React – Django 세션기반

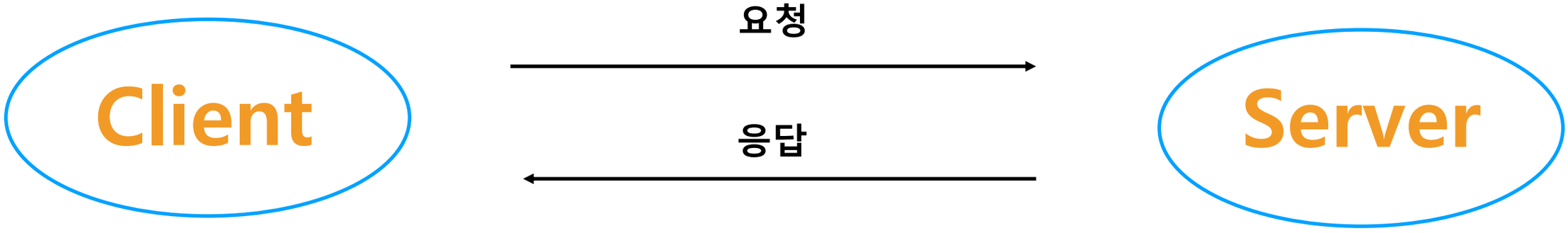
Login

정지원

중앙대 멋사 10기

세션 기반 인증 vs 토큰 기반 인증
쿠키? 세션? 토큰?
CORS
LocalStorage vs sessionStorage
리액트-장고 로그인

React(3000) – Django(8000)



1.

세션 기반 인증 vs 토큰 기반 인증

HTTP stateless.

서버로 가는 요청이 이전 요청과 독립적으로 다뤄진다

2.

쿠키 & 세션

쿠키(cookie)

사용자의 컴퓨터에 저장하는 작은 기록 정보 파일

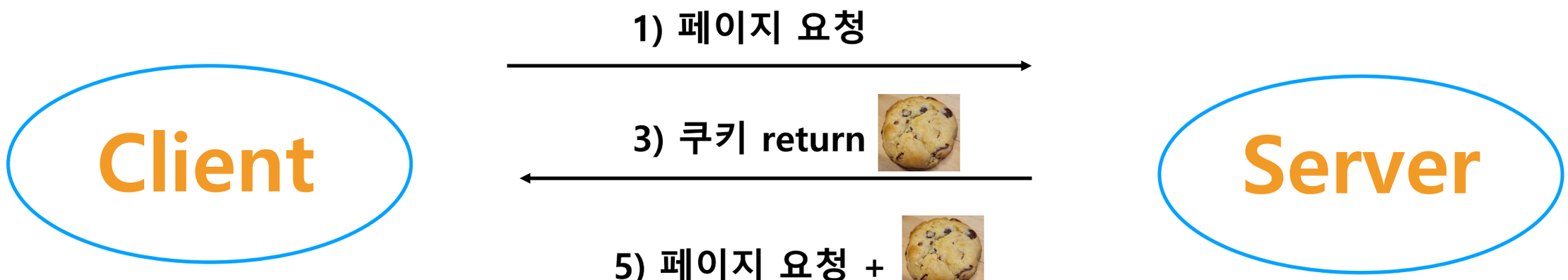
Client

Server

데이터를 클라이언트 측에 저장
자동으로 생성됨
데이터 크기에 제한o
이름, 값, 만료일, 경로 정보로 구성

쿠키(cookie) 동작 순서

사용자의 컴퓨터에 저장하는 작은 기록 정보 파일



1. 클라이언트가 페이지 요청
2. 웹 서버는 쿠키 생성
3. 생성한 쿠키에 정보를 담아 클라이언트에게 return
4. 넘겨 받은 쿠키는 클라이언트가 가지고 있다가(로컬pc에 정보 저장), 서버에 요청과 함께 쿠키 전송
5. 동일 사이트 재방문 시, 해당 쿠키가 있는 경우, 요청페이지와 함께 쿠키 전송

정보 담는 보자기

쿠키(cookie)

! 보안 상 이슈

- 쿠키 안에 담긴 데이터는 브라우저에 저장되어 있어 쉽게 노출 -> 개인정보 유출에 취약
- 서버 성능이 증가함 -> 데이터는 서버에 저장하는 방식으로 발전

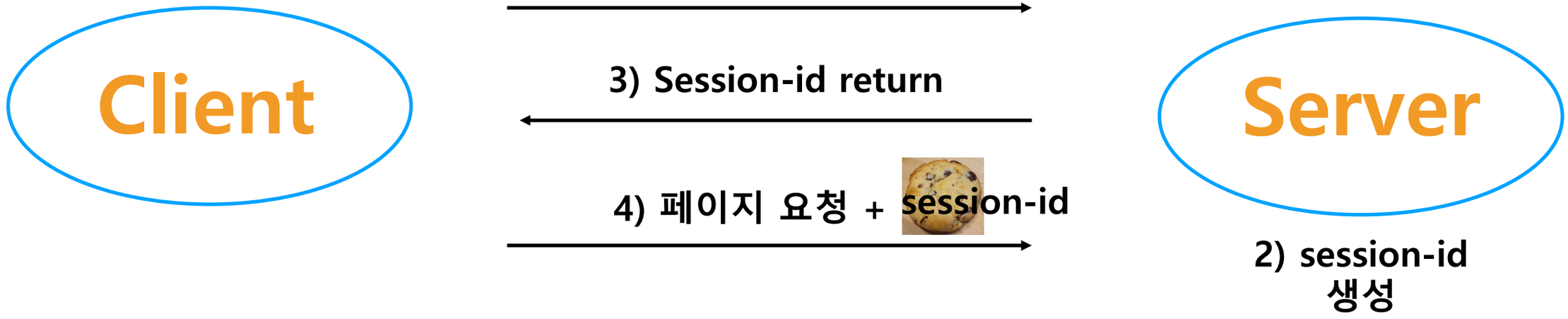
세션(session)

일정 시간 동안 브라우저로 들어오는 요구를 유지시키는 기술

데이터를 서버 측에 저장
일정 시간 : 웹 서버 접속 시점 ~ 웹 브라우저 종료
데이터 크기에 제한 x -> 서버 성능이 가능한 한
브라우저 닫거나, 서버에서 세션 삭제했을 때만 삭제
-> 쿠키보다 비교적 보안이 좋음
각 클라이언트 고유 session Id 부여
-> session Id로 클라이언트 구분

세션(session) 동작 순서

일정 시간 동안 브라우저로 들어오는 요구를 일정 시간 유지시키는 기술



1. 클라이언트가 페이지 요청
2. 클라이언트 쿠키에 session-id가 있는지 확인
3. Sessin-id가 존재하지 않으면, 서버가 생성해서 클라이언트에게 돌려줌
4. 서버에서 준 session-id를 쿠키 이용해 서버에 저장
5. 클라이언트 재접속 시, 쿠키 이용해 session-id 값을 서버에 전달

명찰

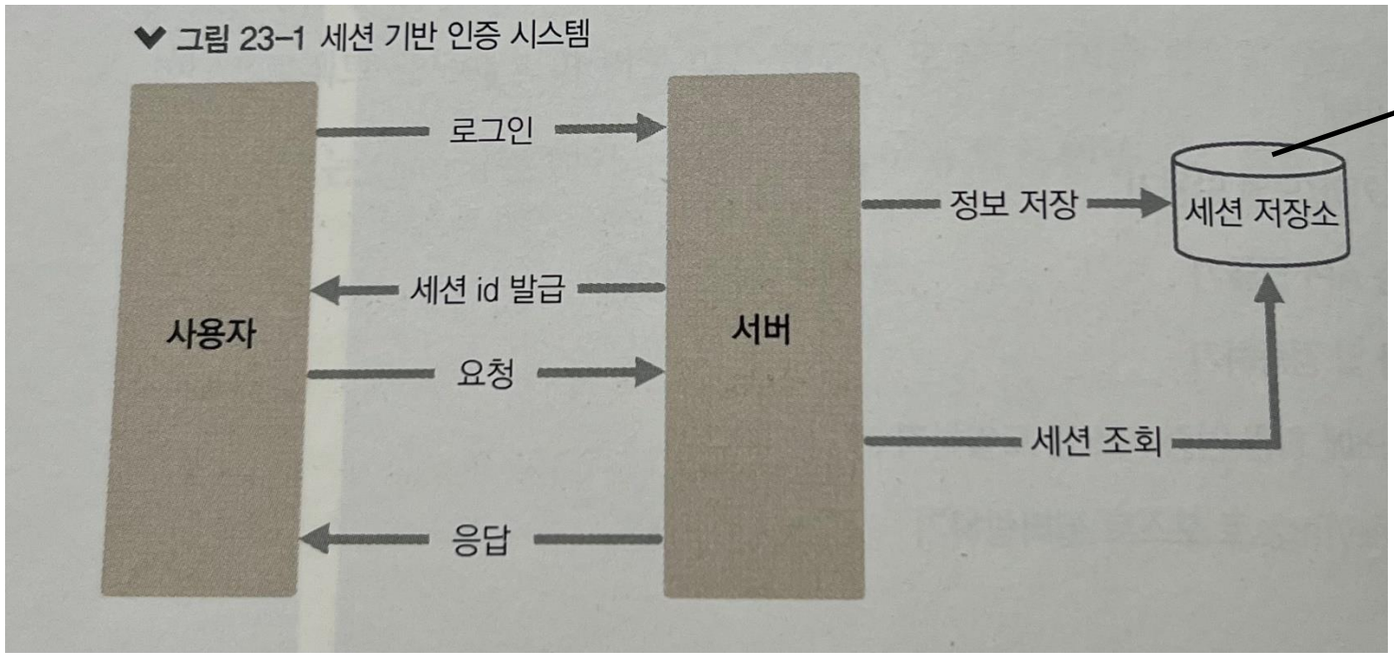
그럼 쿠키는 언제 쓰는데유?

1. Session-id 값을 쿠키에 저장해서 사용
2. 구글, 유투브에서 사용자가 접근한 콘텐츠 기록을 쿠키로 저장 -> 검색 최적화, 추천 알고리즘

	쿠키(Cookie)	세션(Session)
저장 위치	클라이언트(=접속자 PC)	웹 서버
저장 형식	text	Object
만료 시점	쿠키 저장시 설정 (브라우저가 종료되도, 만료시점이 지나지 않으면 자동삭제되지 않음)	브라우저 종료시 삭제 (기간 지정 가능)
사용하는 자원(리소스)	클라이언트 리소스	웹 서버 리소스
용량 제한	총 300개 하나의 도메인 당 20개 하나의 쿠키 당 4KB(=4096byte)	서버가 허용하는 한 용량제한 없음.
속도	세션보다 빠름	쿠키보다 느림
보안	세션보다 안좋음	쿠키보다 좋음

그래서 세션 기반 인증이 뭔데?

서버가 사용자가 로그인 중임을 기억하고 있는 로직!



메모리,
디스크, DB

1. 로그인 하면, 서버는 세션 저장소에 사용자 정보를 조회, 없으면 세션id 발급
2. 발급된 id는 브라우저 쿠키에 저장
3. 클라이언트가 다른 요청 보낼 때마다 서버는 세션저장소에서 세션id로 조회 후 로그인 여부 결정

아!
사용자 정보 확인할 수 있는 key인
session-id는 쿠키타고(저장) 다니는구나!

3.

토큰 & 토큰 기반 인증

토큰(token)

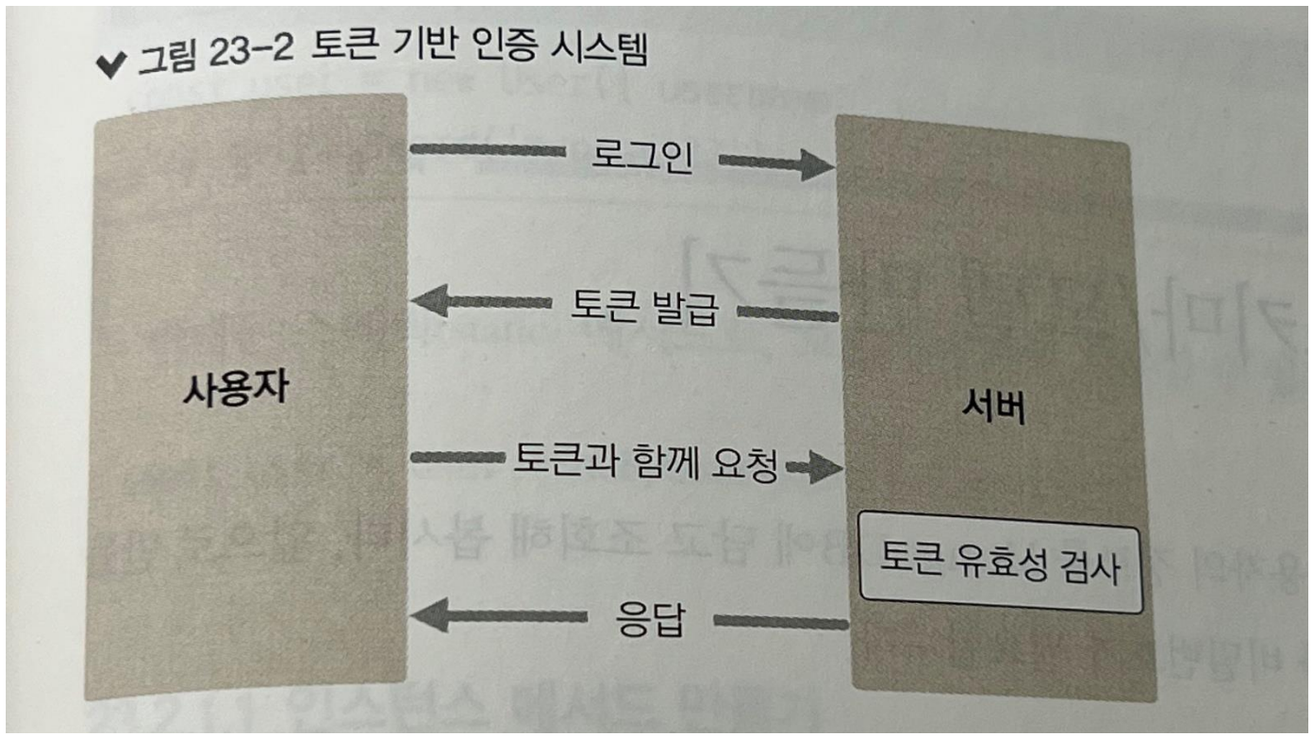
서버가 만들어주는 요상한 문자열

문자열 안에는?

- **사용자의 로그인 정보**
- **해당 정보가 서버에서 발급되었음을 증명하는 서명**

토큰 기반 인증 동작 순서

토큰으로 사용자가 맞다는 걸 인증해주는 로직!



1. 로그인 하면, 서버에서 토큰을 발급
2. 클라이언트가 다른 요청할 때, 발급받은 토큰과 함께 요청
3. 서버는 해당 토큰이 유효한지 검사하고 응답

토큰 기반 인증

토큰으로 사용자가 맞다는 걸 인증해주는 로직!

세션 기반 인증처럼 DB를 확인할 필요 없이 유저 정보 인증 가능!

JWT

JSON Web Token

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKx
wRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

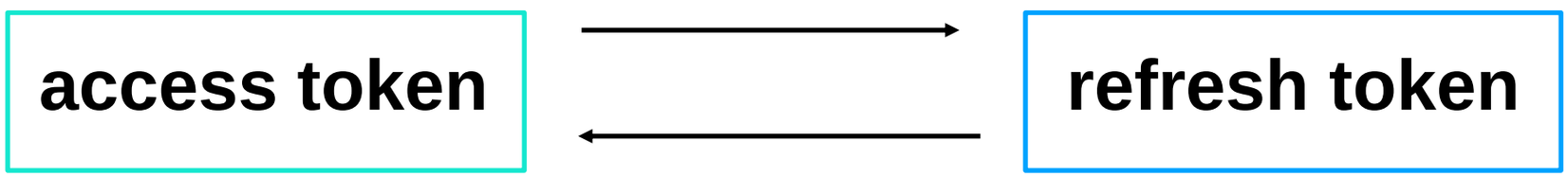
```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

토큰 보안 이슈

할,,,사실 토큰도 1개만 쓰면 보안 문제있지롱...



=> 서로의 유효성을 검증해줍니다

세션 기반 인증

세션 id(명찰)로 DB에 있는 유저 정보 확인!

DB 필요해! 소중해!

토큰 기반 인증

토큰에 있는 유저 고유 정보로 유저 확인!

DB 필요 없어! 토큰 소중해!

4.

CORS

CORS 에러

Cross-origin resource sharing

cross-origin

`cross-origin`이란 다음 중 한 가지라도 다른 경우를 말합니다.

1. 프로토콜 - http와 https는 프로토콜이 다르다.
2. 도메인 - domain.com과 other-domain.com은 다르다.
3. 포트 번호 - 8080포트와 3000포트는 다르다.

장고(8000) – 리액트(3000)

보안에 취약!
Cross-origin하려면 서버와 클라이언트 모두
CORS에러에 대한 처리를 해주어야 함

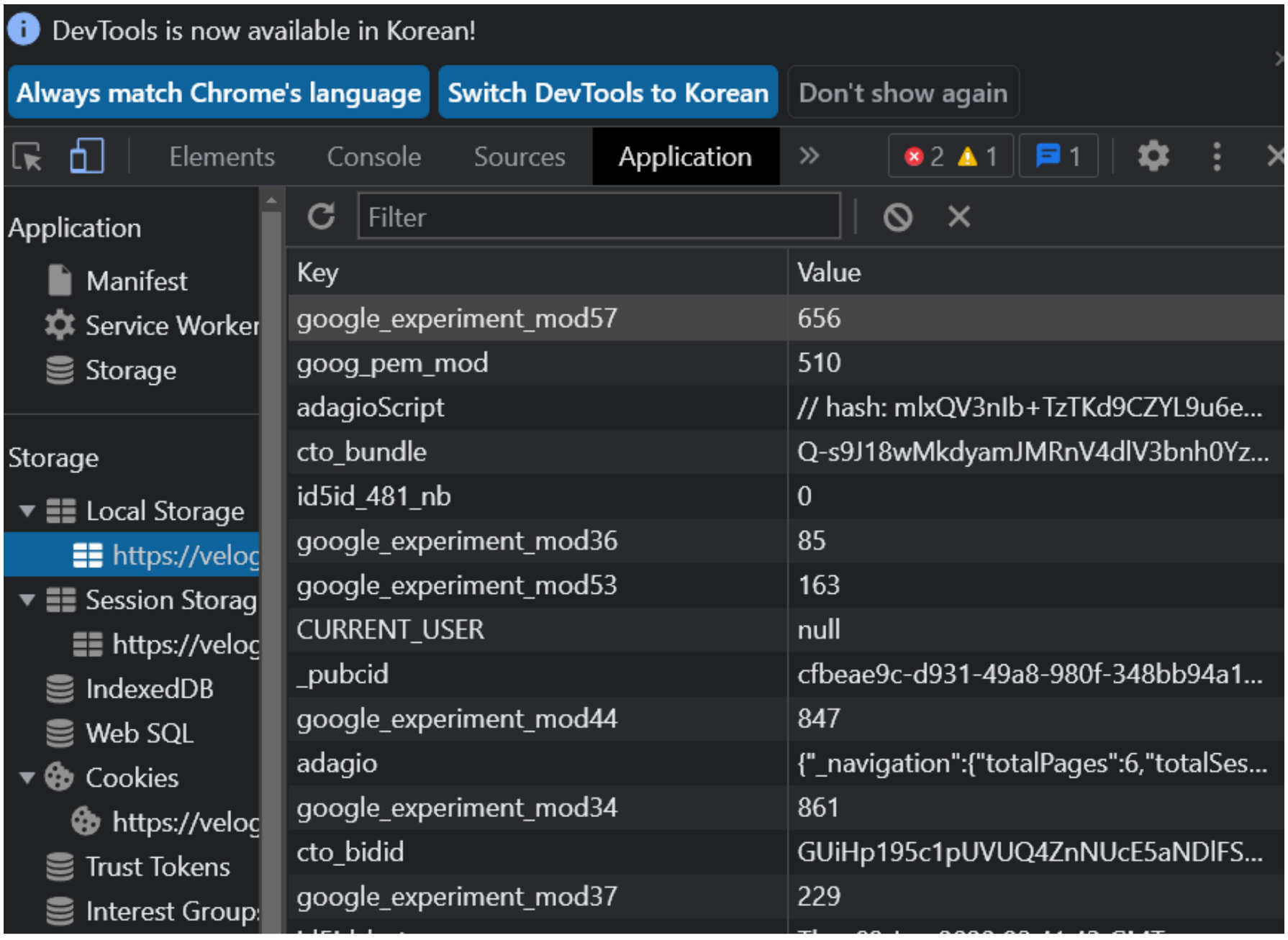
5.

localStorage vs sessionStorage

웹 스토리지

1. key-value 형태
2. 클라이언트에 대한 정보 저장
3. 웹스토리지는 브라우저에만 정보저장
쿠키는 서버, 로컬에 정보 저장
4. 서버에 불필요하게 정보 저장 x

개발자 도구 > Application



데이터가 어떤 범위 내에서 얼마나 오래 보존되느냐

로컬스토리지	세션스토리지
세션이 끝나도 데이터 지워지지 않음 Ex) 자동로그인 저장	세션이 끝날 때, 창이 닫힐 때 데이터가 지워짐 Ex) 은행페이지 입력 폼 정보 비로그인 장바구니

값 가져오는 방법

1. 로컬 스토리지

- localStorage.A **(Key == A)**
- localStorage.getItem("A")

2. 세션 스토리지

- sessionStorage.A **(Key == A)**
- sessionStorage.getItem("A")

3. 쿠키

- getCookie("A") **(Key == A)**

값 세팅하는 방법

1. 로컬 스토리지

- localStorage.A = 1 **(Key == A, Value = 1)**
- localStorage.setItem("A", 1)

2. 세션 스토리지

- sessionStorage.A = 1 **(Key == A, Value = 1)**
- sessionStorage.setItem("A", 1)

3. 쿠키

- setCookie("A", 1, 7) **(Key == A, Value == 1, 유효기간 == 7초)**

6.

리액트-장고 로그인 실습

실습 로직

1. Createsuperuser로 계정을 만들어놓고 (session id 발급)
2. Session id가 있으니 로그인 성공
3. 만들어놓지 않은 계정으로 로그인하면 403 에러

장고에서는 쿠키 안에 유저정보랑 session id 다 담아서 준다

Django 서버 open

1. Django 서버 clone받기

2. 장고 가상환경 venv만들기
Window | python –m venv 이름
Mac | python3 –m venv 이름

3. 가상환경 활성화
Window | source 이름/Scripts/activate
Mac | source 이름/bin/activate

4. 패키지 설치
Window | pip install –r requirements.txt
Mac | pip3 install –r requirements.txt

5.
Window | python manage.py migrate
Mac | python3 manage.py migrate

6. 계정 만들기
python manage.py createsuperuser

7. 서버 돌리기
python manage.py runserver

useEffect	useCallback
값이 바뀌면 어떤 동작을 자동으로 실행하기 위해	원하는 타이밍에 호출시킬 함수를 만들기 위해

```jsx
const [ food, setFood ] = useState('샐러드');
const [ menuAsked, setMenuAsked ] = useState(false);

useEffect(() => {
  if (menuAsked === true){
    printMenu();
    setMenuAsked(false); // state를 초기상태로 돌려준다.
  }
}, [menuAsked])

const printMenu useCallback(() => {
    console.log(`[ Today's Dinner Menu : ${food} ]`);
}, [food]);
```

동작	food	menuAsked	printMenu 함수 변화	출력
최초 상태	'샐러드'	false	X	-
setFood('고추장 삼겹살')	'고추장 삼겹살'	false	O	-
setFood('갈치 구이')	'갈치 구이'	false	O	-
setMenuAsked(true)	'갈치 구이'	true	X	[Today's Dinner Menu : 갈치 구이]

fetch()
브라우저 내장 함수-콜백

Callback함수
다른 함수가 실행 끝낸 뒤 실행되는 걸 callback

그다음~
Error나면~

```
fetch(url, options)
  .then((response) => console.log("response:", response))
  .catch((error) => console.log("error:", error));
```

fetch()
브라우저 내장 함수-콜백

여기에 요청 보내는거

```
fetch("http://127.0.0.1:8000/auth/login", {
  method: "POST",
  mode: "cors",
  credentials: "include",
  headers: {
    // content-type : 해당 형태로
    "Content-Type": "application/json",
  },
  body: JSON.stringify(user),
})
```

Cors 에러처리

서버한테 json형태로 보낼거라고 알려줌

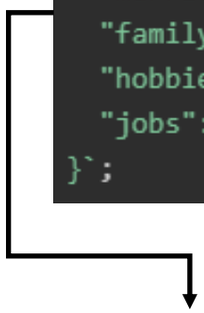
요청 내용 : user를 json string형태로

JSON.parse()

JSON 문자열을 JS 객체로 변환

```
const str = `{
  "name": "홍길동",
  "age": 25,
  "married": false,
  "family": { "father": "홍판서", "mother": "춘섬" },
  "hobbies": ["독서", "도술"],
  "jobs": null
}`;
```

```
const obj = JSON.parse(str);
```



```
{
    name: "홍길동",
    age: 25,
    married: false,
    family: {
        father: "홍판서",
        mother: "춘섬"
    },
    hobbies: [
        "독서",
        "도술"
    ],
    jobs: null
}
```

JSON.stringify()

JS 객체를 JSON 문자열로 변환

```
const obj = {
  name: "홍길동",
  age: 25,
  married: false,
  family: {
    father: "홍판서",
    mother: "춘섬",
  },
  hobbies: ["독서", "도술"],
  jobs: null,
};
```

```
const str = JSON.stringify(obj);
```



```
'{"name":"홍길동","age":25,"married":false,"family":{"father":"홍판서","mother":"춘섬"},"h
```

```js
const str2 = JSON.stringify(obj, null, 2);
console.log(str2);
```



```json
{
  "name": "홍길동",
  "age": 25,
  "married": false,
  "family": {
    "father": "홍판서",
    "mother": "춘섬"
  },
  "hobbies": [
    "독서",
    "도술"
  ],
  "jobs": null
}
```

.preventDefault()

기존의 동작이 진행되지 않고, 결과적으로 해당 이벤트가 발생하지 않는다.

1. a태그를 클릭해도 원하는 href링크로 이동x
2. Submit 태그를 클릭해도 창이 새로고침 안되게 하기

-> 우리는 submit 태그 사용
=> submit태그를 통한 데이터 전달은 정상적으로 작동, 페이지 새로고침은 막는다!!

CORS 에러 처리

Cross-origin resource sharing

django

```
CORS_ALLOW_CREDENTIALS = True
CORS_ORIGIN_ALLOW_ALL = True
```

react

```
mode: 'cors',
credentials: 'include',
```

과 제

8/9(화) 18:00 까지 세션 기반 인증 이론 + 실습 내용 Readme에 정리하기!

Thank You