

# Day 1

Tuesday, December 8, 2020 9:42 AM

Mithril Emmet  
Open in browser  
Prettier - Code Formatter/beautify  
Material Icon Theme

textContent vs innerHTML

less memory used when textContent is used, use textContent when only the text needs to be updated instead of HTML

Microsoft ways  
attachEvent/detachEvent - deprecated

Mozilla ways  
addEventListener/removeEventListener

Placing script block in head vs body

- head - Getting dressed and going to a party
- body - Going to a party and getting dressed

`<script async src="./index.js"></script>` - Load index.js asynchronously  
`<script defer src="./index.js"></script>` - Delay the execution of the index.js once the body is loaded

JS

```
arr.concat(arr2) // merges and creates a new array
arr.slice(startIndex [endIndex]) // does not modify existing array
arr.splice(startIndex, deleteCount, 'newVal') // modifies existing array
arr.sort() // js uses bubble sort
let arr = [2, 4, 56, 78, 8, 10, 88, 9];
console.log(arr.sort()); // [10, 2, 4, 56, 78, 8, 88, 9]

arr.sort((a, b) => {
  if (a < b) return -1;
  else if (a > b) return 1;
  else return 0;
});
console.log(arr); // [2, 4, 8, 9, 10, 56, 78, 88]
```

This is because JavaScript's sort() method converts each item in the array into strings and constructs the sequence by comparing each array item based on the UTF-16 code values when there is no callback specified.

- sort works with string comparison perfectly
- for numbers and objects we need to use comparator functions

caniuse.com -> gives us browser support based on the feature

## var vs let vs const

- var gets hoisted to the top of the block
- let is used for block scope, even though it is block scope, it gets hoisted to temporal dead zone
- const is also used for block scope, even though it is block scope, it gets hoisted to temporal dead zone but the value cannot be assigned/modified again
  - o const value when used as collection(array/object) the properties of collection can be assigned/modified

Fat arrow does not have a scope of its own, it borrows the scope from the parent

## Ways to create objects

- var obj1 = {};
- var obj2 = new Object();
- var obj3 = Object.create(null); // Preferred way because it is immutable
  - o var obj3 = Object.create(null);
  - o Object.defineProperty(obj3, 'title', {  
value: 'Batman'  
}); //args: name of the object, property name
- Creating Immutable object in js and making it mutable using writable in descriptor
- By default writable is false
- Making enumerable true will help the use loop through the keys
- Making configurable true will help us to modify the descriptor

```
var obj4 = Object.create(null);
Object.defineProperty(obj4, 'title', {
  value: 'Joker',
  writable: true,
  enumerable: true,
  configurable: true
}); // args: object name, property name, descriptor of property
Object.defineProperty(obj4, 'titleAccess', {
  get: function() {
    return this.title;
  },
  set: function(value) {
    this.title = value;
  }
}); // args: object name, property name, descriptor of property
Object.defineProperty(obj4, 'power', {
  value: 7,
  writable: true,
  enumerable: true
});
```
- Github repo
  - o <https://github.com/vijaylabfiles/node>

# Day 1 - Assignments

Tuesday, December 8, 2020 10:38 AM

## Assignment:

Difference between body onload, window onload and DOMContentLoaded

- The DOMContentLoaded event will fire as soon as the DOM hierarchy has been fully constructed, the load event will do it when all the images and sub-frames have finished loading.
- DOMContentLoaded will work on most modern browsers, but not on IE including IE9 and above. There are some [workarounds](#) to mimic this event on older versions of IE, like the used on the jQuery library, they attach the *IE specific* [onreadystatechange](#) event

From <<https://stackoverflow.com/questions/2414750/difference-between-domcontentloaded-and-load-events>>

# Day 2

Wednesday, December 9, 2020 9:44 AM

## Multiple promises

- `Promise.all([promise1(), promise2()])` -> if one of the promise in the array is rejected the whole response will not be returned but we get the rejected promise error
- `Promise.race([promise1(), promise2()])` -> it considers the first promise response which gets resolved/rejected and rest is ignored.
- `Promise.any([promise1(), promise2()])` -> it waits for the first resolve response to be returned.

## Installing Node and Typescript

nvm -> Used to maintain different versions of node

Install nvm from <https://github.com/coreybutler/nvm-windows> -> "Download Now!" link

After installing nvm

Type `nvm install latest`

`nvm list`

`nvm use <version to be used>`

`npm install -g typescript`

To check for typescript

`tsc -v`

## Typescript

- Latest version is 4.1.2
- Adv
  - o Flexible
  - o Independent from dev env
  - o Easier to configure for correct compilation
- Use commands
  - o `tsc -v` // for version
  - o `tsc -h` // help
  - o `tsc -t` // target
  - o `tsc -w` // watch
  - o `tsc -outFile` // concat and output to a single file
  - o `tsc -outDir` //
  - o `tsc --pretty` // add styles
  - o `tsc <file-name> -w -t='ES5'` // to run a file with ES5 as target compilation
  - o `tsc --init` // to create a ts json config file
- Types



- Assigning multiple types
  - o `let str : (string | Array<string>) = 'dsfsf';` // ['kgjg', 'dsfsf'] can also be assigned as we have a union of types
- Optional arguments of a function are only assigned at the end
  - o 


```
function adder(num1: number = 0, num2: number = 0, message?: string) {
    return num1+num2;
}
console.log(adder(5, 6));
```
- Classes
  - o All classes generated by functions should have its methods created on the prototype to reduce memory usage
 

```
function Hero(fname, lname) {
    var mission = "Secret Mission"; // private property
    this.firstName = fname; // public property
    this.lastName = lname; // public property
}
Hero.prototype.getFullName = function() {
    return this.firstName + ' ' + this.lastName;
};
```

## NodeJs

- Server side js

# What is NodeJS ?



It is a server-side JavaScript platform  
Helps develop scalable network applications.

Its fast because its mostly C/C++

**NodeJS**  

**LibUV**

**V8 Engine**

Was adapted in windows version of NodeJS to replace UNIX libraries

Google's JavaScript processing engine

<b>LibUV</b>	Was adapted in windows version of NodeJS to replace UNIX libraries
<b>V8 Engine</b>	Google's JavaScript processing engine
<b>JavaScript and C++</b>	Most of builtin modules are developed using JavaScript or C++

## Where can I use it ?



Real time data app that requires frequent updates

Fast file upload

Ads Server application

Multiplayer Games

Web Socket Applications (chat / meeting )

## NodeJS is not



A web framework

Multi threaded ( its a single threaded server )

For everyone

For every project

## Lets the Fun begin



No Window

Global

No Document

Require

No Location

Module

# Lets the Fun begin



No Window

Global

No Document

Require

No Location

Module

# Lets the Fun begin



REPL : Read Evaluate Print Loop  
variables

Blocking v/s Non Blocking

Fetch data

Display content

Continue

-----

Fetch data

Let me know when it comes

continue

Blocking Code

Non Blocking Code

- commonjs - to import and export files as modules
- RequireJs uses amd (asynchronous module definition)
- system - also supports import/export

## OS Module

- `const os = require('os');`
- 1st place that node will look for 'os' into `node_modules`

## Event Emitters

- `const ee = require('events').EventEmitter;`
- `.on` will listen to the event emitted

- `.once` will listen to the event emitted only once
- To stop listening to an event use `removeListener` / `off` event
  
- `process.nextTick` - will happen when the entire file is evaluated



# Day 2 - Assignment

Wednesday, December 9, 2020 4:14 PM

emit an event once every second for 5 seconds

# Day 3

Thursday, December 10, 2020 10:12 AM

## Creating Node Module

- Sign up on npmjs.com
- Create folder in local
  - o Got to folder in cmd and type "npm init" / "npm init --yes"
  - o versioning 1.0.0 (Major.minor.bugfix)
- To update package to npmjs
  - o type npm login
  - o type username and password
  - o npm publish
- Creating server in node
  - o http
    - to get list of status codes: in cmd prompt -> node -> http.STATUS\_CODES
  - o express
    - Any transactions between browser and server will be handled by middleware
    - syntactical sugar on http
    - can leverage templates for html
    - To serve static files using express ->  
`app.use(express.static(__dirname));`
    - To rename images while serving in browser  
`app.use('/assets', express.static(__dirname+'/images'));`
  - o Tools - nodemon ->
- Streams
  - o to read large files we need streams to get data divided in chunks
  - o types
    - read stream - createReadStream
    - write stream - createWriteStream
    - buffer stream
- Templating in express js
  - o Jadelang.com/pugjs.org/ejs/handlebar - > For templating (Pug and ejs preferred)
    - npm i jade pug ejs --save
    - by default for templating it will search for the views folder
  - o For styling -> bootstrap, stylus, foundation, skeleton, pure
  - o ejs
    - Tags

## Tags

- `<%` 'Scriptlet' tag, for control-flow, no output
- `<%=` 'Whitespace Slurping' Scriptlet tag, strips all whitespace before it
- `<%=` Outputs the value into the template (HTML escaped)
- `<%-` Outputs the unescaped value into the template
- `<##` Comment tag, no execution, no output
- `<%%` Outputs a literal '<%'
- `%>` Plain ending tag
- `-%>` Trim-mode ('newline slurp') tag, trims following newline
- `_%>` 'Whitespace Slurping' ending tag, removes all whitespace after it

# Day4

Monday, December 14, 2020 3:02 PM

temp pwd: #94L:/yLlg/i

```
mongod
mongo
show dbs
```

<https://developer.mongodb.com/quickstart/cheat-sheet>

```
use deloittedb
db.createCollection('users')
db.users.insert({name: 'Ram', mail:'rayerra@deloitte.com', place: 'Vizag'})
db.users.find(); // to list all the inserted/update records
```

-----

-----

```
npm i express mongojs mongoose cors
```

# Day 5

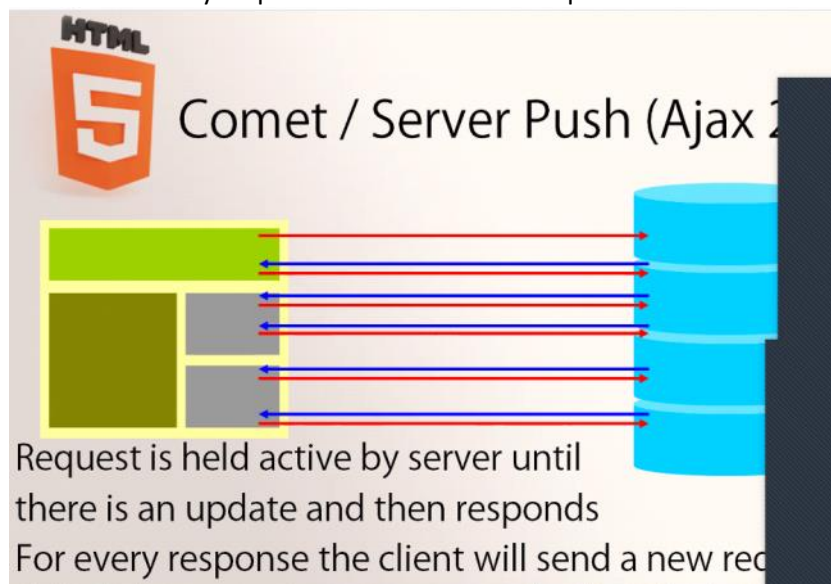
Tuesday, December 15, 2020 2:20 PM

client-sessions -> npm library for session management along with express

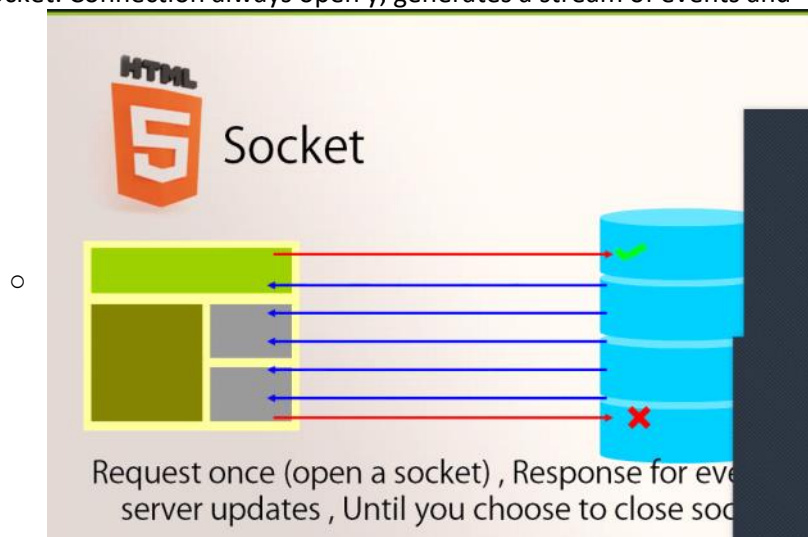
bcryptjs -> for hashing passwords // once hashed original string can never be retained

## Web Sockets

- Follows ws protocol
- Whatwg.org - for all html 5 documentaion
- Socket API provided by HTML5
- Comet: For every response there is another request as shown below



- Socket: Connection always open y, generates a stream of events and





## Socket Methods

```
var ws = new WebSocket("url")
○ ws.send("message");
ws.close(); terminate the socket connection
-----
ws.onopen = openFun;
ws.onclose = closeFun;
ws.onmessage = messageFun;
ws.onerror = errorFun;
```

- Socket.IO for web sockets in node
  - npm i socket.io --save
  - as express is not designed to work with ws, we will use http for handling ws
  - In socket.io
    - we need socket.io.js from client side dist
    - To establish socket connection with server on client side

```
io.connect('http://localhost:6060');
```
  - webrtc.jitsi should be used for media conferences link zoom
    - <https://jitsi.org/blog/a-simple-congestion-test-for-zoom/>
- Webpack
  - npm i webpack webpack-cli --save-dev
  - package.json start: "webpack"