JS Questions LIJIA XU (31)

30. Best way to detect undefined object property in JS

```
ex. var person = {
      name: 'Bob',
      age: 24
    }
    if (typeof person.salary === 'undefined'){
      console.log('salary is undefined');
    }
```

31 ⇒ copy object but not deep clone

```
var objectLit = {foo: 'Bar'};
var cloneObj = Clone(obj);
  console.log(cloneObj === Clone(objectLit));
  // this returns false
  console.log(cloneObj == Clone(objectLit));
  // this returns true
```

```
function Clone(object){
  var newObject = {};
  for(var key in object){
    newObject[key] = object[key];
  }
  return newObject;
}
```

32. what are promises and how are they useful?

handling asynchronous interactions

* do an async operation and then do another async operation

then method ⇒ what to do when the promise fulfilled or rejected

handle errors in the catch()

⇒ async/await

⇒ RxJS observables

* get more details

* Flat is better than nested

33. check a key exist in an object

```
var person = {
  name: "Bob",
  age: 24
}
```

'name' in person // true

'salary' in person // false

'toString' in person // true

```
person.hasOwnProperty('toString');
// False
```

34. NaN? ⇒ not a number

```
NaN !== NaN // use isNaN function to check
.isNaN({}) // true
Number.isNaN(NaN) // true
```

```
console.log(type of NaN); // 'number'
```

isNaN	Number.isNaN
isNaN(NaN) // true	Number.isNaN(NaN) // true
isNaN('Hello') // true	Number.isNaN('Hello') // False
isNaN(['x']) // true	

For Number.isNaN

1. if Type(number) is not number, return false
2. if number is NaN, return true
3. otherwise, return false

* true only if the argument is really NaN

Q35 ⇒ fix a bug (function scope)

36. check if the value of a variable in an array. method 1.

```
function isArray(value){
  return Object.prototype.toString.call(value)
  === '[object Array]';}
```

Method 2.

```
function(value){
  if(typeof Array.isArray === 'function'){
    return Array.isArray(value);
  }else{
    return Object.prototype.toString.call...
```

37. Best way to detect reference vals of any type? In JS Object are called as reference type, Any value other than primitive is definitely a reference type

built-in reference types: is object

Object, Array, Function, Date, null and Error

best way: using instance of

value instance of constructor

```
if(value instance of Array){ }
```

38. Object.create()

```
var empl = Object.create(employee, {
  name:{
    value: "John"
  }
});
```

39. use constructor for inheritance?

```
function Person(name, age, salary){
  this.name = name;
  this.age = age;
  this.salary = salary;
  this.incrementSalary = function(byValue){
    this.salary = this.salary + byValue;
  };
}
```

```
function Employee(company){
  this.company = company;
}
//Prototypal Inheritance
Employee.prototype = new Person(
"Nishant", 24, 5000);
var empl = new Employee("Google");
```

* Constructor inheritance

```
function Person(name){
  this.name = name || "Bob";
}
var obj = {};
Person.call(obj);
console.log(obj); // Object {name: "Bob"}
```

constructor inheritance

40. prevent modification of object in JS? ① Prevent extensions

```
var employee = {
    name: "Bob"
};
object.preventExtensions(employee)
```

lock the object

```
employee.name = 'Jack'; // works
employee.age = 20; // does not work
```

② seal ⇒ not only prevent extensions also prevent existing properties and methods from being deleted

```
var employee = {
    name: "Bob"
};
object.seal(employee)
```

object.isExtensible(employee) // false

Object.isSealed(employee) // true

```
delete employee.name // fails silently
employee.age = 20 // fails silently
```

③ Freeze * test in console

Also prevent existing properties from modified

```
var employee = {
    name: "Bob"
};
Object.freeze(employee)
```