FUNDAMENTAL NODE JS

BEGZAT KIDIRBAEV

Fundamentallıq Node.Js

1-Bólim: Asinxron baǵdarlamalaw tiykarları

- Sinxron hám asinxron aǵın
    - Bloklawshı hámde bloklamaytuǵın baǵdarlama
    - Resurslar ısrapı
- Asinxron baǵdarlamalaw qıyınshiliqları
    - Til imkániyatları hám sheklilik
- Javascipt'te asinxron aǵısdi payda etiw
    - Funkcional baǵıt
    - Qatarlıq baǵıt
    - Parallel hám shekli parallel baǵıt
    - async/await hám Async.Js
    - Promise'lar

2-Bólim: Node.Js

- Node.Js filosofiyası
    - Kishi tp hám modulliliq
    - Ápiwayililiq hám pragmatizm
- Node.Js tiykarları
    - Node.js qanday isleydi
    - Bloklawshi hám bloklamaytuǵini Kiriw/Shiǵiw
    - Node.js arxitekturasi hám dizaynı
    - Node.js ishinde Javascript
    - Modul Sisteması hám Tp modular
    - NPM
    - EventEmitter

3-Bólim: Node.Js'ta baǵdarlama jasaw

- Veb Serverlar hámde HTTP dástrlew

        - Internet jáne veb serverlar
        - Soraw/Juwap dzilisi
        - Klient-Server sxemasi
        - Node.Js'ta HTTP server
    - Maǵliwmatlar menen islesiw
        - Node.Js platformasinda Fayllar Sistemasi
        - Maǵliwmatlar Bazasi
        - Relyatsion maǵliwmatlar modeli
        - RestAPI hám MongoDB
    - Qáteliklerdi gzetiw hám Jurnallastiriw
        - Node.js'ta qáteliklerdi gzetiw texnikalari
        - Qáteliklerdi dzetiw
        - Qátelik hám process jurnali
    - Testlew hám Sapaliliq
        - Node.Js kod strukturasi hám Gózzalliq
        - Node.js baǵdarlamasin testlew usillari
        - Jest/Chai testlew kitapxanasi
    - Baǵdarlamani keńeytiriw
        - Baǵdarlama kodi versiyalarin basqariw
        - Baǵdarlamani Replit'ta isletiw
        - AWS hám Amazon servisleri

4-Bólim: Quramali temalar

- Quramali kontseptsiyalar
    - Event hám Taymerler
    - Serializatsiya hám deserializatsiya
    - Memoizatsiya
    - Factory hám Poll
- Nest.js freymvorki
    - CQRS hám Nest.js arxitekturasi

        - Modullar araliq baylanislar
        - Nest.js’ta baǵdarlama
    - Real-Time baǵdarlamalar
        - Strimlar anatomiyasi
        - Vebsocketler hám strim protokollari
        - Socket.io
    - Qáwipsizlik hám duris praktikalar
        - Klient qáwipsizligi
        - Node.Js baǵdarlamasi qáwipsizligi
        - Soraw hám Juwaplardi qorǵaw

5-Bólim: Javascript baǵdarlamalarin optimizatsiyalaw

- Node.Js’ta lgili kod koncepciyaları
- Keń qollaniliwshi patternńlar
- Mikroservis hám Serversiz baǵdarlamalar

Bonus: **DALL-E menen Node.Js’ta jasalma intellektli proekt**

1-Bólim: Asinxron baǵdarlamalaw tiykarlari

Sinxron hám asinxron barıs

Tezlik hám effektivlik eń tiykarǵı bolǵan búgingi zamanda heshkim qandaydır bir baǵdarlamanı isletiw ushın hátte bir-neshe minut kútiwdi qálemeydi. Usınday baǵdarlamalawda bul trendge maslasıwǵa májbúr. Sebebi paydalanıwshılar tez hám sapalı sheshimlerlerdi qáleydi, tradiciyalıq baǵdarlamalawda bolsa bul ádewir qıyınshılıq tuwdıradı.

Óytkeni tradiciyalıq baǵdarlamalaw tiykarınan sinxron barıstı payda etedi. Bul neni ańlatadı? Kóz aldımısqa on qabatlı minára qurılısına juwapker aktiv brigadani keltireyik. Brigada mináranı joqarıǵa qarap qabatpa-qabat quradı. Birinshi «Birinshi qabat», ol pitkennen soń «Ekinshi qabat» hám usılayınsha dawam etip aqırǵı bolıp sońǵı «Onınshı qabat» qurılıp minára qurılısı tamamlanadı. Bunda keyingi qabatqa, aldınǵı qabat qurılısı tolıq tamamlanbastan turıp ótilmeydi.

Sinxron barıstı minára qurılısına, qabatlardı málim bir operaciyalar tártibine qıyaslasaq boladı. Bunda bir operaciya tamamlanbastan keyingi operaciyaǵa ótiw imkániyatı bolmaydı. Tártipte aldınǵı turǵan operaciyanıń orınlanıwı qanshelli uzaq waqıt dawam etse, tártipte keyingi turǵan operaciyaniń orınlanıwın sonshelli uzaq waqıt bloklap qoyadı. Endi oylap kóreyik, bizdiń baǵdarlamamıs ádette mıńlap operaciyalardan turadı. Eger bunday bloklanıwlar mıńlap márte dawam etetuǵın bolsa, baǵdarlamamıstıń tezligi ádewir páseyedi.

Álbette, eger baǵdarlama kishi kólemli, bıraq kóp resurstı (intensiv) talap etetuǵın bolsa, bul waqıtta mısalımızdaǵı minára qurılısı sıyaqlı, sinxron barıstı payda etip baǵdarlamalaw qol keliwi múmkin. Jáne de bunday baǵdarlama kodların oqıw hám túsiniw ańsat boladı.

Keliń mısalımızǵa ózgeris kiriteyik. Endi brigadanı eki toparǵa bólemis, qabatlar hám pútkil minára qurılısına juwapker etip. Qabatlar qurılısına juwapker toparǵa qabattıń tolıqlıǵınsha pitkiziw tapsırmasın tayınlaymıs, al pútkil minára qurılısına juwapker toparǵa bolsa málim bir qabattıń pitiwin kútpesten, keyingi qabat qurılısın baslay beriwdı tayınlaymıs. Yaǵnıy birinshi topar «Birinshi qabat» qurılısn baslaǵan waqıtta, ekinshi topar “beton ústinler” arqalı «Ekinshi qabat» qurılısın baslap jiberedi, «Birinshi qabat» qurılısı tolıqlıǵınsha tamamlanǵannan soń, birinshi topar «Ekinshi qabat» qurılısın dawam ettiredi, al ekinshi topar usı waqıtta «Úshinshi qabat» qurılısın baslap jiberedi.

Endi mısalımız baǵdarmalawdaǵa asinxron barıs konsepciyasına qamtıydı. Bundaǵı bir operaciya tamamlanbastan turıp keyingi operaciyanıń baslanıwına imkán beriwshi “beton ústinler” baǵdarmalawda hár túrli usıllar arqalı ámelge asırıladı. Ulıwma aytqanda hárqanday asinxron operaciya baslanǵan waqıtta, tiykarǵı aǵın heshqashan bloklanbaydı, óytkeni mısalımızdaǵı sıyaqlı, biz aldınnan operaciyalar orınlanıw processin eki toparǵa bólip qoyǵan bolamıs.

Asinxron barıstı payda etiw, tradiciyalıq baǵdarmalawda kóplegen qıyınshılıqlardı payda etedi, mısalı: qosımsha qurallarǵa júginiw; kodtı shiyelenip ketiwi; qáteler menen islesiwdiń qıyınlıǵı; kod basqarılıw tártibin basqarıp bolmaw;

Juwmaqlap aytatuǵın bolsaq asinxron barısta awır operaciyalar operaciya denesi hám operaciya orınlanıw halatı sıyaqlı eki jumısqa bólinedi. Bunda sırtqı orınlanıw barısı operaciya denesi juwmaqlanǵanlıǵın operaciya halatı arqalı bilip baradı. Al sinxron barısta bolsa operaciyaniń ózi bir jumıs bolǵanlıqtan sırtqı orınlanıw barısı bloklanadı. Bular haqqında tolıqraq keyingi bapta bilip alamıs.

Ótkiziwshi funkciyalar úlgisi

Aldınǵı baptaǵı mısalımızda, asinxron barıstı payda etiwshi mexanizm retinde keltirilgen “beton ústinler”di hár qıylı baǵdarlamalaw tilleri hár qıylı tárizli ámelge asıradı. Javascript bunıń ushın ádette **ótkiziwshi funkciya** dep atalıwshı úlgini qollaydı. Bul funkciya shaqırılǵanında, payda etilgen asinxron barıstaǵı operaciya nátiyjesin qaytaradı.

“Toqtań! Biz Javascriptti bir aǵınlı, sinxron baǵdarlamalaw tili dep ótken joqpedik? Qanday qılıp bir aǵımda asinxronlıqtı támiynlew múmkin?”

Álbette, Javascript negizinde bir aǵınlı sinxron til. Bıraq tilge qosımsha imkániyatlar járdeminde biz Javascriptta konkurrentlikti de támiynley alamıs. Mısalı, ES8 alıp kelgen **async/await** sintaksisi til tarıyxında, asinxron baǵdarlamalawdı keyingi dárejege alıp shiqtı. Al asinxronlıq bolsa, ádette ámellerdi kishi-ámellerge bólip, olardı konkurrent júrgizgen halda ámelge asırıladı. Bunda tolıq parallellikke erisilmesede, bunday usıl tilde jazılǵan kodlardı bir qansha márte optimizaciya qıla aladı. Degen menen házirde kópshilik kompyuterlerde ámeller keminde bir neshe processorda júrgizilgenliginen, Javascriptte-da, óziniń túp imkániyatlarınan paydalanǵan halda tolıqlıǵınsha parallel baǵdarlamalaw múmkin. Bular haqqında keyingi baplarda tolıqraq toqtalıp ótemis.

Qullaslap aytatuǵın bolsaq, ótkiziwshi funkciyalar basqa hárqanday asinxron mexanizmlerdiń tiykarın qurawshı blok bolıp, sinxron funkciyalardan parıqlı túrde basqarıwdı basqa funkciyaǵa jóneldiriwshi funkcialardan basqa nárse emes.

Javascriptte funkciyalardi ózgeriwshilerge teńlew, basqa funkciyalarǵa argument esabında ótkiziw, funkciyalardı qaytarıw imkániyatı bolǵanlıqtan, til ótkiziwshi funkciyalar ushın ideal esaplanadı. Onıń ústine Javascript **anıqlanıw oblastı** konsepciyasında qollap quwwatlaydı.

Anıqlanıw oblastı arqalı biz funkciya qaysı oblastta jaratılǵan bolsa, sol oblastqa siltew bere alamıs, bul bizge asinxron operaciya qaysı kontekstte shaqırılǵanlıǵın anıqlaw imkánin beredi.

`onsom.uz/u/closures`

Ótkiziwshi funkciya hám ápiwayi funkciyanıń parqı nede? Keliń usı sorawǵa juwap tabıw, odan qala berdi ótkiziwshi funkciyalar tábiyatın keńnen túsiniw ushın, bulardıń parqına toqtalıp óteyik. Bunıń ushın tómendegi sinxron funkciyaǵa itibar bereyik:

```
function add(a, b) {
  return a + b
}
```

Kórip turǵanıńızday bul ápiwayı funkciya. Nátiyje **return** instrukciyası arqalı sorawshıǵa qaytadı. Endi bul funkciyaǵa tómendegishe ózgeris kiritsek, ótkiziwshi funkciyaǵa iye bolamıs:

```
function callbackFunction(a, b, cb) {
  cb(a + b)
}
```

Bıraq joqarıdaǵı ótkiziwshi funkciya sinxron tábiyatqa iye. Sebebi, callbackFunction ishki cb tolıq tamamlanǵannan keyin ǵana tamamlanadı. Tómendegi kod arqalı bunı tekserip kórsek boladı:

```
console.log('aldın')
callbackFunction(1, 2, result => console.log('Nátiyje: '
+ result)
console.log('keyin')
```

Kod nátiyjede tómendegi tártipte júredi:

```
aldın
Nátiyje: 3
keyin
```

Endi bul funkciyanı asinxron funkciyaǵa aylandırıw ushın **setTimeout** APIsınan tómendegishe paydalanamız:

```
function asyncCallback(a, b, cb) {
  setTimeout(() => cb(a + b), 0)
}
```

Hámde bul funkciyadan aldınǵı kod arqalı paydalanǵanımısta, endi nátiyje tómendegishe ózgeredi:

```
aldın
keyin
Nátiyje: 3
```

Bunı túsindiriw ushın aǵındaǵı ámellerdiń orınlanıw tártibin izbe-izlikte jazıp shıǵayıq:

1. Basqarıw birinshi qatarǵa beriledi, bunda console.log('aldın') sinxron ámel bolǵanlıqtan, ámel tolıqlıǵınsha orınlanıp nátiyjede konsolǵa «aldın» sózi jazıladı hám basqarıw keyingi qatarǵa ótedi

2. asyncCallback(1, 2, cb) funkciyası **setTimeout** járdeminde asinxronlıqtı támiynlegenlikten, bul funkciya denesi tolıq orınlanıwın kútpesten basqarıwdı keyingi qatarǵa ótkizip jiberedi

3. Úshinshi qatardaǵı sinxron console.log('keyin') funkciyası orınlanıp nátiyjede konsolǵa «keyin» sózi jazıladı.

4. Endi basqarıw, aǵında orınlanbaǵan ámeller bolǵanlıqtan qaytadan ekinshi qatarǵa ótedi

5. Aradan 0 millisekund ótkennen keyin basqarıw sinxron cb ge, yaǵınıy «3» parametri menen result => console.log('Nátiyje: ' + result) funkciyasına ótedi, nátiyjede konsolǵa «Nátiyje: 3» teksti jazıladı.

Bul jerde hárdayım «1 + 3» ámeli ótkiziwshi funkciya shaqırılmastan ámelge asıwı támiynlenedi.

Itibár bergen bolsańız funkciyanıń sinxron yáki asinxronlıǵı, funkciya tábiyatına pútkilley tásirin ótkizedi. Sonlıqtan basqarıwdıń biz qálegenimizdey ótiwi ushın qay jerde qanday funkciyadan paydalanıwdı aldınnan anıqlap alıwımıs kerek boladı. Bunıń ushın sinxron hám asinxron ótkiziwshi funkciyalardıń parqın jaqsı biliw talap etiledi.

Bıraq Javascriptte mısallarda kórgenimizdey funkciyanıń tábiyatın anıqlaw bıraz qıyınshılıq tuwdıradı. Anıqlanbaǵan tábiyatlı funkciyalardan paydalanıw bolsa baǵdarlama kodı iske qosılǵanda biz pútkilley kútpegen halatlardı keltirip shıǵaradı. Mısal retinde tómendegi kodtı qarayıq:

```
const result = [1, 2, 3].map(element => element - 1)
console.log(result)
```

Bundaǵı map metodı ótkiziwshi funkciya alǵanı menen sinxron tárizde isleydi hám keyingi qatarǵa ótiwdi tolıq iteraciya juwmaqlanbaǵanınsha bloklap qoyadı. Usıǵan uqsas anıqsızlıqlar baǵdarlama kodınıń hár qanday jerinde ushrasıwı múmkin. Ádette biz itibarsızlıq etip funkciya denesinde kóp paydalanamıs, mısalı tómendegishe usılda:

```
const cache = new Map()

function readGuest(name, cb) {
```

```
  if (cache.has(atı)) {
     cb(cache.get(atı))
  } else {
     const newGuest = new Request(“url” + name)
     if (newGuest) {
       cache.set(name, newGuest)
       cb(newGuest)
     } else {
       cb(new Error(“Miyman maǵlıwmatları tabılmadı”))
     }
  }
}
```

Bir qarastan túsinikli kóringeni menen, funkciya hár qıyli shárıyatta ózin hár qıylı tutadı. Eger mıyman atı, **kesh**te bar bolsa sinxron tárizinde, al joq bolsa asinxron tárizinde júredi. Bunday tábiyatlı funkciyalar baǵdarlamanı biz kútpegen tárizinde júrgiziwshe májbur qıladı. Sonlıqtanda, baǵdarlamalawda bunday funkciyalardan paydalanıw qatań tárizde másláhát berilmeydi.

Baǵdarlamamısta bunday funkciyalardan qashıw ushın, biz funkciyanı ya tuwırı sinxron ya bolmasa, funkciya denesindegi hár qanday shártli úziliwshi orınlarda asinxron islewshi etip jaratıwımız kerek. Jáne de hár qıylı API lardan paydalanıp atırǵanımızda API tábiyatın anıqlap alıwımıs kerek boladı, ádette bul maǵlıwmatlar API hújjetlerinde beriledi.

Ótkiziwshi funkciyalardı durıs qollaw tájriybeleri

Biz baǵdarlama jaratıw waqıtında derlik barlıq jerde ótkiziwshi funkciyalardan bilip-bilmey paydalanamıs. Sonlıqtan bul funkciyalardan paydalanıwda túsiniksizliklerdiń aldın alıw maqsetinde bazı standart qollanıw shártlerin bilip alıwımıs kerek boladı.

- **Hárqanday ótkiziwshi funkciya argument esabında eń aqırǵı bolıp beriledi**

Mısalı, tómendegishe usılda:

```
readFile(filename, [options], cb)
```

- **Hárqanday qátelik hárdayım birinshi keledi**

```
readFile(‘fayl.txt’, ‘utf-8’, (err, data) => {
  if (err) {
     handleError(err)
  } else {
     processData(data)
  }
}
```

- **Qáteliklerdi ótkiziwshi funkciyalar arqalı uslanadı**

Sinxron barısta ádette qátelikler **throw** bayanlaması arqalı shıǵarıladı, bıraq asinxron barısta bul bayanlama menen qáteliklerdi shıǵarıw nadurıs praktika esaplanadı.

Ádette asinxron barısta qátelikler tómendegishe usılda shıǵarıladı:

```
function readFile(cb) {
   // itimallı qátelik júz beriwshi kod
   cb (error, result)
}

function writeFile(cb) {
   readFile((err, result) => {
      if (err) {
         cb (err)
      } else {
         // nátiyje menen islewshi tiykarǵı kod
      }
   })
}
```

Joqarıda biz qátelikti hesh qanday kontekstti buzpastan, “jumsaqlıq” penen uslap keyingi ótkiziwshi funkciyaǵa uzatıp jiberdik. Bul jerde eger kútilmegen qátelik readFile funkciyasında júz beriwi múmkin bolsa, qátelikti try { … } catch { … } bayanlaması menen uslap shaqırıwshı kontekstke ótkiziwimiz-de múmkin (qátelik júz bermegen haldaǵı ótkiziwde birinshi parametr standart null muǵdarına teńlenedi, keyingi parametrlerde nátiyje jaylasadı.)

Ulıwmalastırıp aytatuǵın bolsaq ótkiziwshi funkciyalar bizge tiykarǵı aǵındı bloklamastan ámellerdi júrgiziw imkániyatın beredi. Olar menen baǵdarlama orınlanıw barısında axbarattıń biz belgilep bergen kontekstlerden tuwırı ótiwin támiynlewdi ańsatlastıra alamıs hám kod formallıǵıda bizge qolaylı bir obrazdı sáwleleydi. Basqa tárepten biz kodımızdı elede ózimizge maslap asinxron barıs penen islesiwdi keyingi basqıshqa alıp shıǵıwımıs múmkin. Bunıń ushın biz baǵdarlamalawdaǵı *gúzetiwshi* úlgisinen paydalana alamıs. Tolıqraq keyingi bapta kórip shıǵamıs.

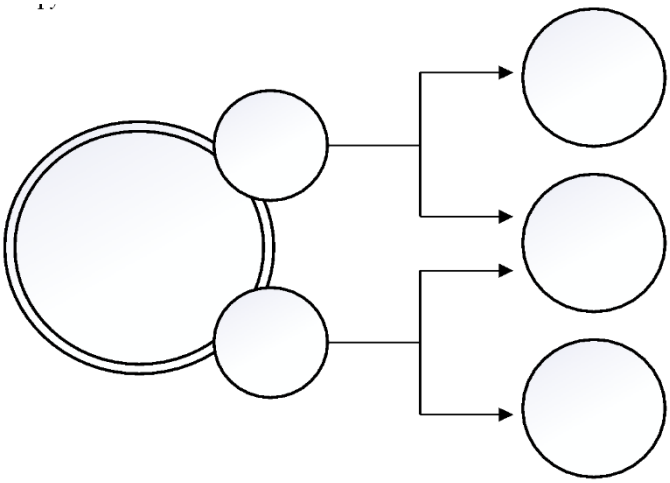
Gúzetiwshi úlgisi

«Jańalıqlar tarqatıwshı telekanal», «Social mediadaǵı paydalanıwshı postı», «Sońǵı hawa-rayı boljawların tarqatıwshı baǵdarlama». Bulardı ne baylanıstırıp turadı? Baylanıstırıwshı nárse axbarattıń dinamik ózgeriwi bolıp, bul ózgeristiń kóplegen qabıl etiwshilerge uzatılıwı. Yaǵınıy, bul jerde axbarat subyekt hám gúzetiwshiler ortasında almasadı. Subyekt bul axbaratlardı saqlawshı oray bolıp, oǵan gúzetiwshiler (qabıl etiwshiler yáki tıńlawshılar) biriktirilgen boladı. Eger subyekt

halatında qandaydır ózgeris júz berse (jańa programma, jańa post yáki boljaw), bul ózgeris haqqında barlıq gúzetiwshilerge bir waqıtta xabar beredi. Gúzetiwshiler bolsa subyektke “aǵza boladı” hám onnan kelgen hárqanday ózgerislerǵa reakciya berediı. Bunday usıldaǵı kommunikaciya baǵdarlamalawda **gúzetiwshi úlgisi** dep ataladı.

Gúzetiwshi úlginiń ótkiziwshi funkciyalardan parqı sonda, gúzetiwshi úlgisinde ózgeristi bir neshe qabıl qılıwshıǵa ótkiziw imkániyatı bar al ótkiziwshi funkciyalarda bolsa bul ádette shınjır tárizli keyingi funkciyaǵa ótedi. Keliń bul haqqında hám ulıwma baǵdarlamalawda, hám NodeJs prizmasında tolıqraq toqtalıp ótsek.

Tradiciyalıq obyektke-jóneldirilgen baǵdarlamalawda, gúzetiwshi úlgisin ámelge asırıw ushın tilge bazı talaplar qoyıladı. Olardan, tildiń konkret klaslarǵa, interfeyslerge hámde anıq ierarxiyaǵa iye bolıwı kereklgin aytsaq boladı. Al NodeJSte bolsa bul ańsatraq. Sebebi úlgi NodeJStıń túp EventEmitter klası (klass events modulınan keledi) arqalı álleqashan ámelge asırılǵan bolıp paydalanıw ushında óte qolaylı interfeys jaratılǵan. Klass bizge birneshe funkciyanı gúzetiwshi (tıńlawshı) túrinde saqlawǵa imkán beredi:



EventEmitter klasınıń tiykarǵı metodları tómendegiler:

- on(waqıya, tıńlawshı): Waqıyaǵa jańa tıńlawshı qosadı.
- once(waqıya, tıńlawshı): Waqıyaǵa, bir márte shaqırılǵannan soń óship ketiwshi jańa tıńlawshı qosadı.
- emit(waqıya, [arg1], [...]): Shaqırılǵanda qosımsha argumentlerdi beriwshi jańa waqıya payda etedi.
- removeListener(waqıya, tıńlawshı): Waqıyadan berilgen tıńlawshını alıp taslaydı.

Keliń bulardı bir mısalda kórip shıǵamıs. Ádette tómendegishe kod arqalı gúzetiwshi úlgisin qamtıwshı klass obyektı jaratıladı:

```
import EventEmitter from 'events'
class WrapperSubject extends EventEmitter {
  constructor() {
     super()
  }

  addChange(data) {
     this.emit('change', data)
     return this
  }
}
const subject = new WrapperSubject()

function firstObserver() {
  console.log('Birinshi gúzetiwshi waqıyani esitti')
}
function secondObserver(data) {
  console.log('Ekinshi gúzetiwshi waqıyani esitti: ',
data)
}
function handleError(err) {
  console.error('Qátelik júz berdi: ', err.message)
}

subject.on('change', firstObserver)
subject.on('change', secondObserver)
subject.on('error', handleError)

subject.addChange('change', 'Malades')
```

Bunda subyektte “change” waqıyası júz bergende oǵan biriktirilgen eki gúzetiwshi funkciyalar shaqırıladı. Axırǵı qatarda bolsa bul waqıyanı ‘Malades’ parametrı menen shaqırdıq, aqıbetinde firstObserver hám secondObserver tıńlawshıları iske qosıladı. Eger kodta qandaydır qátelik júz berse, EventEmitter klası 'error' waqıyası shaqıradı hám sol waqıyanı tıńlawshı orınlanadı, al eger bul waqıyaǵa heshqanday tıńlawshı biriktirilmegen bolsa baǵdarlama qátelik kodı menen toqtatıladı, sonlıqtan durıs tájriybe retinde hárdayım qátelik waqıyasına tıńlawshı funkciya ornatılıwı kerek.

Gúzetiwshi úlgiden paydalanıwdaǵı durıs tájriybeleri

Gúzetiwshi úlgi baǵdarlamashılar ushın qolaylı kod úlgisin jaratıp beredi. Úlgi aldınǵı bapta kórip ótkenimizdey asinxron barıstı payda etiwde kodtı tártipli jazıw imkániyatında jaratadı. Álbette bunday “jeńillikler” óz qıyınshılıqları menen birge keledi. Endi bul qıyınshılıqlardı kórip shıǵayıq.

- **Kereksiz tıńlawshılardı jaratıw yáki málim dáwır ushın ǵana jaratılǵan tıńlawshılar jaramsız axbarattıń qalıp ketiwine soń baǵdarlama ónimliligine zıyanlı tásir tiygizedi**

Gáp sonda málim bir waqıyaǵa tıńlawshı jaratıp atırǵanda tıńlawshınıń anıq qashan alıp taslanıwı (removeListener metodı arqalı) kerekligin kiritiwimiz kerek. Óytkeni tıńlawshı jaratqan leksikalıq oblasttaǵı kereksiz (artıqsha) obyektlerdiń yadtan shıǵarılmawı waqıt ótiwi menen **yad aǵıwı** (baǵdarlama ólsheminiń artıqsha joqarılanıwına) alıp keledi. Mısalı:

```
const hugeData = 'Ósek gápler toplamı ...'
const listener = () => {
  console.log(hugeData)
}
emitter.on('event', listener)
```

Bul jerdegi hugeData ózgeriwshisine listener tıńlawshısınıń ishinde siltew berilgenlikten tıńlawshı alıp taslanaman degenshe yáki emitter **shıǵındı kollektorı** járdeminde óshirilemen degenge (bul tek qana subyektke aktiv siltewler joǵalǵanda ǵana ámelge asadı) shekem yadta saqlanadı. Ádette baǵdarlamada bunday defektlerdiń aldın alıw maqsetinde EventEmitter klası, málim bir waqıyanı tıńlawshılar sanı onnan asıp ketse, eskertiw beredi, yáki biz ózimiz setMaxListeners metodı járdeminde bul limitti sazlasaqta boladı.

- **Sinxron ba Asinxronba?**

Ótkiziwshi funkciyalar sıyaqlı gúzetiwshilerdi de sinxron hám asinxron tárizinde shaqırıwımız múmkin. Bul tıńlawshılardıń shaqırılıs usılına baylanıslı, mısalı sinxron funkciya yáki asinxron funkciya degen sıyaqlı. Bıraq dıqqatlı bolıwımız kerek tárepi, bir subyektte bul ekewin aralastırıp jibermewimiz kerekligi (ótkiziwshi funkciyalardaǵıday).

Gáp sonda, waqıyanı asinxron usılda shaqırǵannan keyinde, bul waqıyaǵa jańa tıńlawshını aǵza qılıp úlgere alamıs. Sebeb, Javascriptte waqıyalar, keyingi waqıyalar ciklı aylanımına shekem shaqırılmawı támiynlenedi (tolıqraq keyingi baplarda).

Aldıńǵı mısalda, addChange metodı tolıq sinxron islegenlikten “this.emit('change', data)” qatarı waqıyalar ciklındaǵa haqıyıy gezeginde orınlanadı, bul bolsa change waqıyasına tıńlawshı biriktirmesten aldın waqıyanı shaqırıw imkániyatın bermeydı. Al eger change waqıyasın asinxron barısqa sala alsaq, bizde waqıyanı shaqırǵannan keyinde oǵan tıńlawshılardı aǵza qılıw imkániyatına iye bolamıs, mıaslı tómendegishe usılda:

```
import EventEmitter from 'events'
class WrapperSubject extends EventEmitter {
```

```
  constructor() {
     super()
  }
  addChange(data) {
     setTimeout(() => {
         this.emit('change', data)
     }, 0)
     return this
  }
}
```

Bunda addChange metodınnıń setTimeout APIsı járdeminde asinxron júriwin támiynledik, sonlıqtanda, addChange shaqırılǵannan keyinde metod ishindegi change waqıyasında tıńlawshı biriktire alamıs:

```
const subject = new WrapperSubject()

subject.addChange('Malades')
subject.on('change', (data) => {
  console.log('Gúzetiwshi waqıyani esitti: ', data)
})
```

Álbette asinxron yáki sinxron shaqırıw bul hárkimniń óz tanlawı, degen menen EventEmitter klasınıń tábiyatı asinxron waqıyalar menen birikken. Sonlıqtan asinxron waqıyalar menen islew kóbirek usınıs beriledi. Haslan eger waqıya sinxron shaqırılǵan bolsa, bul kóbinshe kodımız ushın EventEmitter klası shárt emesliginiń belgisi esaplanadı.

EventEmitter hám Ótkiziwshi funkciyalar

Asinxron API jaratıp atırǵandaǵı tiykarǵı dilemma EventEmitter yáki ótkiziwshi funkciyalardı tańlawda jatadı. Qaysı birin tańlap baǵdarlama kodın jazıw hárkimniń óz qálewinde yáki mashqalaǵa baylanıslı boladı. Bular arasındaǵı ulıwmalıq ózgeshelik semantik bolıp: kóbine ótkiziwshi funkciyalar nátiyjeni asinxron usılda qaytarıw ushın, al gúzetiwshiler málim bir waqıya júz bergende baǵdarlamanıń funkcional blokları arasında kommunikaciyanı támiynlew maqsetinde paydalanıladı. Bıraq qaysı birin tańlaǵanda da, hár eki úlgi arqali bir nátiyjege erisse boladı. Mısalı tómendegi gúzetiwshi úlgisin paydalanǵan halda jaratılǵan kod:

```
import { EventEmitter } from 'events'
function observerWay() {
  const subject = new EventEmitter()
  setTimeout(() => subject.emit('finish', 'tamam'), 0)
  return subject
}
observerWay().on('finish', console.log)
```

Bul kodtı tómendegishe ótkiziwshi funkciyalar menende jazsa boladı:

```
function callbackWay(cb) {
  setTimeout(() => cb(null, 'tamam'), 0)
}

callbackWay((err, msg) => console.log(msg))
```

Eki funkciyada funkcionallıǵı jaǵınan ekvivalent dep aytsaq boladı. Bıraq birinshi kodta, subyekt hám gúzetiwshi erkin jalǵanǵan bolıp bizge modullılıq hám kodtan qaytadan paydalanıw imkániyatın beredi. Al ekinshi usılda funkciyalar bir-birine tuwırıdan-tuwırı jalǵanǵan bolıp, bizge basqarıwdı (kod júriwin) anıq ashıp bere aladı.

Eger eki úlginida birge-bir salıstıratuǵın bolsaq tómendegishe kesteni alsaq boladı:

Qásiyet	EventEmitter	Ókiziwshi funkciya
Kommunikaciya modeli	Birge-kóp	Birge-bir
Ajratılıw	Waqıya subyektı hám tıńlawshı obyekt bir-biri menen erkin baylanısadı	Shaqırıwshı hám ótkiziwshi funkciyalar bir-biri menen tıǵız baylanısadı
Bir neshe tıńlawshılardı júrgiziw	Bar	Joq
Tártip	Tıńlawshılar jaratılǵan waqıttaǵı tártipte ámelge asırıladı	Orınlanıw tártibi funkciya shaqırılıw tártibine baylanıslı
Quramalılıq	Kóp waqıya hám tıńlawshılardı basqarıw qıyın	Kishi ámeller ushın ańsat hám qolaylı
Qáteliklerdi tuwırılaw	Kóp waqıya hám tıńlawshılar ushın qıyın	Kishi operaciyalardaǵı kod orınlanıw tártibin anıqlaw ańsatraq
Shekleniwler	Jaramsız axbarat defektı, shekli masshtablılıq	Asinxron batpaq, kodtan qayta paydalanıwdıń shekliligi, kútilmegen basqarıw tártip
Qolaylı	Bir neshe komponentlerdi xabarlaw, erkin kommunikaciya, málim bir axbarattı basqarıw ushın	Ańsat hám bir mártelik asinxron operaciyalar ushın

Bul qásiyetlerdiń tásiri baǵdarlamamıs qanshelli úlkeygen sayın sezile baslaydı, sonlıqtanda qanday sheshimlerden paydalanıwdı kod jazbastan aldın anıqlap alǵan maqul.

Juwmaqlap aytatuǵın bolsaq, gúzetiwshi bolsın yáki ótkiziwshi funkciyalar bolsın bular algoritmdı kodta qanday usılda jaratıwdıń úlgileriǵana. Axırı qanday sheshimdi shıǵarıw báribir hárkimniń óz talǵamına kiredi. Tek itibar beriwimiz shárt bolǵan jeri, mashqala hámde sheshimler arasındaǵı altın teńlikti tawıp alıwımızda.

Asinxron barıstı ótkiziwshi funkciyalar arqalı basqarıw

Jabayı ótkiziwshi funkciyalar, kóbinshe baǵdarlamalawda túsiniksiz, baqlawsız hám aldınnan aytıp bolmaytuǵın tábiyatlı ótkiziwshi funkcialardı usılayınsha ataymıs. Bunday tábiyat ádette tildegi ótkiziwshi funkciyalardıń qanday júriwin bilmew aqıbetinde kelip shıǵadı. Bunday kod bólekleri baǵdarlamanıń hárqanday jerinde ushırasıwı múmkin. Mısalı, fayllar toplamında operaciyalar ámelge asırǵanda, ámeller iz-izbeliginde yáki konflikt keltirip shıǵarıwshı operaciyalardı orınlaǵanda.

Negizi Javascriptte asinxron kodtıń basqarıwın joǵaltıw óte ańsat. Kóbinshe basqarıwın joǵaltıw, kerek bolmaǵan jerde funkciyalardı jaratıw yáki keńeytiw menen baylanıslı. Bunday funkciyalar waqıt ótiwi menen kodtıń vertikal emes gorizontal keńeyiwine alıpp keledi. Gorizontal keńeyiw bolsa hárqashanda kodtıń biz ushın qolaysız (oqıw, túsiniw hám qáteliklerdi anıqlap tuwırılaw ushın) qılıp qoyadı. Bul ádette **asinxron batpaq** dep atalıwshı funkcional bloklarda kórinedi.

Haslan asinxron batpaq kodta anıqlanıw oblastlardıń konflikt bolatuǵın dárejede kópligi hám ótkiziwshi funkciyalardı ishpe-ish shaqıra beriw nátiyjesinde payda boladı. Bul baǵdarlamalawdaǵı eń tiykarǵı nadurıs úlgilerdiń bir esaplanadı. Bunday úlginiń ózine tán strukturası tómendegishe:

```
asyncFoo(err => {
  asyncBar(err => {
     asyncFooBar(err => {
     //...
  })
}
```

Kórgenimizdey bul óziniń shuqır ishke tartılıwı nátiyjesinde piramidanı esletedi, sonlıqtanda bunday kod bólekleri **apatiya piramidası** depte ataladı.

Bunday kodlardıń tiykarǵı nuqsanlarınan biri, oqıwdıń qıyınlıǵı. Ishke tartılıwdıń sonshelli shuqırlıǵınan funkciyalardıń qay jerde baslanıp, qay jerde tamamlanıwın biliw qıyınshılıq tuwdıradı. Jáne bir nuqsan ózgeriwshilerdiń atları. Biz kóbinshe, uqsas funkciyalı muǵdardı qabıl etiwshi ózgeriwshilerge birdey at qoyamıs. Mıaslımızdaǵı `err` ózgeriwshisi usı nuqsandı kórsetip beredi. Bazı baǵdarlamashılar bul nuqsandı hárbir qátelikke tákirarlanbas at qoyıw menen sheshiwge urınadı. Mısalı, `err`, `err1`, `err2`, `error` sıyaqlı. Hátte usılay hárqıylı at qoysaqta apatiyadan

qutıla almaymıs. Óytkeni bunıń aqıbeti anıqsızlıqqa barıp taqaladı. Odan qala berdi, anıqlanıw oblastları yadta orın alıw úlesiniń kishi bólegin qabıl etedi. Sonlıqtanda anıqlanıw oblastları arqalı jaratılǵan yadtıń aǵıwın anıqlaw ańsat bolmay qaladı. Negizinde biz, shıǵındı kollektorınan aman qalǵan hárbir aktiv anıqlanıw oblastınan siltew berilgen, hárqanday kontekstti umıtpawımıs kerek.

Asinxron batpaq Javascripttegi ótkiziwshi funkciyalardan paydalanǵanda jolıǵatuǵın jalǵız mashqala emes, álbette. Odan basqa, bir neshe asinxron ámellerdiń orınlanıw barısın basqarıwda da qıyınshılıqlar ushıraydı. Mısalı, qanday da bir kollekciyanıń hárbir elementı ushın asinxron operaciyanı ámelge asırıw, bul kóringenindey ańsat emes, sonlıqtan da arnawlı rekursiyaǵa uqsas texnikanı talap etedi. Qanday texnika ekenligin bırazdan kórip shıǵamıs. Házir bolsa, joqarıdaǵıday apatiyalıq kodlardan qorǵanıw maqsetinde ótkiziwshi funkciyalar menen islegende ámel qılıwımız kerek bolǵan bazı “tárbiyalıq qaǵıyda”lardı anıqlap alamıs.

Ótkiziwshi funkciyalardan paydalanıwshi kod strukturasın jaqsılaw maqsetinde tómende bazı principler keltirsek boladı:

- **“Erte shıǵıw principi”**

Princip atınanda bilingenindey, bloktan ilájı barınsha tezirek shıǵıwdı ańlatadı. Shıǵıw jaǵdayǵa qarap, return, break, continue bayanlamaları arqalı ámelge asırıladı. Mısalı tómendegishe kod:

```
if (err) {
  callback(err)
} else {
  // tiykarǵı algoritm
}
```

Bul kodtı tómendegishe usılda “tárbiyalaw”ımız múmkin:

```
if (err) {
 return callback(err)
}
// tiykarǵı algoritm
```

Kórgenimizdey biz ishke tartılıwdıń aldın aldıq. Bunday usıl baǵdarlamalawda **erte shıǵıw principi** dep ataladı.

Bul jerde itibar beriwimiz kerek, ótkiziwshi funkciyalar menen isleskende, kóbinshe bloktan yáki funkciyadan shıǵıwdı umıtıp ketemis. Bul bolsa shárt penen ótkiziwshi funkciyanı shaqırıp, shártsiz bólektiń birge orınlanıwına alıp keledi. Mısal:

```
if (err) {
 callback(err)
}
```

```
// tiykarǵı algoritm
```

Bunda tiykarǵı algoritm qátelik bolsa da bolmasada islep ketedi. Sonıń ushın, artıqsha quramalılasıwdıń aldın alıw maqsetinde, nátiyjeni funkciya shaqırıwshıǵa qaytarıwshı etip mınanday túrde jazsaq boladı:

```
return callback(err)
```

Yáki, nátiyjeni qaldırıp ketiwshi:

```
callback(err)
return
```

- **Anonim ótkiziwshi funkciyalardan iláji barınsha paydalanbaw**

Ádette anonim ótkiziwshi funkciyalardan kóp paydalanamıs, sonıń ushında kóp waqıtımızdı kodtı tuwırılaw menen bánt bolıp ótkizemis. Sebebin mına kod penen bilsek boladı:

```
function readFile(handleContentCallback) {
  try {
     handleContentCallback()
  } catch (error) {
     console.error(error.stack)
  }
}

readFile(function handleContent() {
  console.log("Ótkiziwshi funkciya orınlandı")
  throw new Error("Qátelik júz berdi!")
})
```

Kod iske túsirilgende konsolda tómendegishe nátiyje payda boladı:

```
Ótkiziwshi funkciya orınlandı
Error: Qátelik júz berdi!
   at handleContent (C:\Users\begzat\book\first-sec\12.js:10:9)
   at readFile (C:\Users\begzat\book\first-sec\12.js:3:5)
```

Itibar bergen bolsańız stek izinde qátelik eń birinshi kóringen funkciya handleContent atı biz ushın qolaylı tárizde berilgen. Eger ótkiziwshi funkciyanı anonim tárizde qaldırǵanımızda, stek izinde tek qana qátelik turǵan fayl jolın kórseter edi.

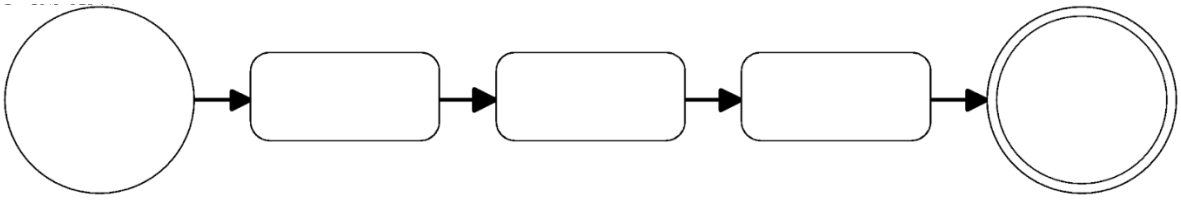
Ulıwmalastırıp aytatuǵın bolsaq, asinxron barıstaǵı ótkiziwshi funkciyalardı jaqsı jolǵa qoyıw baǵdarlamada, potencial qáteliklerdiń aldın alıw, baǵdarlamanıń ónimliligin asırıw, baǵdarlama tezligin optimal halatqa keltiriw ushın baslı faktor

esaplanadı hámde bizdiń 40-50% waqıtımızdı únemleydi. Odan qala berdi anıq bir struktura hám tártipke iye kodtıń tazalıǵıda joqarı boladı. Sonlıqtanda, kod jazıwdı baslamastan aldın, bizdegi mashqala hám jaǵdaylarǵa mas keliwshi usıllardı anıqlap alıwımız kerek. Keliń bunday usıllardıń bazılarına toqtalıp óteyik. Sonda biz kóbinshe ushıraytuǵın mashqalalarǵa qanday qılıp optimal sheshim beriwge bolatuǵınlıǵın bilip alamıs.

Birinshi bolıp toqtalatuǵın usıl **izbe-iz orınlanıw** dep ataladı. Bul usılda ámeller izbe-izlikte, birinen keyin ekinshisi orınlanadı. Bunda ámellerdiń izbe-izliktegi tártibi áhmiyetli bolıp, aldınǵı ámeldiń nátiyjesi keyingi ámeldiń orınlanıwına tásir qılıwı múmkin, mısalı tómendegi kodta kórsek boladı:

```
let x = 5       // 1-Ámel
let y = 10      // 2-Ámel
let sum = x + y // 4-Ámel
```

Bul kod processorda tómendegishe tártipte orınlanadı (bul jerde kishi-dárejeli operaciyalardı ulıwmalastırǵan halda keltirildi):



Izbe-iz orınlanıw usılınıń da bir-neshe haldaǵı implimentaciyaları bar, olardan keń tarqalǵanları:

- *Ápiwayı*, bir-biri menen ulıwma axbarattı almaspaytuǵın ámeller toplamın orınlawshı
- *Shınjırlı*, aldınǵı ámel orınlanıwı keyingi ámelge tásir etiwshi
- *Iterativ*, ámeller toplamınıń hárbir elementi ústinde málim bir instrukciyanı orınlawshı

Bul jerdegi ápiwayi hal, baǵdarlamalaw tilleriniń negizgi sinxron tábiyatı esaplanadı. Al keyingi ekewi kóbinese asinxron barısta effektiv qollanıladı. Sinxron tek qana konkurrent esaplawdı simulyaciya qılıw ǵana múmkin, haslında bári-bir sinxron orınlana beredi. Sonlıqtanda biz bulardıń asinxron haldaǵı implimentaciyaların kórip shıǵamıs.

Shinjırlı haldaǵı izbe-iz asinxron orınlanıwdı mına kod penen ulıwmalastırsaq boladı (asinxronlıqtı támiynlew maqsetinde setTimeout APIsınan paydalanıldı, ádette bunıń ornına asinxron funkciya qoyıladı):

```
function task1(cb1) {
  setTimeout(() => task2(cb1), 0)
```

```
}
function task2(cb2) {
  setTimeout(() => cb2(), 0)
}

task1(function () {
  console.log("Ekiwide orınlandı")
})
```

Bunda biz hárbir keyingi funkciyanı qoldan jazıp shaqırıwmızǵa tuwrı keledi. Bıraq qatań yaǵınıy biz bergen tártip saqlanadı. Hár qanday qáteliklerdi de qoldan hár bir ótkiziwshi funkciyanıń ishine jazıwımız kerek boladı (ádette qátelikti tutıw algoritmi kópshilik orında birdey keledi). Bul álbette ótiw waqıtında bizge kóbirek basqarıwdı bergeni menen aqıbette bir algoritmniń tákirar jazılıwına sebebshi boladı. Áyne usı kemshilikti keyingi izbe-iz orınlanıw usılı toltıradı. Bul iterativ izbe-izlik bolıp, toplamdaǵı barlıq elementler ústinen júrip shıǵıw menen isleydi. Tómende iterativ izbe-izliktiń ulıwmalastırılǵan forması berilgen:

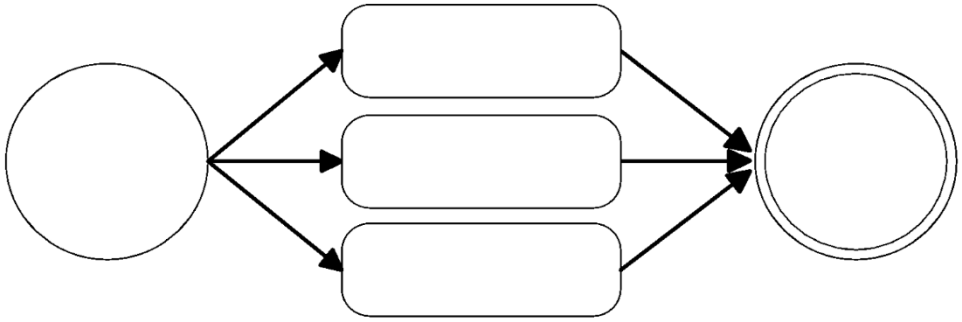
```
const tasks = [
  cb1 => setTimeout(cb1, 2000),
  cb2 => setTimeout(cb2, 1000),
  cb3 => setTimeout(cb3, 3000)
]

function finish () { /* operaciyalar juwmaqlandı */ }

function iterate (index) {
  if (index === tasks.length) return finish()

  const task = tasks[index]
  task(() => iterate(index + 1))
}

iterate(0)
```

Iterativ izbe-izlik joqarıdaǵı koddaǵı sıyaqlı kópshilik qolaylıqlardı beredi, olardan ámeller ushın ulıwma bir qáteliklerdi uslawshı algoritmdi qollansaq boladı, qala berdi ámeller qánshelli kóp waqıt dawam etsede tártip saqlanıp qaladı. Jáne de tasks toplamınıń ornına muǵdarlardı berip iteraciya ámelge asırsaqta boladı. Bıraq, itibarlı bolıwımız kerek jeri, bunday túrdegi algoritmler eger ámeller sinxron operatciyadan ibárat bolsa shuqır rekursiyaǵa ushraydı.

Endi bolsa ekinshi bolıp toqtalatuǵın usılımız parallel orınlanıwdı kórip shıǵamıs.

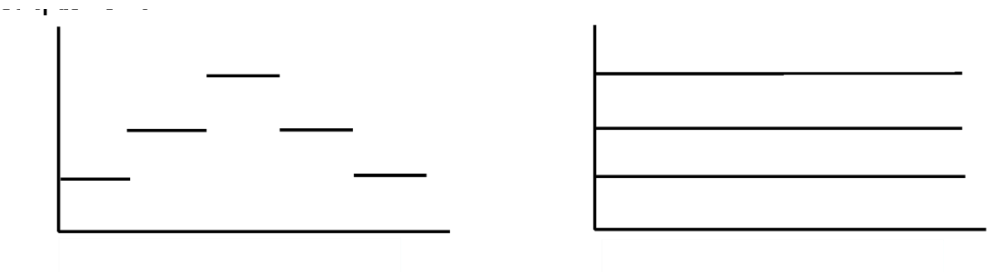
Parallel orınlanıw ámellerdiń orınlanıw tártibi áhmiyetli bolmaǵan halda barlıq ámellerdi shaqırıwdı ańlatadı. Bunda, kod barlıq ámeller orınlanǵannan keyin ǵana

tamamlanadı. Mısalı, tómendegishe tártipte ámeller orınlanadı hám kod juwmaqlanadı:



Túsiniwimiz kerek, Javascript bir aǵında isleydi. Bir aǵında bolsa tolıq parallellikke erisip bolmaydı (álbette egerde bir neshe processorlar ústinde gibrid aǵındı jaratpasańız) bálkim konkurrentlikke ǵana erise alamıs. Biz ańsatraq túsindiriliwi maqsetinde parallellik sózin qollanamıs. Bıraq umıtpań, ádette Javascriptta eki asinxron operaciya bir waqıtta orınlanbaydı, bálkim operaciya kernelge berilip jiberiledi. Operaciya orınlanǵanında kerneldegi aǵın Javascript aǵınına signal beredi hám usı signal kelgende bekitilgen ótkiziwishi funkciya signalda kelgen nátiyje menen iske túsip ketedi (tolıqraq keyingi bapta toqtalamıs).

Tómende Javascript qalayınsha asinxron operaciyalardı parallel júrgiziwidi imitaciya qılıwın kórsek boladı:



Bunda sızıqlar operaciyalar bolıp haqıyqıy parallellikte olar bir waqıtta birdey orınlanadı, al Javascript bir aǵınlı bolǵanlıqtan, túp mánide operaciyalar orınlanıw ushın kútiwge májbur boladı.

Parallellikti tómendegishe usılda kodta keltirsekte boladı:

```
const tasks = [
  cb1 => setTimeout(cb1, 3000),
  cb2 => setTimeout(cb2, 1000),
  cb3 => setTimeout(cb3, 2000)
]

function finish () { /* operaciyalar juwmaqlandı */ }
```

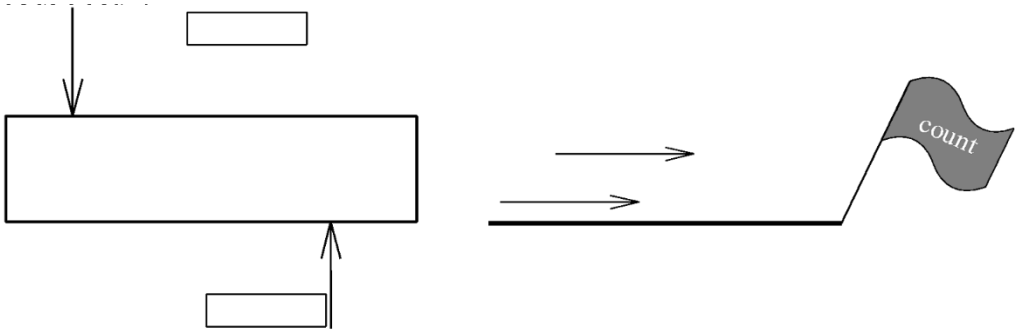
```
let completed = 0
tasks.forEach(task => {
  task(() => {
     if (++completed === tasks.length) return finish()
  })
})
```

Itibar bergen bolsańız bunda toplamda sinxron júrip shıqtıq, bul bizge ámellerde birdey shaqırıw imkániyatın beredi. Sońında, yaǵınıy barlıq ámellerdiń sinxron denesi orınlanǵannan keyin, qaysı biriniń asinxron denesi eń aldın orınlanıp bolǵan bolsa sol birinshi qaytadı jáne usılay dawam etedi, eń aqırında bolsa juwmaqlawshı ótkiziwshi funkciya orınlanadı.

Endi bolsa itibardı jáne bir eń áhmiyetli jerge qaratayıq. Bul kópshlik asinxron kodta jiyi ushrasıwshı úlken mashqala. Talas halatı, parallellizmniń belinen jıǵıwshı mashqala. Ol ne ózi hám qashan júz beredi? Bul sorawlarǵa juwap beriw ushın tolıqraq toqtalmasaq bolmaydı.

Demek, **talas halat** ádette bir neshe parallel islewshi processler, aǵınlar yáki ápiwayı ámeller bir waqıtta, bir resursqa kiriwge urınǵan waqıtta júz beredi. Bul kernelde konfliktke alıp keledi hám kóbinshe axbarattıń buzılıwı yáki jaman halatlarda sistemanıń isten shıǵıwınada sebebshi bolad aladı. Bıraq Javascript bir aǵınlı bolǵanlıqtan jáne de parallel baǵdarlamalaw koncepsiyasında tolıqlayın qollamaǵanlıqtan bul mashqala ádette júz bermeydi (ádette, sebebi keyingi baplarda).

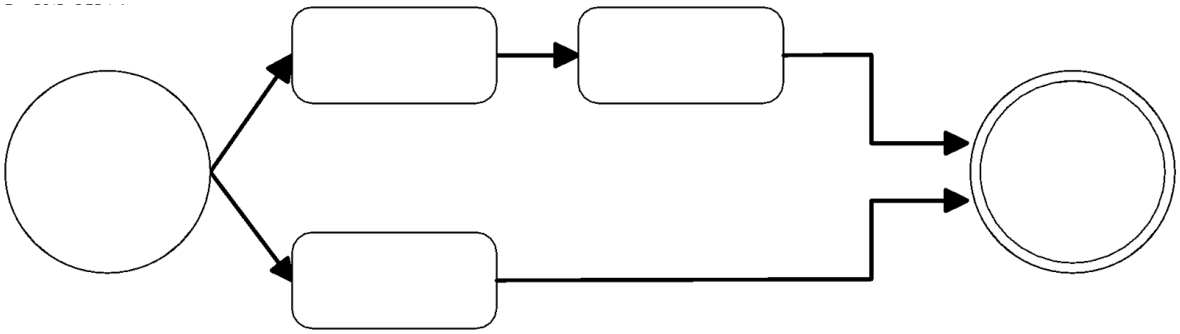
Bıraq, bıraq, bıraq. Biz insánlar bolǵanlıǵımızdan álbette mashqala jarata alamıs. Hátte bir aǵınlı baǵdarlamalawda talas halattı jaratıwımız múmkin. Qanday? Óte ańsat, egerde biz ámellerdi tuwrı sinxronlawdı jolǵa qoymasaq. Mısalı, eki ámel asinxron shaqırılǵan hám ekewinde da global `count` ózgeriwshisi muǵdarın ózgertiwshi operaciya bar dep alayıq. Bunda birinshi ámel `count`tıń baslanǵısh muǵdarı menen al ekinshi ámel `count`tıń birinshi ámel tásir qılǵannan keyingi muǵdarı menen islewi kerek bolsın. Egerde ámeller basqa-basqa aǵınlarda jaylasqanında bul haqıyqıy katastrofaǵa alıp keler edi, degen menen hátte bir aǵında jaylasqanda da konflikt kelip shıǵıwı múmkin. Bul konflikt ádette ámellerdiń shaqırılıw hám nátiyjeniń orınlanǵanlıǵı arasındaǵı keshigiw menen baylanıslı. Óytkeni biz bilmeymis, birinshi ámeldiń orınlanǵanlıǵı haqqındaǵı signal anıq ekinshi ámel orınlanbasınan aldın waqıyalar ciklına jetkiziliwin (sebep, operacion sistemadaǵı biz basqara almaytuǵın tásirler yáki ózimiz jasaǵan tártiptiń buzılıwı múmkin). Onıń ústine ámellerdegi `count++` hám `count--` operaciyaları atomar bolmaǵanlıǵı ushın bul operaciyalar orınlanıw waqıtında úshinshi tárepten úziliwi múmkin aqıbette resurstıń korrupciyalanıwına alıp keledi. Tómende usı mısaldıń illustraciyasın kórsek boladı:



Bunday mashqala birneshe aǵınlar menen islewshi baǵdarlamalaw tillerinde jiyi ushrasadı, ádette qulıplar, muteks hám semafor sıyaqlı konstrukciyalardan paydalanıladı. Ulıwma aytqanda hárqanday algoritmlerde da bul mashqala ushrasıwı múmkin sonlıqtan da parallel ámeller menen isleskende itibarlı bolıwımız kerek.

Sońǵı, úshinshi bolıp toqtalatuǵın usılımız **shekli parallel orınlanıw** dep ataladı. Bunı parallel orınlanıwdıń ayrıqsha halı retinde qarasaq boladı. Óytkeni bul usıldıń atıda, dúziliside parallel orınlanıwdı tolıqtıradı desek boladı.

Parallel orınlanıwdıń eń ázzi jerlerinen biri ol, eger intensiv ámeller kóbeyip ketse bul baǵdarlamanıń tezligine úlken tásir kórsetedi. Al usı mashqalanı shekli parallel orınlanıw sheshe aladı, yaǵınıy bunda biz parallel orınlanıwshı ámellerdiń maksimum sanın belgilep bergen bolamıs, mısalı tómende diagramma usı kórinisti beredi:



Bunda biz eń basta eki ámeldi shaqırıp alamıs (maksimum parallel orınlanıwshı ámeller sanı eki dep belgilengen waqıtta). Ekewiniń birewi tamamlanǵannan keyin barıp úshinshi ámel shaqırıladı.

Tómende kodta ulıwmalastırılǵan forması keltirilgen:

```
const tasks = [
  cb => setTimeout(cb, 2000),
  cb => setTimeout(cb, 1000),
  cb => setTimeout(cb, 3000)
]

const MAX_CONCURRENCY = 2
let running = 0
```

```
let completed = 0
let index = 0

function finish() { /* operaciyalar juwmaqlandı */ }

function next() {
  while (running < MAX_CONCURRENCY && index <
tasks.length) {
    const task = tasks[index++]
    task(() => {
      if (++completed === tasks.length) return finish()
      running--
      next()
    })
    running++
  }
}

next()
```

Bunda next funkciyası keyingi ámeldi shaqırıw ushın qollanıladı. Ondaǵı cikl (MAX_CONCURRENCY – running) dana ámel shaqıradı. Shaqırılǵanlardan qaysıdırları orınlanǵannan keyin ótkiziwshi funkciya iske túsedi. Ol bolsa keyingi next`tı qaytadan shaqıradı. Usılayınsha eń sońǵı ámelge shekem dawam etedi hám sońında finish funkciyası orınlanadı. Egerde ámel júrip atırǵanda qandayda bir qátelik shıqsa onı, ámelden cb(err) kórinisinde shıǵarıp, ótkiziwshi funkciyada task(err) => { ... kórinisinde tutıp alsaq ta boladı.

Juwmaqlap aytatuǵın bolsaq qay waqıtta qanday orınlanıw tártibinen paydalanıw bul álbette hárkimniń óz tańlawı. Sebebi, negizinde hár qanday mashqalaǵa hár túrli jollar menen sheshim berse boladı. Degen menen biz ilaji barınsha sınaqtan ótken jollardı qollawǵa háreket etiwimiz kerek.

Endi bolsa Javascripttiń baǵdarlamalawshılardı ózine altınday tartıwshı qásiyeti bolǵan, úziliwshi wádeler hám async/await sintaksisi asinxron barıstı qanday qılıp ańsatlıq penen basqarıwı haqqında toqtalamıs.

Úziliwshi wádeler hám Async/Await

Aldınǵı bapta kórgenimistey, asinxron barıstı ótkiziwshi funkciyalar járdeminde basqarıw bir qansha tájriybe talap etedi. Sonlıqtan, Javascript baǵdarlamashıları tilde jańa úyreniwshilerge ele de qolaylatıw maqsetinde, bir qansha sheshimler usında. Olardan bir úziliwishi wádeler ekisnshisi async/await sintaksisi bolıp, bular baǵdarlamalawshılardı Javascriptke tartıwshı jańa tolqın jarattı desek boladı. Óytkeni úziliwshi wádeler hám async/await, aldınǵı ótkiziwshi funkciyalar keltirip shıǵarǵan bazı “tesik”lerdi jamay aldı hám tilde asinxron operaciyalardı basqarıw

elede ańsatlastı. Keliń bulardı izbe-iz úyrenip baslayıq. Demek birinshi úziliwshi wádeler.

Úziliwshi wádeler

Keliń mınanday analogiya jasayıq. Siz klient bolıp bir restoranǵa keldińiz hám oficiantqa jaqtırǵan taǵamıńızdı ayttıńıs. Oficiant sizge taǵam atı hám sizdiń stolıńız nomeri jazılǵan token berdi. Bul tokendi restorannıń, taǵamdı sizge islep beriwi haqqındaǵı «**wáde**»si desek boladı. Siz bilmeysiz bul wáde orınlanadıma ya joq, yáki qansha waqıtta orınlanıwı haqqında. Basqa tárepten siz token alǵannan keyin, taǵam stolıńızǵa kelemen degenshe basqa islerdi islewińiz múmkin, mısalı kitap oqıwıńız yáki doslarıńız benen sóylesip otırıwıńız múmkin. Taǵam tayın bolǵanında oficiant sizge taǵamdı alıp keledi hám restoran «**wáde**»sin orınlaǵan boladı.

Javascripttegi úziliwshi wáde usı tokenge uqsaydı. Bul asinxron operaciyanıń tamamlǵanlıǵın bildiriwshi obyekt. Yaǵınıy, asinxron operaciya hám operaciya nátiyjesi menen islesiwshi funkciyanıń arasındaǵı interfeys desekte boladı. Úziliwshi wáde nátiyjesi menen islesiwshi funkciyanı wáde jaratılıp atırǵanında jalǵap qoysańız boladı, bul ótkiziwshi funkciya túrinde isleydi. Egerde úziliwshi wáde jaratılıwı menen birden (sinxron) shaqırılsa, ol *kútiw* halatındaǵı obyektti qaytaradı. Sebebi wáde asinxron operaciyanıń tamamlanıwın kútip atırǵan boladı.

Endi usı sózlerdi, restoran analogiyamıstan paydalanıp psevdokod túrinde jazsaq boladı:

```
funkciya wádeBer(operaciya):
  qaytar jańa Wáde(operaciya)

funkciya taǵamTayarla (stolǵaApar, menejergeXabarBer):
  eger (taǵam ushın kerekli zatlar bolsa):
     tayarlandı(taǵam)
  bolmasa:
     tayarlanbadı(sebep: kerekli zatlar joq)

wádeBer(taǵamTayarla)
  .orınlansa(stolǵaÁkel)
  .orınlanbasa(menejerdiShaqır)
```

Bunda wáde obyekti birinshi wádeBer funkciyası arqalı jaratılıp alınadı. Keyin bul obyekt haqıyqıy asinxron operaciya (taǵamTayarla) hám onı qabıl etiwshi (stolǵaÁkel) arasında baylanıstırıwshı qural bolıp xızmet etedi. Óz náwbetinde, taǵamTayarla funkciyası ózine eki tayarlandı hám tayarlanbadı ótkiziwshi funkciyaların parametr qılıp aladı. Eger taǵam (nátiyje) ushın kerekli zatlar *bar* bolsa taǵam tayarlanadı hám tayarlandı funkciyasına argument esabında beriledi. Óz náwbetinde, wáde obyekti orınlansa interfeysi arqalı tayarlandı funkciyasınan qaytqan taǵamdı, stolǵaÁkel qabıllawshısına parametr qılıp berip jiberedi. Al eger

taǵam ushın kerekli zatlar *joq* bolsa tayarlanbadı funkciyasına argument esabında qátelik obyekti (sebep) beriledi. Óz náwbetinde, wáde obyekti orınlanbasa interfeysi arqalı tayarlanbadı funkciyasınan qaytqan qátelik obyektin, menejerdiShaqır qabıllawshısına parametr qılıp berip jiberedi. Eger wádeBer funkciyası tayarlandı yáki tayarlanbadı funkciyaları shaqırılmastan burın shaqırılatuǵın bolsa, *kútiw* halatındaǵı wáde obyektin qaytaradı.

Psevdokodta kóringenindey, úziliwshi wáde qabıllawshı hámde orınlawshı arasında kópir wazıypasınǵana orınlaydı. Analogiyada klient restoranǵan kelgeninde ol restorannıń wádeBer(taǵamTayarla) psevdokodı arqalı restoran menen baylanısadı hámde nátiyjeni qabıl etiwshi óz funkciyaların wáde obyekti arqalı biriktirip qoyadı.

Endi bolsa keliń bunı Javascript tilinde qalayınsha túsindiriw múmkinligin kórip shıǵayıq.

Úziliwshi wádeler – asinxron operaciya nátiyjesin (yáki qátelikti) ózinde saqlawshı obyektler bolıp, eger operaciya tamamlanbaǵan bolsa «**kútilmekte**» ***(pending)***, operaciya tamamlanǵan bolsa «**turǵın**» **(*settled*)**, operaciya nátiyje menen tamamlanǵan bolsa «**orınlandı**» ***(fulfilled)***, al eger operaciya qátelik penen tamamlanǵan bolsa «**biykarlandı**» ***(rejected*)** halatında boladı.

Orınlanıw nátiyjesin yáki **biykarlanıw sebebin** (**qátelik**) alıw ushın then metodınan paydalanamıs. Metod eki funkciyanı qabıl etedi. Birinshisi orınlanǵan halattı uslawshı funkciya, ekinshisi biykarlanǵan halattı uslawshı funkciya. Tómende sintaksisi berilgen:

```
promise.then(onFullfilled, onRejected)
```

Bunda onFullfilled úziliwshi wáde obyektinen orınlanıw nátiyjesin (nátiyje) alıwshı ótkiziwshi funkciya bolıp, al onRejected bolsa biykarlanıw sebebin (qátelik) aladı. Texnik tárepten bul eki funkciya asinxron operaciyanıń eki túrli shıǵıwı ushın eki ótkiziwshi funkciyanı biriktirgen menen birdey. Mısalı tómendegi eki sandı asinxron qosıw operaciyası nátiyjesin ápiwayı ótkiziwshi funkciya túrinde alıw keltirilgen:

```
asyncAdd(a, b, (err, result) => {
  if (err) {
     // qátelik penen islesiwshi kod
  }
  // nátiyje menen islesiwshi tiykarǵı kod
})
```

Bul kodtı úziliwshi wáde obyekti arqalı tómendegishe jaza alamıs:

```
asyncAddPromise(a, b)
  .then(result => {
```

```
      // nátiyje menen islesiwshi tiykarǵı kod
   }, err => {
      // qátelik penen islesiwshi kod
   })
```

Bul kodta asyncAddPromise úziliwshi wáde obyektin qaytarıwshı konstruktor bolıp, qaytqan obyekttiń (úziliwshi wáde) then metodı arqalı wádeniń orınlanıw nátiyjesi yáki wádeniń biykarlanıwına sebep bolǵan qátelikti ala alamıs.

Úziliwshi wáde obyektin jaratıw ushın ádette (new Promise((resolve, reject) => {})) konstruktorınan paydalanıladı. Bıraq kóbinshe konstruktordı sırtqı funkciya oblastına oraladı bul bizge tiykarǵı orınlanıw barıstan izolyatciyalanıw imkániyatın beredi. Mısalı tómendegishe usılda, bunda orınlanıw barıstı belgilingen millisekundlarǵa shekem toqtatıp qoyıwshı wádeni qaytarıwshı funkciya berilgen:

```
function delay(msec) {
  return new Promise(res => setTimeout(res, msec))
}
```

delay funkciyası msec waqıttan keyin orınlanıwshı úziliwshi wáde jaratadı.

Standartqa kúre Promise.then metodı *sinxron* tárizde isleydi hám avtomatik *jańa* úziliwshi wádeni (wáde halatın saqlawshı) qaytaradı. Metodtıń bul qásiyeti bizge óte qolaylı bolǵan *shınjırlı* orınlanıw tártibin jaratıwǵa imkániyat jaratadı. Bul imkániyat arqalı biz orınlanıw nátiyjesi ústinde bir neshe operaciyalardı izbe-izlikte islewimiz múmkin boladı. Mısalı:

```
const asyncAddPromise = (a, b) => {
  return new Promise((resolve, reject) => {
     if (Number.isFinite(a) && Number.isFinite(b)) {
        return resolve(a + b)
     }
     reject("Sanlıq parametr bolıwı kerek")
  })
}

asyncAddPromise(a, b)
  .then(result => (result < 0 ? 'teris' : 'oń'))
  .then(flag => `Jıyındınıń belgisi ${flag}`)
  .then(console.log, console.error)
```

Bundaǵı asyncAddPromise obyektinen qaytqan nátiyjeni, birinshi bolıp nátiyjeniń oń yáki teris ekenligin tekseriwshi funkciya keyin jıyındınıń belgisi haqqında xabar beriwshi tekstti qaytarıwshı funkciyalar aladı. Sońında axırǵı nátiyjeni konsolǵa shıǵarıwshı funkciya shaqırıladı. Eger bul izbe-izlik dawamında qátelik shıǵarılsa

yáki tiykarǵı wádeden qátelik qaytsa axırǵı console.error funkciyası bul qátelikti tutıp aladı.

Promises/A+ hám Promisifikaciya

Tarıyxtan, úziliwshi wádeler jaratılıwınıń kóplegen implimentaciyaları bolǵan hám olardan kópshiligi bir-birinen parqlanǵan. Yaǵınıy bir kitapxanada wádeler funkcional jolda jaratılsa al basqasında pútkilley basqasha tárizli qılıp jaratılǵan. Bul bolsa Javascriptte islewshi baǵdarlamalawshılardıń úshinshi tárep kodlarınan paydalanıwdı qıyınlastırıp qoyǵan.

Álbette waqıt ótiwi menen Javascript jámiyeti bul mashqalanı sheshiw ushın bazı sheshimlerdi beredi. Mısalı, **Promise/A+** specifikaciyasi. Bul sfecifikaciya then metodınıń xarakterin ashıp beredı hám úziliwshi wádelerdiń basqa kıtapxanalar menen birgelikte islesiwine kómeklesedi. Búgingi kúnge kelip úziliwshi wádelerdiń kópshilik implimentaciyaları bul standartqa mas túsedi.

Promise/A+ specifikaciyasi haqqında usı linkten kóbirek oqıwdı máslәhat beremen, sebebi keyingi bólimler dawamında usı specifikaciyaǵa ǵana túsiwshi kodlar jazıladı.
onsom.uz/u/promise-aplus

Bul standarttıń qabıl etiliwi nátiyjesinde JavaScript APIsındaǵı barliq then metodına iye obyektler **thenable** dep atalına baslandı. Bul xarakteristika úziliwshi wádelerdiń túrli implimentaciyalarına bir-biri menen qıyınshılıqlarsız óz-ara tásir qılıw imkániyatın beredi.

Endi keliń úziliwshi wádeler APIsın kórip shıqsaq:

- **Promise.resolve(obj):** Bul metod berilgen parametrdem jańa úziliwshi wádeni jaratadı. Eger parametr úziliwshi wáde bolsa solayınsha, thenable obyekt bolsa obyektti úziliwshi wádege aylandıradı, al eger muǵdar bolsa úziliwshi wáde usı muǵdar menen orınlanadı.

- **Promise.reject(err):** Bul metod err sebebi menen biykarlawshı jańa úziliwshi wádeni jaratadı.

- **Promise.all(iterable):** Bul metodqa úziliwshi wádeler (yáki thenable, muǵdarda bolıwı múmkin) toplamı parametr qılıp beriledi hám toplamdaǵı barlıq úziliwshi wádelerdiń orınlanıw nátiyjelerinen ibárat toplam menen orınlanıwshı jańa úziliwshi wáde jaratıladı. Eger wádelerden biri err sebebi menen biykarlanatuǵın bolsa, usı sebep (toplamda eń birinshi biykarlanǵan wádeniń sebebi) penen úziliwshi wádede qaytarıladı.

- **Promise.allSettled(iterable):** Bul metod, parametr qılıp berilgen barlıq wádelerdiń tamamlanıwın kútip turadı hám qaytqan nátiyje yáki qátelik sebeplerinen (obyekt tárizli) ibárat bolǵan toplamdı qaytaradı. Toplamdaǵı hár bir obyekttiń *'status'* degen qásiyeti boladı, bul qásiyet ózine, eger wáde orınlansa *'fullfilled'* muǵdarın, al eger biykarlanǵan bolsa *'rejected'* muǵdarın aladı.

- **Promise.race(iterable):** Bul metod paramter qılıp berilgen wádeler toplamı ishindegi eń birinshi orınlanıwshı yáki biykarlanıwshı úziliwshi wádeni qaytaradı.

Úziliwshi wáde obyektiniń eń tiykaǵı úsh metodı:

- **promise.then(onFullfilled, onRejected):** Bul metod, wáde orınlanǵannan keyin onFullfilled ótkiziwshi funkciyasın, al eger wáde biykarlanatuǵın bolsa onRejected ótkiziwshi funkciyasın shaqırıw imkánin beredi.

- **promise.catch(onRejected):** Bul metod eger wáde biykarlansa onRejected ótkiziwshi funkciyası arqalı qaytqan qátelikti uslawǵa imkán beredi. Metod promise.then(undefined, onRejected) sintaksisi menen birdey.

- **promise.finally(onFinally):** Bul metod úziliwshi wáde tamamlanǵanda onFinally ótkiziwshi funkciyasın shaqırıw imkánin beredi.

Bul metodlardıń durıs orında qollanılıw arqalı, asinxron barıstı basqarıwdı ańsatlastıra alamıs. Kórgenimistey úziliwshi wádeler ótkiziwshi funkciyalardıń inkapsulaciya forması dewimiz múmkin. Álbette asinxron barıstı basqarıw ushın hárkim ózine qolayın tańlaydı. Al eger bir neshe baǵdarlamashılar bir jobada islep atırǵanda, birew ótkiziwshi funkciyalar al basqası úziliwshi wádeler menen islesetuǵın bolsa ne boladı?

Bul sorawǵa juwaptı álleqashan Javascript jámiyeti berip qoyǵan. Bunday hallar ushın promisifikaciya (ótkiziwshi funkciyanı úziliwshi wádege aylandırıw) usılı bar. Bul usıl ádette tómendegishe implimentaciya qılınadı:

```
function promisify(fn) {
  return function promisified (...args) {
    return new Promise((resolve, reject) => {
      fn(...args, (err, result) => {
        if (err) return reject(err)
        resolve(result)
      })
    })
  }
}
```

```
function readFile(filename, callback) {
  setTimeout(() => callback(null,'Fayl kontenti'), 1000)
}

const readFilePromise = promisify(readFile)
readFilePromise('myfile.txt').then(console.log)
```

promisify funkciyası tómendegishe tártipte isleydi:

1. Funckiyaǵa parametr qılıp basta, úziliwshi wádege aylandırılıwı kerek bolǵan funkciya beriledi hám bul wádege juwap beriwshi promisified funkciyası qaytarıladı.

2. promisified funkciyası jańa úziliwshi wáde jaratadı hám shaqırılıwshıǵa jaratılǵan wádeni birden qaytaradı.

3. Haqıyqıy funkciyanı (fn) shaqıramıs. Bunda ótkiziwshi funkciya standartqa kóre eń aqırǵı keletinin bilgenlikten, args toplamındaǵı barlıq elementlerdi argument qılıp berip jiberemis. Eger ótkiziwshi funkciyada qátelik shıqsa wádeni biykarlaymıs; bolmasa orınlap nátiyjeni shıǵaramıs.

Endı bolsa keliń, ótkiziwshi funkciyalardaǵı sıyaqlı orınlanıw tártiplerin úziliwshi wádeler járdeminde qanday qılıw múmkinligin kórip óteyik.

Demek, asinxron operaciyalardıń izbe-iz orınlanıw tártibin úziliwshi wádeler menen bir neshe usıllarda jaratıw múmkin. Birinshi hám kóbirek qollanılıwshı usıl, then metodınıń jańa úziliwshi wáde qaytarıwın bilgen halda, úziliwshi wádeler shınjırın jaratıw. Mısalı, tómendegishe:

```
addPromise(1, 3)
  .then(Math.sqrt))
  .then(result => {
    console.log(result)
    return differPromise(2, 2) // jańa operaciya wádesi
  })
  .then(console.log)
```

Keyingi usıl cikllar járdeminde úziliwshi wádelerdi dinamikalıq shınjırlaw. Bunda wádeler shınjırı cikl ishinde dinamik jaratıladı:

```
const additions = [[1, 8], [-2, 7], [3, 0]]
let chain = Promise.resolve()

for (const addition of additions) {
  chain = chain.then(() => addPromise(...addition))
}
```

```
chain.then(console.log)
```

Bul eki usılda da itibar bergen bolsańız then metodınıń qolaylılıǵı menen paydalanıladı. Eger bular ulıwmalastırsaq tómendegishe bir úlgige iye bolamıs:

```
Promise.resolve()
  .then(() => operation1())
  .then(resultOperation1 => operation2(resultOperation1))
  .then(resultOperation2 => { ... })
```

Endi bolsa asinxron operaciyalardı parallel orınlanıw tártibin úziliwshi wádelerde qanday implimentaciya qılıw múmkinligine qısqasha toqtalayıq. Qısqasha, óytkeni úziliwshi wádelerdiń tábiyatı parallel orınlanıwǵa tiykarlanǵan. Sonlıqtanda Javascripttegi Promise obyektiniń Promise.all metodı asinxron operaciyalardı parallel orınlaw ushın jaratılǵan. Yaǵınıy bul metod shaqırılǵanda, barlıq wádeler orınlanǵannan keyin usı wádelerdiń nátiyjelerin ózinde saqlawshı jańa wáde obyektin jaratadı.

```
Promise.all([
  addPromise(1, 2),
  addPromise(2, 3),
  differPromise(3, 4)
]).then(console.log)
```

Eger wádeler arasında bir-birine tásir qılıwshı asinxron operaciyalar bolsa ishke tartılıwshı wáde jaratıwımız múmkin:

```
Promise.all([
  addPromise(6, 3),
  Promise.all([
    sqrtPromise(25), // addPromise(6, 3) ke baylanıslı
    differPromise(5, 0) // Erkin
  ])
]).then(console.log)
```

Bunda birinshi addPromise(6, 3) wádesi basqa wádelerge baylanıssız túrde orınlanadı. Ishke tartılıwshı Promise.all ishtegi eki wádeni addPromise orınlanıp bolǵannan keyin bir-birine konkurrent esab'nda júrgizedi. Sırtqı Promise.all barlıq wádeler orınlanıwın kútedi.

Sońǵı bolıp toqtalatuǵın orınlanı tártibi shekli parallel orınlanıw úlgisi. Kod tómendegishe:

```
const tasks = [
  new Promise((resolve) => setTimeout(resolve, 2000)),
```

```
  new Promise((_, reject) => setTimeout(reject, 1000)),
  new Promise((resolve) => setTimeout(resolve, 3000)),
]

const MAX_CONCURRENCY = 2
let running = 0
let completed = 0

function finish() { /* operaciyalar juwmaqlandı */ }

function next() {
  while (running < MAX_CONCURRENCY && tasks.length > 0) {
     const task = tasks.shift()
     running++
     task.finally(() => {
        completed++
        running--
        if (completed === tasks.length) return finish()
        next()
     }).catch(console.error)
  }
}

next()
```

Bunda asinxron operaciyanıń tamamlanıwın uslawshı ótkiziwshi funkciya keyingi wádeni shaqıradı. Eger operaciya waqıtında qátelik kelip shıqsa catch metodı bul qátelikti uslaydı.

Eger asinxron operaciya nátiyjesi menen islesiw kerek bolsa, finally ornına then metodın nátiyje menen qollanıwǵa boladı. Bunda catch metodına da aldınǵı operaciya juwmaqlanǵanlıǵı hám next funkciyası óz aldına shaqırıladı.

Async/Await

Úziliwshi wádelerde kórgenimizdey, olar menen asinxron kodtı taza hám túsiniwge qolaylı etip jaza alamıs. Bıraq, wádeler izbe-iz asinxron operaciyalardı jazıwda sál qıyınshılıq tuwdıradı. Álbette wádeler shınjırı ótkiziwshi funkciyalar batpaǵınan kóre qolaylıraq, degen menen Javascript kod jazıwdı elede optimallastıra aladı. Álbette EcmaScripttiń standartı bolǵan async/await sintaksisi arqalı.

Async/await funkciyanı asinxron operaciyanıń juwmaqlanıwına baylanıslı túrde izbe-iz orınlanıw tártibin ańsat jaratıw imkánin beredi. Bul sintaksis penen jazılǵan kod oqıwǵada túsiniwgede ańsatlasadı. Sonlıqtan da búgingi kúnde async/await asinxron kod penen islesiwdiń eń optimal jolı dep bilinedi. Endi, keliń bulardıń bárine toqtalsaq.

Eger qısqasha aytatuǵın bolsaq, **async** bayanlaması ápiwayı funkciyanı úziliwshi wáde qaytarıwshı funkciyaǵa aylandıradı, **await** bolsa funkciyanı sol wádeniń tamamlanıwın kútiwge májbúrleydi, sonda **async/await** birgelikte funkciyalardı úziliwshi wádeler menen islewin ańsatlastıradı. Yaǵınıy Javascripttaǵı funkciyalardı jaratıwshı (function add(a, b){ ... }) qatarın (async function add(a, b){ ... }) qatarına ózgertirsek bul funkciya endi úziliwshi wádeni qaytarıwshı funkciyaǵa aylanadı. Endi bul funkciyanı shaqırǵandaǵı (add(1, 2)) qatarın (await add(1, 2)) qatarına ózgertirsek shaqırıwshı funkciya sol wádeniń tamamlanıwın kútiwge májbur boladı. Bul jerde itibar beriwimiz kerek, await giltsózi async giltsózi oblastına oralǵan bolıwı kerek. Mısalı:

```
const delay = msec => new Promise(res => setTimeout(res,
msec))

async function run() {
  console.time('run')
  await delay(1000)
  console.timeLog('run', '1-sekundtan keyin ')
  await delay(3000)
  console.timeEnd('run')
  return 'tamamlandı!'
}
```

Bul mısalda kórgenińistey async/await sintaksisi qolaylı túrde isleydi. Bundaǵı hárbir await giltsózinen keyin run funkciyasınıń halatı saqlanǵan halda, orınlanıw barısı toqtatılıp turıladı hám basqarıw waqıyalar ciklına jiberiledi. delay funkciyasınan qaytqan wáde orınlanıp bolǵanında basqarıw qaytadan funkciyaǵa qaytadı hám usılayınsh dawam etedi. Endı bolsa bul funkciyanıń shaqırılıwın kóreyik:

```
run().then(res => console.log('4-sekundtan keyin ', res))
```

Kodta berilgenindey run asinxron funkciya bolıp ol úziliwshi wáde qaytaradı. Tek parqı jazılıwında ǵana. Ulıwma aytqanda run funkciyasınıń qaytaratuǵın nátiyjesin Promise.resolve('tamamlandı!') kórinisindegi ápiwayi funkciya menende jazsaq boladı.

Al endi keliń, wáde biykarlanǵanda ne bolatuǵınlıǵına yáki qandayda bir qátelik shıqqanda bul qátelikti qanday uslawımız múmkinligin kóreyik. Demek berilgen waqıttan keyin biykarlanıwshı wádeni qaytarıwshı funkciya:

```
const delayError = msec => new Promise((_, reject) => {
  setTimeout(() => {
     reject(new Error('Asinxron qátelik'))
  }, msec)
```

```
})
```

Bul funkciyanı shaqırıwshi kod:

```
async function runError(syncError) {
  try {
    if (syncError) throw new Error('Sinxron qátelik')
     await delayError(1000)
  } catch (err) {
     console.error(`Qátelik: ${err.message}`)
  }
}
```

Bunda runError(true) orınlanǵanda da, runError(false) orınlanǵan waqıttada aǵındaǵı qátelikler birdey uslanadı. Sebebi await funkciya barısın, wáde tamamlanaman degenshe toqtatıp turadı. Eger wáde biykarlanatuǵın bolsa, biykarlanıw sebebi menen catch blogına basqarıwdı uzatıp jiberedi. Bul jerde itibarlı bolıwımız kerek, asinxron funkciya ishinde await giltsózi isltetilgen waqıtta , potencial qátelikler sol funkciyanıń ishinde catch blokı penen uslanadı.

Endi bolsa úziliwshi wádeler hám ótkiziwshi funkciyalarda qılǵanımızday asinxron operaciyalardıń orınlanıw tártibi qalayınsha async/await penen basqarılıwın kórip shıqsaq. Demek birinshi izbe-iz orınlanıw.

Izbe-iz orınlanıw async/await sintaksisiniń negizi bolǵanı ushın buǵan uzaq toqtalıwdıń shárt emes, tómende bir neshe wádelerdi izbe-iz orınlawshı kod bólegi berilgen:

```
let p1 = () => new Promise(r => setTimeout(r, 2000, 1))
let p2 = () => new Promise(r => setTimeout(r, 1000, 2))
let p3 = () => new Promise(r => setTimeout(r, 3000, 3))


async function run() {
  let n1 = await p1()
  let n2 = n1 + await p2()
  return n2 + await p3()
}

run().then(console.log)
```

Bunda ulıwma orınlanıw waqtı 6 sekundtı quraydı. Itibar beriwimiz kerek asinxron funkciyalardaǵı return hám return await tábiyatı ayırmalanadı. return await asinxron operaciya orınlanıw waqıtındaǵı potencial qátelikler funkciya denesindegi try ... catch penen tutıladı.

Al endi parallel orınlanıwdı kórip shıqsaq. Async/Await penen parallel orınlanıw tártibin jaratıwdıń tiykarǵı eki jolı bar: biri Promise.all arqalı; ekinshisi haqıyqıy async/await sintaksisi arqalı. Eki jolda qolaylı, bıraq kóbinshe baǵdarlamashılar tárepinen Promise.all qollanıladı. Sonlıqtan bul joldı kórip shıqsaq.

Promise.all menen tómendegishe parallel orınlanıw jaratıladı:

```
const promises  = [
  new Promise(r => setTimeout(r, 2000, 1)),
  new Promise(r => setTimeout(r, 1000, 2)),
  new Promise(r => setTimeout(r, 3000, 3)),
]

async function run() {
  return await Promise.all(promises)
}

run().then(console.log)
```

Bunda kodtıń ulıwma orınlanıw waqtı 3 sekundtı quraydı. Bul jerde barlıq wádeler sinxron shaqırıladı, await bolsa sol wádelerdiń tamamlanıwın kútedi hám wádelerdiń nátiyjelerin bir toplamda funkciyadan qaytarıladı.

Axırǵı orınlanıw tártibi shekli parallel orınlanıw bolıp, bul tártipti async/await penen jaratıw óte ańsat. Mısalı tómendegishe:

```
const tasks = [
  () => new Promise(r => setTimeout(r, 2000, 1)),
  () => new Promise((_, r) => setTimeout(r, 1000, 2)),
  () => new Promise(r => setTimeout(r, 3000, 3))
]

const MAX_CONCURRENCY = 2

function finish() { /* operaciyalar juwmaqlandı */ }

async function run(tasks, limit) {
  const results = []
  let i = 0

  while (i < tasks.length) {
     const _tasks = tasks.slice(i, i + limit)
     const promises = _tasks.map(async task => task())
     const _results = await Promise.allSettled(promises)
```

```
      results.push(..._results)
      i += limit
    }

    return results
  }

  (async () => {
    try {
      const results = await run(tasks, MAX_CONCURRENCY)
      finish()
    } catch (err) {
      console.error(err)
    }
  })()
```

Bul kodta itibar bergen bolsańız. Asinxron funkciya shaqırılǵanda áwel-basta while ciklı tasks toplamınan shekli limit muǵdardaǵı ámeldi tańlap aladı hám usı ámellerdi Promise.allSettled metodı menen parallel júrgizip jiberedi. Tanlanǵan barlıq ámeller tamamlanıp nátiyjeleri alınǵannan keyin nátiyjeler arnawlı results toplamına qosıladı hám usılayınsha toplamdaǵı barlıq ámeller ústinde júrilip shıǵıladı. Barlıq ámeller tamamlanıp nátiyjeler qaytqannan keyin finish funkciyası orınlanadı, eger bul operaciya orınlanıw barısında qandayda bir qátelik kelip shıqsa bul qátelik catch penen uslanadı. Sonda ulıwma operaciya bizdiń mısalımızda 5 sekund dawam etedi.