DAY-8

Exercises: Level 1

1. Create an empty object called dog

```
let dog = {};
```

2. Print the the dog object on the console

```
console.log(dog);
```

3. Add name, legs, color, age and bark properties for the dog object. The bark property is a method which return *woof woof*

```
dog = {
    name: "Bruno",
    legs: 4,
    color: "Black",
    age: 10,
    bark:function() {
        return "Woof Woof";
    }
};
```

4. Get name, legs, color, age and bark value from the dog object

```
dog.name;
dog.legs;
dog.color;
dog.age;
dog.bark;
```

5. Set new properties the dog object: breed, getDogInfo

```
dog.breed = "Labra";
dog.getDogInfo = function() {
          return dog.name;
}
```

Exercises: Level 2

1. Find the person who has many skills in the users object.

```
const a = Object.keys();
const abb = [];

a.forEach(key => {
    if(users[key].skills.length >= 5)   abb.push(key);
});
console.log(abb);
```

2. Count logged in users, count users having greater than equal to 50 points from the following object.

```
const users = {
  Alex: {
    email: 'alex@alex.com',
    skills: ['HTML', 'CSS', 'JavaScript'],
    age: 20,
    isLoggedIn: false,
    points: 30
  },
  Asab: {
    email: 'asab@asab.com',
    skills: ['HTML', 'CSS', 'JavaScript', 'Redux', 'MongoDB',
'Express', 'React', 'Node'],
    age: 25,
    isLoggedIn: false,
    points: 50
  },
  Brook: {
    email: 'daniel@daniel.com',
    skills: ['HTML', 'CSS', 'JavaScript', 'React', 'Redux'],
    age: 30,
    isLoggedIn: true,
    points: 50
  },
  Daniel: {
    email: 'daniel@alex.com',
    skills: ['HTML', 'CSS', 'JavaScript', 'Python'],
    age: 20,
    isLoggedIn: false,
    points: 40
  },
  John: {
    email: 'john@john.com',
    skills: ['HTML', 'CSS', 'JavaScript', 'React', 'Redux', 'Node.js'],
    age: 20,
    isLoggedIn: true,
    points: 50
  },
  Thomas: {
    email: 'thomas@thomas.com',
    skills: ['HTML', 'CSS', 'JavaScript', 'React'],
    age: 20,
    isLoggedIn: false,
    points: 40
  },
  Paul: {
    email: 'paul@paul.com',
```

```
        skills: ['HTML', 'CSS', 'JavaScript', 'MongoDB', 'Express',
    'React', 'Node'],
        age: 20,
        isLoggedIn: false,
        points: 40
      }
    }
```

```
const s = [];
a.forEach(key => {
   if(users[key].isLoggedIn == true && users[key].points >=50)
         s.push(key);
});
console.log(s);
```

3. Find people who are MERN stack developer from the users object

```
const s= [];
a.forEach(key => {
   if(users[key].skills.includes('MongoDB','Express','React',
'Node') == true)
          s.push(key);
});
console.log(s);
```

4. Set your name in the users object without modifying the original users object

```
users.Priyanka: {
    email: 'priyanka@paul.com',
    skills: ['HTML', 'CSS', 'JavaScript', 'Java'],
    age: 20,
    isLoggedIn: false,
    points: 40
}
```

5. Get all keys or properties of users object

```
const array = [];
array = Object.keys(users);
console.log(array);
```

6. Get all the values of users object

```
const array = [];
array = Object.values(users);
console.log(array);
```

Exercises: Level 3

1. Create an object literal called *personAccount*. It has *firstName, lastName, incomes, expenses* properties and it has *totalIncome, totalExpense, accountInfo,addIncome,*

addExpense and *accountBalance* methods. Incomes is a set of incomes and its description and expenses is a set of incomes and its description.

```
personAccount = {
  firstName: "Priyanka",
  lastName: "Saini",
  incomes: {
    5: "Salary",
    6: "From Friend"
  },
  expenses: {
    2: "Food"
  },
  totalIncome: function() {
    const incomeArray = Object.keys(this.incomes);
    let sum = 0;
    for(i of incomeArray) sum += parseInt(i);
    return sum;
  },
  totalExpense: function() {
    const expenseArray = Object.keys(this.expenses);
    let sum = 0;
    for(i of expenseArray) sum += parseInt(i);
    return sum;
  },
  accountInfo: function() {
    return `${this.firstName} ${this.lastName}`;
  },
  addIncome: function(amount, desc) {
    this.incomes[`${amount}`] = desc;
    return `Added`;
  },
  addExpense: function(amount, desc) {
    this.expenses[`${amount}`] = desc;
    return `Added`;
  },
  accountBalance: function() {
    return `${this.totalIncome()}`-`${this.totalExpense()}`;
  }
}
```

2. **** Questions:2, 3 and 4 are based on the following two arrays:users and products
   ()

```
const users = [
 {
     _id: 'ab12ex',
     username: 'Alex',
```

```
    email: 'alex@alex.com',
    password: '123123',
    createdAt:'08/01/2020 9:00 AM',
    isLoggedIn: false
  },
  {
    _id: 'fg12cy',
    username: 'Asab',
    email: 'asab@asab.com',
    password: '123456',
    createdAt:'08/01/2020 9:30 AM',
    isLoggedIn: true
  },
  {
    _id: 'zwf8md',
    username: 'Brook',
    email: 'brook@brook.com',
    password: '123111',
    createdAt:'08/01/2020 9:45 AM',
    isLoggedIn: true
  },
  {
    _id: 'eefamr',
    username: 'Martha',
    email: 'martha@martha.com',
    password: '123222',
    createdAt:'08/01/2020 9:50 AM',
    isLoggedIn: false
  },
  {
    _id: 'ghderc',
    username: 'Thomas',
    email: 'thomas@thomas.com',
    password: '123333',
    createdAt:'08/01/2020 10:00 AM',
    isLoggedIn: false
  }
  ];

  const products = [
{
  _id: 'eedfcf',
  name: 'mobile phone',
  description: 'Huawei Honor',
  price: 200,
  ratings: [
```

```
      { userId: 'fg12cy', rate: 5 },
      { userId: 'zwf8md', rate: 4.5 }
    ],
    likes: []
  },
  {
    _id: 'aegfal',
    name: 'Laptop',
    description: 'MacPro: System Darwin',
    price: 2500,
    ratings: [],
    likes: ['fg12cy']
  },
  {
    _id: 'hedfcg',
    name: 'TV',
    description: 'Smart TV:Procaster',
    price: 400,
    ratings: [{ userId: 'fg12cy', rate: 5 }],
    likes: ['fg12cy']
  }
]
```

Imagine you are getting the above users collection from a MongoDB database.

a) Create a function called signUp which allows user to add to the collection. If user exists, inform the user that he has already an account.

```
const signUp = (username, email, password) => {
  for(user of users) {
    if(user.email == email || user.username == username) {
      return `User Already Exists`;
    } else {
      continue;
    }
  }
  const id =  generateID();
  const obj = {_id: `${id}`,
               username: `${username}`,
               email: `${email}`,
               password: `${password}`,
               createdAt: `${new Date().toLocaleDateString()}
${new Date().toLocaleTimeString()}`,
               isLoggedIn: false

  }
  users.push(obj);
  return users;
```

```
}
function generateID() {
  const str = "abcdefghijklmnopqrstuvwxyz1234567890";
  let randomStr = "";
  let randomIndex = 0;
  for(let i = 0; i < 6; i++) {
    randomIndex = Math.floor(Math.random()*str.length);
    randomStr += str.charAt(randomIndex);
  }
  for(user of users) {
    if(randomStr == user._id) return generateID();
  }
  return randomStr;
}
console.log(signUp("Priyanka", "priyanka@gmail" ,"111"));
```

b) Create a function called signIn which allows user to sign in to the application

```
const signIn = (email) => {
  for(user of users) {
    if(user.email == email) {
      if(user.isLoggedIn == false) {
        user.isLoggedIn = true;
        return `You are logged in!`
      }
      else return `You are already logged in`;
    } else {continue}
  }
  return `User not found!`;
}
console.log(signIn("thomas@thomas.com"));
```

3. The products array has three elements and each of them has six properties.
   a) Create a function called rateProduct which rates the product

```
const rateProduct = (username, p_id, rating) => {
  let u_id = "";
  let u_obj;
  let p_idValid = false
  let p_obj;
  for(user of users) {
    if(user.username == username) {
      u_id = user._id;
      u_obj = user;
    }
    else continue;
  }
  for(product of products) {
    if(product._id == p_id) {
      p_idValid = true;
```

```
      p_obj = product;
    }
    else continue;
  }
  if(u_id == "") return `User not found!`;
  if(p_idValid == false) return `Product not found!`;
  else {
    const rate = {userId: u_id,
                  rate: rating
    }
    p_obj.ratings.push(rate);
  }
  return products;
}
console.log(rateProduct("Thomas", "hedfcg", 4.5));
```

b) Create a function called averageRating which calculate the average rating of a product

```
const averageRating  = (p_id) => {
  let p_idValid = false
  let p_obj_rating;
  for(product of products) {
    if(product._id == p_id) {
      p_idValid = true;
      p_obj_rating = product.ratings;
    }
    else continue;
  }
  if(p_idValid == false) return `Product not found!`;
  else {
    let sum = 0;
    for(rating of p_obj_rating) {
      sum += rating.rate;
    }
    return sum/p_obj_rating.length;
  }
}
console.log(averageRating("eedfcf"));
```

4. Create a function called likeProduct. This function will helps to like to the product if it is not liked and remove like if it was liked.

```
const likeProduct = (username, p_id) => {
  let u_id = "";
  let u_obj;
```

```
  let p_idValid = false
  let p_obj;
  for(user of users) {
    if(user.username == username) {
      u_id = user._id;
      u_obj = user;
    }
    else continue;
  }
  for(product of products) {
    if(product._id == p_id) {
      p_idValid = true;
      p_obj = product;
    }
    else continue;
  }
  if(u_id == "") return `User not found!`;
  if(p_idValid == false) return `Product not found!`;
  else {
    if(p_obj.likes.includes(u_id)) {
      let index = p_obj.likes.indexOf('u_id');
      p_obj.likes.splice(index, 1);
      return `Like Removed!`;
    } else {
      p_obj.likes.push(u_id);
      return `Like Added!`;
    }
  }
}
console.log(likeProduct("Asab", "hedfcg"));
console.log(products);
```