Javascript osztályok és modulok

Az osztályok és modulok az ES6-ban (ECMAScript 2015) kerültek bevezetésre, hogy modernebb, strukturáltabb és objektumorientált programozást biztosítsanak.

**JavaScript Osztályok**

A JavaScript osztályok

* a megszokott módon a *class* kulcsszóval definiálhatók,
* rendelkezniük kell *constructor* függvénnyel,
* tartalmazhatnak mezőket és metódusokat,
* példányosításuk a *new* kulcsszóval történik,
* támogatják az öröklést az *extends* kulcsszó segítségével.

Az osztályok példányai ugyanolyan objektumok mint bármilyen más módon létrehozott JS objektumok, így ugyanolyan dinamikusan változtathatók.

class Entity {

constructor() {

this.id = ‘’;

}

get label() {

}

}

**JavaScript Modulok**

A modulok használatával a JavaScript kód szétbontható kisebb, újrafelhasználható egységekre, és ezeket az *export* és *import* kulcsszavakkal érhetjük el. Minden modulnak saját szintaktikai környezete van, így elkerülhető a globális névtér szennyezése. A modulokat általában külön fájlokban kezelik, és az import utasítás lehetővé teszi az egyes komponensek szelektív importálását.

**Összehasonlítás C#-al**

A JavaScript osztályai sok tekintetben hasonlítanak a C# osztályokra, mivel mindkettő támogatja az objektumorientált programozási koncepciókat, mint az öröklés és az inkapszuláció. Ugyanakkor a C# típusosan biztonságos (statikusan típusos) nyelv, ahol minden osztály és típus előre definiált típusokkal rendelkezik. JavaScriptben azonban az osztályok dinamikusabbak, és a típusbiztonság hiánya nagyobb rugalmasságot, de kevesebb típusbiztonságot is eredményez.

C#-ban a moduláris megközelítés a névterek (namespace) és az assembly-k használatával valósul meg, míg JavaScriptben a modulok egyszerű exportálási és importálási mechanizmust kínálnak. Az import és export kulcsszavak mellett C#-ban olyan koncepciók is elérhetők, mint az interfészek és absztrakt osztályok, amelyek további struktúrát biztosítanak az osztályok számára, míg JavaScriptben ezeket explicit módon nem támogatja a nyelv.