***2.1)Datatypes in js:***

***1)Basic Mathematical Operations:***

*function basicOp(operation, value1, value2)*

*{*

*return eval(value1+operation+value2);*

*}*

***2)Opposite Number:***

*function opposite(number) {*

*return -number;*

*}*

***3)Printing Array elements with Delimiters:***

*function printArray(array){*

*return array.join(',');*

*}*

***4)Transportation on vacation:***

function rentalCarCost(d) {

let day = d >= 3 && d < 7 ? 20 : 0,

week = d >= 7 ? 50 : 0;

return (d \* 40) - day - week;

}

***2.2 Understanding lexical scope :***

***1)***

function zero(numStr) {

return !numStr ? 0 : parseInt(eval(0 + numStr));

}

function one(numStr) {

return !numStr ? 1 : parseInt(eval(1 + numStr));

}

function two(numStr) {

return !numStr ? 2 : parseInt(eval(2 + numStr));

}

function three(numStr) {

return !numStr ? 3 : parseInt(eval(3 + numStr));

}

function four(numStr) {

return !numStr ? 4 : parseInt(eval(4 + numStr));

}

function five(numStr) {

return !numStr ? 5 : parseInt(eval(5 + numStr));

}

function six(numStr) {

return !numStr ? 6 : parseInt(eval(6 + numStr));

}

function seven(numStr) {

return !numStr ? 7 : parseInt(eval(7 + numStr));

}

function eight(numStr) {

return !numStr ? 8 : parseInt(eval(8 + numStr));

}

function nine(numStr) {

return !numStr ? 9 : parseInt(eval(9 + numStr));

}

function plus(numStr) {

return '+' + numStr;

}

function minus(numStr) {

return '-' + numStr;

}

function times(numStr) {

return '\*' + numStr;

}

function dividedBy(numStr) {

return '/' + numStr;

}

2)Get the Middle Character:

function getMiddle(string) {

var middleIndex = string.length / 2;

if (string.length % 2 == 0) {

return string.slice(middleIndex - 1, middleIndex + 1);

} else {

return string.charAt(middleIndex);

}

}

3)Partition On

function partitionOn(pred, items) {

let evenItems = items.filter(el=>pred(el));

let oddItems = items.filter(el=>!pred(el));

items.length = 0;

items.push.apply(items, oddItems.concat(evenItems));

return items.indexOf(evenItems[0])

}

***2.3)Dynamic Scope:***

1)A function within a function

function always (n) {

return function(){return n};

}

2)Closures and Scopes

function createFunctions(n) {

var callbacks = [];

var fu = function (x) {

return function () { return x };

};

for (var i=0; i<n; i++) {

callbacks.push(fu(i));

}

return callbacks;

}

3)Keep a secret

function createSecretHolder(secret) {

var \_secret = secret;

return {

setSecret: function(s) {

\_secret = s; },

getSecret: function() {

return \_secret;

}

}

}

3)Prefill an array:

function prefill(n, v) {

try {

var arr = Array.apply(null, Array(typeof n=='boolean'? parseInt(n): +n));

return arr.map(function() {

return v;

});

} catch (e) {

throw new TypeError(n+' is invalid');

}

}

***2.4)Object and array prototype:***

*1)Function Composite:*

function compose(...args) {

return function(n){

var result=n;

while(args.length!==0)

{

result = args.pop()(result);

}

return result;

}

};

2)Function Cache:

function cache(func) {

const cachedResults = {};

return (...args) => {

const key = JSON.stringify(args);

if (!(key in cachedResults)) {

const result = func.apply(null, args);

cachedResults[key] = result;

}

return cachedResults[key];

};

}

**3)Using closures to share class states:**

var Cat = (function () {

const cats = []

const constructor = function (name, weight) {

if (!name || !weight) throw Error('invalid parameters')

Object.defineProperty(this, 'name', { get: () => name })

Object.defineProperty(this, 'weight', { get: () => weight, set: value => weight = value })

cats.push(this)

}

constructor.averageWeight = () => cats.reduce((acc, cat) => (acc + cat.weight), 0) / cats.length

return constructor

}())

4)A chain adding function:

function add(n) {

const func = x => add(n + x)

func.valueOf = () => n

return func

}

***2.5)OOPS in JS:***

***1)Santaclause interface:***

*function isSantaClausable(obj) {*

*return ['sayHoHoHo', 'distributeGifts', 'goDownTheChimney'].every(function(methodName) {*

*return typeof obj[methodName] == 'function';*

*});*

***}***

***2)New with Apply:***

*function construct(Class) {*

*var obj = Object.create(Class.prototype);*

*Class.apply(obj, Array.prototype.slice.call(arguments, 1));*

*return obj;*

*}*

**3)Array Helpers:**

Array.prototype.square = function() {

return this.map(function(item) {

return Math.pow(item, 2);

});

}

Array.prototype.cube = function() {

return this.map(function(item) {

return Math.pow(item, 3);

});

}

Array.prototype.average = function() {

if(this.length==0){return NaN}

return this.reduce(function(p, c) {return p+c;}) / this.length;

}

Array.prototype.sum = function() {

if(this.length==0){return 0}

return this.reduce(function(p, c) {

return p+c;

});

}

Array.prototype.even = function() {

return this.filter(function(item) {

return item%2==0;

});

}

Array.prototype.odd = function() {

return this.filter(function(item) {

return item%2==1;

});

}

**4)Extract Nested Object Reference:**

Object.prototype.hash = function(string) {

try {

return eval(`this.${string}`)

}

catch(e){

return undefined

}

}