

Arrays

- La mayor parte de las aplicaciones web gestionan un número elevado de datos.
- Por ejemplo si se quisiera definir el nombre de 180 productos alimenticios:

```
var producto1 = "Pan";  
var producto2 = "Agua";  
var producto3 = "Lentejas";  
var producto4 = "Naranjas";  
var producto5 = "Cereales";  
...  
var producto180 = "Salsa agridulce";
```

Arrays

- Si posteriormente se quisiera mostrar el nombre de estos productos:

```
document.write(producto1);  
document.write(producto2);  
document.write(producto3);  
document.write(producto4);  
document.write(producto5);  
...  
document.write(producto180);
```

- Esto sería una tarea compleja, repetitiva y propensa a errores.
- Para gestionar este tipo de escenarios se pueden utilizar los arrays.

Arrays

- Un array es un conjunto ordenado de valores relacionados almacenados en una misma variable.
- Cada uno de estos valores se denomina elemento y cada elemento tiene un índice que indica su posición numérica en el array
 - Los arrays se referencian con un índice numérico mientras que los objetos se referencian con un nombre
- Puede almacenar diferentes tipos de datos. Los tipos de los valores de cada elemento que conforman el array pueden ser heterogéneos: valores primitivos, objetos, etc..

Arrays

- Declaración de arrays:
 - Al igual que ocurre con las variables, es necesario declarar un array antes de poder usarlo.
 - La declaración de un array consta de seis partes:
 - La palabra clave `var`.
 - El nombre del array.
 - El operador de asignación.
 - La palabra clave para la creación de objetos `new`.
 - El constructor `Array`.
 - El paréntesis final.

Arrays

- Declaración de arrays – Sintaxis:

```
var nombreArray = new Array();
```

```
var nombreArray =[];
```

- Diferentes comportamientos a la hora de crear un Array

```
var array7 = new Array(10); // crea un array con 10 elementos undefined  
var array8 = new Array(10, 20); //crea un array con 2 elementos
```

```
var array9 =[10]; //crea un array con 1 elemento  
var array10 =[10,20]; //crea un array con 2 elementos
```

Arrays

- Inicialización de arrays:
 - Una vez declarado el array se puede comenzar el proceso de inicializar el array con los elementos que contendrá.
 - La sintaxis es la siguiente:

```
nombreArray[índice] = valorElemento;
```

- Es posible declarar e inicializar simultáneamente mediante la escritura de los elementos dentro del paréntesis del constructor.

```
var prod_alimenticios = new Array('Pan', 'Agua', 'Lentejas');
```

Arrays

- Uso de arrays mediante bucles:
 - Si se mezclan las características de los bucles junto a las de los arrays se pueden apreciar las ventajas que proporciona este objeto.
 - Por ejemplo:

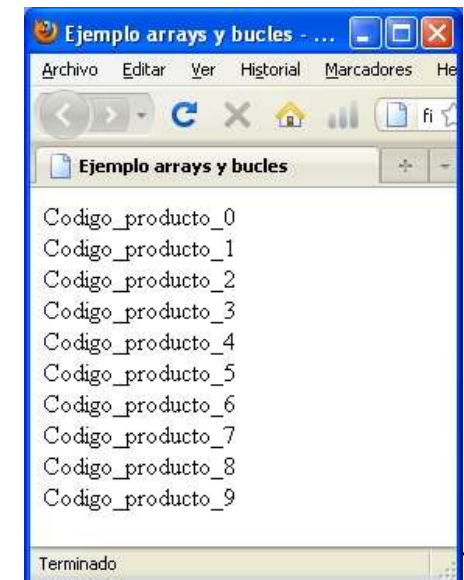
```
var codigosProductos = new Array();  
for (var i=0; i<10;i++){  
    codigosProductos[i] = "Codigo_producto_" + i;  
}
```

Arrays

- Uso de arrays mediante bucles:
 - La inicialización de un array con un bucle funciona mejor en dos casos:
 - Cuando los valores de los elementos se pueden generar usando una expresión que cambia en cada iteración del bucle.
 - Cuando se necesita asignar el mismo valor a todos los elementos del array.

```
for (var i=0; i<10; i++){  
    document.write(codigosProductos[i] + "<br>");  
}
```

- Mediante el uso de un bucle se pueden escribir instrucciones mucho más limpias y eficientes:



Arrays

- Podemos asignar un valor a cualquier posición del array

```
var array3 = [];  
array3[5]="hola";  
  
var array4 = new Array();  
array4[5] = "mundo";
```

El array pasaría a tener longitud 6.

Arrays

- **Características de los arrays en JavaScript**

- Pueden contener distintos tipos de datos

```
var v = [false, 5, "Apple", /(ab)/];
```

- En JavaScript, cuando asignamos a una variable un valor de tipo *String* o *Boolean*, estamos creando una copia de dicho objeto. Sin embargo, cuando asignamos un *Array* o un *objeto*, lo que estamos es **creando una referencia a dicho valor.**

```
var foo = [ "Muzzy", "Bob", "Norman", "King", "Queen" ],  
    bar = foo;  
  
// Truncate the numbers of item to 2  
bar.length = 2;  
  
console.log( foo ); // [ "Muzzy", "Bob" ]  
console.log( bar ); // [ "Muzzy", "Bob" ]
```

Arrays

- Para crear una copia independiente tenemos las siguientes opciones:

- Utilizar un bucle

```
for ( var i = 0, l = foo.length, bar = []; i < l; i++ ) {  
    bar[ i ] = foo[ i ];  
}
```

- Utilizar el método slice() sin parámetros

```
var bar = foo.slice();
```

- Utilizar parámetros SPREAD

```
var bar = [...foo];
```

Arrays

- Propiedades de los arrays:
 - El objeto array tiene dos propiedades:

1. **length:**

```
for (var i=0; i<codigosProductos.length; i++) {  
    document.write(codigosProductos[i] + "<br>");  
}
```

2. **prototype:**

```
Array.prototype.nuevaPropiedad = valor;  
Array.prototype.nuevoMetodo = nombreFuncion;
```

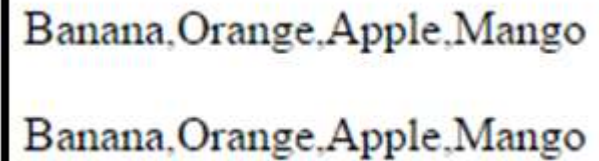
Arrays

- Métodos de los arrays:

Arrays - Métodos	
<code>push(valor)</code>	<code>shift()</code>
<code>concat(valor1,valor2,...)</code>	<code>pop()</code>
<code>join(carácter)</code>	<code>slice(posI, posF)</code>
<code>reverse()</code>	<code>sort()</code>
<code>unshift(valor1,valor2,...)</code>	<code>sort(function(a,b){return a-b})</code>
<code>indexOf(item,start)</code>	<code>splice(índice, número, items,...)</code>
	<code>lastIndexOf(item, start)</code>

Arrays

- Para convertir un array en cadena:



Banana,Orange,Apple,Mango
Banana,Orange,Apple,Mango

- Nombre del array

```
<script type="text/javascript">  
var fruits = new Array ("Banana", "Orange", "Apple", "Mango");  
document.write(fruits);  
</script>
```

- Nombre Array.toString()

```
<script type="text/javascript">  
var fruits = new Array ("Banana", "Orange", "Apple", "Mango");  
document.write(fruits.toString());  
</script>
```

Arrays

- Métodos de los arrays – `join()` :
 - Concatena los elementos de un array en una sola cadena separada por un carácter opcional.

```
<script type="text/javascript">  
  var pizzas = new Array("Carbonara", "Quattro_Stagioni", "Diavola");  
  document.write(pizzas.join(" - "));  
</script>
```

Carbonara - Quattro_Stagioni - Diavola

Arrays

- Métodos de los arrays – `concat()` :
 - Selecciona un array y lo concatena con otros elementos en un nuevo array.

```
<script type="text/javascript">  
  var equipos_a = new Array("Real Madrid", "Barcelona","Valencia");  
  var equipos_b = new Array("Hércules", "Elche","Valladolid");  
  var equipos_copa_del_rey = equipos_a.concat(equipos_b);  
  document.write("Equipos que juegan la copa: " + equipos_copa_del_rey);  
</script>
```

Equipos que juegan la copa: Real Madrid,Barcelona,Valencia,Hércules,Elche,Valladolid

Arrays

- Métodos de los arrays – `push ()` :
 - Añade nuevos elementos al array por el final y devuelve la nueva longitud del array.

```
<script type="text/javascript">
  var pizzas = new Array("Carbonara", "Quattro_Stagioni","Diavola");
  var nuevo_numero_de_pizzas = pizzas.push("Margherita","Boscaiola");
  document.write("Número de pizzas disponibles: " +
    nuevo_numero_de_pizzas + "<br />");
  document.write(pizzas);
</script>
```

Número de pizzas disponibles: 5
Carbonara,Quattro_Stagioni,Diavola,Margherita,Boscaiola

Arrays

- Métodos de los arrays – pop () :
 - Elimina el último elemento de un array y lo devuelve

```
<script type="text/javascript">
  var premios = new Array("Coche", "1000 Euros", "Manual de JavaScript");
  var tercer_premio = premios.pop();
  document.write("El tercer premio es: " + tercer_premio + "<br />");
  document.write("Quedan los siguientes premios: " + premios);
</script>
```

El tercer premio es: Manual de JavaScript
Quedan los siguientes premios: Coche,1000 Euros

Arrays

- Métodos de los arrays – delete:
 - Los elementos de un array pueden ser eliminados con el operador de JavaScript delete (deja huecos).

```
<script type="text/javascript">
  var fruits = ["Banana", "Orange", "Apple", "Mango"];
  document.write("Array: " + fruits + "<br>");
  document.write("Longitud: " + fruits.length + "<br>");
  delete fruits[0];
  document.write("Posición 0: " + fruits[0] + "<br>");
  document.write("Array: " + fruits + "<br>");
  document.write("Longitud: " + fruits.length + "<br>");
</script>
```

```
Array: Banana,Orange,Apple,Mango
Longitud: 4
Posición 0: undefined
Array: ,Orange,Apple,Mango
Longitud: 4
```

Arrays

- Métodos de los arrays – `unshift()`:
 - Añade nuevos elementos al inicio de un array y devuelve el número de elementos del nuevo array modificado.

```
<script type="text/javascript">
  var sedes_JJ00 = new Array("Atenas", "Sydney","Atlanta");
  var numero_sedes = sedes_JJ00.unshift("Pekín");
  document.write("Últimas " + numero_sedes + " sedes olímpicas: " + sedes_JJ00);
</script>
```

Últimas 4 sedes olímpicas: Pekín,Atenas,Sydney,Atlanta

Arrays

- Métodos de los arrays – `shift()`:
 - Elimina el primer elemento de un array y lo devuelve.

```
<script type="text/javascript">
  var pizzas = new Array("Carbonara", "Quattro_Stagioni", "Diavola");
  var pizza_eliminada = pizzas.shift();
  document.write("Pizza eliminada de la lista: " + pizza_eliminada + "<br />");
  document.write("Nueva lista de pizzas: " + pizzas);
</script>
```

Pizza eliminada de la lista: Carbonara
Nueva lista de pizzas: Quattro_Stagioni,Diavola

Arrays

- Métodos de los arrays – `sort()` :
 - Ordena alfabéticamente los elementos de un array. Orden por defecto es alfabético y ascendente.

```
<script type="text/javascript">
  var apellidos = new Array("Pérez", "Guijarro", "Arias", "González");
  apellidos.sort();
  document.write(apellidos);
</script>
```

Arias,González,Guijarro,Pérez

```
<script type="text/javascript">
  var nums = new Array(20, 100, 10,5);
  nums.sort();
  document.write(nums);
</script>
```

10,100,20,5

Arrays

- **Sort- Ordenación numérica y ascendente.**

Para solucionar el problema, a la función sort, debemos de pasarle una función de comparación, la cual tendrá dos parámetros (a y b) y que retornará lo siguiente:

Un valor menor que 0 si a es menor que b.

Un valor mayor que 0 si a es mayor que b.

Un cero si ambos son iguales.

La función de comparación numérica más simple es por lo tanto:

```
function compare(a, b) {  
    return a - b;  
}
```

- **Ejemplo**

```
var points = [40,100,1,5,25,10];  
points.sort(function(a,b){return a-b});
```

1,5,10,25,40,100

Arrays

- **Sort- Ordenación numérica y descendente.**

Un valor menor que 0 si b es menor que a.

Un valor mayor que 0 si b es mayor que a.

Un cero si ambos son iguales.

La función de comparación numérica más simple es por lo tanto:

```
function compare(a, b) {  
    return b - a;  
}
```

Ejemplo:

```
var points = [40,100,1,5,25,10];  
points.sort(function(a,b){return b-a});
```

100,40,25,10,5,1

Arrays

- Métodos de los arrays – `reverse()` :
 - Invierte el orden de los elementos de un array sin ordenarlos.
 - Modifica el array original

```
<script type="text/javascript">  
  var numeros = new Array(1,2,3,4,5,6,7,8,9,10);  
  numeros.reverse();  
  document.write(numeros);  
</script>
```

10,9,8,7,6,5,4,3,2,1

Arrays

- Métodos de los arrays – `slice()` :
 - Devuelve un nuevo array con un subconjunto de los elementos del array que ha usado el método.
 - Ojo que el array empiece en 0 y no incluye el elemento de la posición final.
 - Números positivos: desde el inicio del array.
 - Números negativos: desde el final del array.

```
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];  
var myBest = fruits.slice(1,3);  
//devuelve Orange, Lemon
```

```
var myBest = fruits.slice(-3,-1);  
//devuelve Lemon,Apple
```

```
Devuelve los 3 últimos. fruits.slice(-3)  
Del 3 hacia delante: fruits.slice(3)
```

Arrays

- Métodos de los arrays – `splice()`:
 - Elimina o añade elementos del array dependiendo de los argumentos del método.
`array.splice(índice, número, item1, , itemX)`
 - *Índice*: Obligatorio, un entero que especifica la posición donde quieres agregar o eliminar los items. Si es negativo, desde el final array.
 - *Número*: Obligatorio, el número de elementos, que quieres eliminar. Si es 0 no eliminará.
 - *Item*: opcional. Los elementos a insertar.
 - Modifica el array original

```
<script type="text/javascript">
  var coches = new Array("Ferrari", "BMW", "Fiat");
  coches.splice(2,0,"Seat","Audi"); //add
  document.write(coches);
  coches.splice(1,2); //remove
  document.write(coches);
</script>
```

Arrays

- Métodos de los arrays – `indexOf()` :
 - Buscar en el array el elemento indicado y devuelve su posición.
`array.indexOf(item, start)` .
- La búsqueda comienza en la posición especificada, o al principio, si no se especifica una posición de inicio y fin a la búsqueda en la parte final de la matriz.
- Devuelve -1 si el artículo no se encuentra.
- Si el artículo es presentar más de una vez, el método `IndexOf` Devuelve la posición de la primera ocurrencia.

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];  
var pos = fruits.indexOf("Apple");  
//devuelve 2:  
** No es compatible con IE 8 y anteriores
```

Arrays

- Métodos de los arrays – `lastIndexOf()` :
 - Buscar en el array el elemento indicado y devuelve su posición.
`array.lastIndexOf(item, start)` .
- La búsqueda comienza en la posición especificada, o al principio, si no se especifica una posición de inicio y fin a la búsqueda en la parte final de la matriz.
- Devuelve -1 si el artículo no se encuentra.
- Si el artículo es presentar más de una vez, el método `lastIndexOf` Devuelve la posición de la última ocurrencia.

```
var fruits = ["Banana", "Orange", "Apple", "Mango", "Apple"];  
var pos = fruits.lastIndexOf("Apple");  
//devuelve 4:
```

Arrays

- ¿Cómo sabemos si un objeto es un array?

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
```

```
typeof fruits;           // typeof returns object
```

```
function isArray(myArray) {  
    return myArray.constructor.toString().indexOf("Array") > -1;  
}
```

```
Array.isArray(fruits)
```

```
fruits instanceof Array
```

- Usando el método `Array.isArray(arr)`

Arrays asociativos

- JavaScript no soporta arrays con índices nombrados.
- Si se hace de esta forma, JavaScript redefine el array como un objeto estándar y los métodos y propiedades del objeto array dejan de funcionar.

```
var person = [];  
person[0] = "John";  
person[1] = "Doe";  
person[2] = 46;  
var x = person.length;           // person.length will return 3  
var y = person[0];              // person[0] will return "John"
```

```
var person = [];  
person["firstName"] = "John";  
person["lastName"] = "Doe";  
person["age"] = 46;  
var x = person.length;          // person.length will return 0  
var y = person[0];              // person[0] will return undefined
```

Arrays asociativos

- Para acceder al contenido del array se utiliza:

- Objeto['nombre']
- Objeto.nombre

- Ejemplo:

```
<script type="text/javascript">
    var elArray = new Array();
    elArray.primer0 = 1;
    elArray.segundo = 2;

    document.write(elArray['primer0'] + "<br>");
    document.write(elArray.primer0 + "<br>");
    document.write(elArray[0] + "<br>"); //Undefined

    document.write(elArray.length);
    elArray.sort();
    document.write(elArray);
</script>
```

} No funcionan

Arrays asociativos

- Para conocer la longitud de un array asociativo tendremos que hacerlo de la siguiente forma:

```
<script type="text/javascript">
    var elArray = new Array();
    elArray.primeros = 1;
    elArray.segundo = 2;
    Array.longitud = function(obj){
        return Object.getOwnPropertyNames(obj).length - 1;
    }

    var long = Array.longitud(elArray);
    document.write("La longitud del vector es: " + long + "<br>");

    for(i in elArray){
        document.write(elArray[i] + "<br>");
    }
</script>
```

Arrays asociativos

- Ejemplo de cómo simular sort()- *por índice*:

```
<script type="text/javascript">
```

```
function ordenarPorIndice(tabla) {  
    var indicesOrdenados = new Array();  
    var objectOrdenado = [];    // o var objectOrdenado = {};  
    for (var indice in tabla){    // extraemos los índices y los ordenamos  
        indicesOrdenados.push(indice);  
    }  
  
    indicesOrdenados.sort();    // o .sort(funcion)  
    for (var indice in indicesOrdenados){    // recreamos la tabla según  
                                                el nuevo orden de los índices  
        objectOrdenado[indicesOrdenados[indice]] = tabla[indicesOrdenados[indice]];  
    }  
    return objectOrdenado;  
}
```

```
</script>
```

Arrays asociativos

- Ejemplo de cómo simular sort()- *por valor*:

```
<script type="text/javascript">
```

```
function ordenarPorValor(tabla)      {
    var valoresOrdenados = new Array();
    var objectOrdenado = {}; // o var objectOrdenado = [];
    for (var indice in tabla){ // extraemos los valores y los ordenamos
        valoresOrdenados.push(tabla[indice]);
    }
    valoresOrdenados.sort(); // o .sort(funcion)
    for (var indice in tabla){ // recreamos la tabla reasignando
                                los valores de cada índice
        objectOrdenado[indice] = valoresOrdenados.shift(); //extraemos por la
                                                            izquiera
    }
    return objectOrdenado;
}
```

```
</script>
```

Arrays multidimensionales

- No existe un objeto llamado array multidimensional.
- Hay que crear un array, en el que en cada una de sus posiciones debemos crear a su vez otro array.
- Ejemplo:

```
var tmciudad0 = new Array(2);  
tmciudad0[0] = 21;  
tmciudad0[1] = 25;  
var tmciudad1 = new Array (2);  
tmciudad1[0] = 15;  
tmciudad1[1] = 12;  
var tmciudades = new Array (2);  
tmciudades[0] = tmciudad0;  
tmciudades[1] = tmciudad1;
```

```
tmciudades = new Array(new Array (21,25), new Array(15,12));
```

```
tmciudades = new Array([21,25], [15,12]);
```

```
tmciudades = [[21,25], [15,12]];
```

Arrays multidimensionales

- Los arrays no tienen porque ser uniformes

```
var v = [1,  
  ["hola", "que", 'tal', ["estas", 'estamos', "estoy"], ["bien", "mal"], "acabo"],  
  2,  
  5];
```

```
alert (v[0])
```

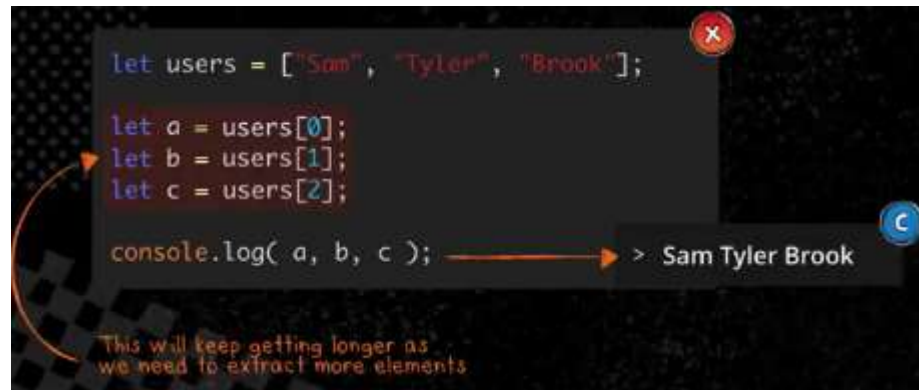
```
alert (v[1][2])
```

```
alert (v[1][3][1])
```

Novedades ES6 - arrays

- **Arrays - asignar valores a variables locales (destructuring)**

- ES5



let users = ["Sam", "Tyler", "Brook"];
let a = users[0];
let b = users[1];
let c = users[2];
console.log(a, b, c);

> Sam Tyler Brook

This will keep getting longer as we need to extract more elements

The image shows a code editor with the above code. A red 'X' icon is in the top right corner. A blue 'C' icon is in the bottom right corner. An orange arrow points from the console output to the code. A red circle with a white 'X' is in the top right corner.

- ES6



let users = ["Sam", "Tyler", "Brook"];
let [a, b, c] = users;
console.log(a, b, c);

> Sam Tyler Brook

Still easy to understand AND less code

The image shows a code editor with the above code. A green checkmark icon is in the top right corner. A blue 'C' icon is in the bottom right corner. An orange arrow points from the console output to the code. A thumbs up icon is at the bottom.



Values can be **discarded**

let [a, , b] = users;
console.log(a, b);

> Sam Brook

Notice the blank space between the commas

The image shows a code editor with the above code. A green checkmark icon is in the top right corner. A blue 'C' icon is in the bottom right corner. An orange arrow points from the console output to the code.

Novedades ES6 - arrays

- Arrays - Operador spread

```
var numeros = [2, 3, 4];  
  
console.log([1, ...numeros, 5]); // [1, 2, 3, 4, 5]
```

```
var parts1 = ['shoulder', 'knees'];  
var parts2 = ['chest', 'waist'];  
var lyrics = ['head', ...parts1, ...parts2, 'and', 'toes'];  
//lyrics = ['head', 'shoulder', 'knees', 'chest', 'waist', 'and', 'toes']
```

Novedades ES6 - arrays

- **Arrays** – combinación de destructuring con parámetros rest



```
let users = ["Sam", "Tyler", "Brook"];  
let [ first, ...rest ] = users;  
console.log( first, rest );
```

> Sam ["Tyler", "Brook"]

Groups remaining arguments in an array

Novedades ES6 - arrays

- **Arrays – valores devueltos por la función**

- Podemos devolver múltiples variables de una vez

//ES5

```
function activeUsers(){  
  let users = ["Sam", "Alex", "Brook"];  
  return users;  
}
```

Returns an array, as expected...

```
let active = activeUsers();  
console.log( active );
```

> ["Sam", "Alex", "Brook"]

//ES6

```
let [a, b, c] = activeUsers();  
console.log( a, b, c );
```

> Sam Alex Brook

Novedades ES6 - arrays

- **¿Cuál de las siguientes opciones hace un uso correcto de array destructuring?**

☐ `let {a, b, c} = ["Programming", "Web", "JavaScript"];`

☐ `let (a, b, c) = ["Programming", "Web", "JavaScript"];`

☐ `let [a, b, c] = ["Programming", "Web", "JavaScript"];`

Novedades ES6 - arrays

- ¿Cuál/es de las siguientes opciones devolverá el primer objeto topic que no está bloqueado?

```
let recentTopics = [  
  {  
    title: "Semi-colons: Good or Bad?",  
    isLocked: true  
  },  
  {  
    title: "New JavaScript Framework Released",  
    isLocked: true  
  },  
  {  
    title: "ES2015 - The Shape of JavaScript to Come",  
    isLocked: false  
  }  
];
```

☐ recentTopics.find([topic] => !topic.isLocked);

☐ recentTopics.find(topic => !topic.isLocked);

☐ recentTopics.find((topic) => !topic.isLocked);

☐ recentTopics.find((topic) => {topic.isLocked});