Video 11

Date :
Page No.

10 Popular Memory Leaks in JS

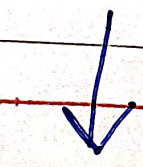
1. Accidental global variables

```
b=15;
function abcc(){
    c = "Hi Charde system design";
}

abcc();

function abcd(){
    this.d = "love to charde SD";
}

abcd();

console.log(b,c,d);
```

// we are able to access all those variable as they are declared globally.

- accidentaly it may override
- normal JS function behave differently i.e it also create global variable.

How we can fix this



Date :
Page No.

```
"use strict"
var b = 15;
function abcc(){
    var c = "Hi Chakde system design";
}

abcc();
console.log(window.c);

function abcd(){
    this.d = "Love to Chakde SD";
}
new abcd()

console.log(c, d)
```

↓ ↓

undefine

2. Forgotten timers or callbacks

```
var timer = setInterval(function(){
    var node = document.getElementById('Node');
    if(node){

        node.innerHTML = JSON.stringify(
                        someResource);

        clearInterval(timer);
    }
}, 1000);
```

Date :
Page No.

3 Remove Listeners if not required

```
var ele = document.getElementById('button');
var text = document.getElementById('text');

function onClick(){
    console.log('clicked');
    text.innerHTML = 'text';
}

function removeListener(){
    element.removeEventListener('click', onClick);
}

function addListener(){
    removeListener();
    element.addEventListener('click', onClick);
}

function recender(){
    setInterval(()=>{
        addListener();
    }, 500);
}
recender();
```

Date :
Page No.

Other way.

```
var ele = document.getElementId('button');
var text = document.getElementById('text');

function onClick() {
    console.log('Clicked');
    text.innerHTML = 'text';
}

function addListener() {
    element.addEventListener('click', onClick,
                                { once: true })
}

addListener();
```

4 out of DOM Reference

```
var a = document.getElementById('xyz');
console.log(a);
```

→ some html

```
document.body.removeChild(document.getElementById('xyz'));
```

```
console.log(document.getElementById('xyz'));
console.log(a);
```

→ some html // somehow it stores that data.

→ null

What to do manually clear it

Date :
Page No.

```
document.body.removeChild(a)
a = null
```

5 **Closures**

```
var theThing = null;

var replaceThing = function () {
  var originalThing = theThing;

  var unused = function () {
    if (originalThing)
      console.log ("hi!");
  };

  theThing = {
    longStr : new Array(1000000),
    someMethod : function () {
      console.log ('Bye');
    }
  };
};
setInterval(replaceThing, 1000);
```

It will increase the memory

See JS VM instance in Memory tab

Date :
Page No.

(6) Reference Object cycle

```
let x = {
    a: {
        b: 2,
    },
};
var y = x;
console.log(x, y);
console.log('----------');
x = 1;
console.log(x, y);
console.log('----------');
var z = y.a;
console.log(x, y, z);
y = 2;
console.log(x, y, z);
console.log('----------');
z = 3;
console.log(x, y, z);
```

Date :
Page No.

(7) Detached window

```
let notesWindow;

function showNotes() {
notesWindow = window.open("/example 7/index.html");
notesWindow.document.addEventListener("click",
                                      nextSlide);
}

document.getElementById("notes").onClick =
                                      showNotes;

let slide = 1;
function nextSlide() {
   slide += 1;
 notesWindow.document.title = 'Slide ${slide}';
}
document.getElementById("title").onClick =
                                      nextSlide;

function closeNotes() {
   notesWindow.close();
   notesWindow = null;
}

document.getElementById("close").onClick =
                                      closeNotes;
```

Date :
Page No.

8 Forget to stop listening.

When we create the promise but it didn't resolving or rejecting or you can say not returning anything.

```
async function randomString(length) {
  await new Promise((resolve) =>
                    setTimeout(resolve, 1));
  :
  return result;
}

(async function main() {
  const unsettled = new Promise(() => {});
  for (let i = 0; ; i++) {
    await Promise.race([unsettled, randomString(10000)]);
  }
})();
```

This will increase the memory.

Date :
Page No.

Forget to disconnect

Resize Observer
Intersection Observer
Mutation Observer

In the above observer don't forget to call .disconnect() or .unobserve()

Virtualization of DOM

Windowed or Virtualization

→ only the visible rows are rendered.

~~Adjust its configurable properties - below to see.~~

→ Also take care of the data which is fetching from the api. It should also not cause memory leak.