CSC309H Cheat Sheet

by Abisco via cheatography.com/27378/cs/7925/

Internet Basics

What is the Internet

- Collection of computer networks that use a protocol to exchange data
- IETF (Internet Enforcement Task Force): Internet protocol standards

IP (Internet Protocol): Simple protocol for exchanging data between computers

TCP: Adds multiplexing and reliability on top of IP

- Multiplexing: Multiple programs using same IP address
- Reliability: Guaranteed, ordered and error-checked delivery

DNS (Domain Name Server): Set of servers that map(translate) written names to ip addresses

URL (Uniform resource locator): Identifies the path to a document on the web server

HTTP (Hypertext transport protocol): Set of commands understood by a web server and sent from a browser

HTTP Commands:

- GET filename: Download
- POST filename: send a web form response
- PUT filename: upload

Security

Understanding Threats: Defacement, Infiltration, Phishing (Spoofed site that looks real, retrieve login credentials), Pharming(Like phishing, get user to enter sensitive data into spoofed site, no conscious action required by the victim), DNS Cache Poisoning (Attacker is able to compromise DNS tables so as to redirect legitimate URL to their spoofed site), DNS translates URL to IP Addresses

SQL Injection:Untrusted input inserted into query or command

Solutions: Defence in Depth, Whitelisting over Blacklisting, Input validation and Escaping, Use prepared statements and Bind variables

Mitigation: Prevent schema and information leaks, Limit privileges (defence in depth), Encrypt sensitive data stored in Database, Harden DB server and Host O/S, Apply input validation

Password Protection: Straw man Proposal, Salting(Include additional info in hash), Honeypots(Simple username/password combos as 'honey' to attract attackers), Aging passwords(Encourage/require users to change passwords every so often)

HTTP Is stateless: Cookies(-Browser can refuse cookies, -size limit/ expiration policy), Hidden Variables (-Following hyperlinks causes a loss of state, -Current submitted page represents current state independent of what was done previously), URL Rewritting (-Current submitted page represents current state independent of what was done previously)

Web Security: Same Origin Policy (A webpage may include some JavaScripts to access its DOM and send AJAX msgs to its backend, try to steal information from another website), XSSI (Cross-site script inclusion, making sure scripts aren't dynamically created with sensitive data.

Security (cont)

Do not support GET requests for scripts returning URLS) XSS (Enables attackers to inject scripts into webpages viewed by other users, which can steal cookies, change appearances of web sites...Do validation and HTTP only option for cookies), XSRF (Makes a user to submit requests on behalf of the attacker. Protection: Give a secret token to a user and tell the user to submit it along with cookie on following requests).

Web Performance

HTML Techniques: Lazy load content, Use idle time to pre-load content, Batch DOM updates, Set image sizes before loading, Reduce DOM depth

CSS Techniques: Stylesheets at the top, Remove unused CSS Rules, Avoid universal selectors, Don't abuse border-radius & transform, Prefer selectors with native JS Support

Network Techniques: Make fewer HTTP requests, Content delivery network, Split resources across servers -load balance, But avoid too many DNS lookups

Performance

Load Testing: Process of putting demand on a system or device and measuring its response. Performed to determine a system's behaviour under both normal and anticipated peak load conditions.

Locust.io

+: Write simple python code to simulate a user behaviour

Performance (cont)

+: Handled thousands of users on a single machine
-: Results are downloadable in CSV format
Back-end Tips: Increase parallelism of node.js, Caching, DB Index

Express

OUR ASSIGNMENT

```
var express = require('express');
var app = express();
 app.get('/newrecipe', function(req, res) {
        var User = require('../app/models/user');
        res.render('newrecipe.ejs', { message: 'loggedin' });
    });
    app.post('/newrecipe', function(req, res) {
          var newRecipe = new Recipe();
          newRecipe.author_id = req.user._id;
         newRecipe.name = req.body.name;
         newRecipe.description = req.body.description;
         res.render('newrecipe.ejs', { message: 'done'});
    });
```

EXAMPLE OF DEALING WITH A SIMPLE LOGIN FORM

HTML CODE

```
<form action=”form_submitted.php” method=”GET”>
        <h1>Login Form</h1>
        <label> Login:</label> <input type=”text” name=”login”> <br>
        <label> Password: </label>
```

Express (cont)

```
<input type=”password” name=”password”><br>
        <button type=”submit” >Log In </button>
</form>
```

SERVER SIDE CODE

```
var express = require(‘express’);
var router = express.Router();
router.get(‘form_submitted.php*’, function(req, res){
if (req.query.login == req.query.password){
res.send(‘Login Successful’);
} else {
res.send(‘Error: Login Failed’);
}
});
```

Less/Sass

CSS Pre-processor: Converts code written in a preprocessed language in css
Allows us to do:
- Don't repeat yourself principle
- Maintainability
- Readability
- Natural extension

Less
- Easier to transition from CSS
- Resembles CSS
- Syntax not as jarring as SASS
- Newer Than SASS, inspired by it

SASS

Less/Sass (cont)

- Syntax is quite different from CSS
- Symbols used are similar to bash
- More functionality/capability than LESS
- Complex tasks are more pragmatic than LESS

Databases

RDBMS (Relational Database Management System): Has Concurrent access, Fault Tolerance, Data Integrity, Scalability
NoSQL: Flexible Schema, Cheaper to setup, massive scalability (Integrated Caching and Auto sharing), relaxed consistency BUT no declarative query language, and fewer guarantees due to ReCo.

Session and Cookies

HTTP Is stateless
- Simply allows a browser to request a single document from web server
- It remembers nothing between invocations, thus short lived
- When we started using web applications, we started ad hoc states
**Adding state to HTTP*
- Client Mechanisms:
1. Cookies *Size limit/ expiration policy, browser can refuse*
2. Hidden variables *hyperlinks leads to loss of state*

Session and Cookies (cont)

3. URL Rewriting *Current submitted page represents current state independent of what was done previously*
4. Local Storage
- Server Mechanisms
1. Sessions (Persistent Storage) - In a file or database

Canvas Coding

ASSIGNMENT CODE

```
var canvas =
document.getElementById("game");
var context =
canvas.getContext("2d");
    document.getElementById("main")
.innerHTML = "<canvas id='"game "'
width = 400 height = '"600"'>
</canvas> ;
canvas =
document.getElementById("game");

// Add Mouse down listener
canvas.addEventListener("mousedown"
, mouseDidPressDown, false);
canvas.addEventListener("mouseup",
mouseDidRelease, false);

context = canvas.getContext("2d");
function mouseDidPressDown(event) {
var WIDTH = HEIGHT * 0.65;
var mousePosition =
mousePositionInCanvas(event.clientX
, event.clientY);
       //DO WHATEVER with
mousePosition.x and
mousePosition.y
}
```

EXAM CODE

Canvas Coding (cont)

```
<body>
    <canvas id='myCanvas'
width='500' height='400'> Canvas
not supported </canvas>
 </body>
</html>
<script>
  var canvas =
document.getElementById('myCanvas')
;
  var context =
canvas.getContext("2d");
  function getMousePos(canvas, evt)
{
      var rect =
canvas.getBoundingClientRect();
return {
x: evt.clientX - rect.left *
(canvas.width / rect.width),
y: evt.clientY - rect.top *
(canvas.height / rect.height)
};
  }
 canvas.addEventListener('click',
function(evt){
   var temp = getMousePos(canvas,
evt);
   context.translate(temp.x,
temp.y);
       drawCoolShape(context);
       context.translate(-temp.x,
-temp.y);
  }, false);
</script>
```

AngularJS

Why

```
- Lightweight, free
- Modularity
- Reusable components
```

What we used previously

```
- Allows for DOM manipulation
- Does not provide structure to
your code
<div ng-app="">
<p>Input something in the input
box:</p>
<p>Name : <input type="text" ng-
model="name" placeholder="Enter
name here"></p>
<h1>Hello {{name}}</h1>
```

XML vs JSON

Some Basics

```
- XML is easy to read and make
automation easy, but bulky
structure makes files large, can be
hard to structure data into good
xml format
- Javascript XML has properties and
methods to structure well
```

Something in XML

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New"
onclick="CreateNewDoc()" />
    <menuitem value="Open"
onclick="OpenDoc()" />
    <menuitem value="Close"
onclick="CloseDoc()" />
  </popup>
</menu>
```

Same in JSON

```
{"menu": {
```

XML vs JSON (cont)

```
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

Navigating JSON

```
var data = JSON.Parse(file)
var fileId = data.menu.id;
var firstMenu = data.menu.popup.menuitem[0];
```

Mocha

What is it: Testing for Node

Example:

```
var assert = require('assert');
var calc = require('./calc.js');
describe('Calculator Tests', function() {
it('returns 1+1=2', function(done) {
assert.equal(calc.add(1, 1), 2);
done();
});
it('returns 2*2=4', function(done) {
assert.equal(calc.mul(2, 2), 4);
done();
});
});
```

AJAX

Asynchronous Javascript and XML: Not a programming language, just a way of using Javascript, Downloads data from server in background, Avoids dynamically updating a page without making the user wait

XMLHttprequest (and why it sucks): Javascript includes an XMLHttprequest object that can fetch files from a web server, BUT clunky and browser incompatibilities

JQuery: Cross browser, simplifies javascript

```
$(document).ready(function(){
    $("p").click(function(){
        $(this).hide();
    });
});
```

Simple Web Request

Basic Structure:

Request: GET /HTTP/1.1

Reply:HTTP/1.1 301 moved permanently

Big Picture

- Client-server model: A client process wants to talk to a server process
- DNS Lookup: Client must find server
- Ports: Clients must find service on server
- Finally establish a connection so they can talk

Types of connection (TCP/UDP)

- Connection oriented model: Use Transmission control protocol (TCP)
- Connectionless Model: Uses user datagram protocol (UDP)

GIT

Difference between CVC and DVC:

- Centralized Version Control: Repository goes straight to each working copy/pc
- Distributed Version Contol: Each computer has it's own repository, which can pull and push to server repository. **WHAT GIT USES**

Working with remote repository

- git remote abu link *Creates a reference called abu to the link*
- git clone https://blah.com/csc309.git *clone remote rep and create local one*
- git fetch mashiyat *Download changes from mashiyat's repository to my local repository*
- git pull mashiyat *Downloaded changes and merges them to my local repository*
- git push origin master
- git push mashiyat master
- git merge blah *Merge changes made in blah branch to current branch*

HTML5 and CSS3

HTML5: New features

- Semantic elements and markups
- Audio and video support
- Canvas
- Drag and drop
- Local data storage: Unlike cookies, the storage limit is far larger

CSS3

HTML5 and CSS3 (cont)

- Allows a lot of new things, such as border-radius
- Viewport (vary with device size)

Responsive Web Design

@media (max-width: 600px) { .facet_sidebar { display: none; } }

Example of how to style the media for phones

Web Architectures

Data independence in Rel. DBMS

- Logical Independence: The Ability to change the logical schema without changing the external schema or application programs.
- Physical Independence: The ability to change the physical schema without changing the logical schema

Significance of Tiers

- N-Tier architectures try to separate the components into different tiers/layers. Tier: physical separation, Layer: logical separation
- 1-Tier architecture: All 3 layers on the same machine - All code and processing kept on a single machine
- 2-Tier Architecture: Database runs on server
- 3-Tier Architecture: Each layer can potentially run on a different machine

MVC Design Pattern: Chang look and feel without changing the core/logic, Maintain multiple views of the same data

MongoDB Schema

```
var mongoose = require('mongoose');
var bcrypt = require('bcrypt-nodejs');
var userSchema = mongoose.Schema({
        firstname : String,
        lastname : String,
        password : String,
        phonenumber : Number,
        fav_cuisine :[String],
        admin : Boolean
});
// generating a hash
userSchema.methods.generateHash =
function(password) {
    return
bcrypt.hashSync(password,
bcrypt.genSaltSync(8), null);
};
// checking if password is valid
userSchema.methods.validPassword =
function(password) {
    return
bcrypt.compareSync(password,
this.password);
};
module.exports =
mongoose.model('User',
userSchema);
```

REST API Code

```
var requestBody = '';
var http = require("http"),
url = require("url"),
path = require("path"),
fs = require("fs");
PORT = 3000;
function handleRequest(request,
response) {
var rest = url &&
url.parse(request.url).pathname;
   var filePath = __dirname +
request.url;
var favs =
fs.readFileSync('js/favs.json');
if (chooseFile()) return;
//For each of the possible return
paths, send the json file
if (request.url == "/allTweets") {
returnTweets();
} else if (request.url ==
"/allUsers") {
returnTweets();
}
//Function to return Json
function returnJson(json) {
response.writeHead(200);
response.end(JSON.stringify(json,
null, 8));
}
//Provide the file in accordance
with the
function returnFile(path, type) {
var file = fs.readFileSync(path,
'utf8');
response.writeHead(200, type);
response.end(file);
}
```

REST API Code (cont)

```
function chooseFile() {
if (path.extname(filePath) == '.js') {
returnFile(filePath, {"Content-Type":
"text/javascript"});
return true;
}
if (rest.length <= 1) {
returnFile('./index.html', {"Content-Type":
"text/html"});
return true;
}
return false;
}
}
http.createServer(handleRequest).listen(PORT);
console.log("Nodejs Server running at
http://127.0.0.1:" + PORT + "/");
```

JQuery Selecting Code

```
function change(){
    $("body").find("*").hide();
    var images = $("body").find("img");
    images.parents().show();
    images.show();
  }
```