**JAVASCRIPT MAP Y SET.**

**MAP (Mapas).**

**Concepto y características.**

- Estructura de datos nativa que permite almacenar una colección de pares clave-valor.
- La clave permite identificar a los valores.
- Se diferencia de un objeto en que las claves pueden ser de cualquier tipo de datos, mientras que, en los objetos, las propiedades o claves son cadenas de caracteres (string).
- No obstante, lo más común es que las calves de un mapa sean cadenas de caracteres.
- Un mapa puede convertirse en un array y viceversa.

**Creación.**

- **Mapa vacío.**
  - El constructor de objeto Map no incluye valores.
  - Sintaxis:
    - var / let / const nombre del objeto map = new Map();
    - El objeto iterable puede ser
  - Ejemplo:
    - let miMapa0 = new Map();
- **Mapa con datos.**
  - El constructor de objeto Map incluye valores.
  - Sintaxis:
    - var / let / const nombre del objeto map = new Map(iterable);
    - El objeto iterable puede ser un array o cualquier objeto compuesto de pares claves-valor.
  - Ejemplos:
    - let miMapa1 = new Map([[“nombre”,”Cristina”],[“edad”,24]]); // Las claves son string y los valores cadenas y números.
    - let miMapa2 = new Map([[1,”JavaScript”],[2,”PHP”],[3,”C++”]]); // Las claves son números y los valores son cadenas.

**Propiedades del objeto Map.**

- **size.**
  - Propiedad que muestra cuantos pares clave-valor tiene un mapa determinado.
  - Si el mapa está vacío devuelve 0.
  - Sintaxis:
    - instancia map.size;
  - Ejemplos.
    - alert(miMapa0.size); // devuelve 0.
    - alert(miMapa1.size); // devuelve 2.
    - alert(miMapa2.size); // devuelve 3.

**Métodos del objeto Map.**

- **set ().**
  - Añade un elemento a la colección de pares clave-valor de un mapa determinado.
  - Se pueden añadir varios pares clave-valor encadenado varios métodos set() con el operador punto (.)
  - Si se repite una clave se sobrescribe su valor.
  - Permite cargar un mapa vacío o añadir nuevos elementos a un mapa con contenido.
  - Sintaxis:
    - Añadir un valor:
      - instancia map.set (“clave”, “valor”); // Con o sin comillas según tipo de dato.
    - Añadir varios valores:
      - instancia set.set(“clave1”, “valor1”).set(“clave2”, “valor2”).set(“claveN”, “valorN”); // Con o sin comillas según tipo de dato
  - Ejemplos.
    - Añadir un valor:
      - miMapa0.set (“Idioma1”, “Inglés”); // Con o sin comillas según tipo de dato.
    - Añadir varios valores:
      - instancia Mapa0.set(“Idioma2”, “Francés”).set(“idioma3”,”Alemán”); // Con o sin comillas según tipo de dato.
    - miMapa2.set (4, ”Perl”);
    - document.write(miMapa0.size); // devuelve 3.
    - document.write(miMapa2.size); // devuelve 4.
    -
- **get ().**
  - Muestra el valor del par especificado con su clave.
  - Si la clave no existe, devuelve *undefined*.
  - Sintaxis:
    - instancia map.get (“clave”); // Con o sin comillas según tipo de dato.
  - Ejemplos.
    - document.write(miMapa1.get(“edad”)); // Devuelve 24.
    - document.write(miMapa1.get(“dirección”)); // Devuelve undefined.
    - document.write(miMapa2.get(2)); // Devuelve PHP.
    - let clave = miMapa2(miMapa0.get(3));
    - document.write(clave); // Devuelve C++.
- **has().**
  - Indica si existe el elemento con la clave especificada.
  - Devuelve un valor booleano (true, false).
  - Sintaxis:
    - instancia map.has (“clave”); // Con o sin comillas según tipo de dato.
  - Ejemplos.
    - document.write(miMapa1.has(“nombre”)); // Devuelve true.
    - document.write(miMapa2.has(12)); // Devuelve false.

- Con if …else:

```
if(miMapa1.has(“teléfono”))
{
        document.write("La clave teléfono está incluida en miMapa1.");
}
else
{
        document.write("La clave teléfono no está incluida en miMapa1.");
}
```

- **delete()**
  - Elimina el elemento (clave-valor) asociado a la clave especificada.
  - Devuelve un valor booleano:
    - Si el elemento existe y es eliminado devuelve *true*.
    - Si el elemento no existe devuelve *false*.
  - Sintaxis:
    - instancia map.delete (“clave”); // Con o sin comillas según tipo de dato.
  - Ejemplos.
    - document.write(miMapa1.delete(“nombre”)); // Devuelve true.
    - document.write(miMapa2.delete(10)); // Devuelve false.
    - document.write(miMapa1.has(“nombre”)); // False. Confirmación de eliminación.
    - document.write(miMapa2.has(10)); // False. Confirmación de eliminación.
    - Con if …else:

```
if(miMapa1.delete(“email”))
{
    document.write("Elemento asociado a la clave email en miMapa1 eliminado.");
}
else
{
    document.write("Clave email y su valor asociado no existen en miMapa1.");
}
```

- **clear()**
  - Elimina todos los elementos (clave-valor) del mapa especificado.
  - No devuelve nada.
  - Sintaxis:
    - instancia map.clear ();
  - Ejemplos.
    - miMapa2.clear();
    - document.write(miMapa2.size); // Devuelve 0.
- **keys()**
  - Devuelve un iterador que contiene todas las claves del mapa especificado.

- Este iterador se puede, o no, guardar en una variable.
- Si se guarda en una variable iteradora, al mostrar su contenido solo aparecerá la última clave, por eso es necesario usar el método next() y la propiedad value para ir mostrando las claves devueltas una a una:
  - next().value;
- Si no se utiliza una variable iteradora, se puede usar un bucle for…of para recorrer y mostrar las claves (no for…in o for clásico).
- Sintaxis:
  - instancia map.keys();
- Ejemplos:
  - Con variable iteradora:
    - let iterador = miMapa1.keys();
    - document.write(iterador.next().value); // Devuelve nombre.
    - document.write(iterador.next()value); // Devuelve edad.
  - Sin variable iteradora y bucle for…of:

```
for(let i of miMapa1.keys())
{
        document.write("Clave: " + i + "<br>");
}
```

- **values()**
  - Devuelve un iterador que contiene todos los valores del mapa especificado.
  - Este iterador se puede, o no, guardar en una variable.
  - Si se guarda en una variable iteradora, al mostrar su contenido solo aparecerá el último valor, por eso es necesario usar el método next() y la propiedad value para ir mostrando los valores devueltos uno a uno:
    - next().value;
  - Si no se utiliza una variable iteradora, se puede usar un bucle for…of para recorrer y mostrar los valores (no for…in o for clásico).
  - Sintaxis:
    - instancia map.values();
  - Ejemplos:
    - Con variable iteradora:
      - let valores = miMapa1.values();
      - document.write(valores.next().value); // Devuelve Cristina.
      - document.write(valores.next()value); // Devuelve 24.
    - Sin variable iteradora y bucle for…of.

```
for(let i of miMapa1.values())
{
        document.write("Valor: " + i );
}
```

- **Mostrar todos los pares clave-valor contenidos en un mapa.**
  - Se usa un bucle for…of especificando sólo el nombre del mapa.
  - Sintaxis:
    - Opción 1

```
for(var/let/const variable iteradora of Instancia Map)
{
        Instrucciones;
}
```

- Opción 2**:**

```
for(var/let/const [identificador clave, identificador valor] of Instancia Map)
{
        Instrucciones;
}
```

- Ejemplo:
  - Opción 1**:**

```
for(let i of miMapa1)
{
        document.write("Elemento: " + i );
}
```

  - Opción 2**:**

```
for( let [posición, lenguaje] of miMapa2)
{
        Instrucciones;
}
```

- **forEach().**
  - Permite aplicar una función a cada uno de los elementos (par clave-valor) del mapa especificado.
  - En la función a aplicar se incluyen los siguientes parámetros en este orden:
    - Un parámetro para representar al valor.
    - Un parámetro para representar a la clave.
    - Un parámetro para representar al mapa (opcional).
  - Sintaxis:
    - Función a aplicar:

```
function nombre de la función (valor, clave, mapa)
{
        Instrucciones;
}
```

      Método forEach():

```
instancia map.forEach(función a aplicar a cada elemento);
```

  - Ejemplo:
    - Función a aplicar:

```
function mostrarElementos(valor, clave)
{
   document.write("La clave “ + clave + “ tiene el valor” + valor);
}
```

        - Función forEach:
            miMapa3.forEach(mostrarElementos);

**SET (Conjuntos).**

**Concepto y características.**

- Estructura de datos nativa que permite almacenar una colección de valores únicos de cualquier tipo.
- No permite datos repetidos, si se intenta introducir en un set un dato que ya existe, éste no se insertará.
- De esta forma, utilizando este tipo de objeto, uno se asegura de que no va a haber datos duplicados.
- Un set puede convertirse en un array y viceversa.

**Creación.**

- **Set vacío.**
    - El constructor de objeto Set no incluye valores.
    - Sintaxis:
        - var / let / const nombre del objeto set = new Set();
        - El objeto iterable puede ser
    - Ejemplo:
        - let miSet0 = new Set();
- **Mapa con datos.**
    - El constructor de objeto Set incluye valores.
    - Sintaxis:
        - var / let / const nombre del objeto set = new Set(iterable);
        - El objeto iterable puede ser un array o cualquier objeto compuesto de pares claves-valor.
    - Ejemplos:
        - let miSet1 = new Set([“Cristina”,24]]); // Los valores son cadenas y números.
        - let miSet2 = new Set([[”JavaScript”,”PHP”,”C++”]); // Todos los valores son cadenas.

**Propiedades del objeto Map.**

- **size.**
    - Propiedad que muestra cuantos valores tiene un set determinado.
    - Si el set está vacío devuelve 0.
    - Sintaxis:
        - instancia set.size;
    - Ejemplos.
        - alert(miSet0.size); // devuelve 0.
        - alert(miSet1.size); // devuelve 2.
        - alert(miSet2.size); // devuelve 3.

**Métodos del objeto Map.**

- **add ().**
  - Añade un nuevo elemento a la colección de valores de un set determinado.
  - Se pueden añadir varios valores encadenado varios métodos add() con el operador punto (.)
  - Permite cargar un set vacío o añadir nuevos elementos a uno con contenido.
  - Si el valor ya existe, no lo agregará.
  - Sintaxis:
    - Añadir un valor:
      - instancia set.add (“valor”); // Con o sin comillas según tipo de dato.
    - Añadir varios valores:
      - instancia set.add (“valor1”).add(“valor2”).add(“valorN”); // Con o sin comillas según tipo de dato.
  - Ejemplos.
    - miSet0.set (“Inglés”);
    - miSet2.set ( ”PHP”);  // No lo agrega porque ya existe.
    - miSet0.set (“Japonés”).add(“Francés”);
    - document.write(miSet0.size); // devuelve 3.
    - document.write(miSet2.size); // devuelve 3.
- **has().**
  - Indica si existe el elemento con el valor especificado.
  - Devuelve un valor booleano (true, false).
  - Sintaxis:
    - instancia set.has (“valor”); // Con o sin comillas según tipo de dato.
  - Ejemplos.
    - document.write(miSet1.has(“Cristina”)); // Devuelve true.
    - document.write(miSet2.has(5)); // Devuelve false.
    - Con if …else:

```
if(miSet1.has(“28967”))
{
        document.write("El valor 28967 está incluido en miSet1.");
}
else
{
        document.write("("El valor 28967 no está incluido en miSet1.");
}
```

- **delete()**
  - Elimina el elemento con el valor especificado.
  - Devuelve un valor booleano:
    - Si el elemento existe y es eliminado devuelve *true*.
    - Si el elemento no existe devuelve *false*.
  - Sintaxis:
    - instancia set.delete (“valor”); // Con o sin comillas según tipo de dato.
  - Ejemplos.
    - document.write(miSet1.delete(“Cristina”)); // Devuelve true.

        - document.write(miSet2.delete(10)); // Devuelve false.
        - document.write(miSet1.has("Cristina")); // False. Confirmación de eliminación.
        - document.write(miSet.has(10)); // False. Confirmación de eliminación.
        - Con if …else:

```
if(miSet1.delete("ordenador"))
{
    document.write("El elemento asociado al valor ordenador en miSet1 ha sido eliminado.");
}
else
{
    document.write("El elemento asociado al valor ordenador en miSet1 no existe.");
}
```

- **clear()**
    - Elimina todos los elementos del set especificado.
    - No devuelve nada.
    - Sintaxis:
        - instancia set.clear ();
    - Ejemplos.
        - miSet2.clear();
        - document.write(miSet2.size); // Devuelve 0.
- **values()**
    - Devuelve un iterador que contiene todos los valores del set especificado.
    - Este iterador se puede, o no, guardar en una variable.
    - Si se guarda en una variable iteradora, al mostrar su contenido solo aparecerá el último valor, por eso es necesario usar el método next() y la propiedad value para ir mostrando los valores devueltos uno a uno:
        - next().value;
    - Si no se utiliza una variable iteradora, se puede usar un bucle for…of para recorrer y mostrar los valores (no for…in o for clásico).
    - Sintaxis:
        - instancia set.values();
    - Ejemplos:
        - Con variable iteradora:
            - let valores = miSet1.values();
            - document.write(valores.next().value); // Devuelve Cristina.
            - document.write(valores.next()value); // Devuelve 24.
        - Sin variable iteradora y bucle for…of.

```
for(let i of miSet1.values())
{
        document.write("Valor: " + i );
}
```

- **forEach().**
  - Permite aplicar una función a cada uno de los elementos del set especificado.
  - En la función a aplicar se incluyen los siguientes parámetros en este orden:
    - Un parámetro para representar al valor.
    - Un segundo parámetro no devuelve una clave, ya que éstas no existen en un set, sino que repite el valor (opcional).
    - Un parámetro para representar al mapa (opcional).
  - Sintaxis:
    - Función a aplicar:

```
function nombre de la función (valor, clave, mapa)
{
        Instrucciones;
}
```

      Método forEach():

```
instancia set.forEach(función a aplicar a cada elemento);
```

  - Ejemplo:
    - Función a aplicar:

```
function mostrarValores(valor)
{
   document.write("Valor: “ + valor);
}
```

    - Función forEach:

```
miSet3.forEach(mostrarValores);
```