**TEST**

1. Cuál/es de las siguientes formas sería correcta a la hora de acceder a la propiedad edad del objeto persona?
2. persona[edad]
3. persona[‘edad’]
4. persona::edad
5. persona.edad
6. Cuál de las siguientes opciones sería la forma correcta de definir e invocar un método que visualice la propiedad ‘nombre’ del objeto ‘persona’.
7. persona.mostrarNombre = function () {alert(this.nombre);}

persona.mostrarNombre();

1. persona.mostrarNombre = new Function() {alert(nombre);}

persona.mostrarNombre();

1. persona.mostrarNombre = new Function ("return this.nombre")

persona.mostrarNombre();

1. A y c son correctas
2. Sea el siguiente código:

var arr = [

{

nombre: 'Ana',

edad: 29,

hobbies: ['Tennis', 'Swimming', 'Painting']

},

{

nombre: 'Juan',

edad: 60,

hobbies: ['Football', 'Golf', 'Programming']

}

];

Cual de las siguientes órdenes permite recuperar el último hobbie de Juan?

1. arr[1][2][2];
2. arr[2].hobbies[3];
3. arr.hobbies[2];
4. arr[1]["hobbies"][2];
5. arr["Juan"].hobbies["Programming"];
6. Qué devuelve el siguiente código?

var f = function g(){

return 23;

};

typeof f();

1. Number
2. Function
3. Undefined
4. Error
5. Sea el siguiente código:

function cero () { return {}; }

var x = cero;

Tras ejecutarlo, qué almacena la variable x?

1. 0
2. Un objeto Funcion
3. Un objeto {}
4. Undefined
5. Se produce un error
6. Qué devuelve el código siguiente:

(function(x, f = () => x)

{

x;

var y = x;

x = 2;

return [x, y, f()];

})(1)

1. [2,1,2]
2. [2,undefined,2]
3. [2,undefined,1]
4. [2,1,1]
5. Indicar el resultado de evaluar la siguiente expresión, después de haber ejecutado las siguientes sentencias:

var pan = {blanco:6, de\_ajo:5, de\_semillas:{de\_avena:3, de\_lino:4}};

delete pan.de\_ajo;

pan.de\_queso = 3;

var p = "blanco";

for (var x in pan.de\_semillas) { pan.de\_semillas[x] = 6;};

Expresión: pan.de\_semillas

* 1. {3,4}
  2. {6,6}
  3. {de\_avena:6,de\_lino:6}
  4. Undefined

1. Sea el siguiente código:

var length = 10;

function fn() {

console.log(this.length);

}

var obj = {

length: 5,

method: function(fn) {

fn();

arguments[0]();

}

};

undefined

obj.method(fn, 1);

Que devuelve el código anterior?

**EJERCICIOS**

1. Objetos y clases.
2. Crea un objeto literal alumno con las siguientes propiedades y valores:
   * Nombre: “juan”
   * Apellidos: “perez”
   * Dni: “11222333A”,
   * Modulos: “prog”,”code”,”dwcs”,”dwcc” (será un array).
   * Provincia: “Ourense”

Visualiza el nombre y el primero de los módulos en los que está matriculado.

1. Crea el objeto alumno empleando un constructor (function):
2. Crea la clase Alumno en un fichero .js independiente. Los atributos son los indicados en el apartado anterior.

Define los métodos que creas adecuados para trabajar con este objeto (getters/setters para poder cambiar los valores de los atributos de las instancias). En la definición del constructor la provincia será “Ourense” por defecto.

1. Instancia la clase definida en el apartado anterior, con los siguientes valores:

Nombre: “juan”

Apellidos: “perez”

Dni: “11222333A”,

Modulos: “prog”,”code”,”dwcs”,”dwcc”

Recupera esos valores empleando los métodos de la clase y visualizalos en la página html

La clase y funciones relacionadas deberán situarse en un fichero .js independiente.

1. Implementa la clase **NumeroRacional** en un fichero .js independiente. Además del constructor añade los getters/setters que necesites. Incluye también en la clase el método simplificar, que se encargará de reducir el número racional.

En un fichero html, instancia la clase e invoca a los distintos métodos.

1. Añade a la clase creada en el apartado anterior métodos que permitirán realizar las operaciones de suma, resta, multiplicación y división de dos números racionales. ¿De qué tipo serán estos métodos?
2. Se desea crear un array de objetos de tipo NumeroRacional.
3. Para ello, realiza una función que permita la introducción de datos,
   * pidiendo el numerador y denominador de cada uno los números racionales que vayan a almacenarse en el array.
   * Instanciando la clase que has creado en el apartado 4, mediante una llamada a su constructor.
   * La función deberá devolver el objeto anterior
4. Implementa una función que cree un array y que solicitará al usuario el número de elementos a almacenar e invocará a la función que permite introducir los datos definida en el apartado anterior.
5. Herencia.
   * 1. Crear la clase Persona. Las propiedades para esta clase serán: nombre, apellidos y fechaNacimiento. Añadir los métodos getters/setters necesarios para modificar las propiedades y crear un método CalcularEdad (que devolverá la edad a partir de la fecha de nacimiento del individuo).
     2. Crear la clase Empleado que heredará de la clase anterior. Añadir además el método TiempoHastaLaJubilación (que devolverá los años que le quedan para jubilarse).
     3. Crea la clase contrato cuyas propiedades serán (fecha de alta, descripción, tipo, duración y fecha de baja). Añade los métodos getters/setters correspondientes a la clase.
     4. Crea un objeto de tipo Empleado y llama al método TiempoParaJubilarse. Se considera que la edad de jubilación son 67 años.
     5. Añade varios contratos al objeto anterior.