PROMESSAS

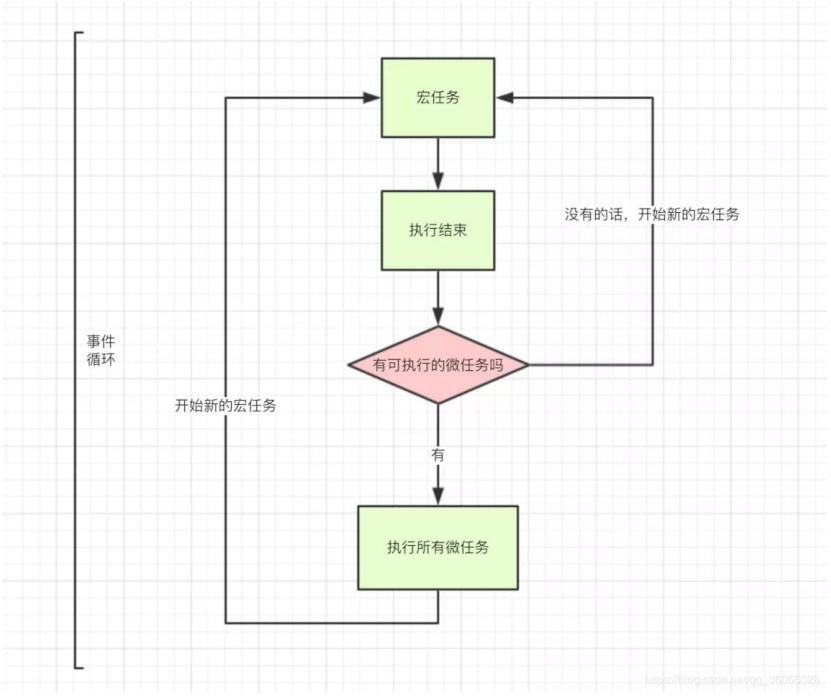
```
const pa = delay('a').then(console.log).catch(console.error)
const pe = delay('e').then(console.log).catch(console.error)
const pi = delay('i').then(console.log).catch(console.error)
const po = delay('o').then(console.log).catch(console.error)
const pu = delay('u').then(console.log).catch(console.error)
```

```
const inc = x => x + 1

const p1 = Promise.resolve(1)
const e1 = Promise.reject(1)

const p2 = p1.then(inc)
const e2 = e1.then(inc)
```

PROMESSAS



- Event loop
- Microtask

PROMESSAS

```
const p = new Promise((res, rej) => {
  /**
   * ????
   */
}).then(console.log)
  .catch(console.error)
```

- Exportar res/rej
- Broadcast
- AsyncCallBack

PROMESSAS

Promessa(promessa)

Promessa(rejeição)

PROMESSAS

```
;(async () => {})()
  .then(console.log)
  .catch(console.error)
```

```
;(async () => {
  await delay(5000)
  throw
})()
  .then(console.log)
  .catch(console.error)
```

PROMESSAS

```
async () => {
  const [err, data] = await maybe
    .then(data => [undefined, data])
    .catch(err => [err])

  if (err) {
    throw err
  } else {
    return data
  }
}
```