|  |  |
| --- | --- |
| Examiner: | T. Mkwaira |
| Moderator | S. Zengeni |

Web Programming 27(8)1 Examination

* This assessment must be submitted on AssessmentQ.
* This test includes a practical section. See the attachments.
* No instructions or directives of the invigilator shall be disregarded; if any of the instructions are disobeyed, candidates shall expose themselves to disqualification from future assessments.
* Observe the Honor Code[[1]](#footnote-1).
* Each candidate is expected to maintain academic integrity. Any form of plagiarism or cheating that is detected can lead to disciplinary measures being taken.
* The allocated time includes time for uploading/ saving your solution.
* Should any work need to be uploaded, ensure naming conventions are followed as instructed.
* Unless otherwise specified, the required values in your programs must be provided literally in code.
* Organise your code well. If you are using an HTML file that links to a JS script, please create a separate project for each of the tasks; If you are using pure JS files, please create a single project with different files for each of the tasks.
* Any work that is uploaded must be in zip format ONLY. Other file types may give problems.
* At the end of this assessment, you should only upload a single zip file. See the end of this document.
* Penalties may be applied for late work, or work that is not packed and/or submitted correctly.
* No explanation of the assessment tasks shall be given.
* Keep your webcam on at all times.
* Any technical issue related to the examination platform being used needs to be reported to the invigilator immediately, and evidence of errors must be emailed to support@belgiumcampus.ac.za and venter.f@belgiumcampus.ac.za with a *cc* to your lecturer.
* Do not use MS Teams for submissions for whatever reason. In case of technical difficulties, you should submit your test to your lecturer with a proper subject line that follows the naming convention *WPR27(8)1 Examination: StudentSurname StudentName.*
* *It is your responsibility to ensure that you have saved your test correctly!*
* Marks are reserved for best practices including commenting the important parts of your code etc.
* This mark allocation is subject to change.

Section A: Theory [This section will be created on AQ]

|  |  |  |  |
| --- | --- | --- | --- |
| **Task 1: True or False** | | **Subtotal:** | **[10]** |
| **Select the correct option by selecting whether the given statements are true or false, and mark your answers on the special answer sheet provided.** | | | |
| **Negative marking applies to this section.** | | | |
|  | | | |
|  | JavaScript was primarily created for use in the web browser. | | |
|  | JavaScript is strongly typed. | | |
|  | It is possible to convert between data types in JavaScript. | | |
|  | A function cannot be passed as an argument to another function. | | |
|  | Arrays can only contain items of the same data type. | | |
|  | JavaScript can be used to manipulate properties of elements on a webpage. | | |
|  | JSON is a notation that is used to structure data. | | |
|  | Events can be handled using JavaScript. | | |
|  | *innerHTML* can is used to change the value of a textbox. | | |
|  | The toString method gives a string representation of an object | | |

**Section B: Practical (Fundamentals) [15 Marks]**

* IMPORTANT. If the task states that you should use literal values, it means you must write the values in code without having to receive them from the user or generating them.
* Create a single project folder with a file for each of the tasks. Each of the files should be named according to the task number, i.e., *a.js, b.js* etc.
* Do not use inbuilt functions, except for part d.

|  |  |  |
| --- | --- | --- |
| Task 2: | Subtotal: | [15] |
| 1. Write a simple JavaScript function called *Populate*(). The function should use a looping structure to populate an array with all values from 10 to 50. Pass an array as your argument to the function. | | [3] |
| 1. Create another function that doubles every value in an array of three numbers. Print out all the doubled values. You should create a literal array that contains any three numbers of your choice. | | [3] |
| 1. Write a JavaScript program that will filter *marks* that are greater than 49 from an array of marks. Find a way to return a new array with these marks. You must **not** consider taking input from the console or from a form. Rather, you should create your array literally in code. Log the result to the console. | | [5] |
| 1. Write a JavaScript function to display the current day and time in the following format. Create a second function to find the difference this date and today. Call the function **dateDifference(dateA, dateB)**  **Sample Output**: Today is Tuesday 17 September 2019. The current time is 08:30:15. | | [4] |

**Section C: Practical [30 Marks]**

* Create a separate project for each of the tasks.
* You may use any inbuilt functions.

|  |  |  |
| --- | --- | --- |
| **Task 3:** | **Subtotal:** | **[6]** |
| MapReduce is a processing technique and a program model for distributed computing. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs).  Given the data below, use these two functions to solve the problem that follows this description. Please note that the addenda contains the code displayed below. | | |
| *Data*:  let debtors = [    { name: "John Doe", amount: 1200000 },    { name: "Jack Sparrow", amount: 300000 },    { name: "Judy Graham", amount: 7784 },    { name: "Ted Talks", amount: 44212 },    { name: "Ben Dover", amount: 600 },  ]; | | |
| Problem: Calculate the average amount owed by the debtors. | | |

|  |  |  |
| --- | --- | --- |
| **Task 4:** | **Subtotal:** | **[12]** |
| Create a JavaScript function that gives the output shown below. You may display the output in the console or webpage. Do not use arrays for this task. | | |
|  | |  |
| Table  Description automatically generated | | |

|  |  |  |
| --- | --- | --- |
| **Task 5:** | **Subtotal:** | **[12]** |
| See the form below: | | |
|  | | |
| Complete the application so that you can change the background colour and displayed image when the button is clicked. You have already been provided with the illustrated skeleton application, images, and a partial function. Do not assume that there are no errors in the code. | | |

HOW TO SUBMIT: ZIP ONLY! ZIP ONLY!

# NAMING CONVENTION

You must submit a ZIP file that uses the Campus standard naming convention. Any assessment that a student creates for electronic submission must conform to the following naming convention:

*WPR27(8)1 Examination: StudentSurname StudentName.zip*

The reason for this is to ensure that your assessment is received correctly by your lecturer; this prevents any inconvenience for you as the student and the lecturer.

# PACKAGING YOUR SOLUTION

1. Create a folder that follows the naming convention explained above.
2. Move the Tasks folder into the named folder you created in Step 1.
3. Finally, create a zip archive of the folder you created in Step 2.
4. Upload your zip archive.

1. The Honor Code is a statement addressing issues such as cheating, stealing, and misrepresentation, made by a school or other institution in which its participants pledge to adhere to. Honor Codes are self-regulating because under an honor code, students are required to turn in other students in violation of the code.

   [<https://cs.stanford.edu/people/eroberts/cs181/projects/honor-code/honorcodes.html>](https://cs.stanford.edu/people/eroberts/cs181/projects/honor-code/honorcodes.html) [↑](#footnote-ref-1)