**PSG COLLEGE OF TECHNOLOGY**

**Department of Applied Mathematics and Computational Sciences**

**Problem sheet on Java Script**

1. Write a JavaScript program to sort a list of elements using Comb sort.   
   The Comb Sort is a variant of the Bubble Sort. Like the Shell sort, the Comb Sort increases the gap used in comparisons and exchanges. Some implementations use the insertion sort once the gap is less than a certain amount. The basic idea is to eliminate turtles, or small values near the end of the list, since in a bubble sort these slow the sorting down tremendously. Rabbits, large values around the beginning of the list, do not pose a problem in bubble sort.

function comb(arr) {

for(let i = arr.length - 1; i > 0; i--) {

for(let j = 0, k = i; k < arr.length; j++, k++) {

if(arr[j] > arr[k]) {

let temp = arr[j];

arr[j] = arr[k];

arr[k] = temp;

}

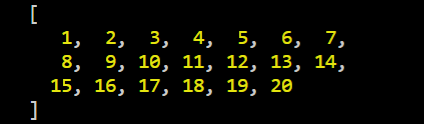
}

}

return arr;

}

console.log(comb([5, 4, 6, 3, 7, 2, 8, 1, 9, 10, 11, 19, 20, 12, 18, 13, 17, 14, 16, 15,]));

  
In bubble sort, when any two elements are compared, they always have a gap of 1. The basic idea of comb sort is that the gap can be much more than 1.

1. Write a JavaScript program to test the first character of a string is uppercase or not.

function firstIsUppercase(str) {

if (typeof str !== 'string' || str.length === 0) {

return false;

}

if (str[0].toUpperCase() === str[0]) {

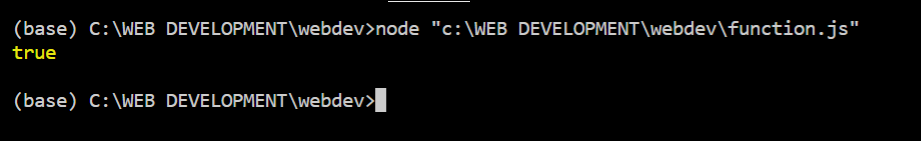
return true;

}

return false;

}

console.log(firstIsUppercase('Hello'));



1. Write a pattern that matches e-mail addresses. The personal information part contains the following ASCII characters.

* Uppercase (A-Z) and lowercase (a-z) English letters.
* Digits (0-9).
* Characters ! # $ % & ' \* + - / = ? ^ \_ ` { | } ~
* Character . ( period, dot or fullstop) provided that it is not the first or last character and it will not come one after the other.

1. Use HTML Geolocation API to locate the user's position.

<html>

<body>

<h1>Javascript Geolocation API</h1>

<button onclick="getloc()">Click the button to get the coordinates</button>

<p id="demo"></p>

<script>

function getloc()

{

if(navigator.geolocation)

{

navigator.geolocation.getCurrentPosition(showpos);

}

else

{

document.getElementById("demo").innerHTML="Geolocation not supported";

}

}

function showpos(position)

{

document.getElementById("demo").innerHTML="Latitude " + position.coords.latitude + " Longitude : " + position.coords.longitude;

}

</script>

</body>

</html>



1. A web worker is a JavaScript running in the background, without affecting the performance of the page. Write a code to create web worker object and try to terminate and reuse the worker object.

<!DOCTYPE html>

<html>

<body>

<p>Count numbers: <output id="result"></output></p>

<button onclick="startWorker()">Start Worker</button>

<button onclick="stopWorker()">Stop Worker</button>

<script>

var w;

function startWorker() {

if(typeof(Worker) !== "undefined") {

if(typeof(w) == "undefined") {

w = new Worker("demo\_workers.js");

}

w.onmessage = function(event) {

document.getElementById("result").innerHTML = event.data;

};

} else {

document.getElementById("result").innerHTML = "Sorry, your browser does not support Web Workers...";

}

}

function stopWorker() {

w.terminate();

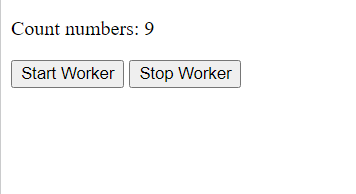
w = undefined;

}

</script>

</body>

</html>



1. In HTML, any element can be dragged and dropped. Write a code to drag an element and drop an element.

<html>

<head>Drag and drop API</head>

<style>

#div1{

width:150px;

height:150px;

padding:10px;

border:1px solid #aaaaaa;

</style>

<script>

function dropallow(event)

{

event.preventDefault();

}

function drag(event)

{

event.dataTransfer.setData("text",event.target.id);

}

function drop(event)

{

event.preventDefault();

var data=event.dataTransfer.getData("text");

event.target.appendChild(document.getElementById(data));

}

</script>

</head>

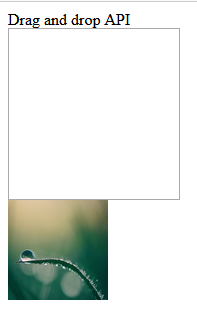
<body>

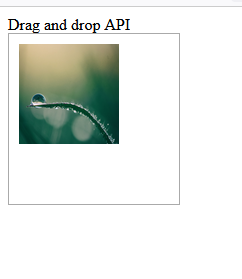
<div id="div1" ondrop="drop(event)" ondragover="dropallow(event)"></div>

<img id="drag1" src="https://images.unsplash.com/photo-1471879832106-c7ab9e0cee23?ixlib=rb-1.2.1&ixid=MnwxMjA3fDB8MHxleHBsb3JlLWZlZWR8Mnx8fGVufDB8fHx8&w=1000&q=80" draggable="true" ondragstart="drag(event)" width="100px" height="100px">

</body>

</html>





1. With web storage, web applications can store data locally within the user's browser. Write a code to access the local storage and session storage.

<!DOCTYPE html>

<html>

<head>

<script>

function clickCounter() {

if (typeof(Storage) !== "undefined") {

if (localStorage.clickcount) {

localStorage.clickcount = Number(localStorage.clickcount)+1;

} else {

localStorage.clickcount = 1;

}

document.getElementById("result").innerHTML = "You have clicked the button " + localStorage.clickcount + " time(s).";

} else {

document.getElementById("result").innerHTML = "Sorry, your browser does not support web storage...";

}

}

</script>

</head>

<body>

<p>Local Storage </p>

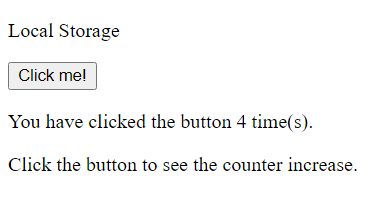
<p><button onclick="clickCounter()" type="button">Click me!</button></p>

<div id="result"></div>

<p>Click the button to see the counter increase.</p>

</body>

</html>



Session Storage

<!DOCTYPE html>

<html>

<head>

<script>

function clickCounter() {

if (typeof(Storage) !== "undefined") {

if (sessionStorage.clickcount) {

sessionStorage.clickcount = Number(sessionStorage.clickcount)+1;

} else {

sessionStorage.clickcount = 1;

}

document.getElementById("result").innerHTML = "You have clicked the button " + sessionStorage.clickcount + " time(s) in this session.";

} else {

document.getElementById("result").innerHTML = "Sorry, your browser does not support web storage...";

}

}

</script>

</head>

<body>

<p>Session storage</p>

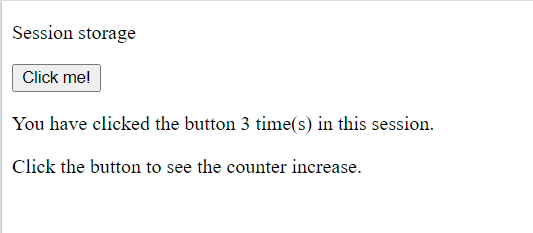
<p><button onclick="clickCounter()" type="button">Click me!</button></p>

<div id="result"></div>

<p>Click the button to see the counter increase.</p>

</body>

</html>



1. Hangman is one of our favourite games, and children and adults love it alike. You will be amazed to know that hangman can be developed in a jiffy using JavaScript, HTML, and CSS. Note that the main functionality is defined using JS. HTML is for display, and CSS does the job of beautifying the contents. Create a hangman game using the above said terms.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<title>Hangman Game </title>

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />

<style>

\* {

box-sizing: border-box;

}

body {

padding: 16px;

font-family: 'Arial Black', sans-serif;

font-size: 100%;

}

header {

text-align: center;

}

h1 {

display: inline-block;

position: relative;

margin: 0 0 2rem 0;

text-align: center;

text-transform: uppercase;

letter-spacing: 2px;

cursor: default;

user-select: none;

transition: all 180ms ease;

}

h1:hover {

color: rgb(248, 165, 57);

letter-spacing: 4px;

transform: scale(1.1) rotate(-4deg);

}

h1:before {

height: 4px;

content: ' ';

display: block;

position: absolute;

bottom: 0;

left: 100%;

right: 0;

transition: all 180ms ease;

}

h1:hover:before {

left: 0;

background: rgb(25, 215, 35);

}

footer {

margin: 5rem 0 3rem;

font-family: Arial, Arial, Helvetica, sans-serif;

color: gray;

line-height: 1.5;

text-align: center;

}

footer a {

color: rgb(0, 62, 230);

text-decoration: none;

}

footer a:hover {

text-decoration: underline;

}

</style>

<style>

.chars {

display: flex;

flex-wrap: wrap;

align-items: flex-end;

justify-content: center;

padding: 3rem 1rem;

}

.chars\_\_char {

width: 2.6rem;

height: 3rem;

flex-shrink: 0;

display: block;

margin: 0 0.5rem 0.5rem 0;

padding: 0.25rem;

border: 0 solid rgb(219, 0, 230);

border-bottom-width: 3px;

font-size: 1.5rem;

text-align: center;

}

.chars\_\_char--is-letter {

border-width: 2px 2px 4px 2px;

}

</style>

<style>

.keyboard {

width: 100%;

max-width: 30rem;

display: flex;

flex-wrap: wrap;

align-items: flex-end;

justify-content: center;

margin: 0 auto;

padding: 1rem;

}

.keyboard\_\_key {

width: 3rem;

margin: 0 0.5rem 0.5rem 0;

padding: 0.5rem 0;

border-radius: 3px;

outline: none;

border: none;

background-color: rgba(0, 62, 230, 0.07);

color: rgb(230, 119, 0);

font: inherit;

font-size: 1.3rem;

text-align: center;

cursor: pointer;

}

.keyboard\_\_key--found {

background-color: rgba(16, 16, 1, 0);

color: rgb(102, 226, 213);

cursor: default;

}

.keyboard\_\_key--is-faulty {

background-color: rgba(254, 0, 0, 0.07);

color: rgb(254, 0, 0);

cursor: default;

}

</style>

<style>

.drawing {

max-width: 80rem;

margin: 0 auto;

border: 3px solid rgb(0, 62, 230);

box-shadow: 5px 5px 0 1px rgb(0, 47, 179);

padding: 1rem;

text-align: center;

}

.drawing\_\_container {

width: 20rem;

max-width: 100%;

height: 18rem;

display: inline-block;

position: relative;

border-bottom: 3px solid black;

}

.drawing\_\_message {

margin-top: 1.5rem;

font-size: 1.3rem;

text-align: center;

}

.drawing\_\_part {

position: absolute;

}

.drawing\_\_part-1 {

width: 3px;

top: 0;

bottom: 0;

left: 0;

background-color: black;

}

.drawing\_\_part-2 {

height: 3px;

top: 0;

right: 50%;

left: 0;

background-color: black;

}

.drawing\_\_part-3 {

width: 3px;

height: 15%;

top: 0;

left: 50%;

background-color: black;

}

.drawing\_\_part-4 {

width: 2.5rem;

height: 2.5rem;

top: 15%;

left: calc(50% - 1.25rem + 1.5px);

border-radius: 100%;

border: 3px solid rgb(16, 58, 3);

}

.drawing\_\_part-5 {

width: 3px;

height: 30%;

top: calc(15% + 2.5rem);

left: 50%;

background-color: purple;

}

.drawing\_\_part-6a,

.drawing\_\_part-6b {

width: 10%;

height: 3px;

top: calc(15% + 2.5rem + 1.5rem);

background-color: yellow;

}

.drawing\_\_part-6a {

left: calc(50% - 10% + 1.5px);

transform-origin: right center;

transform: rotate(20deg);

}

.drawing\_\_part-6b {

left: calc(50% + 1.5px);

transform-origin: left center;

transform: rotate(-20deg);

}

.drawing\_\_part-7a,

.drawing\_\_part-7b {

width: 25%;

height: 3px;

top: calc(45% + 2.5rem - 3px);

background-color: black;

}

.drawing\_\_part-7a {

left: calc(50% - 25% + 1.5px);

transform-origin: right center;

transform: rotate(-55deg);

}

.drawing\_\_part-7b {

left: calc(50% + 1.5px);

transform-origin: left center;

transform: rotate(55deg);

}

</style>

<style>

.modal {

position: fixed;

top: 0;

right: 0;

bottom: 0;

left: 0;

display: flex;

align-items: center;

justify-content: center;

}

.modal\_\_background {

position: fixed;

top: 0;

right: 0;

bottom: 0;

left: 0;

background-color: rgba(0, 0, 0, 0.6);

}

.modal\_\_main {

width: 24rem;

z-index: 1;

box-shadow: 5px 5px 0 1px rgb(0, 47, 179);

border: 3px solid rgb(0, 62, 230);

padding: 2rem;

background-color: white;

}

.modal\_\_content {

font-size: 1.2rem;

line-height: 2rem;

text-align: center;

}

.modal\_\_content em {

color: rgb(0, 62, 230);

}

.modal\_\_actions {

margin-top: 2rem;

text-align: center;

}

.modal\_\_actions button {

margin: 0;

padding: 0.5rem 1rem;

outline: none;

border: none;

background-color: rgb(0, 62, 230);

color: white;

font: inherit;

text-transform: uppercase;

letter-spacing: 1px;

cursor: pointer;

}

</style>

</head>

<body>

<header>

<h1>Hangman Game</h1>

</header>

<div id="hangman">

<div class="drawing" id="drawing">

<div class="drawing\_\_container"></div>

<div class="drawing\_\_message"></div>

</div>

<div id="result"></div>

</div>

<div id="board">

<div id="chars" class="chars"></div>

<div id="keyboard" class="keyboard"></div>

</div>

<div id="modal" class="modal" style="display: none">

<div class="modal\_\_background"></div>

<div class="modal\_\_main">

<div class="modal\_\_content"></div>

<div class="modal\_\_actions">

<button onclick="reinitGame()">Play New Game</button>

</div>

</div>

</div>

<footer>

</footer>

<script>

let game = null;

const MAX\_FAULTS = 9;

const wordList = [

'communication',

'communicators',

'freedom',

'jamstack',

'version control',

'open source',

'get technical',

'have a great day',

'encyclopedia'

];

function reinitGame() {

game = initNewGame();

}

function hideModal() {

const modal = document.getElementById('modal');

modal.style.display = 'none';

}

function showModal(content) {

const modal = document.getElementById('modal');

const modalContent = modal.querySelector('.modal\_\_content');

modalContent.innerHTML = content;

setTimeout(() => (modal.style.display = ''), 300);

}

function gameEndHandler() {

const content = game.hasWon()

? `Great, you are a winner!`

: `Oh no! We were looking for <em>${game.getWord()}</em>.`;

showModal(content);

}

function initNewGame() {

const hangman = new Hangman(getRandomWord(), gameEndHandler);

hideModal();

drawGame(hangman);

return hangman;

}

function getRandomWord() {

const index = Math.floor(Math.random() \* wordList.length);

return wordList[index];

}

function guessLetter(letter) {

if (game && !game.isFinished()) {

game.pickedLetter(letter);

drawGame(game);

}

}

function drawHangman(faults) {

const parts = [

'<div class="drawing\_\_part drawing\_\_part-1"></div>',

'<div class="drawing\_\_part drawing\_\_part-2"></div>',

'<div class="drawing\_\_part drawing\_\_part-3"></div>',

'<div class="drawing\_\_part drawing\_\_part-4"></div>',

'<div class="drawing\_\_part drawing\_\_part-5"></div>',

`<div class="drawing\_\_part drawing\_\_part-6a"></div>`,

`<div class="drawing\_\_part drawing\_\_part-6b"></div>`,

`<div class="drawing\_\_part drawing\_\_part-7a"></div>`,

`<div class="drawing\_\_part drawing\_\_part-7b"></div>`,

];

const visibleParts = parts.splice(0, faults);

const parent = document.querySelector('#drawing .drawing\_\_container');

parent.innerHTML = visibleParts.join('');

}

function drawResult(faults) {

const texts = [

*/\* 0 \*/* "Guess the word",

*/\* 1 \*/* "Sorry, it's not right.",

*/\* 2 \*/* "Try again.",

*/\* 3 \*/* "you have 6 more tries.",

*/\* 4 \*/* "you have 5 more tries.",

*/\* 5 \*/* "you have 4 more tries.",

*/\* 6 \*/* "you have 3 more tries!",

*/\* 7 \*/* "Think carefully.",

*/\* 8 \*/* "Make a good guess now!",

*/\* 9 \*/* "Better luck next time! ;-)",

];

const container = document.querySelector('#drawing .drawing\_\_message');

container.innerText = texts[faults];

}

function drawKeyboard(foundLetters, faultyLetters) {

const letters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'.split('');

const keys = letters.map(letter => {

const found = foundLetters.includes(letter);

const faulty = faultyLetters.includes(letter);

return { letter, found, faulty };

});

const parent = document.getElementById('keyboard');

parent.innerHTML = '';

keys.map(key => {

const el = document.createElement('button');

el.innerText = key.letter;

el.disabled = key.found || key.faulty;

el.className = 'keyboard\_\_key';

key.found && el.classList.add('keyboard\_\_key--found');

key.faulty && el.classList.add('keyboard\_\_key--is-faulty');

el.onclick = () => {

guessLetter(key.letter);

};

parent.appendChild(el);

});

}

function drawCharList(chars) {

const parent = document.getElementById('chars');

parent.innerHTML = '';

chars.forEach(char => {

const charEl = document.createElement('div');

charEl.className = `chars\_\_char ${char.isLetter ? 'chars\_\_char--is-letter' : ''}`;

charEl.innerText = char.show ? char.value : '';

parent.appendChild(charEl);

});

}

function drawGame(hangman) {

drawResult(hangman.getNumberOfFaults());

drawHangman(hangman.getNumberOfFaults());

drawCharList(hangman.getCharList());

drawKeyboard(hangman.getFoundLetters(), hangman.getFaultyLetters());

}

function listenForInputs(callback) {

window.addEventListener('keydown', event => {

const pressedOtherKey = event.altKey || event.ctrlKey || event.metaKey; *// Shift key is allowed*

const key = event.key.toUpperCase();

const isLetter = key.match(/[A-Z]/);

!pressedOtherKey && isLetter && callback(key);

});

}

class Hangman {

constructor(word, onEndGame) {

*this*.word = word.toUpperCase();

*this*.pickedLetters = [];

*this*.faults = 0;

*this*.onEndGame = onEndGame;

}

getWord() {

return *this*.word;

}

getCharsOfWord() {

return *this*.word.split('');

}

getLettersOfWord() {

const chars = *this*.getCharsOfWord();

return chars

.filter(char => char.match(/[A-Z]/))

.filter((char, index, list) => list.indexOf(char) === index);

}

getNumberOfFaults() {

return *this*.faults;

}

getFaultyLetters() {

return *this*.pickedLetters.filter(letter => {

return !*this*.word.includes(letter);

});

}

getFoundLetters() {

return *this*.pickedLetters.filter(letter => {

return *this*.word.includes(letter);

});

}

getCharList() {

const chars = *this*.getCharsOfWord();

return chars.map(char => {

const isLetter = char.match(/[A-z]/);

const pickedLetterAlready = *this*.pickedLetters.includes(char);

const show = pickedLetterAlready || !isLetter;

return { isLetter, show, value: char };

});

}

pickedLetter(letter) {

const alreadyPicked = *this*.pickedLetters.includes(letter);

const contains = *this*.word.includes(letter);

if (alreadyPicked) {

console.log('Letter already chosen');

return;

}

*this*.pickedLetters.push(letter);

if (contains) {

console.log('Yeah, you are great!');

} else {

console.log('Try again');

*this*.pickedLetters.push(letter);

*this*.faults += 1;

}

if (*this*.isFinished()) {

*this*.onEndGame();

}

}

hasWon() {

const letters = *this*.getLettersOfWord();

return (

!*this*.hasLost() &&

letters.every(char => {

return *this*.pickedLetters.includes(char);

})

);

}

hasLost() {

return *this*.faults >= MAX\_FAULTS;

}

isFinished() {

return *this*.hasLost() || *this*.hasWon();

}

}

game = initNewGame();

listenForInputs(letter => {

guessLetter(letter);

});

</script>

</body>

</html>

