JS Questions LIJIA XU (2/)

22 continue * use for...of **Not** for...in for **Arrays**

```
function hi(name){
  return "Hi," + name;
}
hi("Joe");
```

Methods in JS are nothing more than object properties that are functions

```
var obj = {
   hi: function(){
     return "Hi," + this.name;
   },
   name: 'Joe'
}
obj.hi();
```

* prints hi Joe

* copy a reference to the same function in another obj will get different result

```
var obj2 = {
    hi: obj.hi,
    name: "Bob"
}
obj2.hi() // prints hi Bob
```

constructors are defined with function

```
function Employee(name, age){
      this.name = name;
      this.age = age;
}
var empl = new Employee('Joe', 28)
empl.name;
empl.age;
```

Primary role of the constructor function is to initialize the object

In JS, these are three different usage patterns of one single construct.

23. Output?

```
function User(name){
    this.name = name || "Bob";
}

var person = new User("xyz")["location"] =
"USA";
console.log(person); // outputs USA
```

24 what are service workers? (known as offline first)

use cached resources first, and provide default experience.

service workers actively use promises

25. difference between a method and function

function ⇒ called by name and itself

not associated with any object

not defined inside any object

```
(function(){
})();
```

Arrow func

```
const myFunc = arg => {
     console.log("Hello", arg)
}
```

Method ↓ called by its name and that is associated with the object

* It is not how you declare a function It's the way we call a function

26 what is IIFE (Immediately Invoked Function Expression)?

IIFE ⇒ a function runs as soon as it's defined, usually anonymous, but can be named ⇒ helps debugging

it can return a value:

```
var result = (function myIIFEFunc(param)
{ console.log("Hi," + param); })("Bob");
  return 1;
//store 1 to result and print Hi Bob
```

27. Singleton Pattern in JavaScript

* only one instance of an object is needed throughout the lifetime of an application. object can be accessed anywhere in the page

key feature: Global variable

⇒ A singleton as a Namespace

```
var  MyNameSpace = {
     findUserName : function(id){},
          ⋮
 } //console.log(MyNameSpace.findUserName());
```

* Singleton Design Pattern Implementation

```
var MyNameSpace = {};
MyNameSpace.Singleton = (function(){
  var singletonInstance;
  function constructor(){
    var privateVar1 = "Bob";
    var privateVar2 = [1, 3, 5, 7];
    function privateMethod1(){
      //code
    }
    function privateMethod2(){
      //code
    }
    return{
      attribute1 : "Bob",
      publicMethod: function(){
        alert ("Bob");
      }
    }
  }
  return{
    getInstance : function(){
      if(!singletonInstance){
        singletonInstance=constructor();
      }
      return singletonInstance;
    }
  }
})();
console.log(MyNameSpace.Singleton.getInstance().
publicMethod());
```

objects in JS

28. ways of creating ① function based

```
function Employee (name, age){
    this.name =name;
    this.aye =age;
}
var employee1 = new Employee("Bob", 24);
```

Method ②: Object Literal ⇒ best way to create object

```
var employee ={
    name : "Bob",
    salary : 2567,
    getName: function(){
        return this.name
    },
    address :{
        Line1 : "ABC"
        phone : {
            work : 234,
            home : 567
        }
    }
}
```

Method ③ From Object using new

```
var employee= new Object();
employee.name ="Bob"
```

Using ④ Object.create

Object.create(obj)
⇓
create a new object and set obj as its prototype

** Object.crect doesnt run the constructor

object.create(null)
⇓
do not inherit properties of object

29. Write a function takes an object and creates a object copy of it.

```
var newObject = deepClone(obj);
function deepClone(object){
    var newObject ={};
    for (var key in object){
      if (typeof object[key] === 'object'
    && object[key] !== null){
        newObject[key] = deepclone(
        object[key]);
      }else {
        newObject[key] = object[key];
      }//if
    }// for
  return newObject;
}//whole function
```