ACTIVITAT AVALUABLE AC2

Mòdul: MP06- Desenvolupament web en entorn client

UF: UF1 – Servidors web i de transferència de fitxers

Professor: Albert Guardiola

Data límit d'entrega: 7/10/2024 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-AC2-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. Escriu una funció per sumar tots els salaris inclosos en un objecte *salaries* com aquest (l'objecte pot emmagatzemar un nombre indefinit de salaris):

```
let salaries = {
  John: 100,
  Ann: 160,
  Pete: 130
}
```

Tasca 2. Escriu un fragment de codi que demostri la diferència entre copiar un objecte:

a) per referència

b) per clonat

c) per clonat estructurat o recursiu

Tasca 3. Donada aquesta família *family*:

```
function marry(man, woman) {
  woman.husband = man;
  man.wife = woman;

  return {
    father: man,
    mother: woman
  }
}

let family = marry({
  name: "John"
}, {
  name: "Ann"
});
```

a) Què farà què esborrarà de memòria el *garbage collector* si fem:

```
delete family.father;
delete family.mother.husband;
```

b) I si fem:

```
family = null;
```

Tasca 5. a) A quin objecte fa referència *user.ref*?

```
function makeUser() {
    return {
        name: "John",
        ref: this
    };
}

let user = makeUser();

console.log(user.ref);
```

b)Com podríem fer que *user.ref* apunti sempre a *user*?

Tasca 6. Crea una calculadora amb els mètodes read, sum i mul.

- `read()` demana dos valors i els desa com a propietats d'objecte amb noms `a` i `b` respectivament.
- `sum()` retorna la suma dels valors desats.
- `mul()` multiplica els valors desats i retorna el resultat.

El mètode *read()* pot fer servir la funció predefinida *prompt()*.

```
let calculator = {
  // ... your code ...
};

calculator.read();
alert( calculator.sum() );
alert( calculator.mul() );
```

Tasca 7. Heus aquí un objecte *ladder* que et permet anar amunt i avall:

```
let ladder = {
  step: 0,
  up() {
    this.step++;
  },
  down() {
    this.step--;
  },
  showStep: function() {
    alert( this.step );
  }
};
```

Si hem de fer diverses crides sequencialment, les fem així:

```
ladder.up();
ladder.up();
ladder.down();
ladder.showStep(); // 1
ladder.down();
ladder.showStep(); // 0
```

Modifica les funcions *up*, *down* i *showStep* perquè es puguin fer les crides encadenades, així:

```
ladder.up().up().down().showStep().down().showStep(); // shows 1 then 0
```

Tasca 8. Per què *str.test* retorna *undefined*?

```
let str = "Hello";

str.test = 5; // (*)

alert(str.test);
```

Tasca 9. Per què aquest bucle és infinit?

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

Tasca 9. Escriu una funció *checkSpam(str)* que retorni *true* si *str* conté 'viagra' o 'XXX', i *false* en cas contrari.

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

Tasca 10. 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)) );
```

Tasca 11. 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;
}
```

b)Per fer un *bechmarking* més complet, programa una funció que faci la mesura N vegades i promitgi els resultats.