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.

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

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
```

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.

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?

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.

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

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);
            }();
```

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);
            }();
```

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.

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.