THE UNIVERSITY OF WESTERN AUSTRALIA

Department of Computer Science and Software Engineering

SEMESTER 1, 2018 EXAMINATIONS

CITS3403
Agile Web Development

FAMILY NAME:_____ GIVEN NAMES: _____

STUDENT ID: | | | | | | | | | SIGNATURE: _____

This Paper Contains: **18** pages **(including title page)**
Time allowed: **2 hours**

INSTRUCTIONS:

This exam contains 10 questions worth 5 marks each.

Candidates must attempt ALL questions.

The questions should be answered in the space provided in this examination paper.

PLEASE NOTE

Examination candidates may only bring authorised materials into the examination room. If a supervisor finds, during the examination, that you have unauthorised material, in whatever form, in the vicinity of your desk or on your person, whether in the examination room or the toilets or en route to/from the toilets, the matter will be reported to the head of school and disciplinary action will normally be taken against you. This action may result in your being deprived of any credit for this examination or even, in some cases, for the whole unit. This will apply regardless of whether the material has been used at the time it is found.

Therefore, any candidate who has brought any unauthorised material whatsoever into the examination room should declare it to the supervisor immediately. Candidates who are uncertain whether any material is authorised should ask the supervisor for clarification.

Supervisors Only – Student left at:

This page is intentionally left blank

1. (5 marks)

Describe using sample code three different ways of registering a JavaScript event handler.

Briefly explain the advantages and disadvantages of each approach.

2. (5 marks)

Expand and define the following acronyms:

- DOM
- REST
- CRUD
- MEAN
- SPA

3. (5 marks)

Consider the following HTML and JavaScript code, answer the questions on the following page:

```
<html>
<head>
<script>
function countTags(n) {
  var numtags = 0;
  if (n.nodeType == 1 /*Node.ELEMENT_NODE*/)
    numtags++;
    var children = n.childNodes;
    for(var i=0; i < children.length; i++) {
      numtags += countTags(children[i]);
  }
  return numtags;
}

function count(str){
  var matched = str.match(/(\w+) +and +(\w+)/i);
  for(var i=1; i < matched.length; i++){
    var numtags = countTags(document.getElementsByTagName(matched[i])[0]) - 1;
    console.log("The first " + matched[i] + " has " + numtags + " children tag(s)");
  }

}
</script>
</head>

<body onload="count('P and div')">
  <p>This is a <i>sample</i> document.
  <div>
    <p>Some section that is <i>very</i> important.
  </div>
</body>
</html>
```

(a) What does of the function `countTags(n)` do?

(b) What's the console output after this page is loaded?

(c) Can you identify any errors or problems or limitations with the `count(str)` function as implemented?

4. (5 marks)

Consider the JavaScript code below, and answer the questions on the next page.

```
function Person() {}

Person.prototype.walk = function(){
  console.log ('I am walking!');
};
Person.prototype.sayHello = function(){
  console.log ('hello');
};

function Student() {}

Student.prototype = new Person();

Person.prototype.sing=function(){
  console.log('Rock and roll');
};

Student.prototype.sayHello = function(){
  console.log('hi, I am a student');
}

var student1 = new Student();
student1.walk();
student1.sayHello();
student1.sing();
```

(a) What is the console output?

(b) Describe how JavaScript's protoype-based inheritance worked in this case.

5. (5 marks)

Consider a web application using a MEAN stack with server side rendering. Taking your project Stage II as an example, a typical user experience would be: when they visit your app, they need to enter username and password to login, and be taken to a list of "projects" they have previously created. Describe the processes involved in creating and serving the login page and the listing page.

6. (5 marks)

A typical Express project will include the following directories and files:

- `app.js`
- `package.json`
- `views` directory
- `public` directory
- `routes` directory

If we would like to turn these into a proper MVC architecture, describe the minimum steps to achieve it. No coding is required.

7. (5 marks)

Use one entity and its collection from your project Stage II as an example, describe how RESTful API works together with HTTP verbs to map URLs to perform the CRUD actions for both collections and individual elements.

8. (5 marks)

Picnic.com is a site that organises family gatherings such as BBQs, playdates or picnics, and suggests possible venues that may suit the purpose.

For each user, Picnic.com records a name, email address, postcode, and a list of venues that they have been to.

Each venue has a name, location, BBQ facility availability or not, and a list of reviews with each review records the user, his/her personal rating (1-5 stars) and the comments on the venue.

Specify a set of mongoose schemas to describe the necessary models for this data.

9. (5 marks)

Given the Picnic.com scenario described in the previous question, use either Pug(Jade) or embedded JavaScript to write a view that takes a JavaScript object corresponding to a Picnic.com user. The page should display the user's name and then list the names of the venues they have rated, along with the number of stars they gave the venue.

You may assume that there is a JavaScript object provided with properties `name` and `venues`. There is also an image file `star.jpg` you can use to draw the star rating.

10. (5 marks)

Explain the difference between one way data binding and two way data binding in the context of MEAN stack web application development.

What are the advantages and disadvantages of each.

EXTRA BLANK PAGE

EXTRA BLANK PAGE

EXTRA BLANK PAGE

EXTRA BLANK PAGE

End of Paper

THE UNIVERSITY OF WESTERN AUSTRALIA

CSSE

SEMESTER 1, 2018 EXAMINATIONS

CITS3403
Agile Web Development

FAMILY NAME:_____ GIVEN NAMES: _____

STUDENT ID: | | | | | | | | | SIGNATURE: _____

This Paper Contains: **7** pages **(including title page)**
Time allowed: **2:00 hours**

Cheat - sheet

PLEASE NOTE

Examination candidates may only bring authorised materials into the examination room. If a supervisor finds, during the examination, that you have unauthorised material, in whatever form, in the vicinity of your desk or on your person, whether in the examination room or the toilets or en route to/from the toilets, the matter will be reported to the head of school and disciplinary action will normally be taken against you. This action may result in your being deprived of any credit for this examination or even, in some cases, for the whole unit. This will apply regardless of whether the material has been used at the time it is found.

Therefore, any candidate who has brought any unauthorised material whatsoever into the examination room should declare it to the supervisor immediately. Candidates who are uncertain whether any material is authorised should ask the supervisor for clarification.

Supervisors Only - Student left at:

This page has been left intentionally blank

JavaScript

Methods

Object
toString
toLocaleString
valueOf
hasOwnProperty
isPrototypeOf
propertyIsEnumerable

String
charAt
charCodeAt
fromCharCode
concat
indexOf
lastIndexOf
localeCompare
match
replace
search
slice
split
substring
substr
toLowerCase
toUpperCase
toLocaleLowerCase
toLocaleUpperCase

RegEx
test
match
exec

Array
concat
join
push
pop
reverse
shift
slice
sort
splice
unshift

Number
toFixed
toExponential
toPrecision

Date
parse
toDateString
toTimeString
getDate
getDay
getFullYear
getHours
getMilliseconds
getMinutes
getMonth
getSeconds
getTime
getTimezoneOffset
getYear
setDate
setHours
setMilliseconds
setMinutes
setMonth
setSeconds
setYear
toLocaleTimeString

XMLHttpRequest

Safari, Mozilla, Opera:

```
var req = new XMLHttpRequest();
```

Internet Explorer:

```
var req = new
ActiveXObject("Microsoft.XMLHTTP");
```

XMLHttpRequest Object Methods

abort()
getAllResponseHeaders()
getResponseHeader(header)
open(method, URL)
send(body)
setRequestHeader(header, value)

XMLHttpRequest Object Properties

onreadystatechange
readyState
responseText
responseXML
status
statusText

XMLHttpRequest readyState Values

0	Uninitiated
1	Loading
2	Loaded
3	Interactive
4	Complete

JAVASCRIPT IN HTML

External JavaScript File

```
<script type="text/javascript"
src="javascript.js"></script>
```

Inline JavaScript

```
<script type="text/javascript">
  <!--
     // JavaScript Here
  //-->
</script>
```

Functions

Window
alert
blur
clearTimeout
close
focus
open
print
setTimeout

Built In
eval
parseInt
parseFloat
isNaN
isFinite
decodeURI
decodeURIComponent
encodeURI
encodeURIComponent
escape
unescape

REGULAR EXPRESSIONS - FORMAT

Regular expressions in JavaScript take the form:

var RegEx = **/pattern/modifiers**;

REGULAR EXPRESSIONS - MODIFIERS

/g	Global matching
/i	Case insensitive
/s	Single line mode
/m	Multi line mode

REGULAR EXPRESSIONS - PATTERNS

^	Start of string
$	End of string
.	Any single character
(a|b)	a or b
(...)	Group section
[abc]	Item in range (a or b or c)
[^abc]	Not in range (not a or b or c)
a?	Zero or one of a
a*	Zero or more of a
a+	One or more of a
a{3}	Exactly 3 of a
a{3,}	3 or more of a
a{3,6}	Between 3 and 6 of a
!(pattern)	"Not" prefix. Apply rule when URL does not match pattern.

EVENT HANDLERS

onAbort	onMouseDown
onBlur	onMouseMove
onChange	onMouseOut
onClick	onMouseOver
onDblClick	onMouseUp
onDragDrop	onMove
onError	onReset
onFocus	onResize
onKeyDown	onSelect
onKeyPress	onSubmit
onKeyUp	onUnload
onLoad	

FUNCTIONS AND METHODS

A method is a type of function, associated with an object. A normal function is not associated with an object.

DOM Methods

Document
clear
createDocument
createDocumentFragment
createElement
createEvent
createEventObject
createRange
createTextNode
getElementsByTagName
getElementById
write

Node
addEventListener
appendChild
attachEvent
cloneNode
createTextRange
detachEvent
dispatchEvent
fireEvent
getAttributeNS
getAttributeNode
hasChildNodes
hasAttribute
hasAttributes
insertBefore
removeChild
removeEventListener
replaceChild
scrollIntoView

Form
submit

DOM Collections
item

Range
collapse
createContextualFragment
moveEnd
moveStart
parentElement
select
setStartBefore

Style
getPropertyValue
setProperty

Event
initEvent
preventDefault
stopPropagation

XMLSerializer
serializeToString

XMLHTTP
open
send

XMLDOM
loadXML

DOMParser
parseFromString

Pug cheatsheet

Basic document

```
doctype html
html(lang='en')
  h1.class#id(name='hi')
    | This is some text, hello there,
    = name

  - javascript()
```

Comments

```
// This comment will appear in the HTML
```

```
//- This is a silent comment
```

```
//-
  Nesting inside a comment creates
  a comment block
```

See: Comments

Multiline text

Elements

```
div
  | Just a div
```

```
.search
  | A div, with class 'search'
```

```
h1 A heading with text
```

```
h1= page.title
```

```
div.class
div.class1.class2
h1.header
```

Includes (partials)

```
include ./includes/head.pug
```

```
include:markdown article.md
```

See: Includes

Attributes

```
input(type='text' name='q' autofocus)
```

```
- var authenticated = true
body(class=authenticated ? 'authed' : 'anon')
```

See: Attributes

Iteration

```
ul
  each user in users
    li= user
```

Layouts

```
//- page.pug
extends layout.pug

block title
  | hello

block content
  | hello
```

```
This is text that doesn't need to
be prefixed by pipes.
```

```
// It's great for raw
// JavaScript and stuff
alert('hello')
```

Conditionals

```
a(href='/logout') Sign out

a(href='/login') Sign in
```

See: Conditionals

```
//- layout.pug
title
  block title
body
  block content
```

Mixins

Mixins

```
ul
  ...
```

```
+list
```

Mixins allow you to create reusable code blocks. See: Mixins

Mixin attributes

```
span.pet= name
```

```
+pet('cat')
```

See: Mixin attributes

Mixin blocks

```
article
  h2.title= title
```

```
+article('hello there')
  p Content goes here
```

See: Mixin blocks

▶ **0 Comments** for this cheatsheet. Write yours!

Install

```
$ npm install mongoose --save
```

Connect

```
const mongoose = require('mongoose');

const uri = process.env.MONGO_URI || 'mongodb://localhost/test';

mongoose.connect(uri, function(err, res) {
  ...
});
```

Defining a schema

```
const userSchema = new mongoose.Schema({
  name: {
    first: String,
    last: { type: String, trim: true }
  },
  age: { type: Number, min: 0 },
  posts: [ { title: String, url: String, date: Date } ],
  updated: { type: Date, default: Date.now }
});
```

SchemaTypes

- String
- Number
- Date
- Buffer
- Boolean
- Mixed
- ObjectId
- Array

Instantiating a model

A model is a constructor compiled from a schema. Model instances represent documents.

```
const User = mongoose.model('User', userSchema);

var u = new User({
  name: {
    first: 'Tony',
    last: 'Pujals'
  },
  age: 99
});
```

Query

query

$where

```
query.$where('this.comments.length > 10 || this.name.length > 5')

// or

query.$where(function() {
  return this.comments.length > 10 || this.name.length > 5;
});
```