8 Must known JavaScript tricks for web developer

Follow

1. Remove false values from an Array

```
const array = [1, 0, undefined, 6, 8, "", false, false];
console.log(array.filter(Boolean)); //[1,0,6,8]
```

We can combine **Boolean** map method to quickly remove false values from an array. It is very useful when we expect null or other false values in an array and need to clean it.

Follow

2. Shuffle items in an Array

```
const numbers = [1, 2, 3, 4, 5, 6, 7, 8];

const numbersShuffle = numbers
  .slice() //we use slice to copy an array
  .sort(function () {
    return Math.random() - 0.5;
  });
console.log(numbersShuffle(numbrs)); //[2, 6, 4, 3, 1, 8, 7]
```

We can use **sort**() to randomly shuffle items in an array. It is very useful for browser games, when we want to randomize something. Have a glace at our code.

Follow

3. Short circuit evaluation

```jsx
const componentShow = true;
const Component = () => <div>Our Content</div>;

//Component will be rendered
componentShow && <Component />;

const isLogIn = false;
const Login = () => <div>Login</div>;

//Component will not be rendered
isLogIn || <Login />;
```

&& or || can be used to quickly check for conditions and return results in a single line. It is a common pattern in **React** for conditional rendering. (&& is more popular than ||).

Follow

4. Truncate arrays

App.js

```
let array=[1,2,3,4,5,6];

console.log(array); //[1,2,3,4,5,6]
console.log(array.length); //6
array.length=3 //deleted elements from
index 3 to the actual length of array
console.log(array); //[1,2,3]

//we can also empty the array completely
array.length=0;
console.log(array); // []
```

length is very popular property for getting the total numbers of elements in the array. But not many people realize we can use it as a setter too.

Follow

5. Getting unique values from Array

```
const array=[1,1,2,2,3,3,3];
const uniqueArray=[ ... new Set(array)];
console.log(uniqueArray); //[1,2,3]
```

It is very useful trick when we get in situation to get **unique** values from an array. We can use a Set to achieve this objective.

Follow

6. Readable numbers

```
const normalNumber=1000000000;
const readableNumber=1000_000_000;
```

Sometimes we are not able to read the numbers properly but with the help of this property in javascript we can increase our **readability** of numbers javascript allow you to put this _ between numbers.

Follow

7. Nullish Coalescing (Combining) (??)

It is a logical operator, that will return the right-hand side operand, only when it's left-hand side operand is either null or undefined. It is extremely useful, when we want to provide a default value when the value expected is **nullish**.

```
function getUserId(user){
    return user?.name ?? "Anonymous";
}
```

Here **??** says to javascript whether value on left hand side is null or not. If the value is null, then store the value beyond **??** to right hand side.

Follow

8. Optional Chaining (?)

This feature allows you to read a property from object, without checking if object exists. This trick is very helpful for getting deeply nested properties from object without checking validity of each step in the **chain**.

```
user.data?.id;
user.data?.name;
user.data?.addressList?.[0];
user.greet?.();
```

Here **?** says to javascript whether the left side value is null or not if it is not null try to fetch the value **beyond ?.** else just stop here. Just put it when you are not sure whether property exist or not.

Follow

FOLLOW