〈Easy Save〉

1. 한 줄 요약 / 핵심 장점

– Easy Save (Easy Save 3 등)는 Unity에서 데이터 저장/불러오기를 매우 간편하고 강력하게 구현할 수 있게 해 주는 상용 에셋입니다.

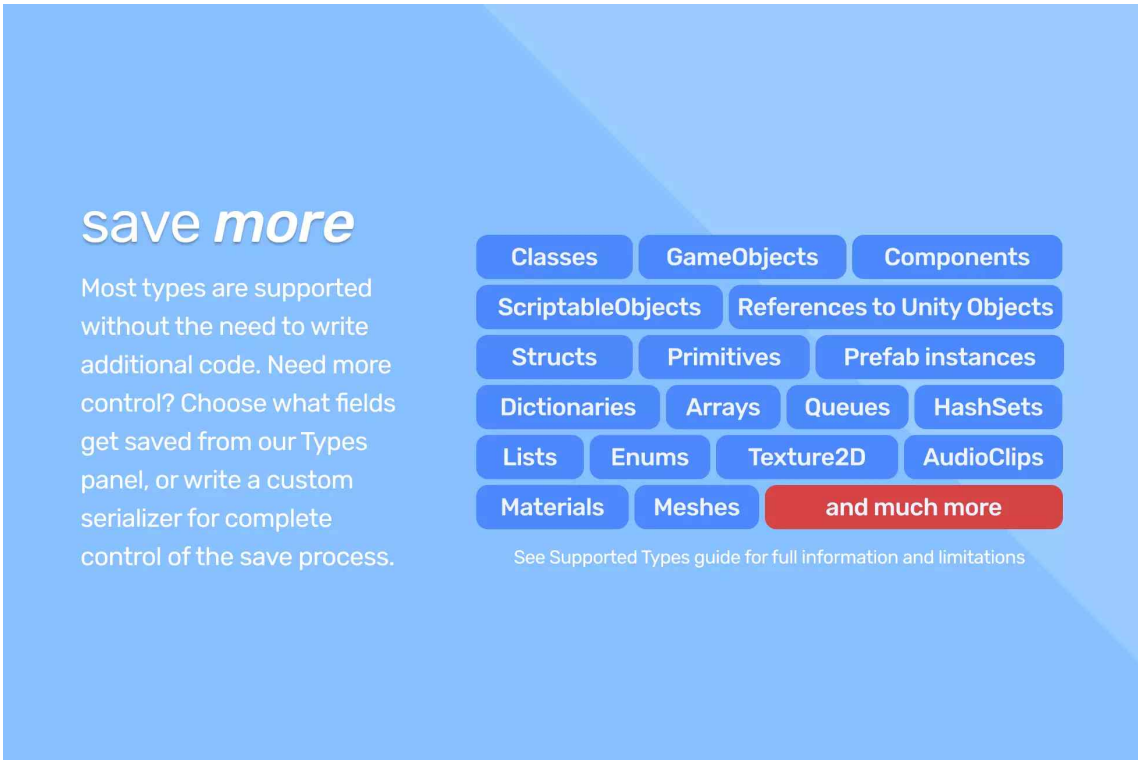
– 기본 타입(int, string 등)은 물론이고 클래스, 배열, 리스트, 날짜 등 복합 데이터까지 JSON 직렬화, 암호화, 파일 분할 등을 지원해 줍니다.

– 기존 'PlayerPrefs'나 'Resources.Load' 방식보다 훨씬 유연하고 안전한 저장 구조를 제공합니다.

2. 주요 기능 & 특징

핵심 기능들을 정리하면 다음과 같습니다:

기능	설명
기본 저장 & 불러오기	ES3.Save(key, value) 와 ES3.Load<Type>(key) 형태로 간단 저장/로드 가능
키 존재 여부 확인	ES3.KeyExists(key) 로 특정 키가 저장되어 있는지 체크
AES 암호화 기능	저장 데이터를 암호화 설정 가능. 암호 키를 지정해 AES 방식 암호화 적용 가능함
파일 분할 저장	기본 파일 외에 별도 .es3 파일 이름을 지정해서 여러 파일로 나누어 저장 가능
고급 저장 방식(ES3File 사용)	ES3File 인스턴스를 생성하여 여러 데이터를 하나의 파일 안에 묶어서 저장 / 불러오기 가능
TryLoad 패턴	키 존재 여부 체크 + 로드 실패 방지 패턴 (예: if (file.KeyExists...))
주의사항	저장할 클래스는 public 무파라미터 생성자 필요, Transform 등의 Unity 참조는 직접 저장 불가 등



3. 기본 사용법 & 코드 예제

3.1 가장 단순한 저장 / 불러오기 예제

```
using UnityEngine;

public class EasySaveBasic : MonoBehaviour
{
    void Start()
    {
        string profile = "Hello";
        ES3.Save("profile", profile);

        if (ES3.KeyExists("profile"))
        {
            string loadProfile = ES3.Load<string>("profile");
            Debug.Log(loadProfile);
        }
    }
}
```

- ‘key’는 문자열 식별자
- ‘ES3.KeyExists’ 로 존재 여부 먼저 체크
- ‘ES3.Load<T>’ 제네릭 방식으로 타입 지정

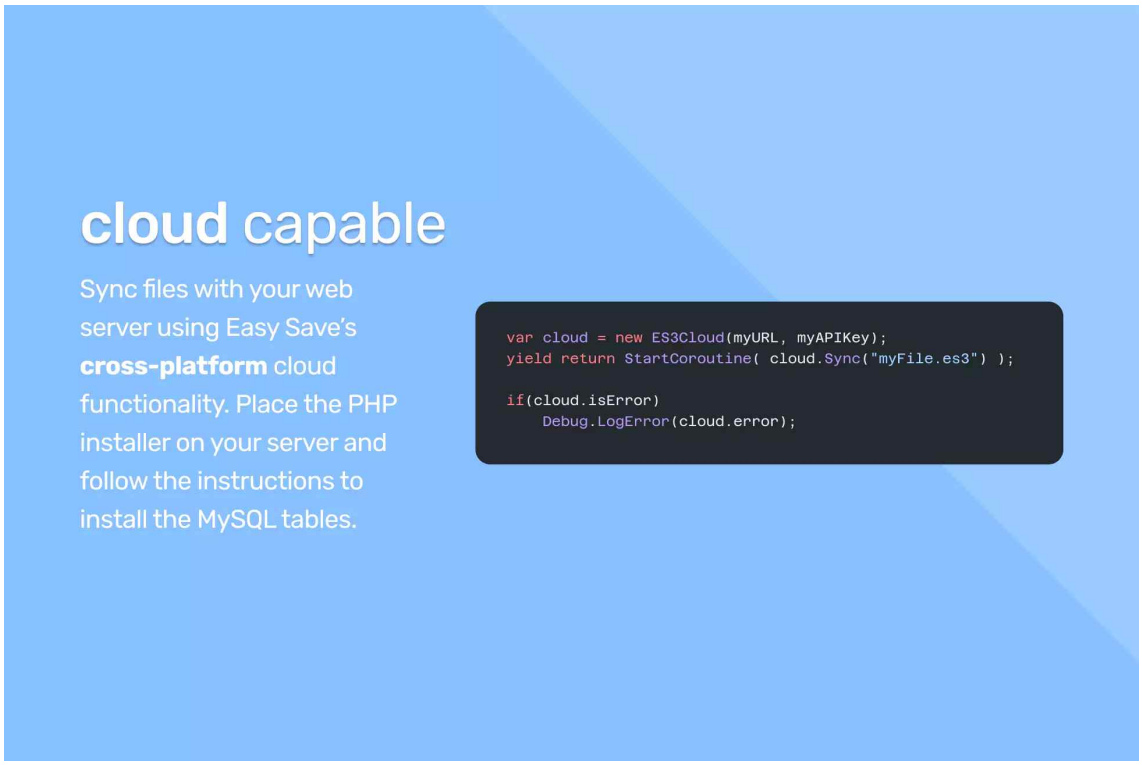
3.2 파일 분할 저장 예제

```
using UnityEngine;

public class EasySaveFileSplit : MonoBehaviour
{
    void Start()
    {
        string profile = "Hello";
        ES3.Save("profile", profile, "SaveFile_Profile.es3");

        if (ES3.KeyExists("profile", "SaveFile_Profile.es3"))
        {
            string loadProfile = ES3.Load<string>("profile", "SaveFile_Profile.es3");
            Debug.Log(loadProfile);
        }
    }
}
```

- 세 번째 인수로 파일명 '.es3' 지정
- 해당 파일 내에서 키/값 저장 & 불러오기



3.3 복합 클래스 + ES3File + TryLoad 예제

```
using UnityEngine;
using System;

[Serializable]
public class Profile
{
    public string name;
    public int age;
    // 반드시 무파라미터 생성자 제공
    public Profile() { }
    public Profile(string name, int age)
    {
        this.name = name;
        this.age = age;
    }
}

public class EasySaveAdvanced : MonoBehaviour
{
    [SerializeField] Profile[] loadProfiles;

    void Start()
    {
        Profile[] profiles = new Profile[]
        {
            new Profile("aa", 20),
            new Profile("bb", 30)
        };

        DateTime nowTime = DateTime.Now;

        // 저장
        {
            ES3File es3File = new ES3File("SaveFile_Profile.es3");
            es3File.Save("profiles", profiles);
            es3File.Save("nowTime", nowTime);
            es3File.Sync();  // 파일 디스크 쓰기 동기화
        }

        // 불러오기
        {
            ES3File es3File = new ES3File("SaveFile_Profile.es3");
            if (TryLoad("profiles", es3File, out Profile[] _profiles))
                loadProfiles = _profiles;

            if (TryLoad("nowTime", es3File, out DateTime _nowTime))
                nowTime = _nowTime;

            Debug.Log(nowTime);
```

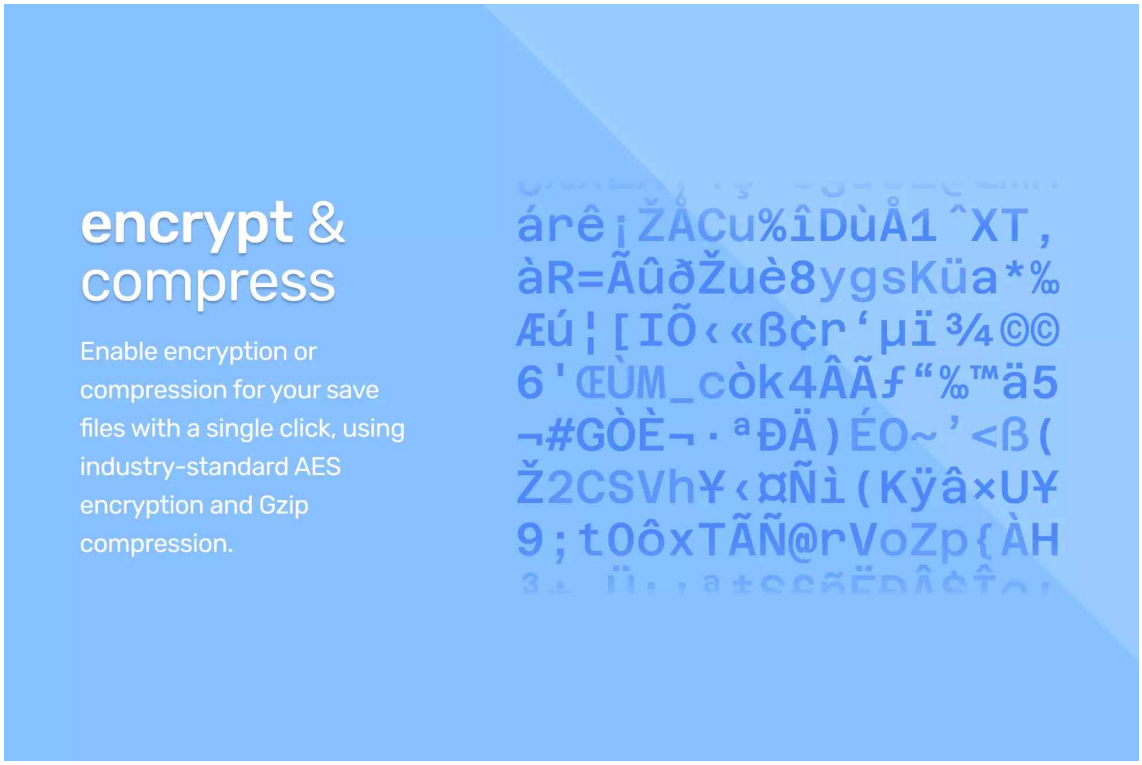
```
        }
    }

    bool TryLoad<T>(string key, ES3File file, out T t)
    {
        if (file.KeyExists(key))
        {
            t = file.Load<T>(key);
            return true;
        }
        t = default;
        return false;
    }
}
```

– ‘ES3File’ 클래스를 사용해 여러 키/값을 같은 파일 내에 저장

– ‘Sync()’ 메서드를 호출해 변경 내용을 디스크에 한번에 반영

– ‘TryLoad<T>’ 패턴으로 키 존재 여부 체크 + 안정성 높이기



4. 설치 & 설정 안내

– Unity Asset Store에서 “Easy Save – The Complete Save Data & Serializer System” 구매 및 Import

– 프로젝트에 임포트한 후, 메뉴〉 Window〉 Easy Save 3〉 Tools 혹은 Settings 창 접근 가능

– 저장 위치:

기본은 ‘Application.persistentDataPath’ 내에 ‘.es3’ 파일로 저장됨

– 키/값 저장 시 내부적으로 JSON 구조로 직렬화됨 (예: ‘SaveFile.es3’ 내부 JSON 구조)

– Settings 창에서 Encryption (None 또는 AES) 설정 가능 (AES 선택 시 암호 입력 필요)

5. 베스트 프랙티스 & 주의사항

– 클래스 저장 시 무파라미터 생성자 (default constructor) 필요 (아예 기본 생성자 정의)

– Unity 컴포넌트 또는 ‘Transform’ 등 참조형 객체는 직접 저장 불가능 — 별도 식별자/데이터로 저장해야 함

– 많은 데이터를 파일 하나에 몰아서 저장하지 말고 파일 분할 & 그룹화 권장 (성능 향상 목적)

– 저장 / 불러오기 반복 시 Sync 또는 배치 처리 권장

– 암호화를 사용할 경우 Key 관리 주의 (암호 유출 시 보안 위험)

– 큰 저장 파일을 빈번히 사용할 경우 디스크 I/O 병목 고려