Describe an instance of prototypal inheritance in JavaScript?

1. JavaScript differensial inheritance inheritance modelindən istifadə edir. Bu o deməkdir ki, metodlar parent-dən child-a copy olmur, amma child-ların parentlə görünməz bağı olur ki, bu da bizə imkan verir ki, parent object-dən(JS-də funksialar da object olaraq verilir) istədiyimiz datanı götürüb child-larda istifadə edək.

function Dog() {

}

Dog.prototype.bark = function() {  
 console.log(‘woof!’);  
};

var fido = new Dog();

fido.bark(); // ‘woof!’

Bu misalda fido-nun bark adlanan bir metodu yoxdur. Lakin, new decleration-u vasitəsilə burada prototype chain yaranır. Fido.bark() yazdıqda isə JS fido-da bark metodun axtarmır, onu Dog funksiyasından götürür.

Real olaraq, belə bir funksiya yazaraq yuxaırda göstərilənləri aydınlaşdırmaq olar:

Function Rectangle(width, height

){

this.width=width;

this.height=height;

}

Var rect = new Rectangle(3,4);

Rectangle.prototype.area = function(){

Return.this.width\*this.height

};

Rect.area()//12

Burda baş verənləri aydınlaşdırsaq, Rectangle adında bir konstruktor funksiyamız var, və onun 2arqumenti var, this. deklarasiyası ilə arqumentləri götürüb yeni width, height properity-lər yaradırıq ki, bu da bizə sonradan prototype vasitəsilə bunlardan istifaəd edərək yeni object yaratmağa imkan verir. This decleration-u bizə imkan verir ki, bizə lazım olan arqumentləri götürüb istifadə edə bilək, və iki funksiya arasında əlaqə yaradır. Prototype-a baxsaq burada return görərəik ki, bu da konstruktor funksiyadakı this vasitəsilə gpötürdüyümüz arqumentləri bizə qaytarır.

1. What are closures and how are they used?

Ümumilikdə internetə nəzər salsaq closure-ları JS-in bug-ı kimi qiymətləndirirlər əslində isə closure sərbəst dəyişənlərdir ki, yaradıldığı mühiti yadda saxlayaraq sonradan olara dəyişənlər təyin etməklə istifadə etmək mümkündür.

Function name(){

Var name=”myname”

Function displayName(){

Console.log (name);

}

displayName();

}

Name()

Name funksiyası daxilində name adında dəyişən daşıyır və funksiyanın yazıılışından bizə elə gələ bilər ki, displayName funksiyası işə düşən zaman undefined çıxacaq lakin elə deyil, burada iki aspect var biri scope chain aspektidir ki, funksiya daxilindəki dəyişəni əvvəl öz daxilində sonar bir üst səviyyədəki mühitdə orda da tapılmasa globala çıxaraq tapir, ikinci aspect isə inheritance-dır ki, displayName funksiyası daxilində olduğu funksiyadan varislik götürür.

Function counter(x){

Return function(y){

Return x+y;

}

}

Var add=counter(5)

Console.log(add(5)) //10

Bu nümunədə var add = counter(5) counter funksiyasından istifadə edərək yeni deklarasiya yardır və bu zaman add özü də çevrilib funksiya olur. əgər add dəyişənin console-a versək console-da bizə counter funksiyasının içindəki funksiya çıxacaq, add(5) isə y-ə yeni dəyər verir və nəticə olaraq console bizə 10 qaytarır.

1. Can you use x === “object” to test if x is an object?

Bu yazılış x-in object olub olmadığın yoxuluyr. X-dən əvvəl typeof yazsaq x array və ya null olsa belə console bizə true qaytarır. X-in object olub olmamasın yoxlamnaq üçün object.is() metodundan istifadə etmək lazımdır. === funksiyası JS-də müqayisə operatorudur və bu yazılışda bizə true qaytarırş əgər x-in dəyəri NaN və ya +0/-0 deyilsə cavab həmişə true olacaq.

function duckType(obj) { // if it looks like a duck, quacks like a duck, walks like a duck, its a duck.

var t = {}.toString;

return t.call(obj).slice(8, -1).toLowerCase();

}

typeof 1; // number

typeof function () {} // function

typeof 'string'; // string

typeof {}; //"object"

typeof []; //"object"

(function() {console.log(typeof arguments)})(); //object

typeof new Date; //"object"

typeof /d/gmi; //"object"

typeof Math; //"object"

typeof JSON; //"object"

typeof new Number(4); //"object"

typeof new String("abc"); //"object"

typeof new Boolean(true); //"object"

1. What happens when you don’t declare a variable in Javascript?

myObj = 1; və var myObj = 1; fərqli şeylərdir. Var istifadə edib dəyişən təyin edən zaman bu funksiyanın scope-unda local olaraq tanınır. Və digər yazılna funksiyalar bu dəyişəni undefined olaraq qəbul edir. Var istifadə etmədən dəyişən təyin edildikdə isə artıq scope-dan çıxaraq qlobal dəyişən halına gəlir.

Var myObj = 1; əvvəlcədən global-da təyin edib sonar funksiyanın içində myObj=2 yazsaq bu zaman funksiya onu 2 kimi görəcəy.

var foo = "I'm global";

var bar = "So am I";

function () {

var foo = "I'm local, the previous 'foo' didn't notice a thing";

var baz = "I'm local, too";

function () {

var foo = "I'm even more local, all three 'foos' have different values";

baz = "I just changed 'baz' one scope higher, but it's still not global";

bar = "I just changed the global 'bar' variable";

xyz = "I just created a new global variable";

}

}

1. What is the difference between == and === ?

== dəyişənlərin dəyərini yoxlayır, onların string və ya number olmasının önəmi yoxdur məsələn: “comp”==4 bizə true qaytaracq birincinin string olmasına baxmayaraq çünki, bu halda stringin length götürülür və 4-ə = olub olmadığı yoxlanır. === isə dəyişənlərin həm dəyərini həm də növünü yoxlayır yəni, yuxarıda qeyd olunan yazıda cavab false olaraq qayıdacaq çün ki, dəyişənin biri string digəri isə number-dır. Yəni “4”===4 false, “4”==4 true olacaq.

1. What datatypes are supported in Javascript?

JS-də null, undefined, string, number, boolean, symbol, function, object, array kimi data type-lar var.

1. What would “1”+2+3 and 1+2+“3” evaluate to, respectively

1-cinin cavabı “123” 2-cinin cavbı isə “33” olacaq səbəb isə, JS-də 1-ci dəyişən string kimi verilibsə yerdə qalanların da string kimi götürülməsidir, sonuncuda isə + operatoru number type-lər gördüyü üçün birinci riyazi operator rolun oynayır “3” string olduğuna görə isə riyazi əməldən sonra + connector rolu oynamağa başlayır.

1. Explain the concept of unobtrusive Javascript?

Unobtrusive oldschool developerlərdən gələn bir prinsipdir, belə ki, js-kodların html kodlarında yarı tutmaq və saytın dinamik hala gəlməsinə yönəlmişdir. Prinsip olaraq html kodları içində js kodları yazmaq kodun oxunaqlığın azaldır. Digər tərəfdən isə mouseeventləroindən mümkün qədər az istifadə etmək tövsiyyə olunur. Çünki saytın istifadəçiləri nəzərdə tutulan kütlənin istifadə etdiyi device-ları da nəzərə almalıdır. Digər tərəfdən isə istifadəçilərin iradəsindən aslı olaraq və ya olmayaraq broüserlərdə js dəstəyi dayandırıla bilər bu zaman isə js-lə sayta verilən dinamkilik itəcək və sayt istədiyimiz kimi işləməyəck.

1. What is the difference between a null value and an undefined value?

Undifined o mənaya gəlir ki, biz dəyişəni təyin etmişik lakin ona nəsə mənimsətməmişik. Bir tərəfdən isə null da əslində bir dəyişəndir və onu sonradan dəyişənin value-su olmadığını bildirmək üçün dəyişənə mənimsədə bilərik.

1. **var** x;
2. console.log(x);
3. => **undefined**

Null JS-də xüsusi dəyişəndir və JS-də onun qarşılığı object-dir:

1. console.log(**typeof** **null**);
2. => **object**
3. Which conditional statements will JavaScript support?

JS şərt operatoralrı olaraq bunları dəstəkləyir:

İf, if... else,else, switch

if (condition)

{

code to be executed if condition is true

}///

if (condition)

{

code to be executed if condition is true

}

else

{

code to be executed if condition is false

} ///

switch (expression)

{

case label1:

code to be executed if expression = label1

break

case label2:

code to be executed if expression = label2

break

default:

code to be executed

if expression is different

from both label1 and label2

}////

1. What is NaN?

Not-a-Number bu property dəyərin rəqəm olmadığın bildiri. Dəyərin number olub olmamasın yoxlamaq üçün globalda isNaN() funksiyasında istifadə olunur. NaN ilə edilən bütün riyazi əməliyyatlar NaN qaytarır. Lakin typeof NaN // number qayıdır.

1. Explain the meaning of the keyword ‘this’ in JavaScript functions

this js-də xüsusi yer tutur. This-I tək işlətəsk onun bizə nə qaytaracağını bilmək çətin olacaq ama genel olaraq this window obyektin qaytarır. əslində this-in bizim üçün əhəmiyyəti var və funksiya daxilində dəyərləri mənimsətəmək sonrada onu rahat istifadə etmək üçün istifadə edə biləri:

Function Rectangle(width, height

){

this.width=width;

this.height=height;

}

1. What is the difference between undefined and not defined in JavaScript?

undefined data tipidir not defined ise Error.

1. What is function hoisting in JavaScript?

Termin olaraq axtarsaq hoisting-ə js dokumentasiyası daxilindərast gələ bilmərik. Hoisting jsdə execution konteksinin necə reallaşmasını göstərir. JS-də təyin edilən dəyişənlər və funksiyalar yazılan kodalrın yuxarısına qaldırılır yəni hoisting olunur və execute olunurş bunula biz aşağəda təyin etdiyimiz dəyişən və funksiyaları yuxarıda istifadə edə bilərik.

1. Can you name two programming paradigms important for JavaScript app developers?

OOP və Imperative paradigm

1. What is functional programming?

Funksional programming-də eventlər funksiyalar üzərində aparılır. Yəni riyazi əməliyyatlar, funksiyalara verilən arqumentlərə parametrlərin ötürülməsi belə funksiyalarla ötürülür. Functional programming kompleks olmasına abaxmayaraq, daha detallı düşünmə və verilən tasklar üzərində plana uyğun davam etmə və yaddaşdan daha az istifadə imkanı verir çünki, funksiyalar execute olunana qədər yaddaşda yer tutumur.

1. What is the difference between classical inheritance and prototypal inheritance?

JS-də hər ikisinin xüsusi bir fərqi yoxdur yəni mahiyyət etibarilə eyni şeylərdir.

Classical inheritance:

function Shape() {

this.x = 0;

this.y = 0;

}

//superclass method

Shape.prototype.move = function(x, y) {

this.x += x;

this.y += y;

console.info("Shape moved.");

};

// Rectangle - subclass

function Rectangle() {

Shape.call(this); //call super constructor.

}

//subclass extends superclass

Rectangle.prototype = Object.create(Shape.prototype);

Prototypal inheritance:

function Circle(radius) {

this.radius = radius;

}

Circle.prototype.area = function () {

var radius = this.radius;

return Math.PI \* radius \* radius;

};

Circle.prototype.circumference: function () {

return 2 \* Math.PI \* this.radius;

};

var circle = new Circle(5);

var circle2 = new Circle(10);

1. What are the pros and cons of functional programming vs object-oriented programming?

Adları çəkilən programming paradigmaları hərəsi öz sahəsində yaxçıdır. Onları biri digərindən pisdir deyə tanımlamaq düzgün deyil. Sadə olaraq oların hər biri problemin həllinə fərqli yanaşma sərgiləyir. Functional-da algoritmlər rekursiya və closure-lar üzərinə olur. Burada data və funksiyalar ümumi istifadədə olur. Bu paradiqmada hər şey funksiyalar üzərinə qurulduğu üçün kod oxunaqlığı çətin olur. OOP-də isə hər şey obyekt olaraq götürülür və obyekt kimi davranılır. Götürdüyümüz obyektin istənilən obyekt olduöğun bilmək üçün this-dən istifadə olunur, burda da bir anlaşılmazlıq this-in çağırılanadək nə olduöğun dəqiq bilmədiymizdir, bu problem aradan götürmək üçün this-I həmişə konsola verib yoxlamaq lazımdır.

1. When is classical inheritance an appropriate choice?

Kod oxunaqlığı baxımımndan protoype inheritance daha məqbuldur. Classical isə uzun yazı metodudur.

Iş müsahibəsi zamanı eşidilmək istəyən cavablara baxsaq heç vaxt classical inheritance-dan istiadə etməmək cavabı ən çox eşidilmək istəyən cavabdır. əslinə qalsa js-da class sözü keçmir lakin class məntiqi yer alır.

1. When is prototypal inheritance an appropriate choice?

Prototypal inheritance-in bir neçə növü var:

* on (i.e., the prototype chain).
* Concatenative (i.e. mixins, `Object.assign()`).
* Functional (Not to be confused with functional programming. A function used to create a closure for private state/encapsulation).

Bunların hər birinin öz istifadə yeri olsa da hər birinin istifadəsi useful-dur. Istənilən vaxt bizim inheritance-a ehtiyacımız ola bilər, əgər obyektləri müxtəlif source-lardan götürüb birləşdirmək bizə lazımdırsa bu zaman istifadə daha yaxşıdır. Classical inheritance-məntiqi ilə işləsə də classical inheritance deyil.

1. What does “favor object composition over class inheritance” mean?

Buaradaa əsas məqsəd kodun yenidən istifadə oluna bilmə məntiqidir. Classic inheritance-lə varislik almaq əvəzinə yeni objectin içinə kodu assemble etməkdir. Class varisliyoindən qaçmaq üçün, tez dağıla bilən base class problemin-dən qaçmaq üçün, kodu flexible etmık üçün.

1. What are two-way data binding and one-way data flow, and how are they different?

Data binding-in 2 tərəfli və ya 1 tərəfli axışı o deməkdir ki, məsələn inputa daxil olunan verilənlərin və ya oradan silinən verilənlərə görə əməliyyatların aparılmasıdır. Hazırda REACT stək tərəfli data axımını təmin edir. Cycle.js isə başqa bir populyar tək tərəfli data axımını tımin edən freymvorkdür. Angular isə iki tərəfli daxa axımın təmin edir.

1. What are the pros and cons of monolithic vs microservice architectures?

Monolithic arxitekturanın mənası, app-in vahid kodları birlikdə işləmək üçün yazılır və onlar eyni source-dan və yaddaşdan istifadə edir. Microservice arxitekturası qurulan app-in kiçik kiçik digər app-larin birlikdə execute olunması ilə çalışması və onların da hər birinin kodlarının müxtəlid yaddaş və source-dan istifadə edir.

Monolithic ən böyük üstünllüyü app-lərin DOS-lardan qorunması və security probleminin həll olunmasıdır. Çatışmayan cəhəti isə bir source-dan və yaddaşdan istifadə daha yavaş işləmə ilə nəticələnə bilməsidir.

1. What is asynchronous programming, and why is it important in JavaScript?