**Q1.Create a hierarchy of person, employee and developers.**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<title>Title</title>**

**</head>**

**<body>**

**<script>**

**var *person*=function(){**

**this.name="Kanchan";**

**}**

***person*.prototype.company="TTN";**

**var *employee*=function(){**

**this.dept="Feen";**

**}**

***employee*.prototype=*Object*.create(*person*.prototype);**

***employee*.prototype.constructor=*employee*;**

**var *e*=new *employee*();**

***document*.write(*e*.company +"<br>");**

***//console.log(e.company);***

***employee*.prototype.id=2;**

**var *developers*=function () {**

**this.tech="React";**

**}**

***developers*.prototype=*Object*.create(*employee*.prototype);**

***developers*.prototype.constructor=*developers*;**

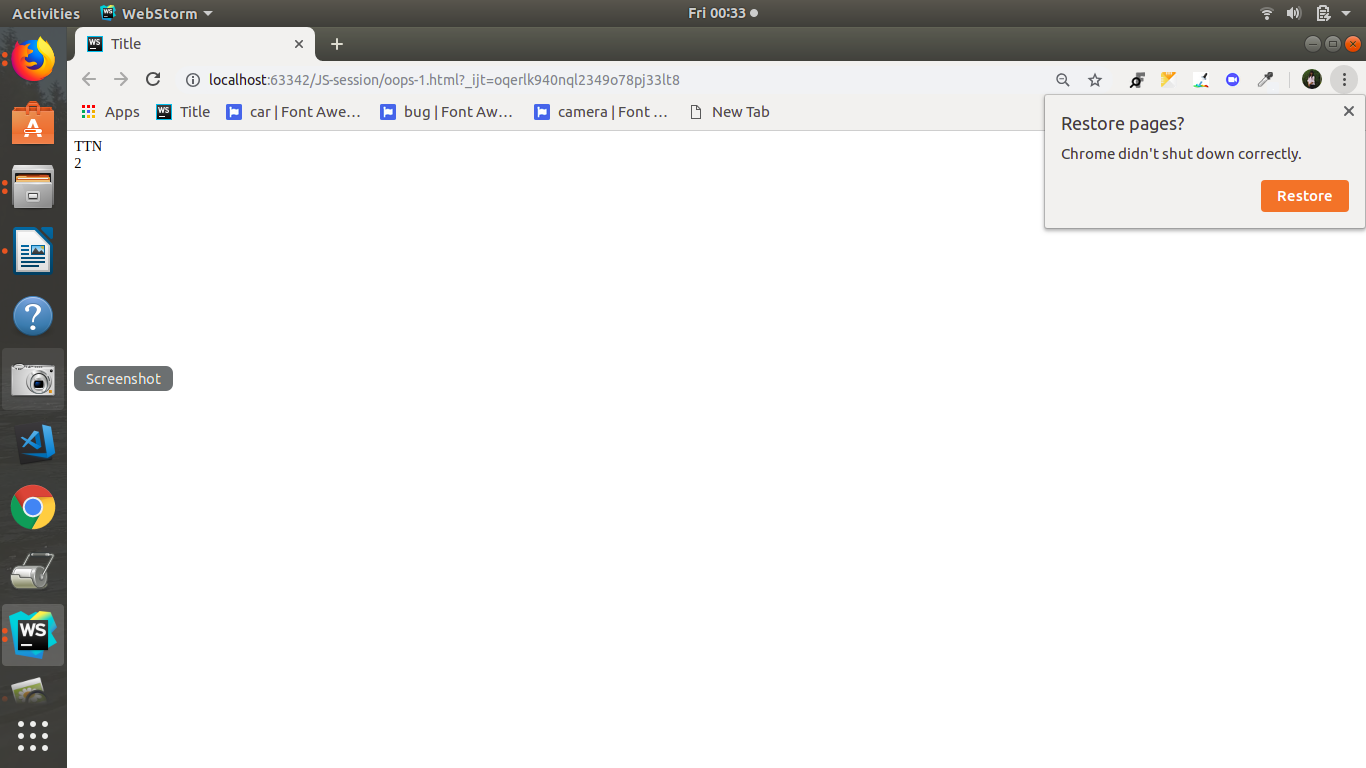
**var *d*=new *developers*();**

***document*.write(*d*.id);**

**</script>**

**</body>**

**</html>**

****

**Q2 Given an array, say [1,2,3,4,5]. Print each element of an array after 3 secs.**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<title>Title</title>**

**</head>**

**<body>**

**<script>**

**var arr= [1,2,3,4,5];**

**var i = 0;**

**(function loop() {**

**if (i < arr.length) {**

**console.log(arr[i]);**

**setTimeout(loop, 3000);**

**i++;**

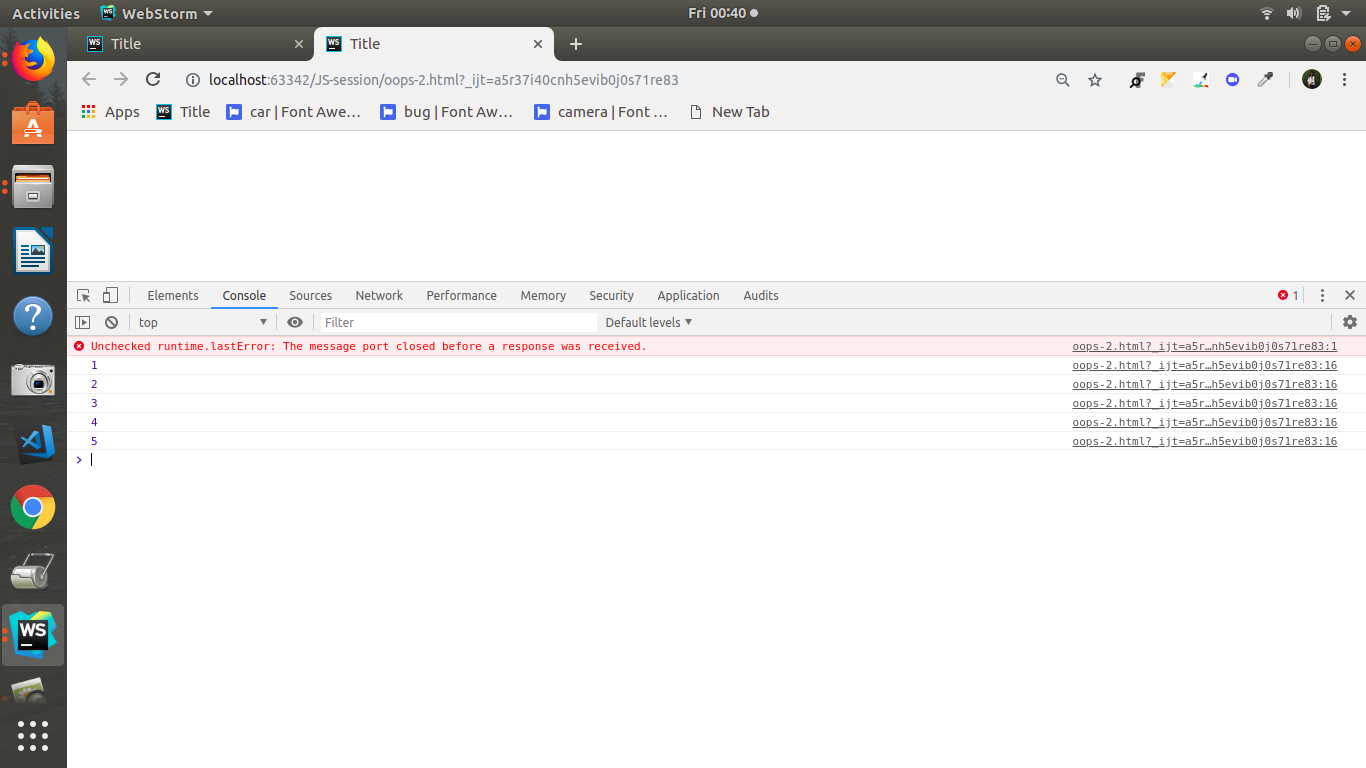
**}**

**})();**

**</script>**

**</body>**

**</html>**

****

**Q3. Explain difference between Bind and Call (example).**

#### **Call()**

call(this [, arg1, arg2...]): Calls a function with a provided this. Further arguments are provided as a comma separated list.

function logThis() {

console.log(this);

}

var obj = { val: 'Hello!' };

logThis(); // -> Window {frames: Window, postMessage: ƒ, …}

logThis.call(obj); // -> { val: 'Hello!' };

#### **Bind()**

bind(this): Returns a new function whose this value is bound to the provided value.

function addArguments(arg1, arg2) {

console.log(this);

console.log(arg1);

console.log(arg2);

}

var obj = { val: 'Hello!' };

var fnBound = addArguments.bind(obj, 'First arg', 'Second arg');

console.log(fnBound);

fnBound();

**Q4.Explain 3 properties of argument object.**

**a) [arguments.callee](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/arguments/callee)**

Reference to the currently executing function that the arguments belong to.

var global = this;  
  
var sillyFunction = function(recursed) {  
 if (!recursed) { return arguments.callee(true); }  
 if (this !== global) {  
 alert('This is: ' + this);  
 } else {  
 alert('This is the global');  
 }  
}  
  
sillyFunction();

**b)[arguments.caller](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/arguments/caller)**

Reference to the function that invoked the currently executing function.

function whoCalled() {  
 if (arguments.caller == null)  
 console.log('I was called from the global scope.');  
 else  
 console.log(arguments.caller + ' called me!');  
}

**c)[arguments.length](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/arguments/length)**

The number of arguments that were passed to the function.

function adder(base /\*, n2, ... \*/) {  
 base = Number(base);  
 for (var i = 1; i < arguments.length; i++) {  
 base += Number(arguments[i]);  
 }  
 return base;  
}

**Q5. Create a function which returns number of invocations and number of instances of a function.**

var myFuncCalls = 0;

function myFunction()

{

myFuncCalls++;

console.log( "I have been called " + myFuncCalls + " times" );

}

myFunction();

**Q6.Create counter using closures.**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width">

<title>JS Bin</title>

</head>

<body>

<button type="button" onclick="myFunction()">Count!</button>

<p id="demo">0</p>

<script>

var add = (function () {

var counter = 0;

return function () {counter += 1; return counter;}

})();

function myFunction(){

document.getElementById("demo").innerHTML = add();

}

</script>

</body>

</html>