EJERCICIOS DE EXPRESIONES 2

Strings

1. Dadas las siguiente expresiónes en JavaScript, para cada una aplicando las reglas de orden de agrupación (precedencia y asociatividad) y orden de evaluación:
   - pon paréntesis de forma que, sin modificar su funcionalidad, haga claro en qué orden se evalúa.
   - indica el valor resultante de la expresión y de las variables, si las hay

a)
```
console.log('una línea \n otra
línea');
```

b)
```
console.log("una línea \n otra
línea");
```

c)
```
let a = "hola mundo".length ** 2;
```

d)
```
console.log(`En JavaScript, las cadenas
plantilla pueden recorrer
varias líneas`);
```

e)
```
'Hola mundo'[2];
```

f)
```
'Hola mundo'.split(" ")[1][0];
```

g)
```
`Hola
mundo`.split(" ")[1][0];
```

h)
```
const nombre = 'Juan', ciudad =
'Madrid';
console.log(`Don ${nombre} es de
${ciudad}`);
```

i)
```
console.log('una línea \'otra
línea\'');;
```

j)
```
console.log('Su directorio es c:\\temp');
```

k)
```
console.log('Varias \
líneas en una\
cadena');
```

Arrays

2. Dadas las siguiente expresiónes en JavaScript, para cada una aplicando las reglas de orden de agrupación (precedencia y asociatividad) y orden de evaluación:
   - pon paréntesis de forma que, sin modificar su funcionalidad, haga claro en qué orden se evalúa.
   - indica el valor resultante de la expresión y de las variables, si las hay

a)
```
let a1 = ['Lucas', 'Ainoa', 'Xian'];
let r = a1[0] + " " + a1[2];
```

b)
```
let a2 = [1, '', '3'];
let r = a2[0] + a2[1] + a2[2];
```

c)
```
let a3 = [1, '', '3'];
let r = a3[0] - a3[1] - a3[2];
```

d)
```
let a4 = [1, '0', '3'];
let r = a4[0] + a4[1] - a4[2];
```

e)
```
let a5 = [1,  , '3'];
let r = a5[0] + a5[1] + a5[2];
```

f)
```
let a6 = [1,  , '3'];
let r = a6[0] + a6[1] - a6[2];
```

g)
```
let a7 = [1,  ,  , 3, ];
let r = a7.length;
```

h)
```
let a8 = [1, 3 , '3'];
let r = a8.length;
```

i)
```
let a9 = [1,  ,  , 3, ];
let r = [...a9 ];
```

j)
```
let a10 = [1,  ,  , 3, ];
let o = {...a10};
```

k)
```
let a11 = [ [1,2,3],
            [ [3,2] ,[5,6]] ,
            [7, ,8], ];
let r1 = a11[0];
let r2 = a11[1][1][1];
let r3 = a11[2][1];
let r4 = [...a11];
let o  = {...a11};
```

l)
```
let [a, b, c] = [1, 2, 3];
```

m)
```
let {x, y, z} = {1, 2, 3};
```

n)
```
let o = {...["juan", "luis", "moni"] };
o[0]
o[1]
```

Objetos

3. Dadas las siguiente expresiónes en JavaScript, para cada una aplicando las reglas de orden de agrupación (precedencia y asociatividad) y orden de evaluación:
   - pon paréntesis de forma que, sin modificar su funcionalidad, haga claro en qué orden se evalúa.
   - indica el valor resultante de la expresión y de las variables, si las hay

```
a) let o1 = {};

b) let o2 = {e1:{}, e2:{}, e3:{}};

c) let o3 =
     {e1:{e11:{}, e12:{}},
      e2:{},
      e3:{e3e1:{}, e3e2: { e3e2e1:{}}}
     };

d) let o4 = {0: "hola", 1: "adios", 3:
   "mañana", c4: "hoy" };
   console.log(o4[1]);
   console.log(o4.1);
   console.log(o4.c4);
   delete o4.c4;

e) let o5 = {0: "hola", 1: "adios", 2:
   "mañana", 4: "hoy" };
   for (let i = 0; i<= 4; i++) {
      console.log(o5[i]);
   }

f) let o6 = {3: "hola",
             1: "adios",
             4: "mañana",
             2: "hoy" };
   for (let i = 0; i<= 4; i++) {
      console.log(o6[i]);
   }
   delete o6[4];
```

```
g) let o7 = { "hola": 34,
              '': [1, 3],
              ":": {c1: 3, c2: 1},
            };
   o7."hola"
   o7["hola"]
   o7.''
   o7[""]

o) let o8 = [ {c1:[1,2], 2: [3, 4]} ,
              {c1:"hola", "": "adios"} ,
              { 3: [ { c1: [1,2]}, 3] },
              { 'clave' : "hola"},
            ];
   o8[0].c1[1]
   o8[0][2][1]
   o8[1].c1[2]
   o8[1][""][1]
   o8[2][3][0].c1[1]
   delete o8[2][3][0].c1[1]
```

Igualdades

4. Dadas las siguiente expresiónes en JavaScript, para cada una aplicando las reglas de orden de agrupación (precedencia y asociatividad) y orden de evaluación:
   - pon paréntesis de forma que, sin modificar su funcionalidad, haga claro en qué orden se evalúa.
   - indica el valor resultante de la expresión y de las variables, si las hay

```
a) [] == false

b) [] == ''

c) [] === false

d) [] === ''

e) (0 && "perro") == false
```

```
f) (0 && "perro") == false

g) (true && "perro") == "perro"

h) (true && "perro") === "perro"

i) ([] && "perro") == "perro"
```

Casos raros

5 Dadas las siguiente expresiónes en JavaScript, para cada una aplicando las reglas de orden de agrupación (precedencia y asociatividad) y orden de evaluación:
   - escríbela de nuevo con paréntesis de forma que, sin modificar su funcionalidad, haga claro en qué orden se evalúa.
   - indica el valor resultante de la expresión y de las variables, si las hay
   - en caso de que no sea una expresión válida, indica el por qué.

a) ({}).toString();
 	
b) {}.toString();
 	
c) [].toString();
 	
d) ([]).toString();
	e) (+[])+1
	
	f) ([])*-1
	
	g) +({}) != +({})
	
	h) typeof (Array instanceof Object);