

~~Video 7~~

Autocomplete / Typeahead

Front End Machine Coding Mock Interview

Statement

Autocomplete (Low-level Design)

Tech Stack

vanilla JS, HTML & CSS

10 Must have

→ On typing inside the search box, it should suggest option.

→ 15 User should be able to select one of the option

Good to have :-

→ 20 Search should be performant enough.

→ It should avoid unnecessary network call.

→ It should be reusable & customizable

→ It should persist previously fetch data

25

reusable

as possible

transaction

distributed

30

Debounce

```

const debounce = (func, delay) => {
  let inDebounce;
  return function () {
    const context = this;
    const args = arguments;
    clearTimeout(inDebounce);
    inDebounce = setTimeout(() => func,
      inDebounce = setTimeout(() => func,
        apply(context, args),
        delay));
  };
}

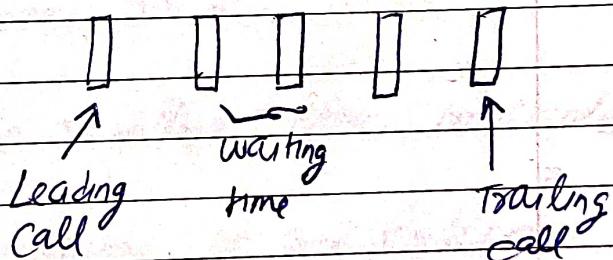
```

Throttle

Event Stream



Callbacks



```
const throttle = (func, limit) => {

```

let lastFunc;

let lastRun;

```
return function () {

```

const context = this;

const args = arguments;

if (!lastRun) {

func.apply(context, args);

lastRun = Date.now();

} else {
 clearTimeout(lastFunc);

lastFunc = setTimeout(function() {

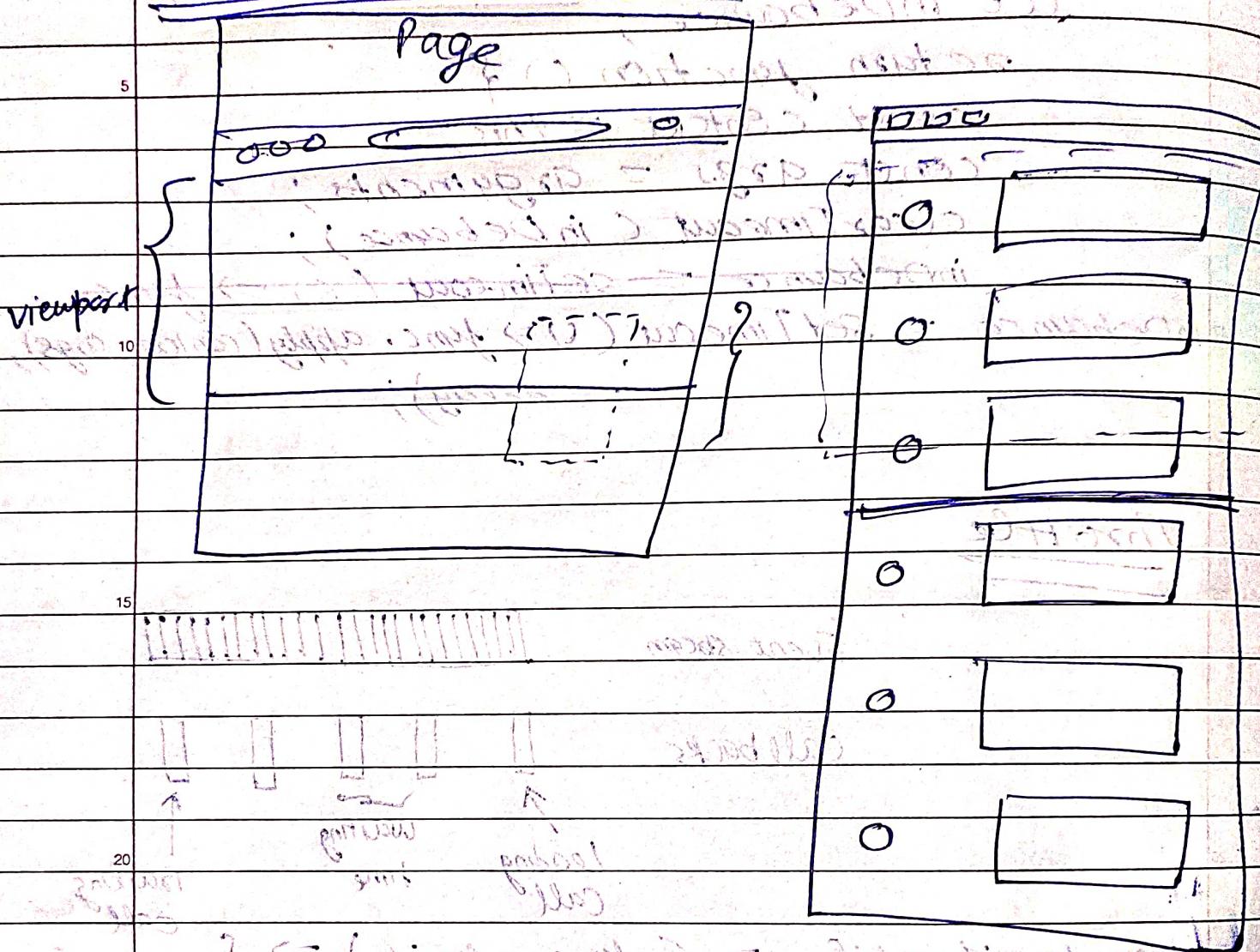
if ((Date.now() - lastRun) >= limit) {

func.apply(context, args);

lastRun = Date.now();
}

3, limit - (Date.now() - lastSeen))

Intersection observer



25

if (intersection) {
 const entry = intersection.entries[0];

if (entry.isIntersecting) {

30

console.log(`Element \${entry.target.id} is now intersecting the viewport!`);
 // do something here

} else {
 console.log(`Element \${entry.target.id} is no longer intersecting the viewport.`);
 // do something here

JavaScript Snippet

```

var scroller = document.querySelector('#scroller');
var sentinel = document.querySelector('#sentinel');
var counter = 1;

function loadItems(n) {
    for (var i = 0; i < n; i++) {
        var newItem = document.createElement('div');
        newItem.classList.add('item');
        newItem.textContent = 'Item' + counter++;
        scroller.appendChild(newItem);
    }
}

```

```

var intersectionObserver = new IntersectionObserver(entries => {

```

If the browser is busy while scrolling happens, multiple entries can accumulate between invocation of this callback.

As long as any one of the notifications affect the sentinel within the scrolling viewport, we add more content.

```

if (entries.some(entry => entry.intersectionRatio > 0)) {
    loadItems(10);
}
```

appendChild will move the existing element, so there is no need to remove it first.

```
scroller.appendChild(sentinel);
```

```
loadItems(5);
```

```
changeable.setStatus('loaded up to item' + counter);
```

intersectionObserver.observe (sentinel);
running Task 2

Abort Controller

```
const controller = new AbortController();
const ocs = fetch('1', { signal: controller.signal });
controller.abort();
```

console.log(ocs)

(AbortError) { message: 'aborted' }
+ '1' + '1'

15

wait for ocs = wait for sentinel now

20

called with value and to break it

it calls Abort, it破壞了繼承性質的

25

task 2 send the val

task 1 interupted because val (= value) must wait for it

(or) and then

task 1 handle success with this kind of code

30

on read or write point to normal

(normal) write and read point