JavaScript Coding Examples

Clear Array of Duplicates and empty values 2

How to loop through an array 2

Array Updating adding and removing array items 4

JavaScript Map Method for Arrays new array 5

How to concat arrays and duplicate an array 6

Array iterator entries using entries and looping items 7

Array Filter Method Example for JavaScript Arrays 8

Array includes Method and index methods JavaScript coding Example 9

This exercise will provide examples of code you can use to accomplish common tasks with array items.

1. Create an array with some values, add empty values into the array items.
2. Using the filter method then remove the duplicates in as few lines of code as possible.
3. Provide several ways to loop through the items of the array and output the values into the console.
4. make updates to the array, add new items to the beginning and end.
5. Remove the first and last items from the array list
6. Using Map create a new array from a previous array
7. Try the callback within the map to add more functionality to the block of code.
8. Create several additional arrays
9. Join them together
10. Duplicate the array contents to a new array.
11. Use the entries() method to create an iterator object of the array contents, output the results in the console.
12. Create new arrays using the filter to only add selected items by condition.
13. Check for values within the array, get the index value of the first occurrence, last occurrence, and the boolean result if it exists.

Clear Array of Duplicates and empty values

1. Create an array with some values, add empty values into the array items.
2. Using the filter method then remove the duplicates in as few lines of code as possible.

```
const arr =
['Laurence','Jack','Jane','',,,,'Sam','Laurence','Jack','Jane','',null,false,undefin
ed,0,'Sam'];
console.log(arr);

const arr1 = [... new Set(arr)];
console.log(arr1);

const arr2 = arr.filter(Boolean);
console.log(arr2);

const arr3 = [... new Set(arr.filter(Boolean))];
console.log(arr3);
```

How to loop through an array

1. Create an array with some values, add empty values into the array items
2. Provide several ways to loop through the items of the array and output the values into the console.

```
▶ (4) ['Laurence', 'Linda', 'Joe', 'Jane']                app1.js:2
****FOR                                                   app1.js:3
Laurence                                                  app1.js:5
Linda                                                     app1.js:5
Joe                                                       app1.js:5
Jane                                                      app1.js:5
****WHILE                                                 app1.js:7
Laurence                                                  app1.js:10
```

```
const arr = ['Laurence','Linda','Joe','Jane'];
console.log(arr);
console.log('****FOR');
for(let i=0;i<arr.length;i++){
  console.log(arr[i]);
}
console.log('****WHILE');
let i=0;
while(i<arr.length){
  console.log(arr[i]);
  i++;
}
console.log('****ForEach');
arr.forEach((item,index,array)=>{
  console.log(item);
})

console.log('****MAP');
const arr1 = arr.map((item,index)=>{
  console.log(item);
  return `${index} ${item}`;
```

```
})

console.log('****Filter');
const arr2 = arr.filter((item)=>{
  console.log(item);
  return item;
});
console.log(arr2);
```

Array Updating adding and removing array items

1. create an array with values
2. make updates to the array, add new items to the beginning and end.
3. Remove the first and last items from the array list

```
const arr = ['Laurence','Next'];
arr.push('End Push'); //adds to end
const val1 = arr.shift(); //removes first
const val2 = arr.pop(); //removes last
arr.unshift(val1); // adds to start
console.log(val1);
console.log(val2);
console.log(arr);

const str1 = arr.toString();
console.log(str1);

const str2 = arr.join('*-*');
console.log(str2);
```

JavaScript Map Method for Arrays new array

1. Using Map create a new array from a previous array
2. Try the callback within the map to add more functionality to the block of code.

```
▼ (3) [{…}, {…}, {…}] i
  ▶ 0: {first: 'Laurence', last: 'Svekis'}
  ▶ 1: {first: 'John', last: 'Smith'}
  ▶ 2: {first: 'Sam', last: 'Jones'}
    length: 3
  ▶ [[Prototype]]: Array(0)
```

```
▼ (3) [{…}, {…}, {…}] i                                app1.js:33
  ▶ 0: {fullName: 'Laurence Svekis'}
  ▶ 1: {fullName: 'John Smith'}
  ▶ 2: {fullName: 'Sam Jones'}
```

```
const arr = ['Laurence',4,545,false,'Test'];
const arr2 = arr;
arr2.push('NEW');
console.log(arr);

const arr1 = arr.map((item,index,array)=>{
  const temp = `${index} ${item}`;
  return temp;
})
arr2.push('NEW');
console.log(arr1);

const arr3 = [3,54,62,4334,1232,444];
const arr4 = arr3.map(val => val*2);
```

```
console.log(arr4);

const arr5 = arr3.map(callback1);
console.log(arr5);

function callback1(item){
  console.log(item);
  return item*2;
}

const arr6 =
[{first:'Laurence',last:'Svekis'},{first:'John',last:'Smith'},{first:'Sam',last:'Jo
nes'}]
console.log(arr6);
const arr7 = arr6.map(({first,last})=>{
  //console.log(first,last);
  //return `${first} ${last}`;
  return {fullName:`${first} ${last}`}
})

console.log(arr7);

const arr8 = arr6.map(({first,last})=> ({full:`${first} ${last}`}));
console.log(arr8);
```

How to concat arrays and duplicate an array

1. Create several additional arrays
2. Join them together
3. Duplicate the array contents to a new array.

```
const arr1 = ['Laurence','Svekis'];
const arr2 = [1,2,3,4,5];
const arr3 = arr1.concat(arr2);
console.log(arr3);
const arr4 = arr1.concat(arr2,['new1','new2']);
console.log(arr4);
const arr5 = arr2.concat(arr2,arr2);
console.log(arr5);

const arr6 = arr1;
arr6.push('END');

const arr7 = arr1.concat();
arr6.push('more');
console.log(arr7);
console.log(arr1);
console.log(arr6);
```

Array iterator entries using entries and looping items

1. Use the entries() method to create an iterator object of the array contents, output the results in the console.

```
const arr1 = ['Laurence','Svekis',100];
const adder = arr1.entries();


console.log(adder);
//console.log(adder.next());
//console.log(adder.next().value);
```

```
for(let item of adder){
  console.log(item[1]);
}

const adder1 = arr1.entries();
let val = adder1.next();
while(val.value){
  console.log(val.value[1]);
  val = adder1.next();
}
```

Array Filter Method Example for JavaScript Arrays

1. Create new arrays using the filter to only add selected items by condition.

```
const arr = ['Svekis','Laurence','test','new',5,23,54,5,1213343];
const arr1 = arr.filter(val => val.length >= 4);
console.log(arr1);

const arr2 = arr.filter(checker);
console.log(arr2);

function checker(item,index,array){
  //console.log(item);
 /// console.log(index);
  //console.log(array);
  return index >=2;
}
```

```
const arr3 = arr.filter((item,index)=>{
  //console.log(typeof(item) );
  return typeof(item) == 'string';
});
console.log(arr3);
arr.push(null);
arr.push(false);
console.log(arr);
const arr4 = arr.filter(Boolean);
console.log(arr4);

const arr5 = arr.filter((str)=>{
  if(typeof(str)=='string'){
     const first = str[0].toUpperCase();
     return first === str[0];
  }
})

console.log(arr5);
```

Array includes Method and index methods JavaScript coding Example

1. Check for values within the array, get the index value of the first occurrence, last occurrence, and the boolean result if it exists.

```
const arr = ['Laurence','Svekis','Svekis',100,'Svekis',1000];
const boo1 = arr.includes('Svekis');
console.log(boo1);
const boo2 = arr.includes('test');
console.log(boo2);

const ind1 = arr.indexOf('Svekis');
console.log(ind1);
const ind2 = arr.indexOf('test');
console.log(ind2);

const lind1 = arr.lastIndexOf('Svekis');
console.log(lind1);
const lind2 = arr.lastIndexOf('test');
console.log(lind2);

const find1 = arr.findIndex(call1);
console.log(find1);
const find2 = arr.findIndex(call2);
console.log(find2);

function call1(item){
  //console.log(item);
  if(item === 'Svekis' ){
     return item;
  }
}
```

```
function call2(item){
  if(typeof(item)=='number'){
     return item;
  }
}
```