# Solutions-Exercises: Objects, Inheritance and Prototypes

## 1.Array extension

(function extendArray(){

Array.prototype.last = function() {

return this[this.length - 1];

};

Array.prototype.skip = function(n) {

let skippedArray = [];

for (let i = n; i < this.length; i++) {

let elemenToReturn = this[i];

skippedArray.push(elemenToReturn);

}

return skippedArray;

};

Array.prototype.take = function(n){

let takedArray = [];

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

let elementToreturn = this[i];

takedArray.push(elementToreturn);

}

return takedArray;

};

Array.prototype.sum = function() {

let sum = 0;

for (let i = 0; i < this.length; i++) {

let elementToSum = this[i];

sum += elementToSum;

}

return sum;

};

Array.prototype.average = function() {

let sum = 0;

for (let i = 0; i < this.length; i++) {

let elementToSum = this[i];

sum += elementToSum;

}

let average = sum / this.length;

return average;

};

})()

|  |
| --- |
| (function solve() { |
|  | Array.prototype.last = function () { |
|  | return this[this.length - 1]; |
|  | }; |
|  | Array.prototype.skip = function (n) { |
|  | let results = []; |
|  | for (let i = n; i < this.length; i++) { |
|  | const element = this[i]; |
|  | results.push(element); |
|  | } |
|  | return results; |
|  | } |
|  | Array.prototype.take = function (n) { |
|  | let results = []; |
|  | for (let i = 0; i < this.length || i < n; i++) { |
|  | const element = this[i]; |
|  | results.push(element); |
|  | } |
|  | return results; |
|  | } |
|  | Array.prototype.sum = function () { |
|  | return this.reduce((a, b) => a + b); |
|  | } |
|  | Array.prototype.average = function () { |
|  | return (this.sum() / this.length); |
|  | } |
|  | })() |

|  |
| --- |
| (function arrayExtension() { |
|  | Array.prototype.last = function () { |
|  | return this[this.length - 1]; |
|  | }; |
|  |  |
|  | Array.prototype.skip = function (n) { |
|  | return this.slice(n); |
|  | }; |
|  |  |
|  | Array.prototype.take = function (n) { |
|  | return this.slice(0, n); |
|  | }; |
|  |  |
|  | Array.prototype.sum = function () { |
|  | return this.reduce((a, b) => a+b); |
|  | }; |
|  |  |
|  | Array.prototype.average = function () { |
|  | return this.sum()/this.length; |
|  | } |
|  | })() |

## 2.Construction Crew

function modifyWorker(worker){

if(worker.handsShaking === true){

worker.bloodAlcoholLevel += 0.1 \* worker.experience \* worker.weight;

worker.handsShaking = false;

}

return worker;

}

console.log(modifyWorker({ weight: 80, experience: 1, bloodAlcoholLevel: 0, handsShaking: true }))

console.log(modifyWorker({ weight: 120, experience: 20, bloodAlcoholLevel: 200, handsShaking: true }))

console.log(modifyWorker({ weight: 95, experience: 3, bloodAlcoholLevel: 0, handsShaking: false }))

|  |
| --- |
| function solve(worker) { |
|  |  |
|  | if (worker.handsShaking) { |
|  | let amount = 0.1 \* (worker.weight \* worker.experience); |
|  | worker.bloodAlcoholLevel += amount; |
|  | worker.handsShaking = false; |
|  | } |
|  |  |
|  | return worker; |
|  | } |
|  |  |
|  | console.log(solve({ |
|  | weight: 80, |
|  | experience: 1, |
|  | bloodAlcoholLevel: 0, |
|  | handsShaking: true |
|  | })) |

|  |
| --- |
| function constructionCrew(worker) { |
|  | if(worker.handsShaking){ |
|  | worker.bloodAlcoholLevel += 0.1 \* worker.weight \* worker.experience; |
|  | worker.handsShaking = false; |
|  | } |
|  | return worker; |
|  | } |

## 3.Car Factory

function assembleCar(carRequirements){

let car = {};

car.model = carRequirements.model;

car.engine = takeEngine(carRequirements.power);

car.carriage = {

type: carRequirements.carriage,

color: carRequirements.color

};

let wheelsize = carRequirements.wheelsize % 2 !== 0 ? carRequirements.wheelsize : carRequirements.wheelsize - 1;

//let wheelsize = Math.trunc(carRequirements.wheelsize) % 2 !== 0 ? Math.trunc(carRequirements.wheelsize) : Math.trunc(carRequirements.wheelsize - 1);

car.wheels = [wheelsize, wheelsize, wheelsize, wheelsize];

return car;

function takeEngine(power){

let engine = {};

if (power <= 90){

engine = {

power: 90,

volume: 1800

}

}

else if (power <= 120){

engine = {

power: 120,

volume: 2400

}

}

else {

engine = {

power: 200,

volume: 3500

}

}

return engine;

}

}

console.log(assembleCar({ model: 'VW Golf II', power: 90, color: 'blue', carriage: 'hatchback', wheelsize: 14 }))

console.log(assembleCar({ model: 'Opel Vectra', power: 110, color: 'grey', carriage: 'coupe', wheelsize: 17 }))

|  |
| --- |
| function solve(blueprint) { |
|  | const engines = [{ |
|  | power: 90, |
|  | volume: 1800 |
|  | }, |
|  | { |
|  | power: 120, |
|  | volume: 2400 |
|  | }, |
|  | { |
|  | power: 200, |
|  | volume: 3500 |
|  | } |
|  | ]; |
|  |  |
|  | function getEngine(power) { |
|  | return engines.filter(e => e.power >= power)[0]; |
|  | } |
|  |  |
|  | function getCarriage(type, color) { |
|  | return { |
|  | type: type, |
|  | color: color |
|  | }; |
|  | } |
|  |  |
|  | function getWheels(size) { |
|  | let results = []; |
|  | for (let j = 0; j < 4; j++) { |
|  | results.push(Math.trunc(size) % 2 !== 0 ? Math.trunc(size) : Math.trunc(size - 1)); |
|  | } |
|  |  |
|  | return results; |
|  | } |
|  |  |
|  | return { |
|  | model: blueprint.model, |
|  | engine: getEngine(blueprint.power), |
|  | carriage: getCarriage(blueprint.carriage, blueprint.color), |
|  | wheels: getWheels(blueprint.wheelsize) |
|  | } |
|  | } |
|  |  |
|  | console.log(solve({ |
|  | model: 'VW Golf II', |
|  | power: 90, |
|  | color: 'blue', |
|  | carriage: 'hatchback', |
|  | wheelsize: 14 |
|  | })) |

|  |
| --- |
| function carFactory(wantedCar) { |
|  | let engines = [{ power: 90, volume: 1800 }, { power: 120, volume: 2400 }, { power: 200, volume: 3500 }]; |
|  | let carriages = [{ type: 'hatchback', color: wantedCar.color }, { type: 'coupe', color: wantedCar.color }]; |
|  | let wheelsize = wantedCar.wheelsize %2 == 1 ? wantedCar.wheelsize : wantedCar.wheelsize - 1; |
|  |  |
|  | return { |
|  | model: wantedCar.model, |
|  | engine: engines.filter(e => e.power >= wantedCar.power)[0], |
|  | carriage: carriages.filter(c => c.type == wantedCar.carriage)[0], |
|  | wheels: [wheelsize, wheelsize, wheelsize, wheelsize] |
|  | } |
|  | } |

## 4.Extensible object

function extendObject(){

let objectToExtend = {

extend: function(template){

for(let parentProperty of Object.keys(template)){

if(typeof template[parentProperty] === 'function'){

//Object.getPrototypeOf(objectToExtend)[parentProperty] = template[parentProperty];

Object.getPrototypeOf(this)[parentProperty] = template[parentProperty];

}

else {

//objectToExtend[parentProperty] = template[parentProperty];

this[parentProperty] = template[parentProperty];

}

}

}

};

return objectToExtend;

}

|  |
| --- |
| function extensibleObject() { |
|  | return obj = { |
|  | extend: function (template) { |
|  | for (let parentProp of Object.keys(template)) { |
|  | if (typeof (template[parentProp]) == "function") { |
|  | Object.getPrototypeOf(obj)[parentProp] = template[parentProp]; |
|  | } else { |
|  | obj[parentProp] = template[parentProp]; |
|  | } |
|  | } |
|  | } |
|  | }; |
|  | return obj; |
|  | } |

|  |
| --- |
| function extensibleObject() { |
|  | let obj = { |
|  | extend: function(template){ |
|  | for(let parentProp of Object.keys(template)){ |
|  | if(typeof(template[parentProp]) == "function"){ |
|  | Object.getPrototypeOf(obj)[parentProp] = template[parentProp]; |
|  | } else { |
|  | obj[parentProp] = template[parentProp]; |
|  | } |
|  | } |
|  | } |
|  | }; |
|  |  |
|  | return obj; |
|  | } |

## 5.String extension

(function extendString() {

String.prototype.ensureStart = function(str){

let stringToReturn = this.toString();

if (!stringToReturn.startsWith(str)) {

stringToReturn = str + stringToReturn;

}

return stringToReturn;

};

String.prototype.ensureEnd = function(str){

let stringToReturn = this.toString();

if(!stringToReturn.endsWith(str)){

stringToReturn += str;

}

return stringToReturn;

};

String.prototype.isEmpty = function(){

if(this.length === 0){

return true;

}

return false;

};

String.prototype.truncate = function(n){

let stringToReturn = this.toString();

if(n < 4){

return '.'.repeat(n);

}

if(stringToReturn.length <= n){

return stringToReturn;

}

stringToReturn = stringToReturn.substr(0, n - 2);

let lastWhitespaceIndex = stringToReturn.lastIndexOf(' ');

if(lastWhitespaceIndex === -1){

return stringToReturn.substr(0, stringToReturn.length - 1) + '...';

}

return stringToReturn.substr(0, lastWhitespaceIndex) + '...';

};

String.format = function(string, ...params){

for (let i = 0; i < params.length; i++) {

let index = string.indexOf(`{${i}}`);

while(index !== -1){

string = string.replace(`{${i}}`, params[i]);

index = string.indexOf(`{${i}}`);

}

}

return string;

};

})()

let str = 'my string'

str = str.ensureStart('my')

console.log(str)

str = str.ensureStart('hello ')

console.log(str)

str = str.truncate(16)

console.log(str)

str = str.truncate(14)

console.log(str)

str = str.truncate(8)

console.log(str)

str = str.truncate(4)

console.log(str)

str = str.truncate(2)

console.log(str)

str = String.format('The {0} {1} fox',

'quick', 'brown');

console.log(str)

str = String.format('jumps {0} {1}',

'dog');

console.log(str)

|  |
| --- |
| (() => { |
|  | String.prototype.ensureStart = function (str) { |
|  | let result = this + ""; |
|  | if (!result.startsWith(str)) { |
|  | result = str + result; |
|  | } |
|  | return result; |
|  | } |
|  | String.prototype.ensureEnd = function (str) { |
|  | let result = this + ""; |
|  | if (!result.endsWith(str)) { |
|  | result += str; |
|  | } |
|  | return result; |
|  | } |
|  | String.prototype.isEmpty = function () { |
|  | return this.length < 1; |
|  | } |
|  | String.prototype.truncate = function (n) { |
|  | let str = this + ""; |
|  | if (str.length < n) { |
|  | return str; |
|  | } else { |
|  | while (str.length > n && Array.from(str).filter(c => c === " ").length > 0) { |
|  | let words = str.split(/\s+/g); |
|  | words.pop(); |
|  | str = words.join(" ") + "..."; |
|  | } |
|  | if (str.length > n && Array.from(str).filter(c => c === " ").length === 0) { |
|  | if (str.replace(/**\.\.\.**$/g, "").length < 4) { |
|  | str = str.replace(/**\.\.\.**$/g, ""); |
|  | str = ".".repeat(n); |
|  | } else { |
|  | str = str.replace(/**\.\.\.**$/g, ""); |
|  | str = (str.substr(0, (str.length - n)) + "..."); |
|  | } |
|  |  |
|  | return str; |
|  | } |
|  | } |
|  |  |
|  | return str; |
|  | } |
|  | String.format = function () { |
|  | let formatString = arguments[0]; |
|  | let params = Array.from(arguments); |
|  | params.shift(); |
|  |  |
|  | let result = formatString; |
|  | let placeHoldercount = result.match(/{\d+}/g).length; |
|  |  |
|  | for (let pl = 0, pa = 0; pl < placeHoldercount && pa < params.length; pl++, pa++) { |
|  | const param = params[pa]; |
|  | let regex = new RegExp(`\\{${pl}\\}`, "g"); |
|  | result = result.replace(regex, param); |
|  | } |
|  |  |
|  | return result; |
|  | } |
|  | })(); |
|  |  |
|  | // let str = 'my string' |
|  | // str = str.ensureStart('my') |
|  | // str = str.ensureStart('hello ') |
|  | // str = str.truncate(16) |
|  | // str = str.truncate(14) |
|  | // str = str.truncate(8) |
|  | // str = str.truncate(4) |
|  | // str = str.truncate(2) |
|  | // str = String.format('The {0} {1} fox', |
|  | // 'quick', 'brown'); |
|  | // str = String.format('jumps {0} {1}', |
|  | // 'dog'); |
|  |  |
|  | // console.log(str); |
|  |  |
|  | // var testString = 'quick brown fox jumps over the lazy dog'; |
|  | // var answer = testString.ensureStart('the '); |
|  | // answer = answer.ensureStart('the '); |
|  | // console.log(answer); |

|  |
| --- |
|  |
| (function stringExtension() { |
|  | String.prototype.ensureStart = function (str) { |
|  | if(! this.toString().startsWith(str)){ |
|  | return str + this.toString(); |
|  | } |
|  | return this.toString(); |
|  | }; |
|  |  |
|  | String.prototype.ensureEnd = function (str) { |
|  | if(! this.toString().endsWith(str)){ |
|  | return this.toString() + str; |
|  | } |
|  | return this.toString(); |
|  | }; |
|  |  |
|  | String.prototype.isEmpty = function () { |
|  | return this.toString().localeCompare("") == 0; |
|  | }; |
|  |  |
|  | String.prototype.truncate = function (n) { |
|  | if(n <= 3){ |
|  | return ".".repeat(n); |
|  | } |
|  | if(this.toString().length <= n){ |
|  | return this.toString(); |
|  | } else { |
|  | let lastIndex = this.toString().substr(0, n - 2).lastIndexOf(" "); |
|  | if(lastIndex != -1){ |
|  | return this.toString().substr(0, lastIndex) + "..."; |
|  | } else { |
|  | return this.toString().substr(0, n-3) + "..."; |
|  | } |
|  | } |
|  | }; |
|  |  |
|  | String.format = function (string, ...params) { |
|  | for(let i=0; i<params.length; i++){ |
|  | let index = string.indexOf("{"+i+"}"); |
|  | while (index != -1) { |
|  | string = string.replace("{"+i+"}", params[i]); |
|  | index = string.indexOf("{"+i+"}"); |
|  | } |
|  | } |
|  | return string; |
|  | } |
|  | })(); |
|  |  |

## 6.\*Sorted List

function manipulateSortedList(){

let processSortedList = (function(){

let sortedList = [];

let add = function(element){

sortedList.push(element);

sortedList.sort((a, b) => a - b);

this.size++;

return sortedList;

};

let remove = function(index){

if(index < 0 || index >= this.size){

//return;

throw new Error('Incorrect index!')

}

sortedList.splice(index, 1);

this.size--;

return sortedList;

};

let get = function(index){

if(index < 0 || index >= this.size){

//return;

throw new Error('Incorrect index!')

}

return sortedList[index];

};

let size = 0;

return {add, remove, get, size};

})();

return processSortedList;

}

|  |
| --- |
| function solve(params) { |
|  | let collection = []; |
|  | return { |
|  | add: function (ele) { |
|  | collection.push(ele); |
|  | this.size++; |
|  | collection = collection.sort((a, b) => a - b); |
|  | }, |
|  | remove: function (ind) { |
|  | collection[ind] = undefined; |
|  | collection = collection.filter(i => i !== undefined).sort((a, b) => a - b); |
|  | this.size = collection.length; |
|  | }, |
|  | get: function (ind) { |
|  | return collection[ind]; |
|  | }, |
|  | size: 0 |
|  | } |
|  | } |
|  |  |
|  |  |
|  | // let col = solve(); |
|  | // console.log(col.size); |
|  | // col.add(10); |
|  | // console.log(col.size); |
|  | // console.log(col.get(0)); |

|  |
| --- |
| function sortedList() { |
|  | let obj = (() => { |
|  | let arr = []; |
|  | let sorting = (a,b) => a-b; |
|  | let add = function (element) { |
|  | arr.push(element); |
|  | arr.sort(sorting); |
|  | this.size++; |
|  | return arr; |
|  | }; |
|  | let remove = function (index) { |
|  | if(index >=0 && index< arr.length) { |
|  | arr.splice(index, 1); |
|  | arr.sort(sorting); |
|  | this.size--; |
|  | return arr; |
|  | } |
|  | }; |
|  | let get = function (index) { |
|  | if(index >= 0 && index< arr.length){ |
|  | return arr[index]; |
|  | } |
|  | }; |
|  |  |
|  | let size = 0; |
|  | return {add, remove, get, size} |
|  | })(); |
|  |  |
|  | return obj; |
|  | } |

## 7.DOM Traversal

function traverseDom(selector){

let $target = $(selector).children();

if($target.length === 0){

$(selector).addClass("highlight");

return;

}

let $next = $target;

while($next.length > 0){

$target = $next;

$next = $next.children();

}

$target.first().addClass("highlight");

$target.first().parentsUntil($(selector).parent()).addClass("highlight");

}

|  |
| --- |
| function domTraversal(selector) { |
|  | let $target = $(selector).children(); |
|  | if($target.length == 0){ |
|  | $(selector).addClass("highlight"); |
|  | return; |
|  | } |
|  | let $next = $target; |
|  |  |
|  | while( $next.length ) { |
|  | $target = $next; |
|  | $next = $next.children(); |
|  | } |
|  |  |
|  | $target.first().addClass("highlight"); |
|  | $target.first().parentsUntil($(selector).parent()).addClass('highlight'); |
|  | } |

function solve(selector) {

let selectedEl = [document.querySelector(selector)];

let counter = 0;

changeClass(getMaxDeepEl(selectedEl), counter);

function changeClass(el, num) {

for (let i = 0; i <= num; i++) {

el.classList.add('highlight');

num--;

return changeClass(el.parentNode, num)

}

}

function getMaxDeepEl(array) {

let hasChildren = array.filter(x => x.childElementCount > 0);

if (hasChildren.length > 0) {

let arrChildren = [].concat(...hasChildren.map(x => Array.from(x.children)));

counter++

return getMaxDeepEl(arrChildren)

} else {

return array[0];

}

}

}

1. **function** highlight(selector) {
2. let element = document.querySelector(selector);
4. (**function** changeClass(element) {
5. **if** (element.hasChildNodes()) {
6. element.className += ' highlight';
7. changeClass(Array.from(element.childNodes).sort((a, b) => b.childNodes.length - a.childNodes.length)[0]);
8. }
9. })(element);
10. }

## 8.\* Bug Tracker

function trackBugs(){

let bugTracker = (function(){

let bugReports = [];

let selector;

let idCounter = 0;

let report = function(author, description, reproducible, severity){

bugReports[idCounter] = {

ID: idCounter,

author,

description,

reproducible,

severity,

status: 'Open'

};

idCounter++;

if(selector){

draw();

}

};

let setStatus = function(id, newStatus){

bugReports[id].status = newStatus;

if(selector){

draw();

}

};

let remove = function(id){

bugReports = bugReports.filter(br => br.ID !== id);

if(selector){

draw();

}

};

let sort = function(method){

switch(method){

case 'author':

bugReports.sort((a, b) => a.author.localeCompare(b.author));

break;

case 'severity':

bugReports.sort((a, b) => a.severity - b.severity);

break;

// case 'ID':

// bugReports.sort((a, b) => a.ID - b.ID);

case 'ID':

default:

bugReports.sort((a, b) => a.ID - b.ID);

break;

}

if(selector){

draw();

}

};

let output = function(inputSelector){

selector = inputSelector;

};

let draw = function(){

$(selector).html('');

for(let bugReport of bugReports){

$(selector).append($('<div>').attr('id', 'report\_' + bugReport.ID).addClass('report').append($('<div>').addClass('body').append($('<p>').text(bugReport.description))).append($('<div>').addClass('title').append($('<span>').addClass('author').text('Submitted by: ' + bugReport.author)).append($('<span>').addClass('status').text(bugReport.status + ' | ' + bugReport.severity))));

}

};

return {report, setStatus, remove, sort, output};

})();

return bugTracker;

}

|  |
| --- |
| function bugTracker() { |
|  | let obj = (() => { |
|  | let container = []; |
|  | let selector = undefined; |
|  | let counter = 0; |
|  | let report = function (author, descrition, reproducible, severity) { |
|  | container[counter] = { |
|  | ID: counter, |
|  | author: author, |
|  | description: descrition, |
|  | reproducible: reproducible, |
|  | severity: severity, |
|  | status: 'Open' |
|  | }; |
|  | counter++; |
|  |  |
|  | if(selector){ |
|  | draw(); |
|  | } |
|  |  |
|  | }; |
|  | let setStatus = function (id, newStatus) { |
|  | container[id].status = newStatus; |
|  | if(selector){ |
|  | draw(); |
|  | } |
|  | }; |
|  | let remove = function (id) { |
|  | container = container.filter(el => el.ID != id); |
|  | if(selector){ |
|  | draw(); |
|  | } |
|  | }; |
|  | let sort = function (method) { |
|  | switch(method){ |
|  | case 'author': |
|  | container = container.sort((a,b) => a.author.localeCompare(b.author)); |
|  | break; |
|  | case 'severity': |
|  | container = container.sort((a,b) => a.severity - b.severity); |
|  | break; |
|  | case 'ID': |
|  | container = container.sort((a,b) => a.ID - b.ID); |
|  | } |
|  | if(selector){ |
|  | draw(); |
|  | } |
|  | }; |
|  | let output = function (sel) { |
|  | selector = sel; |
|  | }; |
|  |  |
|  | let draw = function () { |
|  | $(selector).html(""); |
|  | for(let bug of container){ |
|  | $(selector).append($('<div>').attr('id', "report\_" + bug.ID).addClass('report').append($('<div>').addClass('body').append($('<p>').text(bug.description))).append($('<div>').addClass('title').append($('<span>').addClass('author').text('Submitted by: ' + bug.author)).append($('<span>').addClass('status').text(bug.status + " | " + bug.severity)))); |
|  | } |
|  | }; |
|  |  |
|  | return {report, setStatus, remove, sort, output}; |
|  | })(); |
|  |  |
|  | return obj; |
|  | } |