**Asynchronous Programming**

Fucntionq2(){setTimeout( function(){console.log(“apples”);

setTimeout( function(){console.log(“bananas”);

setimeout(function(){console.log(“pears”);}, 1000);},3000);},2000);}

ANS: apples, bananas, pears

**Web workers**

// Post a message to the worker when the button is clicked document.getElementById("send").onclick = function(e) { document.getElementById('out').value = "";

var msg = document.getElementById("inp").value;

console.log("Main script sending " + msg + " to worker");

worker.postMessage(msg); //Send Message to worker };  
// Create the web worker object var worker = new Worker("upper.js"); // create worker  
// Handle any message returned from worker

worker.onmessage = function(e) { Revieve Message from worker

document.getElementById('out').innerHTML = e.data; }

// upper.js

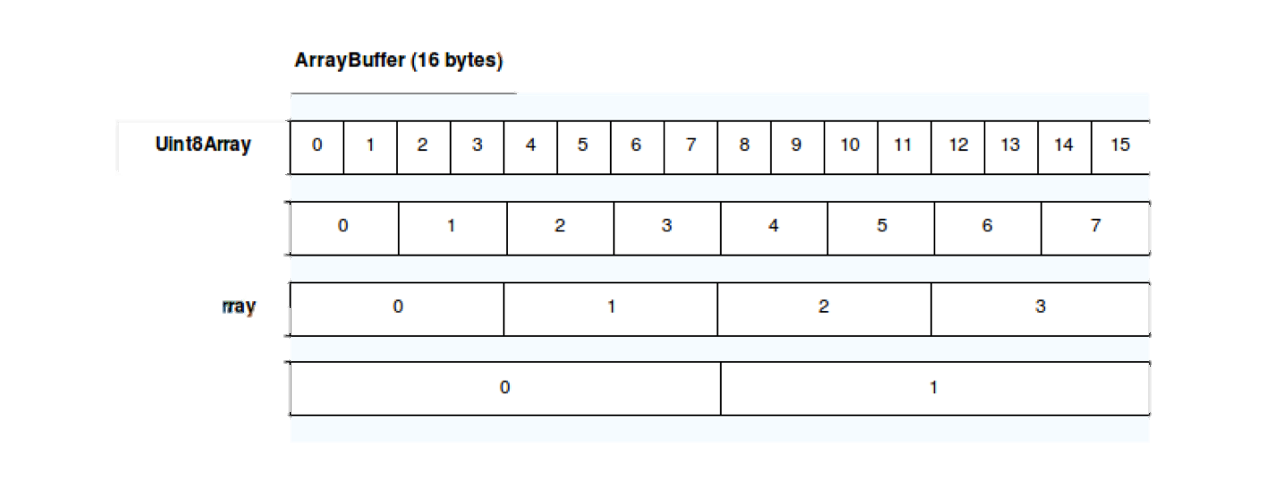
// Convert received strings to upper case

self.onmessage = function(e) { //receive message from main

console.log(self); var msg = e.data;

var msgUpper = msg.toUpperCase();

self.postMessage(msgUpper); }; //send message to main

**Array Buffer**

// Create a raw data buffer

var buf = new ArrayBuffer(64); console.log(buf.byteLength); // 64  
// 16 bit view on buffer

var arr16 = new Uint16Array(buf);  
// Length represents view

console.log(arr16.length); // 32  
// Create 8-bit unsigned view on buffer

var clamped = new Uint8ClampedArray(buf);

console.log(clamped.length); // 64  
// Attempt to assign values outside clamped range

clamped[0] = 256;

console.log(clamped[0]); // 255  
clamped[0] = -1; console.log(clamped[0]); // 0  
// Proof that both views access data in common Arraybuffer

clamped[0] = 100

console.log(arr16[0])

**Context**

Var doit = function(){

Console.log(this.name);};

Var name = “Ned”;

Var ol = {name:”Ollie”, doit:doit};

Ol.doit(); //Ollie

Var fn = ol.doit;

Fn(); //Ned

Var fn3 = doit.bind({name:’Sarah’})

Fn3(); // Sarah

var **Container** = function(vol){

this.maxVol = vol;

this.volume = vol;};

Containter.prototype.**fill** = function(vol){

this.volume += vol;

if (this.volume > this.maxVol) { this.volume = this.maxVol;}};

Container.prototype.**dump** = function(){

this.volume = 0; };

Container.prototype.**drink** = function(){

console.log(“Do not drink. May not be clean.”) };

var **DrinkContainer** = function(vol){

Container.call(this, vol); };

DrinkContainer.prototype = Object.create(Container.prototype);

DrinkContainer.prototype.**drink** = function(vol){

this.volume -= vol;

if (this.volume < 0) { this.volume = 0;} };

var **CoffeeMug** = function(){

DrinkContainer.call(this, 12); };

CoffeeMug.prototype = Object.create(DrinkContainer.protoype);

**Var bucket = new Container(128);**

console.log(bucket.valume); //128

bucket.drink(12);

console.log(bucket.volume); //Do now drink. May not be clean. 128

bucket.dump();

console.log(bucket.volume); //0

**Var cup = new DrinkContainer(16);**

console.log(cup.valume); //16

cup.drink(24);

console.log(cup.volume); //0

cup.dump();

console.log(cup.volume); //0

**Var mug = new CoffeeMug();**

console.log(mug.valume); //12

mug.drink(6);

console.log(mug.volume); //6

mug.dump();

console.log(mug.volume); //0

console.log( Object.getPrototypeOf(cup) === Container.prototype); 🡪false

console.log( Object.getPrototypeOf(cup) === DrinkContainer.porotype); 🡪true

console.log( Object.getPrototypeOf(cup) === CoffeeMug.prototype); 🡪false

console.log( cup instanceof Container); 🡪true

console.log(cup instanceof DrinkingContainer); 🡪true

console.log(cup instanceof CoffeeMug); 🡪false

console.log( Object.getPrototypeOf(bucket) === Container.prototype); 🡪true

console.log( Object.getPrototypeOf(bucket) === DrinkContainer.porotype); 🡪false

console.log( Object.getPrototypeOf(bucket) === CoffeeMug.prototype); 🡪false

console.log( bucket instanceof Container); 🡪true

console.log(bucket instanceof DrinkingContainer); 🡪false

console.log(bucket instanceof CoffeeMug); 🡪false

console.log( Object.getPrototypeOf(mug) === Container.prototype); 🡪false

console.log( Object.getPrototypeOf(mug) === DrinkContainer.porotype); 🡪false

console.log( Object.getPrototypeOf(mug) === CoffeeMug.prototype); 🡪true

console.log( mug instanceof Container); 🡪true

console.log(mug instanceof DrinkingContainer); 🡪true

console.log(mug instanceof CoffeeMug); 🡪true

console.log(mug.\_\_proto\_\_.\_\_proto\_\_.\_\_proto\_\_.\_\_proto\_\_.\_\_proto\_\_); 🡪 null