Job queue

Earlier we said that setTimeout is not a part of JavaScript, it's a part of WEB API

However, with the introduction of Promises Promises let us handle asynchronous code and they are actually a part of Javascript

So we need something to handle these promises just like we need Callback queues to handle setTimeout callback.

// Job queue - Microtask queue

Job queue is a bit smaller compared to callback queue buthas higher priority

So the event loop is going to check the job queue first and then callback queue if it is empty

(guess the output)

```
Promise.resolve('Wohoo').then(()=> console.log('2'))
setTimeout(() => console.log('1'), 0)
setTimeout(() => console.log('3'), 10)
console.log('4')
```

@codeWithSimran

→ 4
2
1
3

How to execute parallel, sequence and race using promises

@codewithsimran

① Parallel

In order to execute promises in parallel (all together) and get their result together we can use

```
Promise.all([promise1, promise2, promise3..])
```

② Race

When we have multiple promises and we only want to return the result of the promise that get's resolved first we can use

```
Promise.race([promise1, promise2....])
```

It will only give result of the promise that gets resolved first

③ Sequence @codewithSimran

What if we have n promises and we want to execute them one after the other ?

```
const p1 = await promise1
const p2 = await promise2
   ⋮
const pn = await promisen
```

In this case after promise1 gets resolved, promise 2 gets executed, once that resolves promise3 and so on....

ES2020 : allSettled. @codewithSimran

We already talked about Promise.all

```
Promise.all([promise1, promise2]).then
        (data => console.log(data))
```

Let's say promise1 gets resolved and promise2 gets rejected.

> Output → Uncaught (in promise) undefined

So we need to put a catch statement to properly handle this

So promise.all resolves only if all the promises resolves. Even if one of them gets rejected, its going to result in an error.

@codewithsimran

for that matter, we have Promise.allSettled which will result in an array of outputs from the promises, if they get resolved or rejected.

Promise.allSettled([promise1, promise2])

> [{status: "fulfilled", value: 'somevalue'},
{status: "rejected", value: undefined}]