Cheatography

JavaScript Cheat Sheet

by Web Dev via cheatography.com/186440/cs/38966/

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	*Key* and *Value* pairs

String Methods

`.length`	returns the number of characters
`string.sl ice (x,y)`	returns a part of the string. **x - index of the start item** **y - count of the last item**
`Number(String)`	converts the string containing numeric values into Number datatype
`.toUpp erC ase()`	converts the string to uppercase
`.toLow erC ase()`	converts the string to lowercase

Arrays

Declaring Array	`let arrayName = []`
Accessing Items	`arrayName[index]`
`.length`	returns the length of the array
`.push(data)`	Adds a new item to the end of the array `arrayN ame.pu sh( " dat a")`
`.pop()`	Removes the last item `arrayn ame.pop()`
`.unshift(data)`	Adds a new item to start of the array `arrayN ame.un shift(data)`
`.shift()`	Removes the first item of the array `arrayN ame.sh ift()`
`.join(seperator)`	Converts the array into string with the seperator
`.slice (x,y)`	Gets a part of the array ***x - index of the start item y - count of the last item***
`.reverse()`	reverse the array items `arrayN ame.re verse()`
`.sort()`	sorts the array items consisting only strings
`.sort( (a,b) => a-b)`	sorts the array items of numbers in ascending order
`.sort( (a,b) => b-a)`	sorts the array items of numbers in decending 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 = requir e("N PMM odu leN ame ")`

NodeJS Basic App

Importing Express	`const express = requir e("e xpr ess ");`
Invoking Express	`const app = express();`
express urlencoding	`app.us e(e xpr ess.ur len cod ed( {ex tended: false}));`
using express's urlencoded	`req.body`
Setting EJS	`app.se t("view engine ", " ejs ");`
Using EJS	`res.re nde r("f ile Nam e", {});`
Dotenv Module	It is a module used to get access to the environment variables from the .env file `npm install dotenv`
configuration of Dotenv	`requir e("d ote nv").co nfig()`

Basic Mongoose Connection

```
const mongoose = require("mongoose");
mongoo se.c on nect(connection string, {useNe wUr lPa rser: true});
const Schema = new mongoo se.S ch ema({
    field1: { type: String, required: true },
    field2: { type: Number, required: true }
})
module.ex ports = mongoo se.m odel('collection name', Schema )
```

Basic NodeJS App

```
const express = require("express");
const app = express();
const bodyParser = requir e("b ody -pa rse r");
app.us e(b ody Par ser.ur len coded({ exteded : true });
app.us e(e xpr ess.st ati c("p ubl ic")); //making a folder to have static files like css, js
```

Javascript Fetch

```
function callAPI() {
    async function getData() {
       let response = await fetch(url, options);
       let data = respon se.j son();
        return data;
   }
    get Dat a().th en(res => {
        con sol e.l og( res);
   })
}
```

Using APIs

```
import https from " htt ps"
```

```
https.r eq uest(url, options, callback() {})
```

```
callba ck(res) {
      let resData = ""
      res.on ("da ta", (chuncks) =>
      resData += chunks
      })
      res.on ("en d", () =>
            consol e.l og( "The data from the API: " + JSON.parse(resData))
      })
}
```

URL Path

`" /ro ot/ :id "`	refers to a url with the value in place of id will be stored in a object
`req.bo dy.p ar ams.id`	access the parameter of the URL Path
`req.bo dy.q ue ry.p Name`	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 (pa ram eters) { return VALUES }
```

conditional statements

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

There are there types of conditional statements

conditional statements (cont)

```
if (condi tion1) {
 // code to be executed if the condition1 is true
 } else if (condi tion2) {
 // code to be executed if the condition1 is true
 } else if (condi tion3) {
 // code to be executed if the condition1 is true
 } else {
 // code to be executed if all the conditions are false
 }
```

Finding HTML Elments

`document`	The whole html document
`getElementById(ID)`	Gets the element with the specified ID
`getEle men tBy Cla ssName(class)`	Gets the elements with the Specified CLASS
`getEle men tBy Tag Name(Tag)`	Gets the specified Tag elements
`queryS ele ctor(Query)`	Returns the first element which satisfy the query
`queryS ele cto rAll(Query)`	Returns all the elements which satisfy the query

NodeJS

NodeJS is a open-source cross-platform runtime 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 = requir e("m ong oos e");
```

Connecting to MongoDB Database:

```
mongoo se.c on nect(connection string, {useNe wUr lPa rser: true})
```

A Schema in mongoDB 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 = mongoo se.S chema(Fields and their info)
const model = mongoo se.m od el( " col lec tio nNa me", 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.se  t( "view engine  ", "  ej s ");
```

Using EJS

```
res.re  nd e r ("f  ile  Na m e ", {});
```

EJS Website

bcrypt

Module	It is used for salting and hashing the passwords
installing bcrypt	`npm install bcrypt`
requiring bcrypt	`const bcrypr = requir e("b cry pt")`
hashing password	`bcrypt.hash(" pas swo rd", hashing rounds).then (hash = { //code })`
comparing password with hash	`bcrypt.co mpare(" pas swo rd", hash).then (result => {})`

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

Bacis Passport.js, Express-Session Code

```
const app = require("express");
const router = expres s.R out er();
const localS trategy = requir e("m ong oos e-l oca l");
const passport = requir e("p ass por t");
const session = requir e("e xpr ess -se ssi on");
const MongoStore = requir e("c onn ect -mo ngo ")
const users = requir e("./ mo del s/m ong o-u ser.js ")
    rou ter.us e(s ess ion({
    secret: "some secret ",
    store: { MongoS tor e.c reate({
      mon goUrl: MongoDB Connection String,
      tou chA fter: 24 * 3600
   },
    cookie: {
      maxAge: 24 60 60 1000
   },
    resave: false,
    sav eUn ini tia lized: true
})
router.us e(p ass por t.i nit ial ize());
router.us e(p ass por t.s ess ion());
passpo rt.s er ial ize User(
    (user, done) => {
```

Bacis 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.isUnauthenticated == true) { return next(); }
)
```

Note

Add `type`key 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

```
requir  e( " d o te  nv").co  nfig()
```