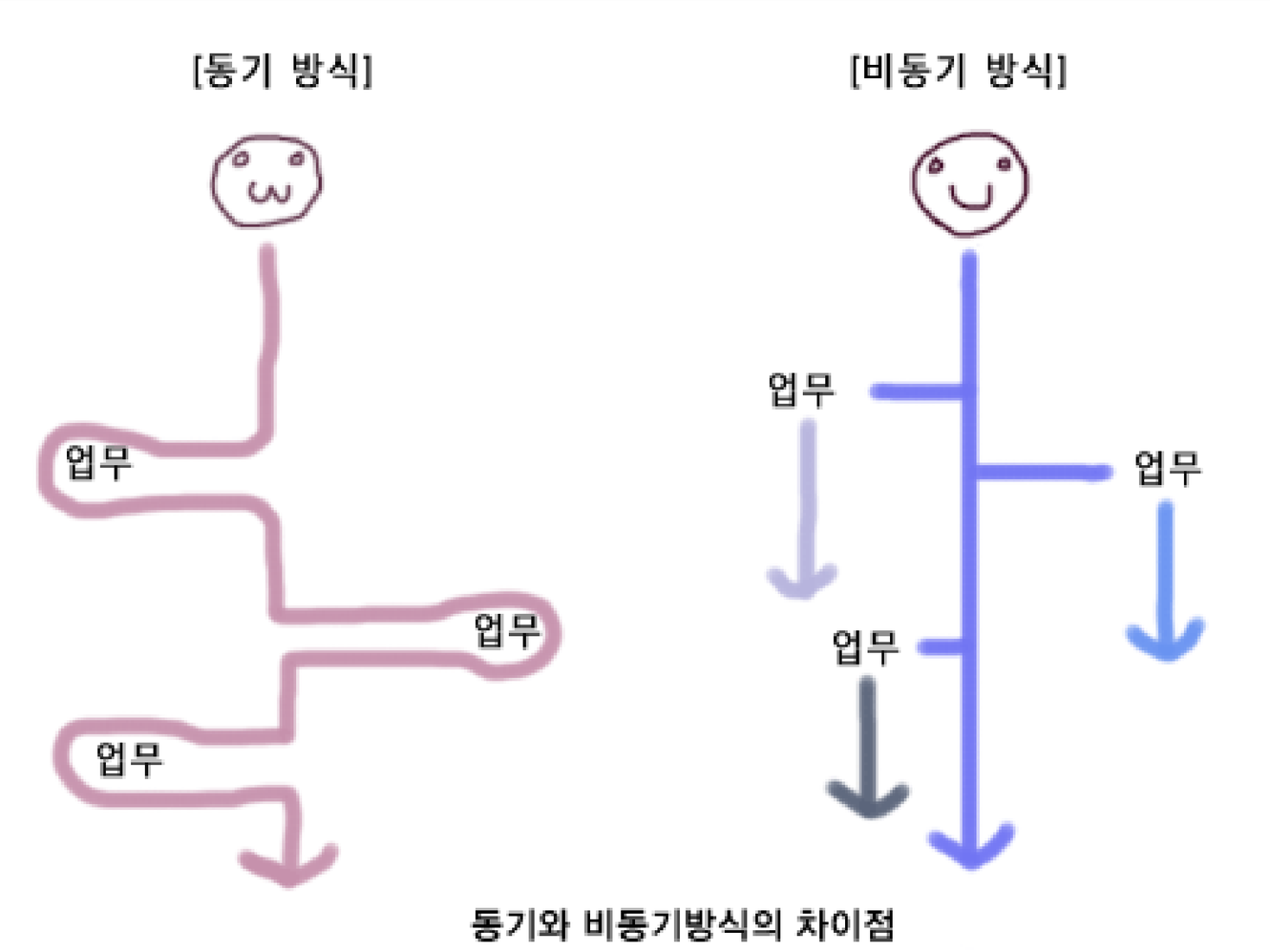
STEPS

☑ AJAX 이론	**AJAX가 무엇인지, 왜 사용하는지 대략적인 이해**
	AJAX 기법의 진행 과정 이해
☑ AJAX 실습	**AJAX 기법을 실제로 실습**

동기 VS 비동기

☑ 클라이언트	서버에서 정보를 가져와서 사용자에게 보여줄 수 있고 사용자와 상호작용할 수 있는 소프트웨어
☑ 서버	클라이언트의 요청에 따라 알맞은 데이터를 보내주는 프로그램
☑ request	클라이언트가 서버로 전달해서 서버에 액션이 일어나게끔 하는 메시지
☑ response	Request에 대한 서버의 응답 메시지

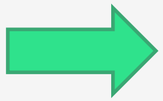
동기 VS 비동기



동기와 비동기방식의 차이점

동기 vs 비동기

Request!



동기 방식

서버에서 요청을 처리하고 응답
이때 페이지가 리로드되면서
전체 리소스를 다시 불러와야 함

비동기 방식

서버에서 요청을 처리하고 응답
이때 페이지의 일부분만 갱신되면서
자원과 시간을 아낄 수 있음

AJAX?

Asynchronous Javascript And Xml (비동기식 자바스크립트와 Xml) 의 약자

브라우저가 가지고 있는 XMLHttpRequest 객체를 이용해서 **전체 페이지를 새로 고치지 않고도, 페이지의 일부만을 데이터 로드**하는 기법

Javascript를 사용하여 클라이언트와 서버 사이에서

XML (JSON) 데이터를 주고 받는 비동기 통신 기술

AJAX?

☑ XML	**데이터를 저장하고 전달한 목적으로 만들어진 마크업 언어**
☑ JSON	**‘키-값’ 쌍 형식의 배열 형태를 이루는 데이터 형식**
☑ XMLHttpRequest	**브라우저가 가지고 있는 객체 중 하나**

AJAX?

XML

```
<dog>
    <name>식빵</name>
    <family>웰시코기<family>
    <age>1</age>
    <weight>2.14</weight>
</dog>
```

JSON

```
{
    "name": "식빵",
    "family": "웰시코기",
    "age": 1,
    "weight": 2.14
}
```

AJAX의 장점

- ☑ Request를 보냈을 때, 서버의 처리가 완료될 때까지 기다리지 않고 처리가 가능
- ☑ 자원과 시간을 아낄 수 있음
- ☑ 서버에서 필요한 데이터만 전송하면 되므로 전체적인 코드의 양 감소

AJAX의 단점

- ☑ 히스토리 관리가 되지 않음
- ☑ 연속으로 데이터를 요청하면 서버 부하가 증가
- ☑ 사용자에게 아무런 진행 정보가 주어지지 않음

AJAX는 언제 사용해요?

페이지 전체를 새로 고침하지 않고

정보를 불러오거나 일부분만 바꾸고 싶을 때!

AJAX 진행 과정

☑ XMLHttpRequest 객체를 생성한다.

```
const httpRequest = new XMLHttpRequest();
```

☑ 서버에 요청을 보낸다.

```
httpRequest.open('GET', 'http://www.example.org/some.file', true);
httpRequest.send(null);
```

AJAX 진행 과정

☑ 서버에서 요청에 대한 처리를 한다.

```python
@csrf_exempt
def like_ajax(request):
    req = json.loads(request.body)
    # request에 대한 처리
```

AJAX 진행 과정

서버의 response를 처리하는 함수를
httpRequest의 onreadystatechange 이벤트 리스너에 붙여준다.

```
function onComplete () {
    if(httpRequest.readyState === XMLHttpRequest.DONE) {
        // 응답 받기 완료된 상태
    }
    else {
        // 응답 받는 중
    }
}

httpRequest.onreadystatechange = onComplete;
```

AJAX 진행 과정

Readystate

0 (uninitialized)	**Request가 초기화되지 않음**
1 (loading)	**서버와의 연결이 성사됨**
2 (loaded)	**서버가 request를 받음**
3 (interactive)	**Request를 처리하는 중**
4 (complete)	**처리가 끝났으며 응답할 준비가 완료됨 (DONE)**

AJAX 진행 과정

☑ 상태값을 확인하고 응답을 처리한다.

```
if(httpRequest.status >= 400) {
    // 일반적으로 에러가 있는 상황
}
else {
    // 일반적으로 에러가 없는 상황
    const data = httpRequest.response;
    // ...
}
```

AJAX 실습

실습을 해봅시다!

Fetch API

☑ API 특정 기능이 어떻게 동작하는지 모르더라도, 요청에 따라 정의된 결과값을 보내주는 서버

Fetch는 Web API 중 하나로,

비동기 네트워크 통신을 보다 직관적으로 사용할 수 있게 도와준다.

```
fetch('http://example.com/movies.json')
    .then(function(response) {
        return response.json();
    })
    .then(function(myJson) {
        console.log(JSON.stringify(myJson));
    });
```

Fetch API

다만, 지원하지 않는 브라우저가 있으니 유의해야 함

(https://caniuse.com/?search=fetch)

Axios

Node.js와 브라우저를 위한 HTTP 통신 라이브러리.

비동기로 HTTP 통신을 가능하게 해주며 사용 방법은 fetch 만큼이나 쉽다.

```
const axios = require('axios');
axios.get('/user', {params: {ID: 1234}})
    .then(function(response) {
        console.log(response);
    })
    .catch(function(error) {
        console.log(error);
    })
    .then(function() {
        // always executed
    });
```

Axios

다만, 모듈 설치를 해주어야 한다.

(웹의 경우 CDN을 사용할 수 있다.)

```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
```

Fetch, Axios 실습

Fetch와 Axios를 사용해서

실습을 해봅시다!


Q&A