Examen Javascript

IES Luis Simarro, 2 DAW, Curs 2021-2022

Alumne:

Penja un document com aquest amb el codi copiar/pegat. No cal que entregues el JS a banda.

1 (1 punt) Observa aquest codi:

```
suma(10,20);
resta(20,10);

let suma = (a,b)=> a+b;
function resta(a,b){return a-b;}
```

Perquè fallarà? Escriu una solució.

Falla la cridada a la funció suma, perquè està declara en una expressió de funció després de ser cridada. Les expressions de funció assignen a una variable una funció i funcionen en temps d'execució, al contrari que les declaracions de funció, que tenen hoisting.

Les solucions poden ser o declarar-la abans o convertir-la en una declaració de funció.

```
let suma = (a,b)=> a+b;
suma(10,20);
resta(20,10);
function resta(a,b){return a-b;}

suma(10,20);
resta(20,10);
function suma(a,b){return a+b;}
function resta(a,b){return a-b;}
```

2 (1 punt) Explica línia per línia què passa en les variables en aquest codi i perquè:

```
function a(b,c){ console.log(b,c,d); return b+c; }
a(b=8,d=40);
console.log(b,c,d);
```

La primera línia és una funció que rep dos arguments: b i c i que intenta imprimir un altres argument (d) abans de retornar la suma de b i c.
En la segona línia s'invoca aquesta funció amb dos arguments: b i d. El problema és que la funció no està esperant a d, sino a c. La funció no falla perquè enten que ha d'agafar els arguments posicionalment. Així, b és b, c és el segon argument i d funciona perquè és declarada com a variable global al no tindre let ni var.
NO SON ARGUMENTS PER DEFECTE, ja que els arguments per defecte es claven al declarar la funció, no al invocar. El que està passant és símplement que estem declarant variables globals i de pas, passant-les a la funció.
En la tercera línia sols falla c, perquè té àmbit de funció. b i d funcionen perquè han sigut declarades de forma global. És a dir, ara són de l'objecte window

3 (1 punt) Observa el codi:

```
class Clock{
   constructor(){
        let hora = new Date();
        this.hour = hora;
        this.getDate = function(){return {hora: hora, hour: this.hour}};
   }
}

let reloj1 = new Clock();
let getDate = reloj1.getDate;
console.log(reloj1.getDate()); // 1er
console.log(getDate());        // 2on
```

Perquè funciona el primer console.log i no el segon?

El primer funciona perquè hora és una closure i this.hour pertany al context d'execcució, que és l'objecte reloj1.

El segon no funciona el this.hour, ja que el seu context d'execució no és un objecte que tinga hour. Si llevem el this.hour sí funciona perquè aquesta funció encara conserva la closure.

3 (7 punts) Amb aquestes dades de la climatología de Xàtiva en 2019:

```
let clima = [
 { month: 1, maxTemp: "16.3", minTemp: "3.7", meanTemp: null, maxTempF: "61.3", minTempF: "38.7", meanTempF: null, raindays: null, rainfall: "58.0", climateFromMemDate: "2019-05-13" },
 { month: 2, maxTemp: "17.7", minTemp: "4.4", meanTemp: null, maxTempF: "63.9", minTempF: "39.9", meanTempF: null, raindays: null, rainfall: "36.1", climateFromMemDate: "2019-05-13" },
 { month: 3, maxTemp: "21.0", minTemp: "6.2", meanTemp: null, maxTempF: "69.8", minTempF: "43.2", meanTempF: null, raindays: null, rainfall: "39.6", climateFromMemDate: "2019-05-13" },
 { month: 4, maxTemp: "23.3", minTemp: "8.3", meanTemp: null, maxTempF: "73.9", minTempF: "46.9", meanTempF: null, raindays: null, rainfall: "39.6", climateFromMemDate: "2019-05-13" },
 { month: 5, maxTemp: "26.7", minTemp: "11.6", meanTemp: null, maxTempF: "80.1", minTempF: "52.9", meanTempF: null, raindays: null, rainfall: "43.3", climateFromMemDate: "2019-05-13" },
 { month: 6, maxTemp: "31.5", minTemp: "15.9", meanTemp: null, maxTempF: "88.7", minTempF: "60.6", meanTempF: null, raindays: null, rainfall: "25.7", climateFromMemDate: "2019-05-13" },
 { month: 7, maxTemp: "34.6", minTemp: "19.0", meanTemp: null, maxTempF: "94.3", minTempF: "66.2", meanTempF: null, raindays: null, rainfall: "10.2", climateFromMemDate: "2019-05-13" },
 { month: 8, maxTemp: "34.4", minTemp: "19.6", meanTemp: null, maxTempF: "93.9", minTempF: "67.3", meanTempF: null, raindays: null, rainfall: "14.7", climateFromMemDate: "2019-05-13" },
 { month: 9, maxTemp: "30.6", minTemp: "16.7", meanTemp: null, maxTempF: "87.1", minTempF: "62.1", meanTempF: null, raindays: null, rainfall: "81.4", climateFromMemDate: "2019-05-13" },
 { month: 10, maxTemp: "25.5", minTemp: "12.5", meanTemp: null, maxTempF: "77.9", minTempF: "54.5", meanTempF: null, raindays: null, rainfall: "83.2", climateFromMemDate: "2019-05-13" },
 { month: 11, maxTemp: "20.0", minTemp: "7.8", meanTemp: null, maxTempF: "68.0", minTempF: "46.0", meanTempF: null, raindays: null, rainfall: "53.1", climateFromMemDate: "2019-05-13" },
 { month: 12, maxTemp: "16.8", minTemp: "5.0", meanTemp: null, maxTempF: "62.2", minTempF: "41.0", meanTempF: null, raindays: null, rainfall: "56.3", climateFromMemDate: "2019-05-13" },
];
```

- Crea una pàgina web generada dinàmicament en Javascript que mostre cada mes en un div.
- Crea la classe ***Mes*** amb un constructor adequat i una funció de renderitzat del mes.
- El color de fons de cada mes serà més o menys càlid en funció de la temperatura mitjana (entre la màxima i mínima).
- No cal mostrar les temperatures en Fahrenheit.
- Si la pluja d'aquest mes ha superat la mitjana de tot l'any, es mostrarà aquest caràcter damunt: 🌧️, en cas contrari, aquest: 🌞
- Utilitza un array amb el nom dels mesos per mostrar el nom en compte del número del mes.
- No pots utilitzar cap while(), for() o foreach(), sols pots utilitzar funcions d'alt ordre com .map(), filter() o reduce().
- Quan l'usuari passa el ratolí per damunt de cada mes, canvia l'estil CSS de manera que es note que està passant. Aquesta funcionalitat també serà declarada en la classe *Mes.*

```
(() => {
// prettier-ignore
const clima = [
    { month: 1, maxTemp: "16.3", minTemp: "3.7", meanTemp: null, maxTempF: "61.3", minTempF: "38.7", meanTempF: null, raindays: null, rainfall: "58.0", climateFromMemDate: "2019-05-13" },
    { month: 2, maxTemp: "17.7", minTemp: "4.4", meanTemp: null, maxTempF: "63.9", minTempF: "39.9", meanTempF: null, raindays: null, rainfall: "36.1", climateFromMemDate: "2019-05-13" },
    { month: 3, maxTemp: "21.0", minTemp: "6.2", meanTemp: null, maxTempF: "69.8", minTempF: "43.2", meanTempF: null, raindays: null, rainfall: "39.6", climateFromMemDate: "2019-05-13" },
    { month: 4, maxTemp: "23.3", minTemp: "8.3", meanTemp: null, maxTempF: "73.9", minTempF: "46.9", meanTempF: null, raindays: null, rainfall: "39.6", climateFromMemDate: "2019-05-13" },
    { month: 5, maxTemp: "26.7", minTemp: "11.6", meanTemp: null, maxTempF: "80.1", minTempF: "52.9", meanTempF: null, raindays: null, rainfall: "43.3", climateFromMemDate: "2019-05-13" },
    { month: 6, maxTemp: "31.5", minTemp: "15.9", meanTemp: null, maxTempF: "88.7", minTempF: "60.6", meanTempF: null, raindays: null, rainfall: "25.7", climateFromMemDate: "2019-05-13" },
    { month: 7, maxTemp: "34.6", minTemp: "19.0", meanTemp: null, maxTempF: "94.3", minTempF: "66.2", meanTempF: null, raindays: null, rainfall: "10.2", climateFromMemDate: "2019-05-13" },
    { month: 8, maxTemp: "34.4", minTemp: "19.6", meanTemp: null, maxTempF: "93.9", minTempF: "67.3", meanTempF: null, raindays: null, rainfall: "14.7", climateFromMemDate: "2019-05-13" },
    { month: 9, maxTemp: "30.6", minTemp: "16.7", meanTemp: null, maxTempF: "87.1", minTempF: "62.1", meanTempF: null, raindays: null, rainfall: "81.4", climateFromMemDate: "2019-05-13" },
    { month: 10, maxTemp: "25.5", minTemp: "12.5", meanTemp: null, maxTempF: "77.9", minTempF: "54.5", meanTempF: null, raindays: null, rainfall: "83.2", climateFromMemDate: "2019-05-13" },
    { month: 11, maxTemp: "20.0", minTemp: "7.8", meanTemp: null, maxTempF: "68.0", minTempF: "46.0", meanTempF: null, raindays: null, rainfall: "53.1", climateFromMemDate: "2019-05-13" },
    { month: 12, maxTemp: "16.8", minTemp: "5.0", meanTemp: null, maxTempF: "62.2", minTempF: "41.0", meanTempF: null, raindays: null, rainfall: "56.3", climateFromMemDate: "2019-05-13" },
  ];
```

```
const nomMes = [
  "",
  "Enero",
  "Febrero",
  "Marzo",
  "Abril",
  "Mayo",
  "Junio",
  "Julio",
  "Agosto",
  "Septiembre",
  "Octubre",
  "Noviembre",
  "Diciembre",
];

class Mes {
  constructor(datos, averageRain) {
    Object.assign(this, datos);
    this.meanTemp = (parseFloat(this.maxTemp) + parseFloat(this.minTemp)) / 2;
    this.nonMes = nomMes[this.month];
    averageRain < this.rainfall ? (this.emoji = "🌧️") : (this.emoji = "🌞");
```

```
  }
  render(container) {
    let divMes = document.createElement("div");
    divMes.classList.add("mes");
    divMes.innerHTML = `<span class="emoji">${this.emoji}</span>
      <h2>${this.nonMes}</h2>
      <p>Max Temp: ${this.maxTemp}</p>
      <p>Min Temp: ${this.minTemp}</p>
      <p>Avg Temp: ${this.meanTemp}</p>
      <p>Rainfall: ${this.rainfall}</p>`;
    divMes.style.backgroundColor = `hsl(${250 - this.meanTemp * 9},100%,70%)`;
    divMes.addEventListener("mouseenter", function () {
      this.classList.add("over");
    });
    divMes.addEventListener("mouseleave", function () {
      this.classList.remove("over");
    });
    container.append(divMes);
  }
}

document.addEventListener("DOMContentLoaded", () => {
  let averageRain =
    clima.reduce(
      (anterior, siguiente) => anterior + parseFloat(siguiente.rainfall),
      0
    ) / 12;
  //console.log(averageRain);
  let meses = clima.map((mes) => new Mes(mes, averageRain));
  let container = document.querySelector("#mesos");
  meses.map((m) => m.render(container));
});
})();
```

El CSS:

```
#mesos {
 display: flex;
 flex-wrap: wrap;
}
```

```css
.mes {
 min-width: 150px;
 padding: 20px;
 color: #111;
 border-radius: 10px;
 margin: 10px;
 font-family: Arial, Helvetica, sans-serif;
}

.emoji {
 font-size: 40px;
}

.over {
 margin: 0;
 border: 10px solid #ddd;
}
```