Web Development Cheatsheet

Abyan Majid, 2023

JavaScript

Concept	Code Snippet

If-Else If-Else	`if (condition1) {`
	`    // code to be executed if condition1 is true`
	`} else if (condition2) {`
	`    // code to be executed if condition2 is true`
	`} else {`
	`    // code to be executed if both condition1 and condition2 are false`
	`}`

Switch	`switch (expression) {`
	`    case value1:`
	`        // code to be executed when expression equals value1`
	`        break;`
	`    case value2:`
	`        // code to be executed when expression equals value2`
	`        break;`
	`    // include additional cases as needed`
	`    default:`
	`        // code to be executed if expression doesn't match any case`
	`}`

While	`while (condition) {`
	`    // code to be executed while the condition is true`
	`}`

Do-While	`do {`
	`    // code to be ran at least once and then while the condition is true`
	`} while (condition);`

For	`// For loop`
	`for (initialization; condition; increment) {`
	`    // code to be executed for each iteration`
	`}`

For-In	`// For-in loop; used for iterating over properties of an object`
	`const object = { a: 1, b: 2, c: 3 };`
	`for (const key in object) {`
	`    console.log(key + ' = ' + object[key]);`
	`}`

For-Of	`// For-of loop; use for iterating over a collection`
	`const array = [1, 2, 3, 4, 5];`
	`for (const value of array) {`
	`    console.log(value);`
	`}`

Object	`const objectName = {`
	`    property1: value1,`
	`    property2: value2,`

```
    method1: function() {
        // code for method1
    }
};
```

Function

```
function functionName(parameter1, parameter2) {
    // function body
    return result;
}
```

Arrow Function

```
const functionName = (parameter1, parameter2) => {
    // function body
    return result;
};
```

Class

```
class ClassName {
  constructor(a, b, c) {
    this.a = a;
    this.b = b;
    this.c = c;
  }
  some_method() {
    // do stuff
    return result
  }
}

const newInstance = new ClassName(1, 2, 3)
```

Inheritance (`extends`, `super`)

```
class Pet {
    constructor(name, age) {
        this.name = name;
        this.age = age;
    }
    eat() {
        return `${this.name} is eating!`
    }
}

class Cat extends Pet {
    // Using `super` to prevent definition of variables already present in
parent
    constructor(name, age, livesLeft = 9) {
        super(name, age)
        this.livesLeft = livesLeft;
    }
    meow() {
        return "MEOW!"
    }
}

class Dog extends Pet {
    // Not adding any new variables, so no need to define constructor
    bark() {
        return "WOOF!"
    }
}
```

Destructuring Objects

```
const person = {
    name: 'John Doe',
    age: 30,
    address: {
        street: '123 Main St',
        city: 'Anytown'
    }
};

// Destructuring properties
const { name, age } = person;

// Destructuring nested properties
const { address: { street, city } } = person;
```

Destructuring Arrays

```
const colors = ['red', 'green', 'blue'];

// Destructuring elements
const [firstColor, secondColor] = colors;

// Skip elements
const [ , , thirdColor] = colors;
```

Destructuring Function Parameters

```
function displayUser({ name, age }) {
    console.log(`Name: ${name}, Age: ${age}`);
}

const user = { name: 'Johan', age: 25 };
displayUser(user); // Name: Alice, Age: 25
```

Destructuring from `this`

```
class ClassName {
  constructor(a, b, c) {
    this.a = a;
    this.b = b;
    this.c = c;
  }
  some_method() {
    const { a, b, c } = this;
  }
}
```

DOM GET-Element Methods

```
// GET element by ID
const elementById = document.getElementById('id');

// GET element by CLASS
const elementsByClassName = document.getElementsByClassName('class');

// GET element by TAG NAME
const elementsByTagName = document.getElementsByTagName('tag');
```

DOM Query Selector

```
// Single element query selector
const element = document.querySelector('.class or #id or tag');

// Multiple elements query selector
const elements = document.querySelectorAll('.class, #id, tag');
```

Create New Promise	`const promise = new Promise((resolve, reject) => {`
	`    if (/* condition */) {`
	`        resolve('Success');`
	`    } else {`
	`        reject('Error');`
	`    }`
	`});`

Promises with Then-Catch	`promise`
	`    .then((result) => {`
	`        // do stuff`
	`    })`
	`    .catch((error) => {`
	`        // do stuff`
	`    });`

Promises with Async Function	`const asyncFunction = async () => {`
	`    try {`
	`        await <code> // use await to wait for this line to finish`
	`        // do stuff`
	`    } catch (error) {`
	`        // do stuff`
	`    }`
	`};`

NodeJS	
Concept	**Code Snippet**

Requiring Files (module.exports)	`// MODULE.JS`
	`const PI = 3.14`
	`const E = 2.72`
	
	`const math = {`
	`    PI: PI,`
	`    E: E,`
	`}`
	
	`module.exports = math;`
	
	`// INDEX.JS`
	`const math = require("./math")`
	`console.log(math.PI)`
	`console.log(math.E)`

Requiring a Directory	`// INDEX.JS`
	`const module1 = require("./module1")`
	`const module2 = require("./module2")`
	`const modules = [module1, module2]`
	`module.exports = modules`
	
	`// APP.JS`
	`const modules = require("./modules")`
	`console.log("REQUIRED DIRECTORY:", modules)`

(NPM) Install package	`// Install locally to project directory`
	`$ npm install <packageName>`
	`or simply, $ npm i <packageName>`
	
	`// Install globally to PC (cannot be `required` in your project directory unless you link it!)`
	`$ npm i -g <packageName>`
	
	`// CD to project directory, then link a globally installed package like so:`
	`$ npm link <packageName>`

(NPM) Install all dependencies from package.json	`$ npm install`

(NPM) Create `package.json`	`$ npm init`
	
	`// Skip all prompts`
	`$ npm init -y`

Express	
Concept	**Code Snippet**

Relevant Installations	`$ npm i express`
	`$ npm i ejs`
	`$ npm i uuid`
	`$ npm i method-override`
	`$ npm i mongoose`
	
	`// or install all of them simultaneously`
	`$ npm i express ejs uuid method-override mongoose`

Auto-Restart with Nodemon	`$ nodemon -L <filename>.js`

Starter Template	`const express = require("express");`
	`const app = express();`
	`const path = require("path");`
	`const mongoose = require("mongoose");`
	
	`const PORT = 8080;`
	
	`mongoose.connect('mongodb://127.0.0.1:27017/<dbName>')`
	`.then(() => {`
	`    console.log("[*] Connected Mongoose to MongoDB!")`
	`})`
	`.catch(err => {`
	`    console.log("[*] Failed to connect Mongoose to MongoDB!")`
	`    console.log(err)`
	`})`
	
	`app.use(express.urlencoded({ extended: true }));`
	`app.use(express.static(path.join(__dirname, "/public")))`

```
app.set("view engine", "ejs");
app.set("views", path.join(__dirname, "/views"));

app.get("/", (req, res) => {
  context = {}
  res.render("home", context);
});

app.listen(PORT, () => {
  console.log(`[*] App hosted at: localhost:${PORT}`);
});
```

Path Parameter

```
app.get("/:sub/:subsub", (req, res) => {
    const {sub, subsub} = req.params
    res.send(`/:${sub}/:${subsub}`);
});
```

Default GET Content for Every Other URL

```
// NEEDS TO BE THE LAST '.GET' CALLBACK; BOTTOM!
app.get("*", (req, res) => {
    res.send("404: Page Not Found!");
});
```

Middleware for request.body (URLENCODED)

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

Middleware for request.body (JSON)

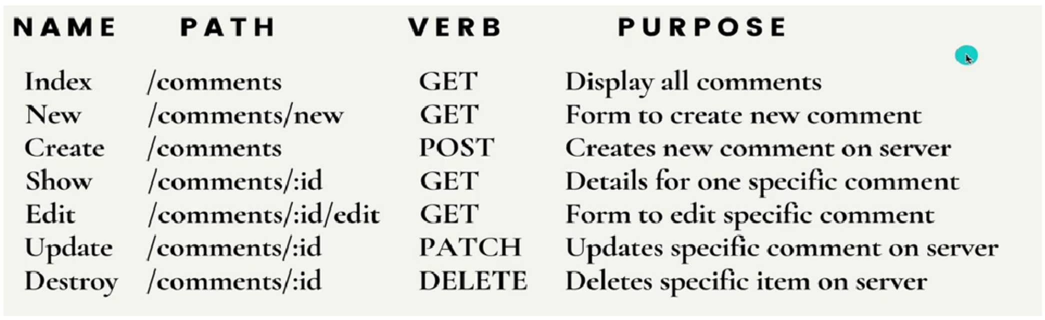
```
app.use(express.json());
```

POST Request

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

app.post("/path", (req, res) => {
  console.log(req.body);
  res.send("POST /path response!");
});
```

RESTful Routing - Comments Example



NAME	PATH	VERB	PURPOSE
Index	/comments	GET	Display all comments
New	/comments/new	GET	Form to create new comment
Create	/comments	POST	Creates new comment on server
Show	/comments/:id	GET	Details for one specific comment
Edit	/comments/:id/edit	GET	Form to edit specific comment
Update	/comments/:id	PATCH	Updates specific comment on server
Destroy	/comments/:id	DELETE	Deletes specific item on server

Redirect

```
app.post("/path", (req, res) => {
  // do stuff
  res.redirect("/path")
});
```

Unique Identifier (UUID Package)

```
const { v4: uuid } = require("uuid");

// call uuid() whenever you want to create a unique identifier
uuid()
```

Method Override	`const methodOverride = require("method-override");`
	`app.use(methodOverride("_method"))`
	
	`// ejsFileName.ejs`
	`<form method="post" action="/path?_method=PATCH">`