

# 12차 강의

## 자바웹프로그래밍

강사 : 최도현



# 오늘의 할일 - 1

- 주 별 기술 트렌드 분석

자바스크립트 - 클래스(ES6)

**ES6**  
**ECMAScript 2015**

# 자바스크립트 - 클래스

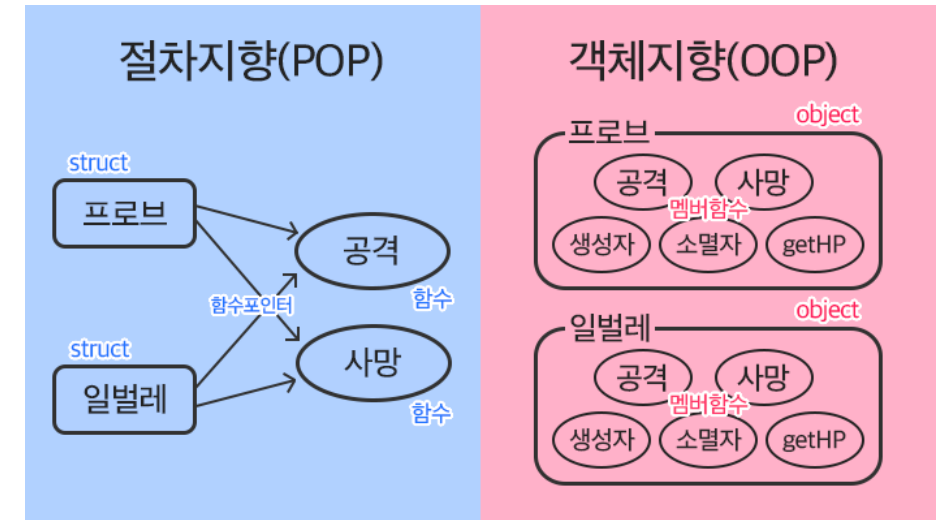
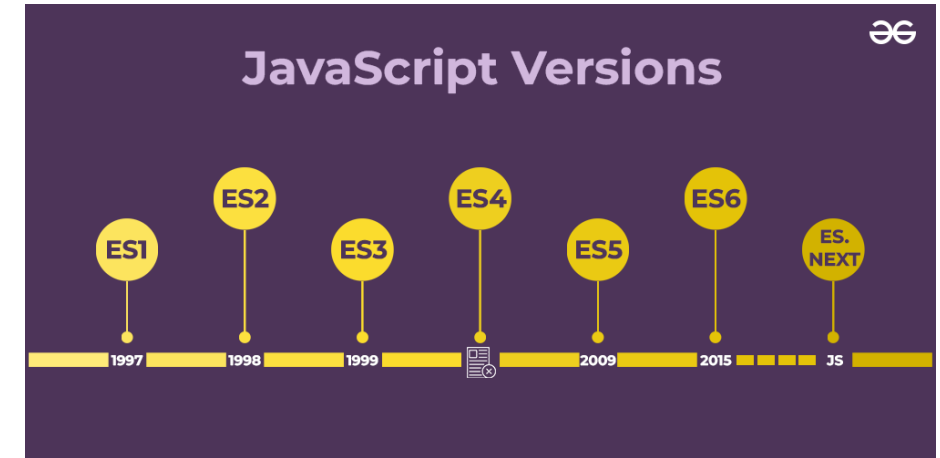
- **모던 자바스크립트**

- 함수 지향적 프로그래밍
  - JS 표준 ES5~ 이후 객체지향 지원
  - 이전 방식 : 함수지향적 구현

- ES6 이후 OOP 기법 지원, but 제한적
  - 캡슐화 및 은닉화 지원(#)
  - 생성자 함수, SET/GET 함수
  - 상속, 오버라이딩 등

- **활용의 우선순위?**

- 일반적 : 재사용 - 유지보수
- 코드 보안 - 캡슐화



# 자바스크립트 - 클래스

- OOP의 클래스란?

- 객체를 생성하기 위한 템플릿(청사진)

- ES6부터 class 키워드로 정의

- 코드 구조 및 활용

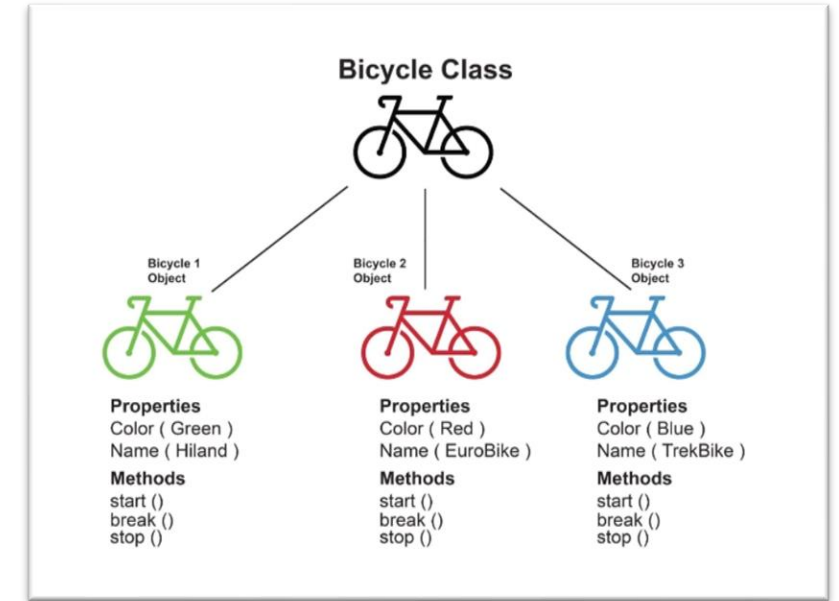
- 생성자 : constructor 지원
    - 객체 생성과 동시에 실행

- 속성(필드), 행동(메서드) 정의
    - SET/GET 함수 등

- New 연산자 - 객체 인스턴스 생성

- 오늘 실습에서

- 회원가입 기능을 클래스로 구현



```
1 class Car {  
2   constructor(color, weight, speed, engine) {  
3     this.color = color;  
4     this.weight = weight;  
5     this.speed = speed;  
6     this.engine = engine;  
7   }  
8 }  
9 const justCar = new Car();  
10 const kia = new Car("gray", 100, 0, "diesel");  
11  
12 console.log(justCar);  
13 console.log(kia);
```

# 오늘의 할일 - 2

- 클래스 활용하기 - 사용자(회원)

자바스크립트 - 모듈화

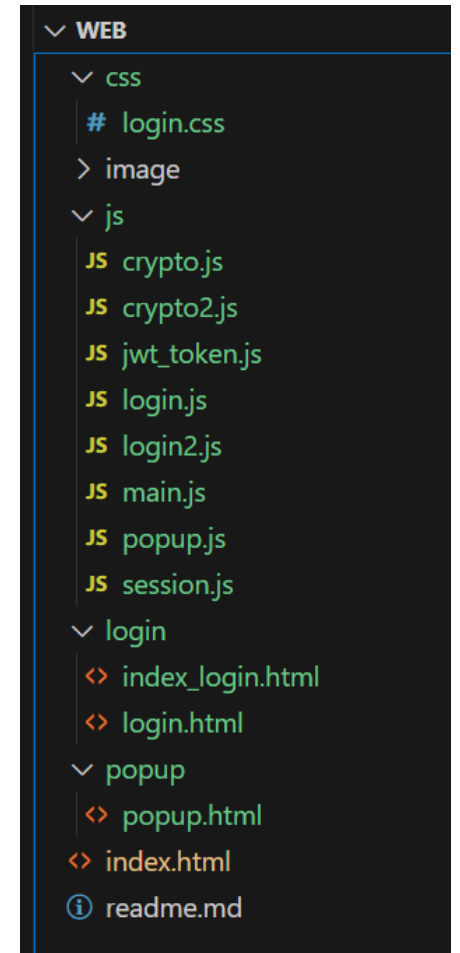
세션 - 객체 저장하기

웹 사이트 - 회원가입 기능



# 자바스크립트 - 모듈화

- 현재 내 프로젝트 폴더 구조는?
  - Login.html의 .js의 연동을 확인한다.
    - 기능이 추가될 때 마다 .js를 추가 연동했다.
  - 개발 및 관리적 문제점은?
    - 전역 변수 충돌 가능성, 파일 구조의 복잡성 증가
    - 가끔 코드 연동 순서 x
- 자바스크립트 Import/Export 모듈화를 구현한다.
  - 다수의 자바스크립트를 통합 관리할 수 있다.
    - 최신 웹 브라우저에서 대부분 지원
  - 특정 함수를 { }에서 지정하여 사용가능
- 자바스크립트를 모듈화 하자.



```

<script type="text/javascript" src="../../js/session.js" defer></script>
<script type="text/javascript" src="../../js/login.js" defer></script>
<script type="text/javascript" src="../../js/crypto.js" defer></script>
<script type="text/javascript" src="../../js/crypto2.js" defer></script>
<script type="text/javascript" src="../../js/jwt_token.js" defer></script>

```

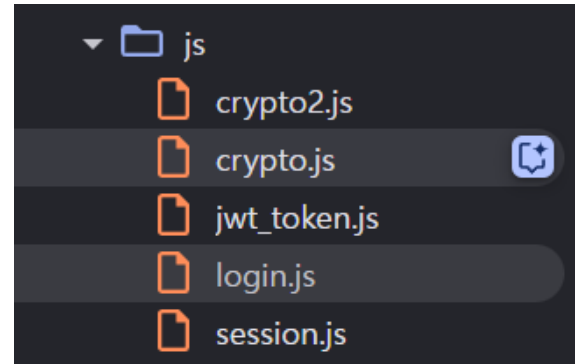
# 자바스크립트 - 모듈화

- 기존 자바스크립트를 주석처리 한다.
  - Ctrl + / 로 주석처리 후 Login.js 연동을 추가한다.
- `<script type="module" src="../js/login.js" defer></script>`
- **module** 기능을 활용하여 모든 파일을 로딩한다.
- Login.js의 맨 위 이외 자바스크립트를 연동한다.
  - Import 구문을 사용, { } 안에 사용하는 모든 함수를 명시

```
import { session_set, session_get, session_check } from './session.js';  
import { encrypt_text, decrypt_text } from './crypto.js';  
import { generateJWT, checkAuth } from './jwt_token.js';
```

- 새로운 기능이 추가되면 위처럼 연동해야 한다.
  - 기존 login.js의 소스코드는 거의 수정이 없다.
- 저장 후 f12 개발자 모드
  - 소스코드 로딩 확인, 에러?

## JS 파일 1개에서 로딩된 결과



쿠키를 요청합니다.

✖ Uncaught ReferenceError: init is not defined  
at onload (login.html:19:48)

# 자바스크립트 - 모듈화

- 다양한 에러가 발생한다. (모듈에서 onload x)
  - 가장 먼저 발생하는 init() 함수 호출에서 에러가 난다.

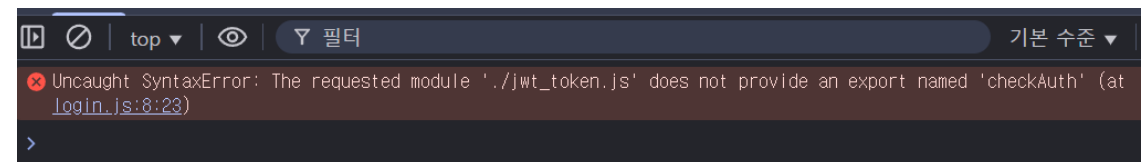
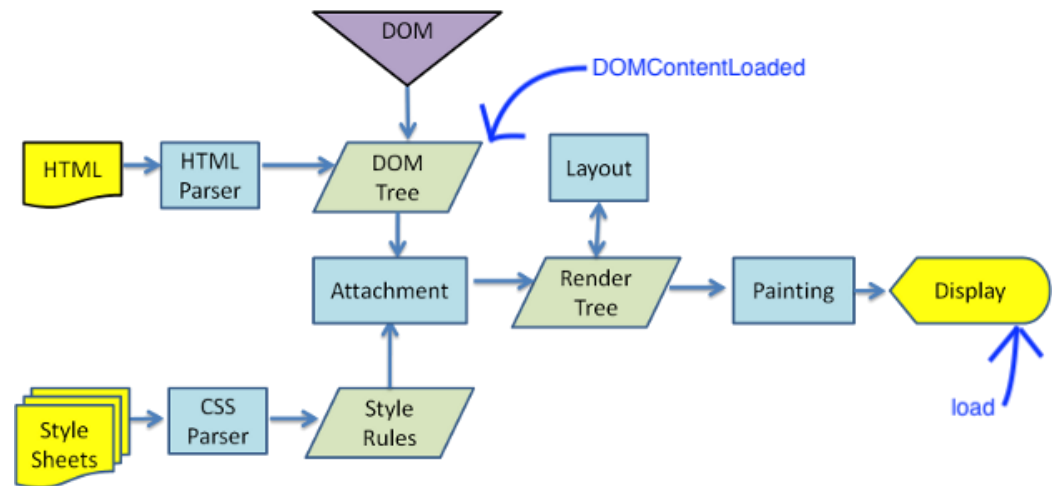
```
<body class="text-center" onload="init();">
```

- 기존 onload 수행방식을 삭제한다.
- Login.js 파일의 init() 함수 아래 추가한다.

```
document.addEventListener('DOMContentLoaded', () => {  
  init();  
});
```

- 저장 후 f12 개발자 모드
  - 소스코드 로딩 확인, 에러?

## 기존 onload보다 먼저 실행됨



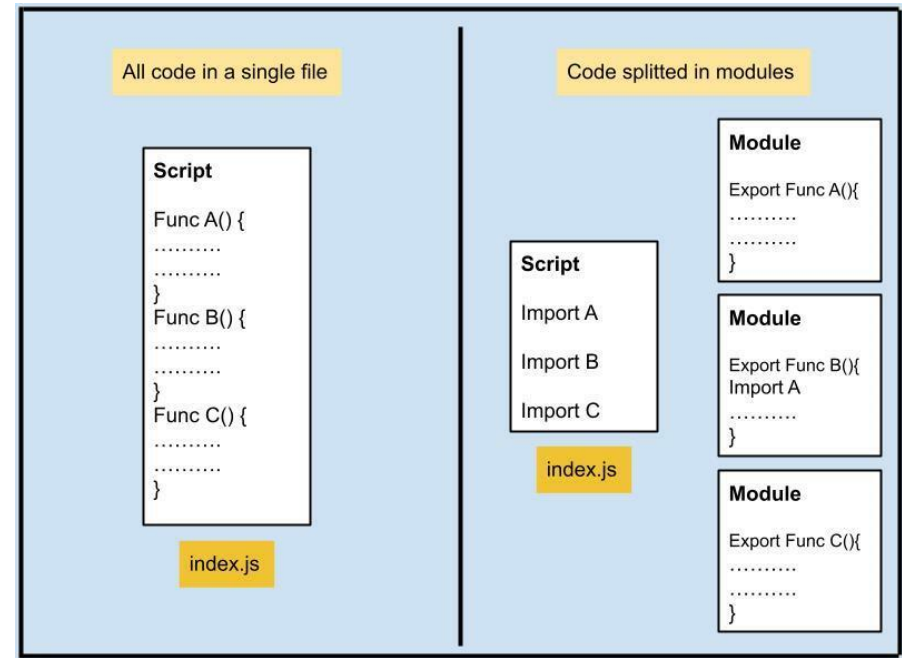


# 자바스크립트 - 모듈화

- login.js의 최상단 { }에 명시된 함수 확인
  - 호출 되는 함수는 모두 **export** 처리 필요
    - 세션, 암호화, 보안토큰까지 연결된 파일이 3개다.
- Js 폴더에 session.js를 열어보자.
  - 세션 set/get, 세션 체크 함수 3개를 확인
- 외부 파일에서 함수 호출을 허가하는 옵션
  - 위 함수 3개 앞에 모두 export 처리 추가
- Js 폴더의 crypto.js와 jwt\_token.js도 추가 처리
  - 그림에 명시된 함수를 모두 export 처리 추가
- 저장 후 f12 개발자 모드
  - 소스코드 로딩 확인, 에러?

그림의 { } 안에 명시된 함수만 처리

```
import { session_set, session_get, session_check } from './session.js';  
import { encrypt_text, decrypt_text } from './crypto.js';  
import { generateJWT, checkAuth } from './jwt_token.js';
```



```
Uncaught (in promise) ReferenceError: session_get is not defined  
    at decrypt_text (crypto.js:52:16)  
    at init_logged (login2.js:37:15)  
    at HTMLDocument.<anonymous> (login2.js:47:5)
```

# 자바스크립트 - 모듈화

- 세션 또는 암호화 자바스크립트 확인
  - session.js 파일 최상위 set 함수 기능?
    - 암호화 한 후 세션에 저장
  - 세션 기능 안에서 암호화를 호출
    - Login.js와 같이 외부 파일(함수)는 import 필요
    - 모듈화 이후 반드시 외부 연결 필요
  - session.js 최상위에 추가한다.

```
import { encrypt_text, decrypt_text } from './crypto.js';
```

- crypto.js 최상위에 추가한다. (동일 개념)
  - 복호화 기능 안에서 세션을 불러온다.

```
import { session_set, session_get, session_check } from './session.js';
```

세션 안에서 암호화를 수행 = 연동 필요

```
JS session.js > session_set
export async function session_set() { //세션 저장
  let session_id = document.querySelector("#typeEmailX"); // DOM 트리에서 ID
  let session_pass = document.querySelector("#typePasswordX"); // DOM 트리에서
  if (sessionStorage) {
    let en_text = await encrypt_text(session_pass.value);
    let en_text2 = await encrypt_text2(session_pass.value);
    sessionStorage.setItem("Session_Storage_id", session_id.value);
    sessionStorage.setItem("Session_Storage_pass", en_text);
    sessionStorage.setItem("Session_Storage_pass2", en_text2);
  } else {
    alert("로컬 스토리지 지원 x");
  }
}
```

```
Uncaught (in promise) ReferenceError: session_get is not defined
    at HTMLDocument.<anonymous> (login2.js:47:5)
```

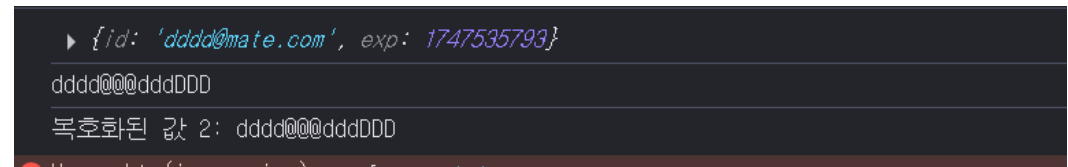
# 자바스크립트 - 모듈화

- 로그인 후 페이지 자바스크립트 추가하기
  - 기존 login.js를 새로운 login2.js로 복사한다.
- 로그인 후 페이지 기능 확인
  - 페이지 로딩 시에 init 호출 x
  - 토큰 인증, 복호화로 수정한다.
    - Auth, init\_logged 호출

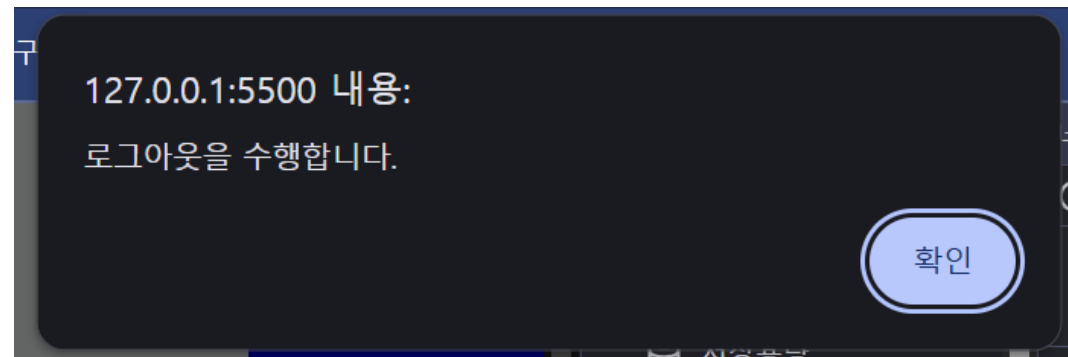
```
document.addEventListener('DOMContentLoaded', () => {  
  checkAuth();  
  init_logged();  
});
```

- 저장 후 f12 개발자 모드
  - 소스코드 로딩 확인

## 로그인 후 페이지의 동작 확인



## 로그 아웃도 확인한다. (세션, 토큰 삭제)



마지막 Login2.js를 정리한다. (필요 없는 함수 제거)

# 세션 - 객체 저장하기

- 웹 세션을 객체로 저장하는 기능을 구현한다.

- js 폴더의 session.js를 수정한다.
  - 기존 소스 코드 : session\_set() 함수는 주석처리
  - 기존 키, 값(문자열)으로 저장

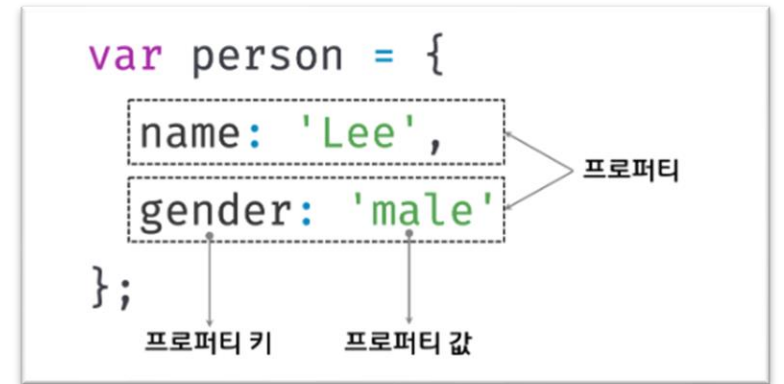
- JS 객체 생성 방법

- 중괄호로 묶고, 속성 값을 콜론 : 으로 구분 입력
- 입력한 아이디(이메일), 타임 스탬프 조합

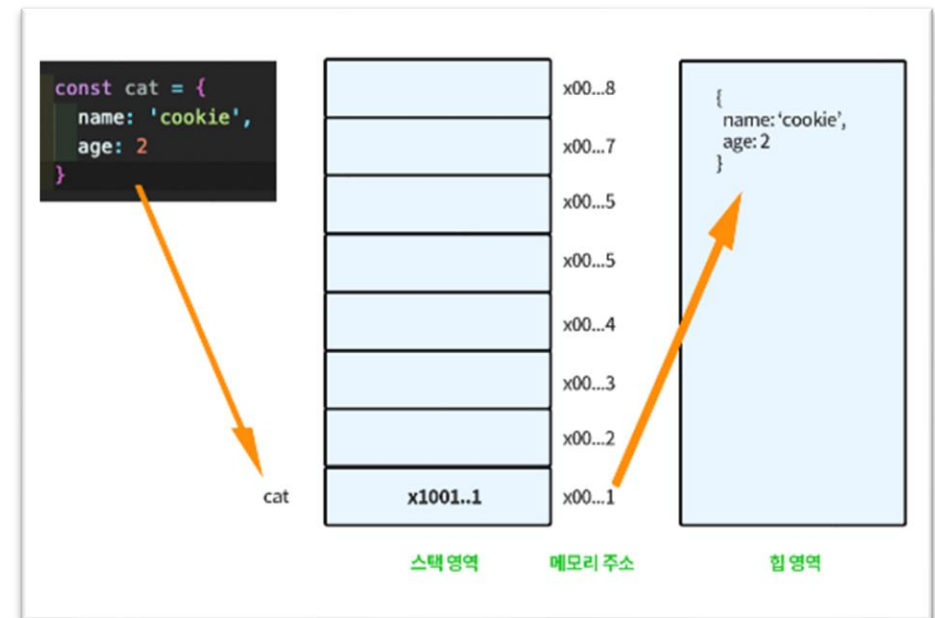
```
export function session_set(){ //세션 저장(객체)  
    let id = document.querySelector("#typeEmailX");  
    let password = document.querySelector("#typePasswordX");  
    let random = new Date(); // 랜덤 타임스탬프
```

```
    const obj = { // 객체 선언  
        id : id.value,  
        otp : random  
    }  
    // 다음 페이지 계속 작성하기
```

## 객체 리터럴 방식



## 객체의 저장 방법



# 세션 - 객체 저장하기

- 웹 세션을 객체로 저장하는 기능을 구현한다.

- JSON 방식으로 문자열 변환
  - .parse 메서드로 다시 역변환 가능

- 세션 저장 및 암호화 기능
  - 이름만 변경

```
if (sessionStorage) {  
    const objString = JSON.stringify(obj); // 객체 -> JSON 문자열 변환  
    let en_text = await encrypt_text(objString); // 암호화  
  
    sessionStorage.setItem("Session_Storage_id", id.value);  
    sessionStorage.setItem("Session_Storage_object", objString);  
    sessionStorage.setItem("Session_Storage_pass", en_text);  
} else {  
    alert("세션 스토리지 지원 x");  
}  
}
```

- 저장 후 결과 확인

## 로그인 - 세션 저장소 내용 확인

키	값
IsThisFirstTime_Log_From_LiveServer	true
Session_Storage_id	dddd@mate.com
Session_Storage_object	{"id":"dddd@mate.com","otp":"2025-05-18T23:55:22.73...
Session_Storage_pass	7vVS5t64gJdGTxy9YDonBX63ySxQZQKmfymO+O0L1r...
Session_Storage_pass2	cVhl2r+ar8L2OHk8:z0/Z8TPdVnpqLlfMw8/SjUCvB3SYhF...

## 로그인 후 - 콘솔 내용 확인

```
▶ {id: 'dddd@mate.com', exp: 1747616121}  
{"id":"dddd@mate.com","otp":"2025-05-18T23:55:22.737Z"}  
복호화된 값 2: dddd@nateDDD@@@
```

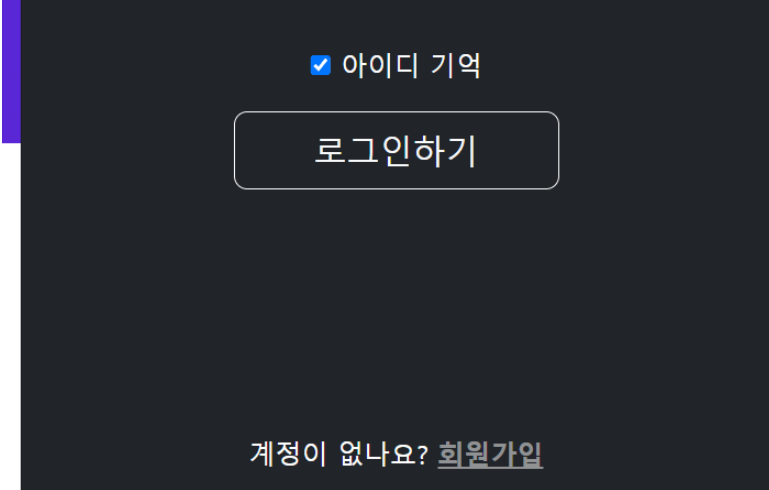
# 회원 가입 기능

- 로그인 페이지의 회원가입 수정하기
  - Login 폴더의 login.html을 수정한다.
    - 로그인 하단 영어 sign up을 한글, 링크 수정

```
<div>  
  <p class="mb-0">계정이 없나요? <a href="/login/join.html" class="text-white-50 fw-bold">회원가입</a>  
</p>  
</div>
```

- 저장 후 로그인 화면 확인
- 회원가입 페이지 추가하기
  - Login 폴더에 join.html 파일을 생성한다.
  - 기본 레이아웃 적용을 위해 소스코드 가져오기
    - Login.html의 전체 소스코드 복사 & 붙여넣기

## 회원가입 버튼 확인

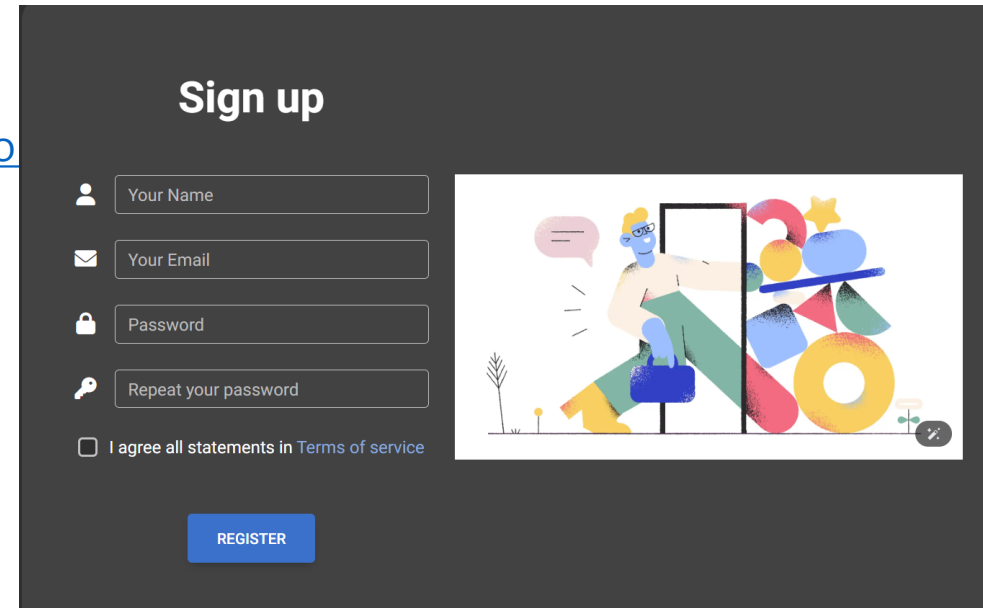


A dark-themed login form mockup. At the top, there is a checked checkbox labeled '아이디 기억'. Below it is a rounded rectangular button with the text '로그인하기'. At the bottom, there is a link that says '계정이 없나요? 회원가입'.

# 회원 가입 기능

- 부트스트랩 기반 회원 가입 폼 찾기
  - 웹 사이트에 접속한다. (업로드 소스코드 확인)
    - <https://mdbootstrap.com/docs/standard/extended/registratio>
    - SHOW CODE 클릭하여 소스코드 확인
  - 다양한 회원 가입 폼 확인
    - 원하는 회원 폼도 사용 가능
    - But, 식별자 id 값 다름(주의)
- Login 폴더에 join.html 파일을 수정한다.
  - 기존 body 태그 사이 내용을 삭제하기
  - 회원가입 소스코드를 사이에 붙여넣기

원본 회원가입 화면



회원가입 화면으로 내용 교체

```
<body class="text-center">  
  <section class="vh-100 gradient-custom"> ...  
  </section>  
</body>
```

# 회원 가입 기능

- Login 폴더에 join.html 파일을 수정한다.

- 헤더 부분의 소스를 정리한다.
  - 기존 login.js을 주석 처리한다.

- 회원가입 한글로 수정하기
  - 기타 레이아웃 수정 가능(기존 디자인)

```
<section class="vh-100 gradient-custom">
```

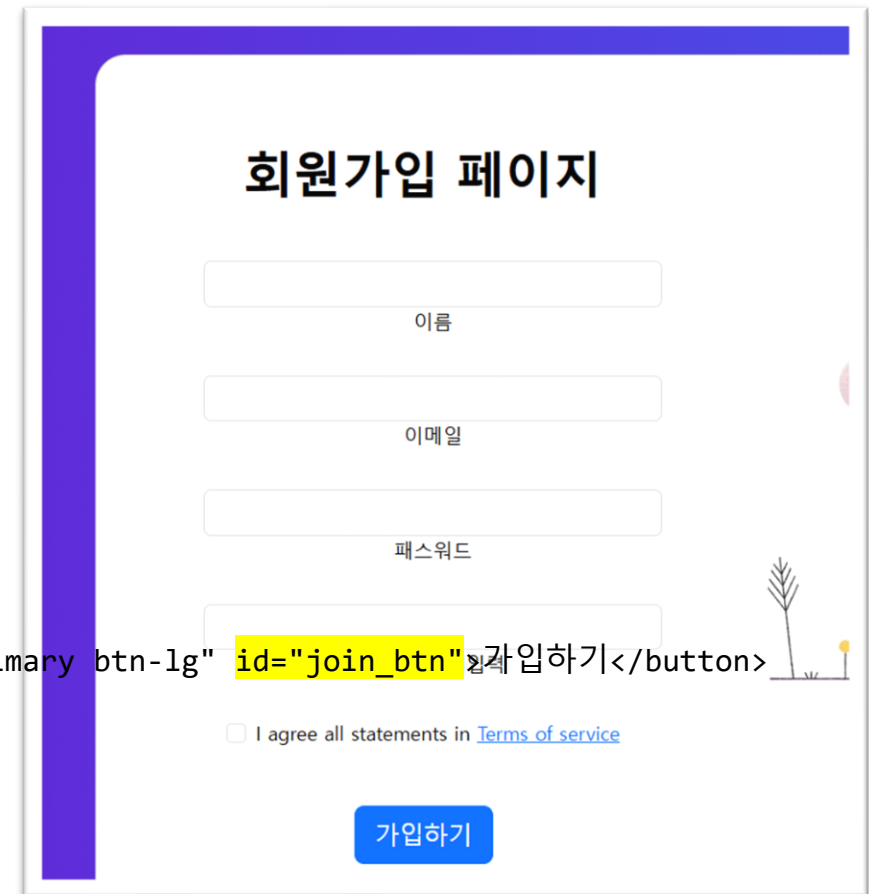
- 회원가입 폼 확인하기
  - 폼과 가입하기 버튼에 식별자 id를 추가한다.

```
<form class="mx-1 mx-md-4" id="join_form">
```

```
<button type="button" data-mdb-button-init data-mdb-ripple-init class="btn btn-primary btn-lg" id="join_btn">가입하기</button>
```

- 저장 후 화면 확인

내용을 한글로 수정한다.



회원가입 페이지

이름

이메일

패스워드

☐ I agree all statements in [Terms of service](#)

가입하기



# 회원 가입 기능

- 회원가입을 위한 JS 작성하기
  - JS폴더에 join.js 회원가입 기능을 작성한다. (주의 : 식별자 확인)

```
function join(){ // 회원가입 기능
    let form = document.querySelector("#join_form"); // 로그인 폼 식별자
    let name = document.querySelector("#form3Example1c");
    let email = document.querySelector("#form3Example3c");
    let password = document.querySelector("#form3Example4c");
    let re_password = document.querySelector("#form3Example4cd");
    let agree = document.querySelector("#form2Example3c");
```

```
    form.action = "../index.html"; // 로그인 성공 시 이동
    form.method = "get"; // 전송 방식
```

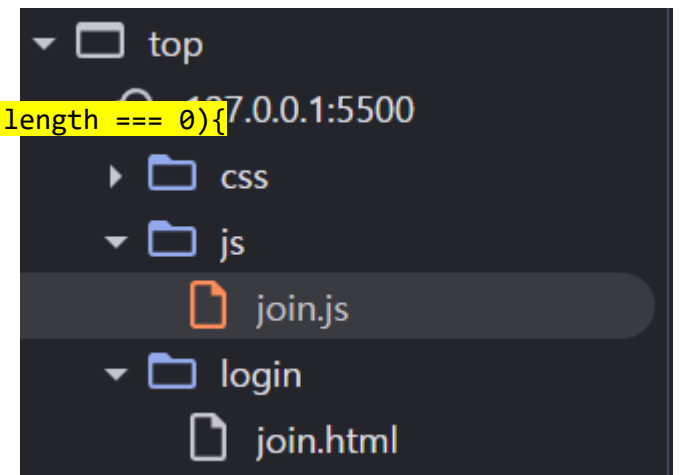
```
    if(name.value.length === 0 || email.value.length === 0 || password.value.length === 0 || re_password.length === 0){
        alert("회원가입 폼에 모든 정보를 입력해주세요.");
    }
    else{
        form.submit(); // 폼 실행
    }
}
```

```
document.getElementById("join_btn").addEventListener('click', join); // 이벤트 리스너
```

## 참고 : 다양한 선택자들

getElementById	<pre>var bar = document.getElementById("bar"); var poo = document.getElementById("poo");</pre>
querySelector (ID)	<pre>var bar = document.querySelector("#bar"); var poo = document.querySelector("#poo");</pre>
jQuery ID Selector	<pre>var bar = \$("#bar"); var poo = \$("#poo");</pre>
getElementsByClassName	<pre>var bar = document.getElementsByClassName("my-bar");</pre>

## Join.html에 연동한다.

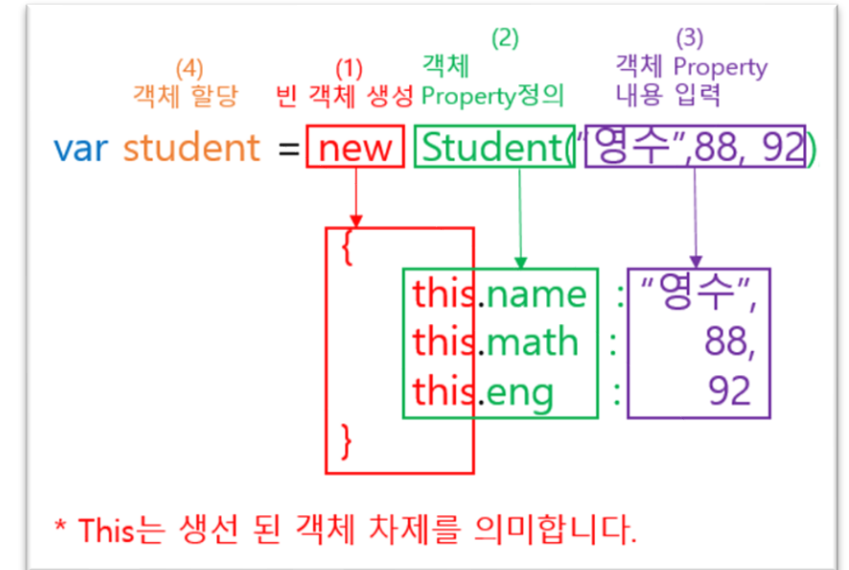


# 회원 가입 기능

- 회원가입 정보를 객체로 저장한다.
  - join.js에 SignUp 클래스를 작성한다. (클래스 구현)
    - 앞서 예제와 같이 객체 리터럴 방식을 사용한다.

```
class SignUp {  
  constructor(name, email, password, re_password) {  
    // 생성자 함수: 객체 생성 시 회원 정보 초기화  
    this._name = name;  
    this._email = email;  
    this._password = password;  
    this._re_password = re_password;  
  }  
}
```

- 기본 값이 없음, 인자값으로 전달 받아 초기화
- 다음 페이지 소스 코드 이어서 작성



## 생성자 : constructor

객체 생성 시에 자동 초기화 담당

C++, C#, 자바 등 대부분 존재

this : 객체 자신 참조

# 회원 가입 기능

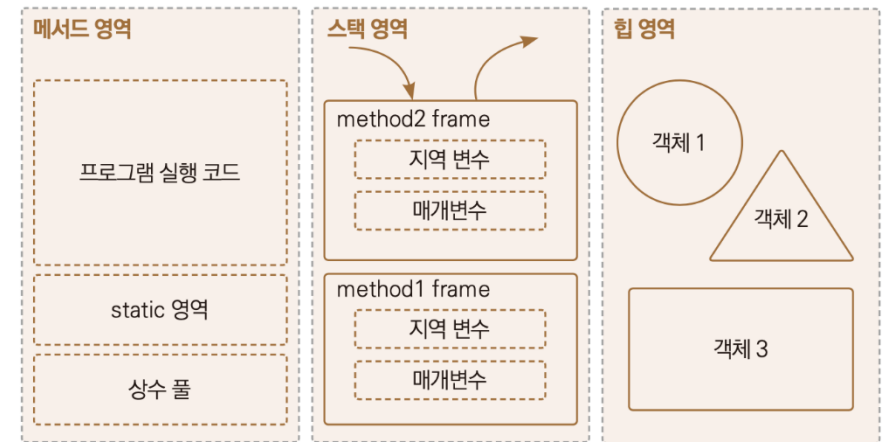
- 회원가입 정보를 객체로 저장한다.
  - SignUp 클래스를 작성을 마무리 한다. (클래스 구현)

```
// 전체 회원 정보를 한 번에 설정하는 함수
setUserInfo(name, email, password, re_password) {
    this._name = name;
    this._email = email;
    this._password = password;
    this._re_password = re_password;
}
```

```
// 전체 회원 정보를 한 번에 가져오는 함수
getUserInfo() {
    return {
        name: this._name,
        email: this._email,
        password: this._password,
        re_password: this._re_password
    };
}
```

## 객체는 힙 영역에 저장됨

▼ 런타임 데이터 영역들



접근자 프로퍼티 : set, get

내부 값에 접근할 때 사용

인자 유무로 호출을 구분

# 회원 가입 기능

- 회원가입과 함께 세션 객체를 생성한다.
  - 기존 join.js 내부의 join() 함수를 수정한다.
    - 회원 가입되면 객체를 생성하고 세션에 저장한다.

```
if(name.value.length === 0 || email.value.length === 0 || password.value.length === 0 || re_password.value.length === 0){
    alert("회원가입 폼에 모든 정보를 입력해주세요.");
}
else{
    const newSignUp = new SignUp(name.value, email.value, password.value, re_password.value); // 회원가입 정보 객체 생성
    session_set2(newSignUp); // 세션 저장 및 객체 전달
    form.submit(); // 폼 실행
}
```

- 세션 함수를 사용해야 한다. 모듈화 이후
  - 최상단에 join.js에 session.js를 연동해야 한다.

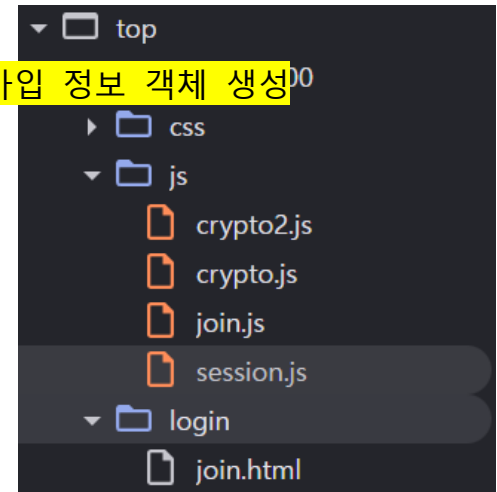
```
import { session_set2 } from './session.js';
```

- Session\_set2를 호출하도록 설정한다.

## • 저장 후 소스 코드 확인

- 세션에 저장된 회원 정보 확인

F12에서 소스 연동 확인



키	값
IsThisFirstTime_Log_From_LiveServer	true
Session_Storage_newJoin	{"_name":"ㅇㅇㅇㅇㅇ","_email":"dwadwadwa","_passv

# 회원 가입 기능

- 회원가입에 필요한 입력 필터링을 구현한다.
  - Js 폴더의 join.js에 기능을 추가한다.
    - Join() 함수 최상단에 변수를 선언한다.

```
const nameRegex = /^[가-힣]+$/;  
const emailRegex = /^[^@]+@[^@]+\.[^@]+$/;  
const pwRegex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[\W_]).{8,}$/;
```

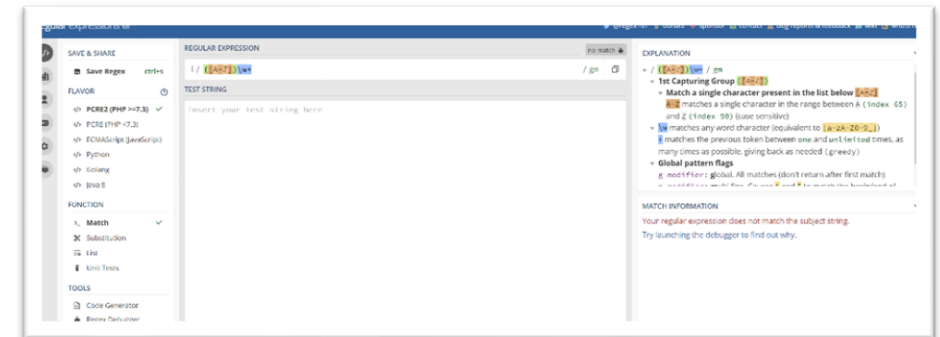
- if()문에 필터링 및 약관 동의 조건을 추가한다.

```
if (!nameRegex.test(name.value)) { // 이름 검사  
  alert("이름은 한글만 입력 가능합니다.");  
  name.focus();  
  return;  
}
```

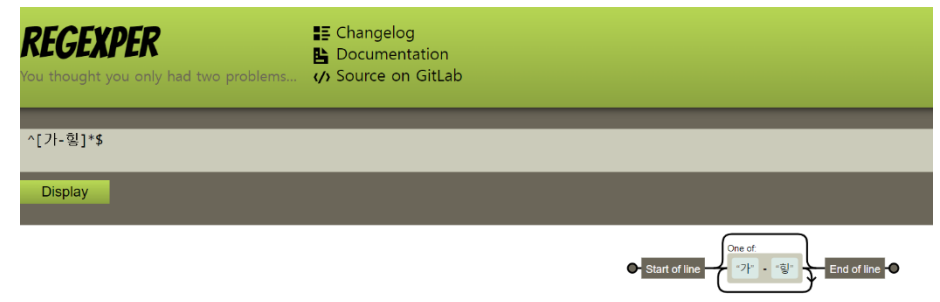
```
if (!emailRegex.test(email.value)) { // 이메일 검사  
  alert("이메일 형식이 올바르지 않습니다.");  
  email.focus();  
  return;  
}
```

## 정규 표현식 테스트

[regex101 \(https://regex101.com/\)](https://regex101.com/)



[egexper \(https://regexper.com/\)](https://regexper.com/)



# 회원 가입 기능

- 회원가입에 필요한 입력 필터링을 구현한다.
  - if()문에 필터링 및 약관 동의 조건을 추가한다.

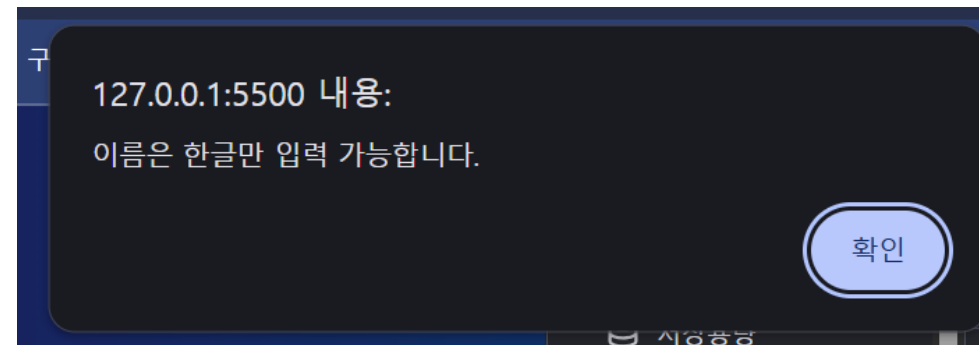
```
if (!pwRegex.test(password.value)) { // 비밀번호 검사
    alert("비밀번호는 8자 이상이며 대소문자, 숫자, 특수문자를 모두 포함해야 합니다.");
    password.focus();
    return;
}
```

```
if (password.value !== re_password.value) { // 비밀번호 일치 검사
    alert("비밀번호가 일치하지 않습니다.");
    re_password.focus();
    return;
}
```

```
if (!agree.checked) { // 약관 동의 확인
    alert("약관에 동의하셔야 가입이 가능합니다.");
    return;
}
```

- 저장 후 필터링 테스트

## 직접 필터링 테스트



- 12주차 연습문제

**회원 가입 - 암호화**

# 회원가입 - 암호화

- 회원가입 후 암호화된 객체 저장하기
  - 세션에 암호화하여 저장
- 로그인 후 복호화된 객체 내용 출력하기
  - 암호화된 회원가입 정보를 복호화
    - 콘솔에 출력
  - 세션에 회원가입 세션이 없다면
    - 복호화 x, 출력하지 않음
- 실습 결과 확인 - Q / A

