21 September 2024 16:08

arr = [10, 20, 30, 40]

arr.filter(element => element);

[10, 20, 30, 40]

arr = [1, 2, 3, 4, 5, 6]

arr.reduce((sum, element) => { c.log //code ... })

arr[0]

a[1]

Higher Order Function

1. takes fun^n as an argument

2. takes fun^n as an argument + return function

map, filter, sort, forEach, some, every, reduce,

OOJS wrapper

- Introduction
- Datatypes — Primitives, Reference
- typeof
- instance of
- Mutability
- Immutability
- Arrays literal
- object literal { }
- G C
- Auto boxing

closures
IIFE
Memoization

Async Program

- Sync, Async
- Call Back Hell
- Promises
- Methods
- Async Await
- API calls
- AJAX

UI –

– Auto boxing

– Wrapper types

– Interview question

– Assignment add

– Parse Int, Is NaN, isFinite

– Rest ~~ab~~ parameters

– Spread operator

– Mutability, Immutability

– spread with objects

– Destructuring – Array, object

– Loops

– Hoisting

– Lexical Scope

– Function & its types

– Article

– Pure fn

– HoF

– Array Methods

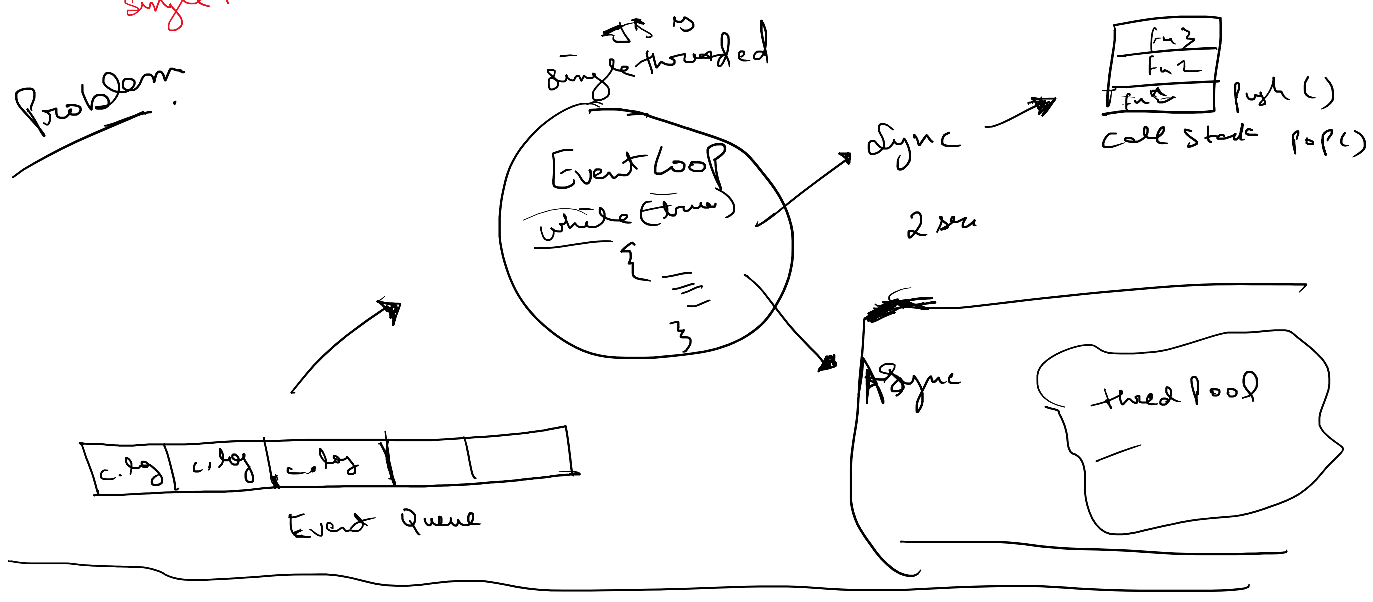
– API calls

– AJAX

– DOM, BOM

$\boxed{\text{OOJS}}$

single thread, Multithreading



Sync

Async Non-Blocking

Sync
Blocking

Async
Non-Blocking

repeat 2 times

Sync style

task 1 → 2 sec
task 2 → 1 sec
task 3 → 3 sec

6 sec

task → 3 sec
4 times repeat
total time → 12 sec

1 → 2 sec
2 → 2 sec

total → 4 sec

Network calls

100 website → 100 sec

1 website → 1 sec

load Half.

3-4 sec

100 sec.

Repeated   Sync task → Memoization

Portion of result → store
Memorize

Cookies
Image, ...

truthy, falsy

~~[illegible]~~

a, b, c

null, NaN, undefined,
0, empty,

if → true

if (a) { c.log(a) }