**Student ID:** 985848\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Full Names:** Van Quy Tran\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Web Application Programming

(CS472)

(September 2017)

Instructor: Obinna A. Kalu

W2D6 – Exam 2

1. The exam duration is 2 hours.
2. The exam is a computer-based exam.
3. You are expected to use a CS lab or your own computer to answer both the Coding questions and the theory/non-coding/knowledge-based questions. You may use the Internet and/or the lecture slides for reference purposes to lookup APIs or code syntax.
4. Make sure to switch-off your cell-phones or simply turn the ringer off.
5. You may use blank sheet(s) of paper for your scratch work, if needed.
6. Exams are copyrighted materials and must not be copied, reproduced or redistributed.
7. All answers to the theory/non-coding/knowledge-based questions should be typed-in, on this document, following the questions.
8. All answers to the Coding questions may be typed-in as source code files, using a Code Editor or IDE. But be sure to copy your finished code for each coding question from your Code Editor and paste it to this document as your answer.
9. Finally, compress/zip your entire code folder into one zip file and upload/submit it to Sakai, along with your typed/pasted answers in this document.

--------------------------------------------------------------------------------------------------------------------

(CS472 - WAP)

(September 2017)

W2D6 – Examination 2

**Part I – Science of Consciousness (SCI):** (5 points)

Given in the table below, are principles from the Science of Consciousness (SCI), related topics in Web Application Programming (WAP) and Connection sentence(s). For each section, fill-in the blank cell(s), by either providing the missing principle from the Science of Consciousness or a Web Programming topic or a connection sentence(s).

**Note**: To get the full credit, make sure your connection sentence(s) provide(s) a reasonable, clear link/relation between the stated SCI principle and the WAP topic.

|  |  |  |
| --- | --- | --- |
| **SCI principle** | **Topic from Web Application Programming (WAP)** | **Connection sentence(s)** |
| Unity in Diversity | Inheritance in JavaScript | Life is structured in layers and when we practice the TM program we experience all the layers of thought from gross thinking to finest feeling. |
| Seek the highest first | Hoisting | Mastery of fundamental web programming concepts gives us an edge. Likewise, when we seek the highest first we achieve more out of life. Through regular practice of Transcendental Meditation we gain maximum advantage. |
| Do Less, Accomplish More | jQuery | jQuery is a tool that allows us to be more efficient in writing code for the Browser, similar to how TM is a tool that allows us to be more efficient in life |

**Part II – Theory (Short answers, True/False, Multiple-choice questions):** (35 points)

1. (8 points) Answer the following questions with True or False. For each answer, give a rationale (i.e. If True state how, if False state why. No rationale, earns you just half of the points if your True/False answer is correct, and zero point if your True/False answer is incorrect).
   1. (2 points) In JavaScript, when a function is defined inside another function, the outer function has access to the inner function’s variables.

False.

The scope of variable is in its function so the scope of variable in the inner function is in the inner function so we can access the inner variable from outer function.

Only the inner function can access the variable of outer function’s variable.

* 1. (2 points) By using the jQuery selector, $(“<div>”), we can select all the div tags present on our webpage.

False.

We should use $(“div”) to select all the div tags present on our webpage.

* 1. (2 points) When using the Web Storage API, the localStorage containing data stored by the website, <http://www.mum.edu/> CANNOT be accessed by your webpage at <http://mumstudents.org/>.

True.

Because the localStorage is storage specifically for a domain. We can not access localStorage by cross domain.

* 1. (2 points) In Javascript, when a variable is declared, (e.g – var dateOfBirth; ) and no value is assigned to it, the default value it gets is, null.

False.

All variables are initially set to *undefined* (Hoisting).

1. (20 points) Give short answers (and examples where required) to the following questions.
   1. (2 points) Does JavaScript support function overloading? Explain your answer + give an example using code snippet.

JavaScript doesn’t support function overloading. If it sees/parses two or more functions with the same name it’ll just consider the last defined function and overwrite the previous ones.

**function** foo() {  
 console.log('No argument');  
}  
  
**function** foo(x) {  
 console.log('1 argument: ' + x);  
}  
  
**function** foo(x, y) {  
 console.log('2 argument: ' + x + ' & ' + y);  
}  
  
  
foo();  
foo(1);  
foo(1,2);

* 1. (4 points) Explain the difference between Obstrusive versus Unobstrusive Javascript. Give an example of each, to illustrate your answer.

Unobtrusive JavaScript is a general approach to the use of JavaScript in web pages. Though the term is not formally defined, its basic principles are generally understood to include:

* Separation of functionality (the "behavior layer") from a Web page's structure/content and presentation
* Best practices to avoid the problems of traditional JavaScript programming (such as browser inconsistencies and lack of scalability)
* Progressive enhancement to support user agents that may not support advanced JavaScript functionality

Obstrusive Javascript is the way we write Javascript that HTML is cluttered with JS code

Example:

Unobtrusive:

<button id="clickme">Click Me</button>  
  
**var** clickMeBtn = document.getElementById("clickme");  
clickMeBtn.onclick = sayHi;  
  
**function** clickMe() {  
 console.log("Clicked");  
}

Obstrusive:

<button onclick="clickMe();">Click Me</button>  
  
**function** clickMe() {  
 console.log("Clicked");  
}

* 1. (6 points) In terms of Javascript programmming:
     1. What is a Closure? Give an example in code snippet.

A closure is an inner function that has access to the outer (enclosing) function’s variables—scope chain. The closure has three scope chains: it has access to its own scope (variables defined between its curly brackets), it has access to the outer function’s variables, and it has access to the global variables.

**function** foo() {  
 **var** name = 'MUM';  
 **function** displayName() {  
 console.log(name);  
 }  
 **return** displayName;  
}  
  
**var** myFunc = foo();  
myFunc();

* + 1. What is an IIFE? Give an example in code snippet.

Declares and immediately calls an anonymous function. Immediately-invoked function expressions can be used to avoid variable hoisting from within blocks, protect against polluting the global environment and simultaneously allow public access to methods while retaining privacy for variables defined within the function.

(**function** () {  
 console.log('IIFE example');  
})();

* + 1. What is the Module Pattern? Give an example in code snippet.

The Module pattern is used to mimic the concept of classes (since JavaScript doesn’t natively support classes - Before ES6) so that we can store both public and private methods and variables inside a single object — similar to how classes are used in other programming languages like Java or Python.

**var** count = (**function** () {  
 **var** counter = 0;  
  
 **return** {  
 add: () => ++counter,  
 reset: () => { counter = 0; },  
 get: () => counter  
 }  
})();

* 1. (2 points) In Web Application programming, what do we mean by **AJAX** (You may give an example to illustrate your answer).

Ajax is not a programming language or a tool, but a concept. Ajax is a client-side script that communicates to and from a server/database without the need for a postback or a complete page refresh. The best definition is “the method of exchanging data with a server, and updating parts of a web page – without reloading the entire page.”

Example with AJAX using jQuery:

$.ajax({  
 url: './studentdata.json',  
 type: 'GET',  
 dataType: 'JSON'  
}).done((result) => {  
 **for** (**const** student **of** result.students) {  
 **let** li = $('<li></li>').text(student.studentID + ' - ' + student.firstName);  
 li.addClass('list-group-item');  
 $('#student-list').append(li);  
 }  
}).fail(**function**(xhr, status, err) {  
 console.log(status);  
}).always(**function**(xhr, status) {  
 console.log(status);  
});

* 1. (4 points) With reference to Inheritance in JavaScript, what is the difference & the relationship between the .\_\_proto\_\_ property and the .prototype property? (You may give an example to illustrate your answer).

Difference:

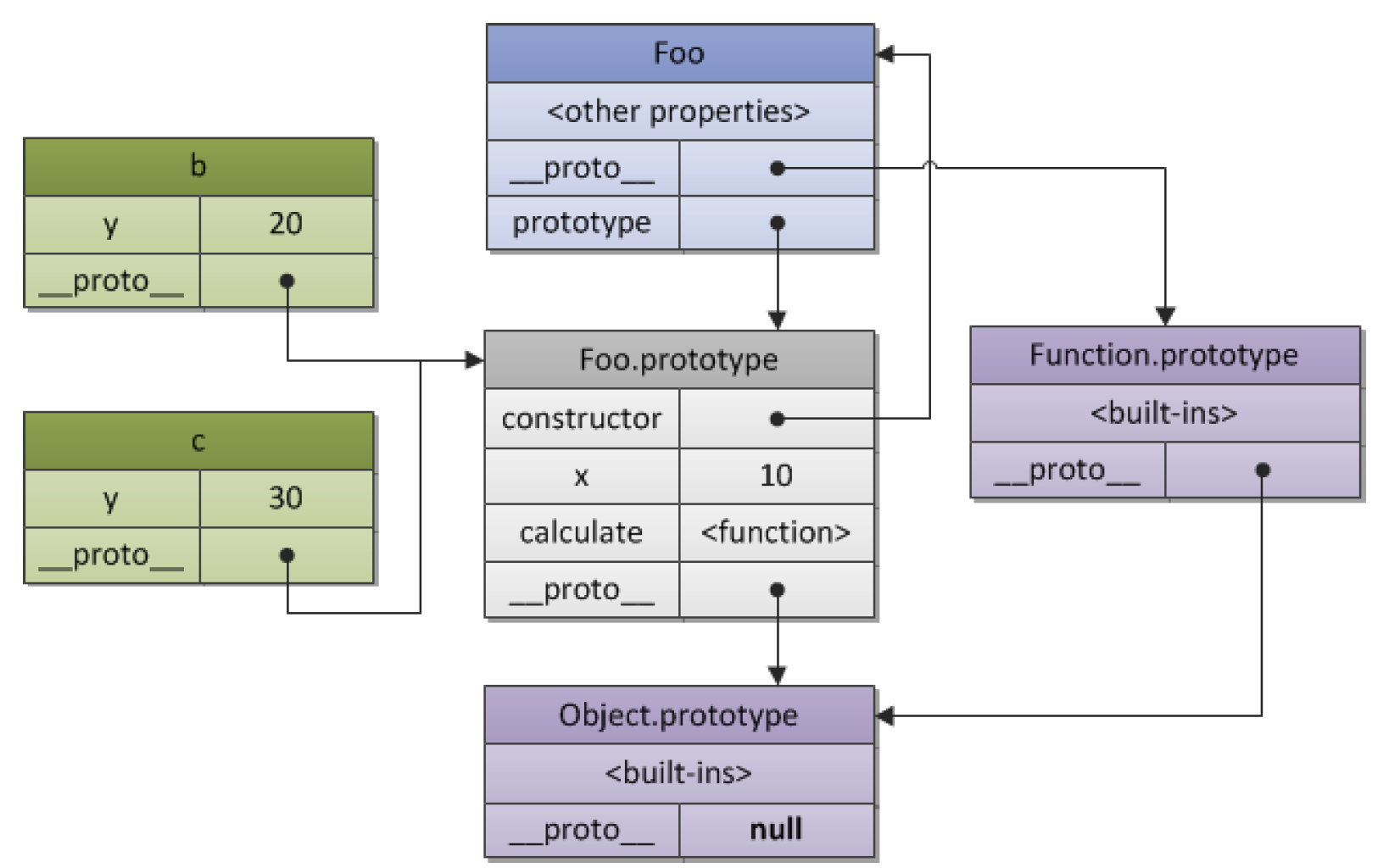
* \_\_proto\_\_ is the actual object that is used in the lookup chain to resolve methods, etc.
* prototype is the object that is used to build \_\_proto\_\_ when you create an object with *new*

Relationship:

When using new the \_\_proto\_\_ of newly created object is set to the prototype to the function constructor.

We can refer to below code snippet & picture to understand more about prototype and \_\_proto\_\_

**function** Foo(y) {  
 **this**.y = y;  
 **this**.name = 101;  
}  
  
Foo.prototype.x = 10;  
Foo.prototype.calculate = **function** (z) {  
 **return this**.x + **this**.y + z;  
}  
  
**let** b = **new** Foo(20);  
**let** c = **new** Foo(30);



* 1. (2 points) With regards to JavaScript’s support for Functional programming, what is a First class function? Give an example.

A language with first-class functions means that it treats functions like expressions of any other type. Functions are like any other object.

Functions as parameters:

**function** runFunction(fn, data) {  
 **return** fn(data);  
}

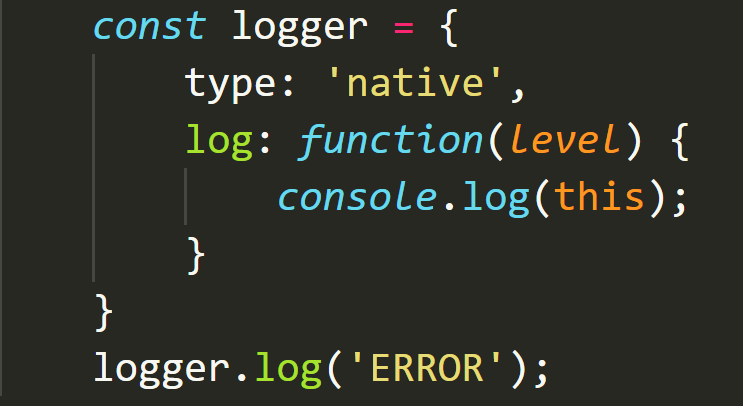
Assign a function as a value to a variable:

**var** myFunc = **function**() {  
 // do something  
};

Return a function:

**var** variable = **function**() {  
 **var** innerFunc = **function** () {  
 // Do something  
 }  
  
 **return** innerFunc();  
}

1. (7 points) The following questions involve multiple choices; choose the correct option by putting a green highlight over, either Option A or Option B or Option C.
   1. (2 points) Consider the code snippet given below:



When the statement, *logger.log(‘ERROR’)* is executed, what object will the ‘this’ reference point to?

**Option A**.

window object

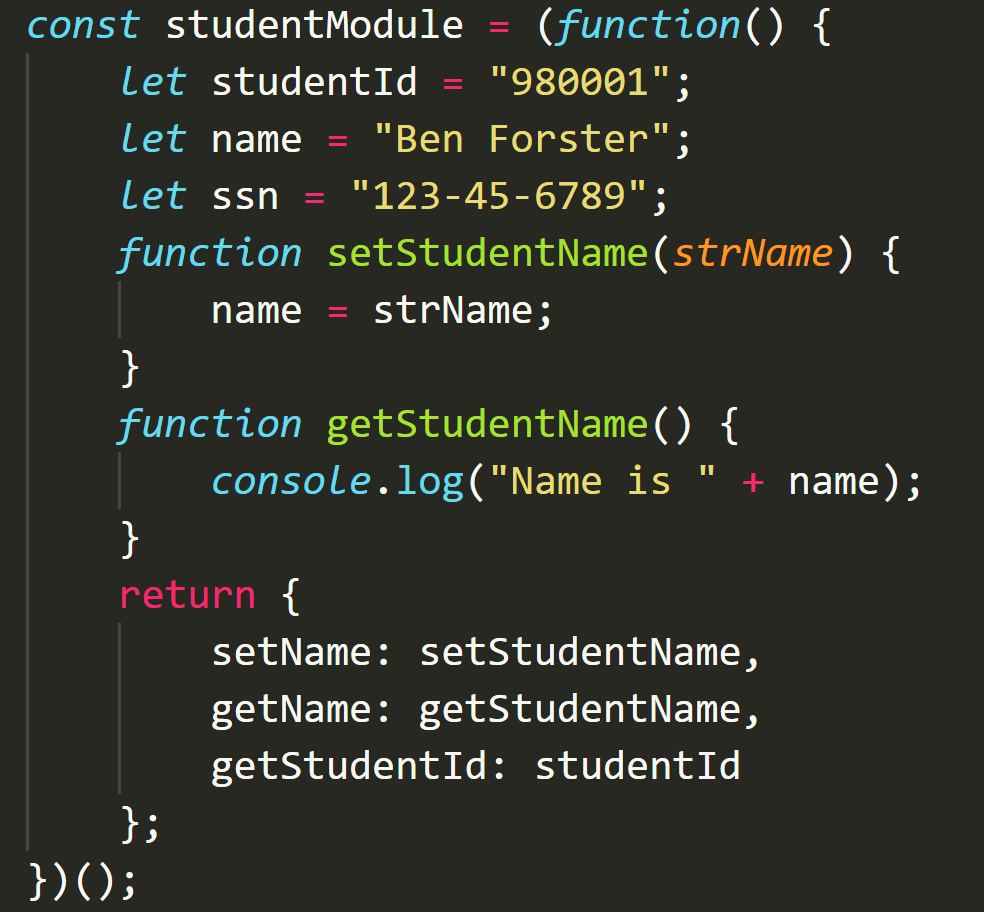
**Option B**.

logger object

**Option C**.

ERROR object

* 1. (3 points) Based on the code snippet given below, which is the incorrect statement:



**Option A**.

getStudentName is a public function

**Option B**.

studentId is a private variable

**Option C**.

setName is a public function

* 1. (2 points) Consider the HTML markup code-snippet for the submit button element given below.

<button id="btnSubmit" type="submit">Submit</button>

Which option is NOT the recommended best practice for adding its click event handler?

**Option A**.

<button id="btnSubmit" type="submit" onclick=”addStudent();”>Submit</button>

**Option B**.

document.querySelector(“#btnSubmit”).addEventListener(“click”, function(event) {…})

**Option C**.

$(“#btnSubmit).on(“click”, function(event) {…})

**Part III – Skill (JavaScript/Web Coding):** (60 points)

1. (5 points) Write a JavaScript function named, *calculateAverage*, that takes any variable amount of numbers, as input, and returns the average. E.g. calculateAverage(1,2,3) should return 2, calculateAverage(1,2,3,4) should return 2.5 and calculateAverage(1,2,3,4,5,16.6) should return 5.26666, and so on.

**function** calculateAverage() {  
 **let** average = 0;  
 **if** (arguments.length > 0) {  
 **let** total = 0;  
 **for** (**let** arg **of** arguments) {  
 total += +arg;  
 }  
 average = total / arguments.length;  
 }  
 **return** average;  
}

1. (5 points) Using JavaScript functional programming style, write a function named, multiplesCount, that takes an array of numbers, nums and an integer base, m, and returns how many of those numbers are multiples of m. e.g. multiplesCount([1,2,3,4,5,6], 3) should return 2, multiplesCount([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15], 5) should return 3 and so on. **[Note: Make sure your solution is a functional programming approach; No use of ‘for…loop’ or ‘if’ statements allowed]**

**function** multiplesCount(nums, m) {  
 **return** nums.filter(num => num % m == 0).length;  
}

1. (10 points) **JavaScript Inheritance – redHondaAccord IS-A Car**:
   1. (5points) Define a base object named, **Car**, as object literal, with the following properties – **make** (set its initial value as ‘default’), **model** (set its initial value as ‘default’), **color** (set its initial value as ‘default’), and include the following two methods:
      1. drive - takes a parameter named, speed, and prints-out the following message to the console – “The [color] [make] [model] is driving at [speed] mph.
      2. stop – prints-out the following message to the console – “The [color] [make] [model] is stopping.

Using inheritance, create a derived object named, redHondaAccord, which inherits from the Car object.

Assign the values ‘red’, ‘Honda’ and ‘Accord’ to the respective properties of color, make, model for the redHondaAccord object.

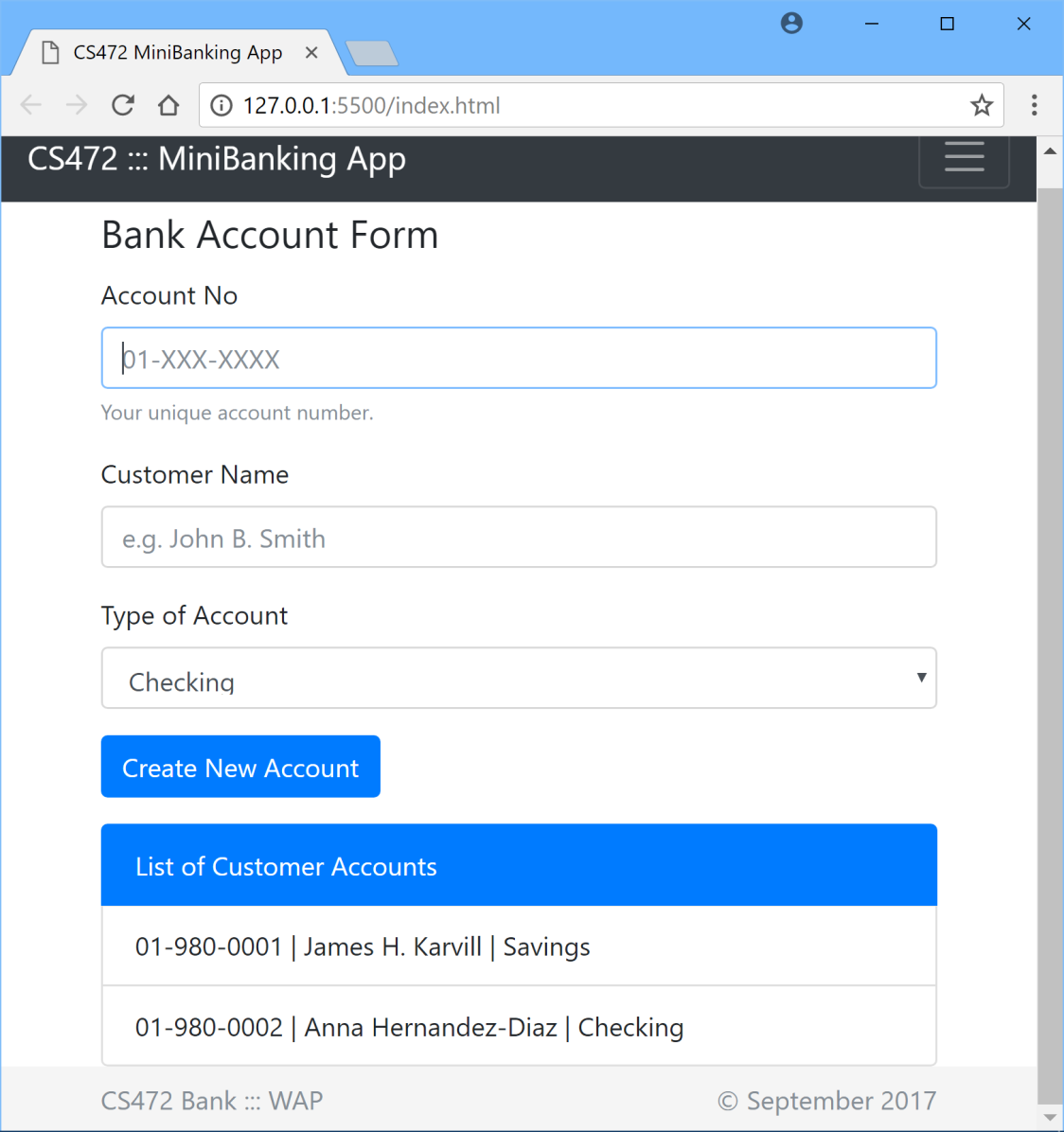
Invoke the drive(…) method on the redHondaAccord object, passing it a **speed** value of 200.

**var** Car = {  
 make: 'default',  
 model: 'default',  
 color: 'default',  
 drive: **function** (speed) {  
 console.log('The ' + **this**.color + ' ' + **this**.make + ' ' + **this**.model + ' is driving at ' + speed + ' mph');  
 },  
 stop: **function** () {  
 console.log('The ' + **this**.color + ' ' + **this**.make + ' ' + **this**.model + ' is stopping');  
 }  
}  
  
**var** redHondaAccord = Object.create(Car);  
redHondaAccord.color = 'red';  
redHondaAccord.make = 'Honda';  
redHondaAccord.model = 'Accord';  
  
redHondaAccord.drive(200);  
redHondaAccord.stop();

* 1. (5points) Re-implement the above described inheritance structure, but this time, using a Constructor function. i.e. define a constructor function named Car and then add to it, a drive(…) method and a stop() method. Next, based on your constructor function, create a derived object named redHondaAccord, with red, Honda and Accord for color, make and model respectively. Finally, invoke its drive method passing in the value of speed, 200.

**function** Car() {  
 make = 'default';  
 model = 'default';  
 color = 'default';  
  
 **function** stop() {  
 console.log('The ' + **this**.color + ' ' + **this**.make + ' ' + **this**.model + ' is stopping');  
 }  
}  
  
Car.prototype.drive = **function**(speed) {  
 console.log('The ' + **this**.color + ' ' + **this**.make + ' ' + **this**.model + ' is driving at ' + speed + ' mph');  
}  
  
Car.prototype.stop = **function** () {  
 console.log('The ' + **this**.color + ' ' + **this**.make + ' ' + **this**.model + ' is stopping');  
}  
  
**var** redHondaAccord = **new** Car();  
redHondaAccord.color = 'red';  
redHondaAccord.make = 'Honda';  
redHondaAccord.model = 'Accord';  
  
redHondaAccord.drive(200);  
redHondaAccord.stop();

1. (40 points) Using HTML, CSS and JavaScript, implement a mini-BankAccount single-page web application, as shown in the UI screenshot below, with the following features and functionalities:



1. Code the User interface of the webapp using standard, semantically correct HTML5 markup, including all the form fields as shown above.
2. Apply styling using Bootstrap or you may apply your own custom CSS styling to produce the same/similar look and layout.
3. Account Number and Customer Name are mandatory to create a new account.
4. Add validation using appropriate regular expression to ensure that any Account Number entered is in the specified format, 01-XXX-XXXX, as shown on the UI above.
5. The user can create 2 types of Bank Accounts – Checking and Savings.
6. Use JQuery and AJAX to load the List of existing Customer Accounts, by making an AJAX HttpGET request to a JSON-formatted data file located at the url – <http://yourApplicationHostAddress/data/customerData.json>.
7. Implement the ‘Create New Account’ functionality to add a new account data to the List on the webpage.
8. Add the Customer Accounts data to localStorage, such that when the browser is closed and re-opened, the webapp still displays all customer Account data, including any newly added accounts.

*(Please note: Points will be awarded based on your adherence to Web programming recommended best practices such as use of unobstrusive CSS, unobstrusive JS etc.*

***Also, make sure all your coding for this question is done in One webapp root folder named P3Q4, contained inside your Exam2 folder****)*

**HTML**

<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>Exam 2</title>  
 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta/css/bootstrap.min.css" integrity="sha384-/Y6pD6FV/Vv2HJnA6t+vslU6fwYXjCFtcEpHbNJ0lyAFsXTsjBbfaDjzALeQsN6M" crossorigin="anonymous">  
 <link rel="stylesheet" href="css/style.css">  
</head>  
<body>  
<div class="container">  
 <div class="row">  
 <div class="col">  
 <h1>Bank Account Form</h1>  
 <form method="get" id="customer-form">  
 <div class="form-group">  
 <label for="account-no">Account No</label>  
 <input type="text" id="account-no" placeholder="01-XXX-XXXX" class="form-control" pattern="01-\d{3}-\d{4}" required>  
 <span class="form-text text-muted">Your unique account number.</span>  
 </div>  
 <div class="form-group">  
 <label for="customer-name">Customer Name</label>  
 <input type="text" id="customer-name" placeholder="e.g John B. Smith" class="form-control" required>  
 </div>  
 <div class="form-group">  
 <label for="account-type">Type of Account</label>  
 <select name="account-type" id="account-type" class="form-control">  
 <option value="Checking">Checking</option>  
 <option value="Savings">Savings</option>  
 </select>  
 </div>  
 <div class="form-group">  
 <button class="btn btn-primary" id="btn-submit" type="submit">Create New Account</button>  
 </div>  
 </form>  
 </div>  
 </div>  
</div>  
  
<div class="container">  
 <div class="row">  
 <div class="col">  
 <ul id="customer-list" class="list-group">  
 <li class="list-group-item active bg-primary border-primary">List of Customer Accounts</li>  
 </ul>  
 </div>  
 </div>  
</div>  
 <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></script>  
 <script src="js/app.js"></script>  
</body>  
</html>

**Javascript:**

// Question 4  
$(**function** () {  
 **if** (!localStorage.getItem('bankAccount')) {  
 $.ajax({  
 url: './data/customerData.json',  
 type: 'GET',  
 dataType: 'JSON'  
 }).done((result) => {  
 localStorage.setItem('bankAccount', JSON.stringify(result.customers));  
 **for** (**const** customer **of** result.customers) {  
 **let** li = $('<li></li>').text(customer.customerID + ' | ' + customer.name + ' | ' + customer.accountType);  
 li.addClass('list-group-item');  
 $('#customer-list').append(li);  
 }  
 }).fail(**function**(xhr, status, err) {  
 console.log(status);  
 }).always(**function**(xhr, status) {  
 console.log(status);  
 });  
 } **else** {  
 **let** customersStr = localStorage.getItem('bankAccount');  
 **let** customers = JSON.parse(customersStr);  
 **for** (**const** customer **of** customers) {  
 **let** li = $('<li></li>').text(customer.customerID + ' | ' + customer.name + ' | ' + customer.accountType);  
 li.addClass('list-group-item');  
 $('#customer-list').append(li);  
 }  
 }  
  
 $('#customer-form').on('submit', (e) => {  
 e.preventDefault();  
 **let** li = $('<li></li>').text($('#account-no').val() + ' | ' + $('#customer-name').val() + ' | ' + $('#account-type').val());  
 li.addClass('list-group-item');  
 $('#customer-list').append(li);  
  
 // Write customer data to localStorege  
 **let** customersStr = localStorage.getItem('bankAccount');  
 **let** customers = JSON.parse(customersStr);  
  
 **const** customer = {  
 "customerID": $('#account-no').val(),  
 "name": $('#customer-name').val(),  
 "accountType": $('#account-type').val()  
 }  
  
 customers.push(customer);  
 **const** customerStrToWriteDown = JSON.stringify(customers);  
 localStorage.setItem('bankAccount', customerStrToWriteDown);  
  
 // Reset form  
 $('#customer-form input').val('');  
 $('#account-no').focus();  
 })  
});

**//-- The End --//**