```
//CONSOLE.TIME:


//Existe dentro do Node uma função que imprime no console o tempo de uma execução. Ela funciona como uma espécie de cronometro onde é possível medir quanto tempo é gastado á partir de uma momento na execução do código até outro momento no futuro.

//Essa função é a "console.time" e suas variáveis, elas são:
/*      * console.time('label') - console.time serve como uma marcação de tempo, ele recebe um label - qualquer nome em string
                                  - que servirá como identificador de tempo, o marco zero;

        * console.timeLog('label') - console.timeLog ele identificará um ponto no tempo depois que uma label foi
                                     pré-determinada, ele pega o momento exato do marco 1 da execução;

        * console.timeEnd('label) - o console.timeEnd finaliza o cronometro de um console.time, ele identifica qual é o
                                     cronometro por através da label e o finaliza. Além de finalizar o cronometro, ele também mostra quanto tempo o cronometro durou no console;
 */

//APLICANDO O CRONOMETRO DO CONSOLE POR ATRAVÉS DA CONSOLE.TIME:
//Temos abaixo uma função que gerará um número aleatório de determinado número inicial até um final em uma quantidade de tempo pré-determinada:
function numbersRandom(firstPoint, finalPoint, seconds) {
    if(firstPoint > finalPoint) [finalPoint, firstPoint] = [firstPoint, finalPoint] //Faz a validação dos números iniciais e finais...
    return new Promise(resolve => {
        setTimeout(function() {
            const randomNum = parseInt(Math.random() * (finalPoint - firstPoint) + firstPoint) //Gerador de números aleatórios dentro do intervalo passado...
            resolve(randomNum)
        }, seconds) //Aqui a quantidade de tempo é passada...
    })
}

//Abaixo temos uma função gerará vários números aleatórios, note que cada um deles tem um time...
function variousNum() {
    return Promise.all([ //Como todos eles estão dentro de um "Promise.all()" a execução só termina depois que áquele que levar maior tempo - que no caso é 4 segundos - tiver terminado a sua execução, retornando um array com todos os valores...
```

```
        numbersRandom(1, 10, 4000),
        numbersRandom(1, 10, 3000),
        numbersRandom(1, 10, 2000),
        numbersRandom(1, 10, 1000),
        numbersRandom(1, 10, 500),
    ])
}

console.time('promise') //Perceba que aqui temos o marco zero, como essa execução está fora da promise, ela é executada primeiro...

variousNum()
    .then(numbers => console.log(numbers))
    .then(() => { //Perceba que só depois da chamada da promise temos os "console.timeLog" - para mostrar o tempo que levou até o término de todas as execuções - e o "console.timeEnd" que finaliza a execução e além disso mostra o valor final do cronometro...
        console.timeLog('promise'),
        console.timeEnd('promise')
    })
```

RESULTADO NO CONSOLE...

```
[Running] node "c:\Users\Almoxarifado\Documents\javascript\arquivos_das_aulas\156-Promise.all__O_executor_de_multiplas_promises.js"
[ 9, 8, 2, 2, 8 ]
promise: 4.027s
promise: 4.028s

[Done] exited with code=0 in 4.192 seconds
```