Първа 1  
Дотук видяхме как файловете се качват, валидират и съхраняват. Но какво става, когато искаме не просто да ги запазим, а да ги обработим – при това в реално време и без да натоварваме паметта? -> Нека се върнем назад за момент – преди потоците, работата с файлове изглеждаше така..."  
Слайд с надпис **"readFile – цял файл в паметта"**  
  
fs.readFile('file.txt', 'utf8', (err, data) => {

if (err) throw err;

const words = data.split(/\s+/).length;

console.log('Words:', words);

});  
Целият файл се зарежда в паметта наведнъж – което при големи файлове води до забавяне, а понякога до срив. Ако файлът е 2GB, RAM-ът се пълни и приложението може да крашне. Да не говорим, че ако искаме да броим думи – чакаме файлът да се прочете целият.

Част 2

Затова се въвеждат потоци . Днес вече използваме readStream и **Transform stream**, за да **обработим файла на части**, още докато се чете. Нека първо видим как изглежда само четенето на файл чрез readStream, без обработка, като той чете файла на части – chunk по chunk. Подготвила съм си код, който ще ми помогне да разберем какъв е размера на всяко парче. В нашата ситуация те са по 116 байта. Ако файлът беше 2GB, нямаше да срине приложението, защото използваме малко памет в даден момент.Чрез highWaterMark можем да зададем максималното количество байтове, които да се четат наведнъж. Нека да го ограничим до 16 байта. Виждате, че всяко парче е с различна дължина – 7, 8, 9, понякога 12 байта. Това е защото някои символи заемат повече байтове, но Node.js е умен – той не прекъсва символи по средата. Но всичко това е само четене, нека да направим нещо реално и да включим и Transform stream като ще броим думи в текстов файл по време на четене.

Transform stream е поток, който приема вход, обработва го и може да върне ново съдържание. В нашия случай – броим думите, но не връщаме нов текст – само статистика.

Сега файлът отново се чете chunk по chunk – но всеки chunk се обработва още по време на четенето. Тук броим думи. Накрая flush връща резултата. Това е реална трансформация по време на поток. Забелязахте ли за колко време имаме резултат, ама вие ще си кажете, е има само 17 думи, лесна работа. Добре, нека да пробваме с доклада по апис, всичко го предадохме тази седмица, знаете колко са големи. В нашия случай са само 45 314 думи. Отново резултата беше бърз. Именно това правят потоците толкова мощни – можем да четем, обработваме и реагираме на данните, още докато текат.