ACTIVITAT AVALUABLE AC3

Mòdul: MP06- Desenvolupament web en entorn client

UF: UF1 – Servidors web i de transferència de fitxers

Professor: Albert Guardiola

Data límit d'entrega: 14/10/2024 23:59

Mètode d'entrega: Per mitjà del Clickedu de l'assignatura. Les activitats entregades més enllà de la data límit només podran obtenir una nota de 5.

Instruccions: S'ha d'entregar un únic document amb el nom:

MP06-UF1-AC3-Nom_Alumne.js

Es valorará la presentació.

Resultats de l'aprenentatge:

RA1. Selecciona les arquitectures i tecnologies de programació sobre clients web, identificant i analitzant les capacitats i característiques de cadascuna.

RA2. Escriu sentències simples, aplicant la sintaxi del llenguatge i verificant la seva execució sobre navegadors web.

Tasca 1. Per què *str.test* retorna *undefined*?

```
let str = "Hello";

str.test = 5; // (*)

alert(str.test);
```

It returns undefined because str or string is a primitive type and we are trying to add test into it. String primitives does not have the ability to hold any data or property.

Tasca 2. Com s'expliquen els resultats d'aquestes comparacions d'igualtat?

```
let str = "Patata";
let str2 = new String("Patata");

console.log(str==str2);
console.log(str===str2);
```

```
true
false
```

== checks for the equality of the values which is the same so it comes out as TRUE

While === checks for both value and type equality, which in rthis case is FALSE because ste is a primitive while str2 is an object making them have different type therefore returning as FALSE.

Tasca 3. Per què aquest bucle és infinit?

```
let i = 0;
while (i != 10) {
  i += 0.2;
}
```

It is due to precision errors. 10 will never be exactly 10 because when we keep adding 0.2 to the i, there might be errors that can be inherited in the floating-point representation.

Tasca 4. Escriu una funció *checkSpam(str)* que retorni *true* si *str* conté 'viagra' o 'XXX', i *false* en cas contrari. La funció ha de ser *case-insensitive*:

```
checkSpam('buy ViAgRA now') == true
checkSpam('free xxxxx') == true
checkSpam("innocent rabbit") == false
```

```
function checkSpam(str) {
    const lowerStr = str.toLowerCase();
    return lowerStr.includes('viagra') || lowerStr.includes('xxxxx');
}

console.log(checkSpam('buy ViAgRA now'));
console.log(checkSpam('free xxxxx'));
console.log(checkSpam('innocent rabbit'));
```

```
true
true
false
```

Tasca 5. Escriu una funció *toAscii(str)* que retorni una llista dels codis ASCII dels caràcters de *str*. Escribiu una altra funció *toChar(chars)* que formi una cadena de text a partir de la llista de codis ASCII. Feu servir els mètodes *codePointAt* i *fromCodePoint*, respectivament.

```
console.log(toAscii("ALBERT"));
console.log(toChar([65, 76, 66, 69, 82, 84]));
```

```
(6) [65, 76, 66, 69, 82, 84]
ALBERT
```

```
function toAscii(str) {
    return Array.from(str).map(char => char.codePointAt(0));
}

function toChar(chars) {
    return String.fromCodePoint(...chars);
}

// Comprobar
console.log(toAscii("ALBERT"));
console.log(toChar([65, 76, 66, 69, 82, 84]));
```

toAscii(str): Converts a string into an array of ASCII values.

toChar(chars): Converts an array of ASCII values back into a string.

```
(6) [65, 76, 66, 69, 82, 84]
ALBERT
```

Tasca 6. Escriu una funció *formatDate(date)* que formategi *date* de la següent manera:

- **Si des de *date* ha passat menys d'1 segon, retornarà "right now".**
- **En cas contrari, si des de *date* ha passat menys d'1 minut, retornarà "n sec. ago".**
- **En cas contrari, si és menys d'una hora, reornarà "m min. ago".**
- **En cas contrari, retornarà la data completa en el format "DD.MM.YY HH:mm". És a dir: "day.month.year hours:minutes", tot en format de 2 dígits, p . ex 31.12.16 10:00.**

```
alert( formatDate(new Date(new Date - 1)) ); // "right now"

alert( formatDate(new Date(new Date - 30 * 1000)) ); // "30 sec. ago"

alert( formatDate(new Date(new Date - 5 * 60 * 1000)) ); // "5 min. ago"

// yesterday's date like 31.12.16 20:00
alert( formatDate(new Date(new Date - 86400 * 1000)) );
```

```
function formatDate(date) {
    const now = new Date();
    const diff = now - date; // Milliseconds

    if (diff < 1000) {
        return "right now";
    } else if (diff < 60 * 1000) {
        const seconds = Math.floor(diff / 1000);
        return `${seconds} sec. ago`;
    } else if (diff < 60 * 60 * 1000) {
        const minutes = Math.floor(diff / (60 * 1000));
        return `${minutes} min. ago`;
    } else {
        // Date format
        const day = String(date.getDate()).padStart(2, '0'); //retrieves the day of the month from the date
        const month = String(date.getMonth() + 1).padStart(2, '0'); //Months
        const year = String(date.getFullYear()).slice(-2); //Last 2 digitsof the year, like 24 in 2024
        const hours = String(date.getHours()).padStart(2, '0'); //retrieves the hours from the date
        const minutes = String(date.getMinutes()).padStart(2, '0'); // retrieves the minutes from the date

        return `${day}.${month}.${year} ${hours}:${minutes}`; //retrieves the full date and time into the specified format
    }
}
```

```
console.log(formatDate(new Date(new Date - 1)));
console.log(formatDate(new Date(new Date - 30 * 1000)));
console.log(formatDate(new Date(new Date - 5 * 60 * 1000)));
console.log(formatDate(new Date(new Date - 86400 * 1000)));
```

```
right now
30 sec. ago
5 min. ago
13.10.24 16:19
```

Tasca 7. a) Utilitza el mètode *Date.now()* per calcular quant de temps (en mil·lisegons) triga a executar-se el següent bucle:

```
for (let i = 0; i < 100000; i++) {
  let doSomething = i * i * i;
}
```

```
const Time = Date.now();

// Loop
for (let i = 0; i < 100000; i++) {
    let doSomething = i * i * i;
}

// Record the end time
const endTime = Date.now();

// Calculate the duration
const total = endTime - Time;

console.log(`It took ${total} milliseconds. :)`);
```

```
It took 1 millisecond/s. :)
```

Tasca 8. Per què es printa *false?*

```
let meetup = {
    title: "Conference",
    room: {
        number: 23,
        participants: ["john", "ann"]
    }
};

console.log(typeof(meetup)===typeof(JSON.stringify(meetup)));
```

Meetup returns an object and typeof(JSON.stringify(meetup)) returns a string. Therefore the two is not the same type making it false.

Tasca 9. Quina és la sortida? Per què?

```
let meetup = {
    title: "Conference",
    participants: [{ name: "John" }, { name: "Alice" }]
};

console.log(JSON.stringify(meetup,["participants"]));
```

```
{"participants":[{},{}]}
```

The outcome is like this because JSON.stringify is a method that is being told to only include only the specified properties when converting ["participants"] into a JSON string. When "meetup" is converted into JSON, only the properties of "participants" were included. But this also means that

the property inside "participants" are not directly included in the conversion. In return, they are converted into empty objects because the replacer didn't directly specify any properties inside the "participants".