CS472 Web Application Programming

2023 October block

Practice (Solution)

Full name_____

Student ID _____

I. True/False	II. Multiple Choice	III. Short Answer	IV. Programming	Total

Q1: const numbers = [1, 5, 18, 2, 77, 108]; print the odd numbers. You're not allowed to use **for, while, do...while, for..of, for..in** loops. You may use forEach method.

```
const numbers = [1, 5, 18, 2, 77, 108]
numbers.filter((e) => e % 2 == 1).forEach(e => console.log(e));
```

Q2: Create a function using function declaration named sum with one parameter of Array type, the returned result is the sum of all elements which are greater than 20. You're not allowed to use **for, while, do...while, for..of, for..in** loops. You may use forEach method.

sum([10, 20, 50, 30, 8]);

```
function sum(arr) {
    return arr.filter(e => e > 20).reduce((acc, cur) => acc + cur, 0)
}
console.log(sum([10, 20, 50, 30, 8]));
```

Q3: Create a function using function expression named getNewArray with one parameter of String Array, return a new array which contains all string, length is greater than and equal to 5, and contains letter 'a'. You're not allowed to use **for, while, do...while, for..of, for..in** loops.

["Hello", "Wonderful", "Happy", "People", "Have a great day"]

```
let getNewArray = function (arr) {
    return arr.filter(e => e.length >= 5).filter(e => e.includes("a"));
}
console.log(getNewArray(["Hello", "Wonderful", "Happy", "People", "Have a great
day"]));
```

Q4-a: (non-strict mode)

```
var a = 2;
let b = 3;
function outer() {
    let c = 5;
    var d = 7;
   return function inner() {
        b = 8;
        let c = 9;

        console.log(a); //2
        console.log(b); //8
        console.log(c); //9
        console.log(d); //7
    }
}
outer()();
```

What will be the output?

```
2
8
9
7
```

Based on the code above,

1. What's the LE of global EC after creation phase finished before execution phase starts?

   `LE: {a: undefined, outer: fn}, tdz{ b }, outer: null, this: window`

2. What's the LE of global EC after execution phase finished?

   `LE: {a: 2, b: 3, outer: fn}, outer: null, this: window`

3. What's the LE of function outer EC after creation phase finished before execution phase starts?

   `LE: { d: undefined, arguments: { length: 0 } }, tdz{ c },`
   `outer: global, this: window`

4. What's the LE of function outer EC after execution phase finished?

   `LE: {d: 7, c: 5, inner: fn, arguments: { length: 0 } },`
   `outer: global, this: window`

5. What's the LE of function inner EC after creation phase finished before execution phase starts?

   `LE: { arguments: { length: 0 } }, tdz{ c }, closure (outer):{d:7},`
   `outer: outer, this: window`

6. What's the LE of function inner EC after execution phase finished?

   `LE: { arguments: { length: 0 },c:9 }, closure (outer):{d:7},`
   `outer: outer, this: window`

Q4-b:

What will be the output?

```
function log(e) {
    console.log(e);
}
let arr = [1, 2, 3];
console.log("start")
setTimeout(() => arr.forEach(log));
console.log("end")
```

start

end

1

2

3

Q4-c:

What will be the output?

```
function log(e) {
    console.log(e);
}
let arr = [1, 2, 3];
console.log("start")
arr.forEach(log)
console.log("end")
```

start

1

2

3

end

Q5: When the HTML is

```
<body>
  <p style="color:red">First</p>
  <div class="central">
    <p class="item">Second</p>
    <ul>
      <li id="item">Third</li>
    </ul>
  </div>
</body>
```

And the CSS is:

```
body {
  background-color: yellow;
  color: blue;
}
p { color: orange; }
.central, .item {
  color: green;
}
#item {
  background-color: white;
}
ul{
 color: purple;
 background-color: beige;
}
```

Specify what colors will show on the screen for the:

	Background-color	color
First	yellow	red
Second	yellow	green
Third	white	purple

Q6: Write a regular expression that matches a string containing a date in the format mm/dd/yyyy.

```
^(0[1-9]|1[0-2])
\/
(0[1-9]|[12][0-9]|3[01])  - 01 – 09, 10-19|20-29, 30|31
\/
\d{4}$
\d\d\/\d\d\/\d\d\d\d
```

Q7: For the given students array below, compute the average grade of all students who took cs303 course which returns an object which key is students' names, value is the average.

Expected result:

```
{
      Quincy: 93.5,
      Sam: 86.5,
      Katie: 71.5
}
```

You're not allowed to use **for, while, do...while, for..of, for..in, forEach method**.

```
const students = [
  { name: 'Quincy', grades: [99, 88], courses:['cs301', 'cs303']}, -
  { name: 'Jason', grades: [29, 38], courses:['cs201', 'cs203']},
  { name: 'Alexis', grades: [79, 78], courses:['cs105', 'cs211'] },
  { name: 'Sam', grades: [91, 82], courses:['cs445', 'cs303'] }, -
  { name: 'Katie', grades: [66, 77], courses:['cs303', 'cs477'] } -
];
```

```
let result = students.filter(e => e.courses.includes("cs303"))
    .reduce((acc, cur) => {
        let avg = cur.grades.reduce((sum, e, cs) => sum + e, 0) / cur.grades.length;
        acc[cur.name] = avg;
        return acc;
    }, {});

console.log(result);
```

Q8: Please work on a project that involves creating various types of vehicles.

Create a `Vehicle` constructor function that has the following properties and methods:

Properties:

`make` - the make of the vehicle

`model` - the model of the vehicle

`year` - the year the vehicle was made

`mileage` - the number of miles the vehicle has been driven

Methods:

`drive(distance)` - a method that takes a distance (in miles) as an argument and increases the vehicle's mileage by that distance

`toString()` - a method that returns a string representation of the vehicle, in the format "YEAR MAKE MODEL (MILEAGE miles)"

Create a `Car` constructor function that extends the `Vehicle` constructor function, with the following additional properties and methods:

Properties:

`numDoors (number)` - the number of doors on the car

`speed (number)` - the current speed of the car, in mph

`topSpeed (number)` - the top speed of the car, in mph

Methods:

`accelerate()` - a method that increases the car's speed by 10 mph

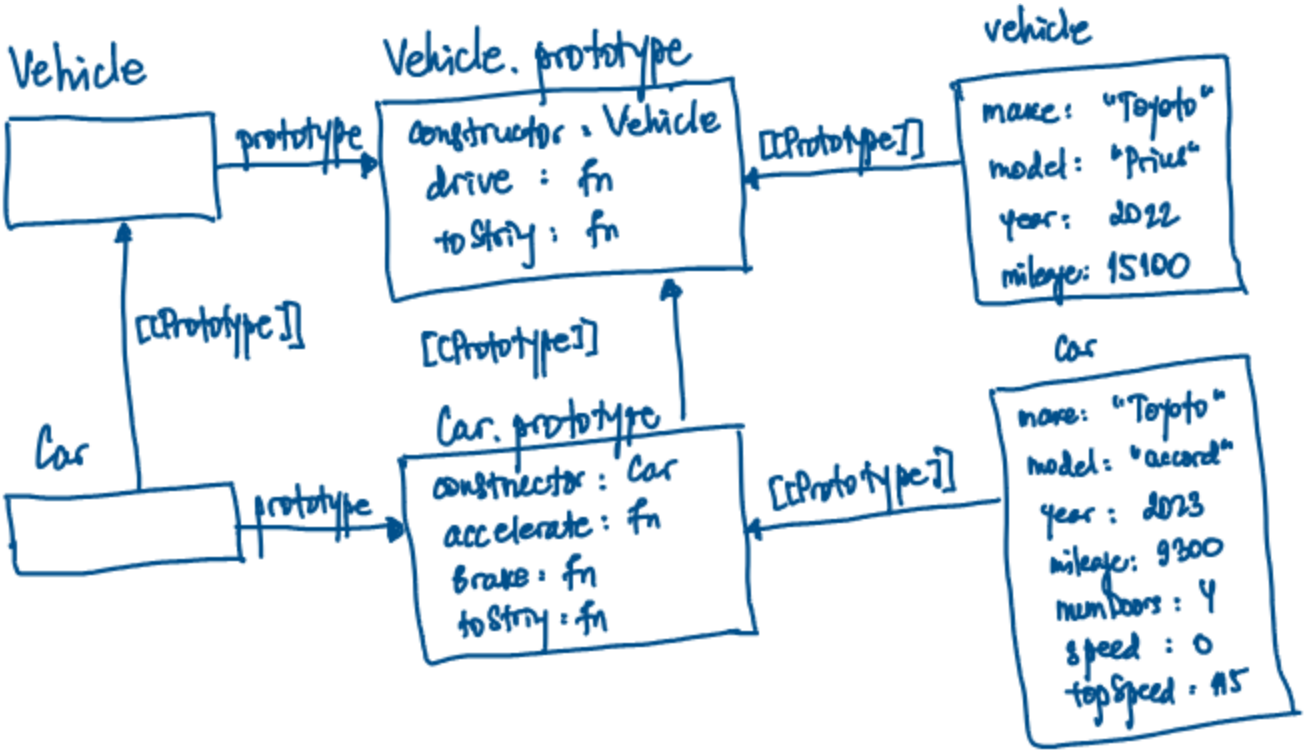
`brake()` - a method that decreases the car's speed by 10 mph

`toString()` - a method that overrides the parent toString() method, and returns a string representation of the sports car in the format "YEAR MAKE MODEL (MILEAGE miles), NUMDOORS doors, TOPSPEED mph"

To do things:

1. Implement it using **constructor function** with the appropriate inheritance,
2. Write code to test their functionality, and
3. Draw a prototypal inheritance diagram (no need to include Function.prototype and Object.prototype).

```
function Vehicle(make, model, year, mileage) {
    this.make = make;
    this.model = model;
    this.year = year;
    this.mileage = mileage;
}
Vehicle.prototype.drive = function (distance) {
    this.mileage += distance;
}

Vehicle.prototype.toString = function () {
    return `${this.year} ${this.make} ${this.model} (${this.mileage} miles)`;
}

Object.setPrototypeOf(Car, Vehicle);
Object.setPrototypeOf(Car.prototype, Vehicle.prototype);

function Car(make, model, year, mileage, numDoors, speed, topSpeed) {
    Vehicle.call(this, make, model, year, mileage);
    this.numDoors = numDoors;
    this.speed = speed;
    this.topSpeed = topSpeed;
}

Car.prototype.accelerate = function () {
    this.speed += 10;
}
Car.prototype.brake = function () {
    this.speed -= 10;
}
Car.prototype.toString = function () {
    return Vehicle.prototype.toString.call(this) + ` ${this.numDoors}
${this.topSpeed}`;
}

let vehicle = new Vehicle("Toyoto", "Prius", 2022, 15000);
vehicle.drive(100);
console.log(vehicle.toString()); // 2022 Toyoto Prius (15100 miles)

let car = new Car("Toyoto", "Accord", 2023, 9000, 4, 0, 115);
car.drive(300);
car.accelerate();
car.accelerate();
console.log(car.toString()); //2023 Toyoto Accord (9300 miles) 4 115
```

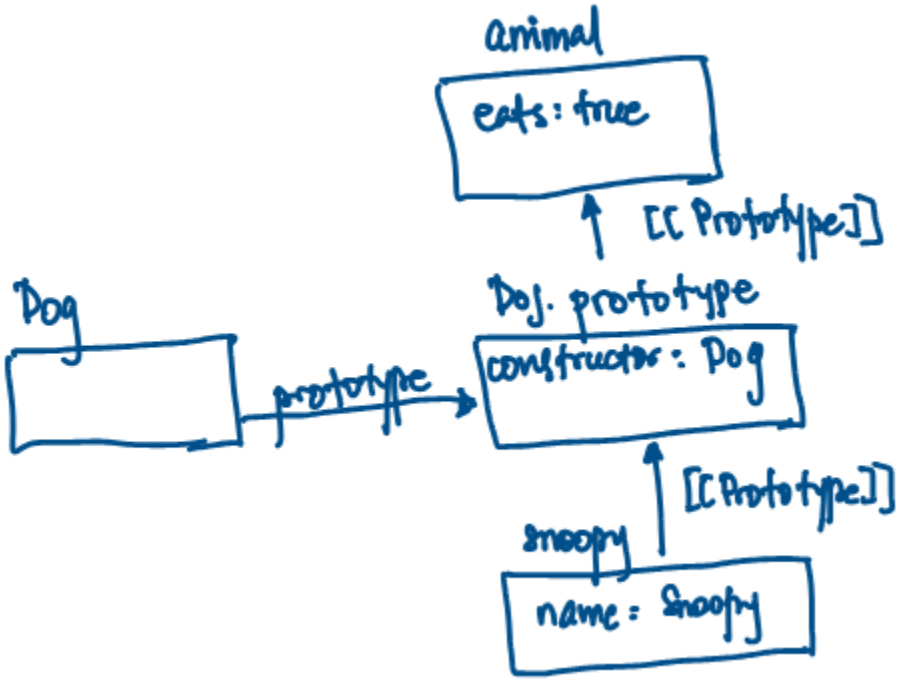

Vehicle
prototype
Vehicle.prototype
constructor: Vehicle
drive: fn
toString: fn
[[Prototype]]
vehicle
make: "Toyota"
model: "Prius"
year: 2022
mileage: 15100
[[Prototype]]
[[Prototype]]
Car
Car.prototype
constructor: Car
accelerate: fn
brake: fn
toString: fn
prototype
[[Prototype]]
Car
make: "Toyota"
model: "accord"
year: 2023
mileage: 9300
numDoors: 4
speed: 0
topSpeed: 115

Q9: Draw the prototypal inheritance diagram based on the code below: (no need to include Function.prototype and Object.prototype)

```
let animal = {
    sleep: function(){
        this.sleeping = true;
    },
    walk: function(){
        if(!this.sleeping){
            console.log('animal walking');
        } else {
            console.log('animal is sleeping');
        }
    }
}
let rabbit = {
    jump: true,
    sleep: function(){
        console.log('Sleeping!');
    }
};
rabbit = Object.create(animal);

animal.eat = true;
```

Q10: Draw the prototypal inheritance diagram based on the code below: (no need to include Function.prototype and Object.prototype)

```
let animal = {
    eats: true
};
function Dog(name) {
    this.name = name;
}
Object.setPrototypeOf(Dog.prototype, animal)
let snoopy = new Dog("Snoopy");
```



Q11: When you run the code snippet below, you will observe the following output in the console:

```
undefined: John
undefined: Pete
undefined: Alice
```

Your task is to correct the code using one of four different techniques (bind, call, apply, and self pattern) to achieve the desired output:

```
Our Group: John
Our Group: Pete
Our Group: Alice
```

Below is the original code:

```
let group = {
    title: "Our Group",
    students: ["John", "Pete", "Alice"],
    showList: function() {
        this.students.forEach(function(student) {
            console.log(this.title + ": " + student);
        });
    }
};
group.showList();
```

```
let groupBind = {
    title: "Our Group",
    students: ["John", "Pete", "Alice"],
    showList: function () {
        this.students.forEach(function (student) {
            console.log(this.title + ": " + student);
        }.bind(this));
    }
};
groupBind.showList();
```