

Variables

Variables are containers of data which we can access anywhere in the program

Value Types

Number	Numeric Values
String	A sequence of characters enclosed within quotes
Boolean	True or False
Array	A special variable, which can hold more than one value.
Object	<i>Key and Value</i> pairs

String Methods

<code>.length</code>	returns the number of characters
<code>string.slice (x,y)</code>	returns a part of the string. x - index of the start item y - count of the last item
<code>Number (String)</code>	converts the string containing numeric values into Number datatype
<code>.toUpperCase()</code>	converts the string to uppercase
<code>.toLowerCase()</code>	converts the string to lowercase

Arrays

Declaring Array	<code>let arrayName = []</code>
Accessing Items	<code>arrayName[index]</code>
<code>.length</code>	returns the length of the array
<code>.push (data)</code>	Adds a new item to the end of the array <code>arrayName.push("data")</code>
<code>.pop()</code>	Removes the last item <code>arrayName.pop()</code>
<code>.unshift (data)</code>	Adds a new item to start of the array <code>arrayName.unshift(data)</code>
<code>.shift()</code>	Removes the first item of the array <code>arrayName.shift()</code>
<code>.join (separator)</code>	Converts the array into string with the separator
<code>.slice (x,y)</code>	Gets a part of the array x - index of the start item y - count of the last item
<code>.reverse()</code>	reverse the array items <code>arrayName.reverse()</code>
<code>.sort()</code>	sorts the array items consisting only strings
<code>.sort((a,b) => a-b)</code>	sorts the array items of numbers in ascending order
<code>.sort((a,b) => b-a)</code>	sorts the array items of numbers in descending order



Node Modules

Node modules provide a way to re-use code in your Node application.

There are internal modules which come along with nodejs and also external modules which we can install using a package manager i.e npm, yarn

NPM

NPM NPM is a package manager for Node.js packages, or modules.

Initializing NPM `npm init -y`

Installing a module `npm install moduleName`

using the modules `const module = require("moduleName")`

NodeJS Basic App

Importing Express `const express = require("express");`

Invoking Express `const app = express();`

express urlencoding `app.use(express.urlencoded({extended: false}));`

using express's urlencoded `req.body`

Setting EJS `app.set("view engine", "ejs");`

Using EJS `res.render("fileName", {});`

Dotenv Module It is a module used to get access to the environment variables from the .env file
`npm install dotenv`

configuration of Dotenv `require("dotenv").config()`

Basic Mongoose Connection

```
const mongoose = require("mongoose");
mongoose.connect(connection string, {useNewUrlParser: true});
const Schema = new mongoose.Schema({
  field1: { type: String, required: true },
  field2: { type: Number, required: true }
});
module.exports = mongoose.model(collection name, Schema )
```

Basic NodeJS App

```
const express = require("express");
const app = express();
const bodyParser = require("body-parser");
app.use(bodyParser.urlencoded({extended: true}));
app.use(express.static("public"));
// Creating a folder to have static files like css, js
```



Javascript Fetch

```
function callAPI() {
  async function getData() {
    let response = await fetch(url, options);
    let data = response.json();
    return data;
  }
  getData().then(res => {
    console.log(res);
  })
}
```

Using APIs

```
import https from "https"

https.request(url, options, callback() {})

callback(res) {
  let resData = ""
  res.on("data", (chunks) => {
    resData += chunks
  })
  res.on("end", () => {
    console.log("The data from the API: " + JSON.parse(resData))
  })
}
```

URL Path

" /root/ :id "	refers to a url with the value in place of id will be stored in a object
req.body.params.id	access the parameter of the URL Path
req.body.query paramName	used to access the path parameters

Functions

A Function is a small block of code which performs a specific task. We can create a new Function using **function** keyword.

```
function myFunc() { // code to be executed }
```

The code inside the function can be executed by calling the function anywhere in the program.

Parameters are the variables defined with the function, but their values are assigned when they are invoked(called).

```
myfunc(4)
```

The return statement stops the execution of a function and returns a value.

```
function myFunc (parameters) { return VALUES }
```

conditional statements

Conditional statements are used to perform different actions based on different conditions.

There are three types of conditional statements



conditional statements (cont)

```
if (condition1) {
  // code to be executed if the condition1 is true
} else if (condition2) {
  // code to be executed if the condition1 is true
} else if (condition3) {
  // code to be executed if the condition1 is true
} else {
  // code to be executed if all the conditions are false
}
```

Finding HTML Elements

document	The whole html document
getElementById(<i>ID</i>)	Gets the element with the specified ID
getElementsByName(<i>className</i>)	Gets the elements with the Specified CLASS
getElementsByTagName(<i>TagName</i>)	Gets the specified Tag elements
querySelector(<i>Query</i>)	Returns the first element which satisfy the query
querySelectorAll(<i>Query</i>)	Returns all the elements which satisfy the query

NodeJS

NodeJS is a open-source cross-platform runtime environment for backend JavaScript

Install NodeJS

Mongoose

Mongoose is an Object Document Mapper (ODM) that is mostly used when the Node.JS server is integrated with the MongoDB database

Requiring Mongoose:

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

Connecting to MongoDB Database:

```
mongoose.connect(connection string, {useNewUrlParser: true})
```

A Schema in mongoose or mongoose is a structure for the data validation. mongoose will return a error if the data is not the same

Schema and Model:

```
const Schema = mongoose.Schema(Fields and their info)
const model = mongoose.model("collectionName", Schema)
```

Using the model variable, we can perform all the CRUD Operations

EJS (Embedded JavaScript templating)

EJS is a simple templating language that lets you generate HTML markup with plain JavaScript, where we can substitute values from JavaScript

Setting EJS

```
app.set("view engine", "ejs");
```

Using EJS

```
res.render("fileName", {});
```

EJS Website



bcrypt

Module	It is used for salting and hashing the passwords
installing bcrypt	<code>npm install bcrypt</code>
requiring bcrypt	<code>const bcrypt = require("bcrypt")</code>
hashing password	<code>bcrypt.hash("password", hashing rounds).then (hash => { //code })</code>
comparing password with hash	<code>bcrypt.compare("password", hash).then (result => {})</code>

Passport.js and Express-Session

Passport is authentication middleware for Node.js. Extremely flexible and modular, Passport can be unobtrusively dropped in to any Express-based web application

Using Passport.js, Express-Session and as well as passport-local as a strategy we can login a user and maintain the login user

Basis Passport.js, Express-Session Code

```
const app = require("express");
const router = express.Router();
const localStrategy = require("passport-local");
const passport = require("passport");
const session = require("express-session");
const mongoose = require("mongoose");
const users = require("../models/user.js");

router.use(session({
  secret: "some secret",
  store: { mongoose.connect(mongoose.connectionString, {
    timeout: 24 * 3600
  }) },
  cookie: {
    maxAge: 24 * 60 * 60 * 1000
  },
  resave: false,
  saveUninitialized: true
}));

router.use(passport.initialize());
router.use(passport.session());
passport.serializeUser((user, done) => {
```



Basis Passport.js, Express-Session Code (cont)

```
> done(null, user);
}
)
passport.deserializeUser(
  (user, null) => {
    users.find({_id: user._id}).then(data => {
      done(null, data);
    })
  }
)
passport.use( new localStrategy(
  (username, password, done) => {
    users.find({username: username}).then( data => {
      if (data.password == password) {
        done(null, data);
      } else if (data.password != password) {
        done(null, false);
      }
    })
  }
)
router.post("/login", (req, res, next) => {
  if (req.isAuthenticated == true) { res.redirect("/") }
  else if ( req.isAuthenticated == false) { return next(); }
})
```

Note

Add `typekey` to the **package.json** to use `import` method to import modules

Dotenv

Dotenv Module It is a module used to get access to the environment variables from the `.env` file

Installation:

```
npm install dotenv
```

Configuration of Dotenv

```
require('dotenv').config()
```



By **Web Dev**

cheatography.com/web-dev/

Published 28th May, 2023.

Last updated 9th December, 2023.

Page 6 of 6.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>