**Exercises: JavaScript Objects**

1. Make a Rectangle class that stores a width and a height. Make a few instances and print out the properties. Modify a few of the properties and print out the results again.

Sol:

*class Rectangle{*

*constructor(length,width){*

*this.length=length;*

*this.width=width;*

*}*

*getLength()*

*{*

*return this.length;*

*}*

*getWidth()*

*{*

*return this.width;*

*}*

*setLength(length)*

*{*

*this.length=length;*

*return ("length Modified "+length);*

*}*

*}*

*const rec=new Rectangle(6,4);*

*console.log(rec.getLength());*

*console.log(rec.setLength(7));*

1. Add a getArea method. Use the prototype property.

Sol:

*class Rectangle{*

*constructor(length,width){*

*this.length=length;*

*this.width=width;*

*}*

*getLength()*

*{*

*return this.length;*

*}*

*getWidth()*

*{*

*return this.width;*

*}*

*setLength(length)*

*{*

*this.length=length;*

*return ("length Modified "+length);*

*}*

*getArea()*

*{*

*return this.length*

*\*this.width;*

*}*

*}*

*const rec=new Rectangle(6,4);*

*console.log(rec.getLength());*

*console.log(rec.setLength(7));*

*console.log("Area : "+rec.getArea());*

**3.** Assuming that the Rectangle constructor takes a width and a height, why does the following output

20 instead of 200? (Hint: if you see an answer that seems too obvious to be what I am looking for,

it probably *is* the answer I am looking for.)

*Rectangle r = new Rectangle(4, 5);*

*r.hieght = 50;*

*r.getArea(); --> 20 // Not 200*

*Sol: class Rectangle{*

*constructor(width,height)*

*{*

*this.width=width;*

*this.height=height;*

*this.hieght=height\*10;*

*}*

*getArea()*

*{*

*return this.width\*this.height;*

*}*

*}*

*const rec1=new Rectangle(4,5);*

*//wrong data feild is called in the question*

*console.log(rec1.hieght);*

*// the data feild having height is height not hieght*

*console.log(rec1.height);*

*console.log(rec1.getArea());*

**4.** Make a variable whose value is an object with firstName and lastName properties, but don’t define

a Person class first. Try looking up the first and last names. Try changing the last name. It seems

very odd to Java programmers to make an object without first defining a class, but JavaScript programmers

do this sort of thing all the time.

Sol: //Defining a object with 2 property

const person1 = {

firstName : "Ram",

lastName : "Bahadur"

};

//Accessing properties

console.log("Name : "+person1.firstName+" "+person1.lastName);

//Updating property

person1.lastName="Singh";

//Accessing updated properties

console.log("Name : "+person1.firstName+" "+person1.lastName);

**5.** Try reading the middleName property from your variable above. Try assigning to the middleName

property. Try reading the property again after you assign to it. Is this behavior a good thing or a bad

thing?

Sol: //Defining a object with 2 property

const person1 = {

firstName : "Ram",

lastName : "Bahadur"

};

//Accessing a undefined properties

console.log("Middle Name : "+person1.middleName);

person1.middleName="Kumar";

//Accessing after defining properties

console.log("Middle Name : "+person1.middleName);

//This property can make the whole code break as it doesn't throw an exception or error when accessing a undefined property

**6.** Create a string that contains what looks like an object with firstName and lastName properties. Use

“eval” to turn it into a real object, and test it the same way you did with the previous object that you

created directly.

Sol: //Defining a object with 2 property using eval function with a string having object definition

var objectStr = '( { firstName : "Ram", lastName : "Bahadur" } )';

const person1=eval(objectStr);

//Accessing a undefined properties

console.log("Name : "+person1.firstName+" "+person1.middleName+" "+person1.lastName);

person1.middleName="Kumar";

//Accessing after defining properties

console.log("Name : "+person1.firstName+" "+person1.middleName+" "+person1.lastName);

**7.** Do the same with JSON.parse. You have to follow strict JSON rules in this case.

Sol: //Defining a object with 2 property using JSON parse

const person1 = JSON.parse('{"firstName" : "Ram", "lastName" : "Bahadur"}');

//Accessing a undefined properties

console.log("Middle Name : "+person1.middleName);

person1.middleName="Kumar";

//Accessing after defining properties

console.log("Middle Name : "+person1.middleName);

//This property can make the whole code break as it doesn't throw an exception or error when accessing a undefined property

**1**. Write a JS program to create object of person with fields as follows:-

fname - string

lname - string

age - int

skills - array

address - object

city - string

pincode - int

dateOfBirth - Date

married - Boolean

profession - string

Create minimum 2 objects and display the object's detail uisng global print method.

CODE::

function person(fname,lname,age,skills,dateofbirth,address,married,profession)

{

this.fname=fname;

this.lname=lname;

this.age=age;

this.skills=skills;

this.dateofbirth=dateofbirth;

this.address=address;

this.married=married;

this.profession=profession;

}

person1=new

person("nikhil","goud",22,["c"],"24/10/1996",{city:"hyderabad",pincode:"521185"},"false","sr analyst")

person2=new

person("harish","chinna",21,"HTML","08/06/1997",{city:"Ameerpet",pincode:"500038"},"false","jr

analyst")

print=function()

{

console.log(person1);

console.log(person2);

}();

Sol*: function person(fname,lname,age,skills,dateofbirth,address,married,profession)*

*{*

*this.fname=fname;*

*this.lname=lname;*

*this.age=age;*

*this.skills=skills;*

*this.dateofbirth=dateofbirth;*

*this.address=address;*

*this.married=married;*

*this.profession=profession;*

*}*

*var person1 = new person("nikhil","goud",22,["c"],"24/10/1996",{city:"hyderabad",pincode:"521185"},"false","sr analyst");*

*var person2=new person("harish","chinna",21,"HTML","08/06/1997",{city:"Ameerpet",pincode:"500038"},"false","jr analyst");*

*print=function()*

*{*

*console.log(person1);*

*console.log(person2);*

*}();*

**2.** **Modify the above program to create 2 objects, amitabh and abhishek, here abhishek has some**

**common properties from amitabh, try to use it such common properties from amitabh instead of**

**creating it in abhishek.**

**CODE::function person(fname,lname,age,skills,dateofbirth,address,married,profession)**

**{**

**this.fname=fname;**

**this.lname=lname;**

**this.age=age;**

**this.skills=skills;**

**this.dateofbirth=dateofbirth;**

**this.address=address;**

**this.married=married;**

**this.profession=profession;**

**}**

**amithab=new**

**person("amithab","bachan",22,["c"],"24/10/1996",{city:"hyderabad",pincode:"521185"},"false","sr**

**analyst")**

**abhisheik=new person("abhisheik",21,"HTML","08/06/1997","false","jr analyst")**

**var abhisheik=Object.create(amithab);**

**print=function()**

**{**

**console.log(amithab);**

**console.log(abhisheik.lname);**

**console.log(abhisheik.address);**

}();

**Sol:**

**3.** Modify the above code to create third object as "Aaradhya", this object shares the common

properties from amitabh as well as abhishek accordingly demostrate on your own.

**Sol:**

**4. Create a JS program to implement below:-**

**BankAccount**

**|(accountNumber)**

**|(accountHolderName)**

**|(accountBalance)**

**-------------------------**

**| |**

**Savings Current**

**(isSalary) (odLimit)**

**Now,**

**create 1 object of savings with accountNumber, accountHolderName, accountBalance and isSalary.**

**create 1 object of currrent with accountNumber, accountHolderName, accountBalance and odLimit.**

**use savings account object to call a function withdraw(amount), that will subtract the amount from the**

**accountBalance, Note that accountBalance should not go below 0.**

**use current account object to calla function withdraw(amount), that will subtract the amount from the**

**accountBalance, Note that accountBalance should not go below negative of odLimit.**

**Now, call getCurrentBalance() from both the object, that will return the accountBalance of the object**

**used to call this method.**

**Sol:** //Parent class with accountNumber , Holder name and Balance

*class BankAccount{*

*constructor(accountNumber,accountHolder,accountBalance){*

*this.accountNumber=accountNumber;*

*this.accountHolder=accountHolder;*

*this.accountBalance=accountBalance;*

*}*

*getBalance()*

*{*

*console.log("Available balance : Rs. "+this.accountBalance);*

*}*

*}*

*//Inherited class for savings account*

*class Savings extends BankAccount{*

*constructor(accountNumber,accountHolder,accountBalance)*

*{*

*super(accountNumber,accountHolder,accountBalance);*

*}*

*withdraw(amount)*

*{*

*if(this.accountBalance-amount >= 0)*

*{*

*console.log("Withdrawal of Rs. "+amount+" successfull.");*

*this.accountBalance=this.accountBalance-amount;*

*}*

*else{*

*console.log("Withdrawal of Rs. "+amount+" failed : Not enough Balance");*

*}*

*}*

*}*

*//inherited class for current account*

*class Current extends BankAccount{*

*constructor(accountNumber,accountHolder,accountBalance,odLimit)*

*{*

*super(accountNumber,accountHolder,accountBalance);*

*this.odLimit=odLimit;*

*}*

*withdraw(amount)*

*{*

*if(this.accountBalance-amount >= -(this.odLimit))*

*{*

*console.log("Withdrawal of Rs. "+amount+" successfull.");*

*this.accountBalance=this.accountBalance-amount;*

*}*

*else{*

*console.log("Withdrawal of Rs. "+amount+" failed : Not enough Balance");*

*}*

*}*

*}*

*//operation on current account*

*var currentAccount = new Current(12,"AB",10000,1000);*

*console.log("Operation of Current account : ");*

*currentAccount.getBalance();*

*currentAccount.withdraw(10000);*

*currentAccount.getBalance();*

*currentAccount.withdraw(500);*

*currentAccount.getBalance();*

*currentAccount.withdraw(1000);*

*currentAccount.getBalance();*

*var savingAccount = new Savings(13,"CD",12000);*

*console.log("Operation of Current account : ");*

*savingAccount.getBalance();*

*savingAccount.withdraw(12000);*

*savingAccount.getBalance();*

*savingAccount.withdraw(1);*

*savingAccount.getBalance();*