Tekion

- Round 1
  - Implement polyfill for filter
  - Given input arr = [1,2,3] n(any number)
    - Generate permutations of n digits combination like [11, 12,13,21,22,23,31,32,33]
  - Implement function to check if 2 nested objects are same
  - Questions on bind, apply, call concepts
  - Var, let , const
  - Prototype chaining

```
function foo() {
        this.a = 'a'
}
hello = new foo();
add getA for `foo` which return property a
```

- Round 2
  - Implement fillColor(x,y) when clicked on any coordinates it should fill the rectangle
  - Event loop
  - Prototype inheritance - achieve call inheritance with function
  - Someother javascript fundamentals based questions
- Round 1
  - Closures
  - Settimeout inside for loop print continuous i
  - Arrow function
  - Arrow function prototype ans: undefined, for function and objects it returns the prototype constructor
  - Array spreading
  - Output of [...[..."abc"]].length
  - React lifecycle
- Round 1
  - Write function composition - promise chaining

CureFit:

1. Front End Interview but with just mobile development experience
    i. Implement slack like application(mostly discussion)
    (HTTP2, websockets, Rest API standards, HTTP response codes)

MindTickle

- Round 1
    - Implement polyfill for Array.sort
    - IMDB Search
- Round 2
    - Best time to sell stocks
    - https://leetcode.com/problems/best-time-to-buy-and-sell-stock/
    - https://leetcode.com/problems/avoid-flood-in-the-city/

EdgeNetworks

- Round 1
    - How browser rendering works
    - From Investment/budget point of view convince why react vs low cost hiring jquery
    - Single Page application vs multi page application
    - Given a system with 1gb ram where memory is very important, implement a page loading carousel which shows 1 image at a point of time.

Mastree.io

- Round 1
    - Implement polyfill for map
    - Create tic tac toe app
    - Given sorted array find index of given element using binary search using iterator approach and return index if element is not found

Spotnana

- Round1
  - Implement polyfil for Posmise.all
  - Build nested foldering UI with static data - expand and show sub folders and files on click on folder

CueMath

- Round1
  - Convert callback to promise
  - Constructor, object inheritance, this based concept questions
  - [a, b, c, d] => {a:b, c: d} using reducer
  - var/let/const
- Round2
  - Implement fibonacci
  - Bind polyfill
  - sum(2)(4)(6).....()
  - Other javascript questions on fundamentals

Rubrik

- Round1
  - Call multiple async tasks, return single formatted resp

```
// Generate data center level aggregated stats of protected vs. unprotected applications

// The following mock helper method returns protection stats for a given application type in a given data center.

// In real implementation this will be an ajax call to `${datacenterUrl}/${applicationType}` or something like that

const getApplicationProtectionStats = (datacenterUrl, applicationType)=> new Promise((resolve, reject) => {

 setTimeout(() => {

   resolve({

     protectedApps: Math.floor(Math.random() * 100),
```

```
      unprotectedApps: Math.floor(Math.random() * 100)
    });
    /
/ reject('error');
 }, 1000);
});
// getApplicationProtectionStats('https://www.olive.com/stat',
'vmware').then(console.log)
const applicationTypes = [
 'vmware',
 'mssql',
 'ahv'
];
const datacenters = [
 {name: 'olive', url: 'https://www.olive.com/stat'},
 {name: 'comet', url: 'https://www.comet.com/stat'},
 {name: 'blah', url: 'https://www.blah.com/stat'}
];
// {
//   olive: {
//     protectedApps: <sum of protected apps counts for each application
type in Olive>,
//     unprotectedApps: <sum of unprotected apps counts for each
application type in Olive>
//   },
//   comet: {..},
//   blah: {...}
// }
```

- **Round 2**
  - Machine coding round - to implement shuffle deck and pick 5 cards from deck
- **Round 3**
  - Design round - slack app (focus is more on frontend components and data management)

- **Round 4**
    - Implement autocomplete - show correct results when API takes varying time correct results should be show

Whitehat.jr

- Round 1
    - Javascript fundamentals based questions
    - Implement reduce polyfill
    - Reduce, map, filter - explain
    - Event loop
    - React - concepts

Cloudera

Machine Coding rounds

1) **Implement tic tac toe**
2) **Given an array of suits and card numbers implement a function which will deal 4 random cards on the table till the deck is exhausted.**
3)

- Round 1
    - How to filter autocomplete array list optimizely without iterative approach

```
["getName", "getValue", "setName", "setValue"] //

find("get") //-> ["getName", "getValue"]

var find = function(param) {
    const source = ["getName", "getValue", "setName",
"setValue"];
```

```
    let filteredList = [];

    for(let i=0; i<source.length; i++) {
        if (source[i].indexOf(param) !== -1) {

        }
    }

};
```

- Is relative > relative and relative > absolute are the same?

```
<div style="position: relative;">
    <div style="position: relative;width: 100%;">Abc</div>
</div>

<div style="position: relative; width: 50px;">
    <div style="position: absolute;width: 100%;">Abc</div>
</div>
```

- What are the differences in width with that?

SalesForce:

Round 1:

- create a parent and child div where child div should be center of parent div
- create new memoization function where it accepts any functions parameter and executes it.
- Create stringfy function to convert json object to string (JSON.stringify)

Round 2:

- Create low level module/libary where different data parser modules are created (json parser,xml parser..etc). Flexibility to add more new modules in future.
- Create Json parser to find pattern “https://” inside json object

Ascendeum:

Round 1: Create stop watch with stop,reset and pause options.

Reducer polyfill

```
Array.prototype.myReduce = function(...args) {
   let arr = this;
   let callback = args[0];
   let acc = arr[0];
   let pending = true;
   let inc = 0;
   do {
```

```
        if (arr.length === inc || !arr[inc + 1]) {
            pending = false;
        } else {
            acc = callback(acc, arr[inc + 1]);
            inc++;
        }
    } while(pending)
    return acc;
}

async function getAllAppCount() {
    let res = {};
    datacenters.map(async (dataCenter) => applicationTypes.map( async (app) => {
        let { url, name } = dataCenter;
        let appType = app;
        let resp = await getApplicationProtectionStats(url, appType);
        if (res[name]) {
            res[name]['protectedApps'] += resp.protectedApps;
            res[name]['protectedApps'] += resp.protectedApps;
        } else {
            res[name] = {
                protectedApps: resp.protectedApps,
                unprotectedApps: resp.unprotectedApps
            }
        }
        console.log(res);
    }));
}



function getAllDataCenterApps(timeout) {
 var allPromises = datacenters.map((datacenter) => applicationTypes.map((appType) =>
getApplicationProtectionStats(datacenter.url, appType)));
 var dataCenterPromises = allPromises.map((dataCenter) => Promise.all(dataCenter));
 return Promise.all(dataCenterPromises).then((resp) => {
```

```
   var finalFormat = resp.reduce((acc, apps, index) => {
     acc[datacenters[index].name] = apps.reduce((acc, app) => {
         acc.protectedApps += app.protectedApps
         acc.unprotectedApps += + app.unprotectedApps
         return acc
     }, {
       protectedApps: 0,
       unprotectedApps: 0
     });
     return acc;
   }, {});
   return finalFormat;
 }).catch((err) => err);
}
// getAllDataCenterApps().then(console.log)
function getAllDataCenterAppsWithTimeLimit(timeout) {
 return new Promise((res, rej) => {
    getAllDataCenterApps().then((data) => res(data))
    setTimeout(() => res('error'), timeout);
 })
}
getAllDataCenterAppsWithTimeLimit(5000).then(console.log)

NighFall AI Round 1
```

Technical Phone Screen Exercise

Prerequisites

Make sure you have your IDE/code editor installed and configured. You may also use CodePen or another similar tool if you prefer.

Expectations

- Feel free to use your preferred JavaScript framework (React, Vue, Angular, etc.) or vanilla JS
- No jQuery

Exercise

1. Create a progress bar that is 200px wide, 20px tall and animates from 0 to 100% of its width in 5 seconds (20% every 1 second). No CSS animations - use JS to advance the loading bar until 100%.
2. Create a button that adds progress bars to the DOM on click
3. Update your code so that only 1 progress bar can be animating at a time, but you can still add progress bars to the DOM on button click. The next progress bar should begin loading once the previous bar has completed loading.
4. Update your code so that 5 progress bars can be animating at a time, but you can still add progress bars to the DOM on button click. A bar should only begin loading if fewer than 5 are currently loading.
5. Add a button next to each progress bar to remove the progress bar from the DOM when clicked