AJAX

Fetch API

Let & Const keyword in JavaScript

Arrow Functions

Asynchronous JavaScript (async/await examples)

Timeout functions

Promises

JSON (Working with JSON)

Revisit JavaScript imp Topics

* JavaScript is a High Level, Multi paradigm, Single Threaded Scripting Language
* JS is a Programming Language of Web in the initial days
* After the introduction of Node (JS Runtime) Java Script code can be used for any purpose.
* Mainly JS was used for DOM manipulation
* JS is a case sensitive programming Language
* It syntax is little similar to Java programming lang, other than that there is no relations between Java and Javascript.
* Official Name of JavaScript Lang is ECMA Script. In Short it is called as ES.
* ECMA – Electronic Computer Manufacturing Association
* Java Script also contains keywords, operators, datatypes, variables, constants
* JavaScript 7 data types are boolean, number, string, null, undefined, object & symbol
* JavaScript is a dynamically typed or loosely typed language. (Meaning variable can hold any values even after declaration)
* “var” keyword is used to declare a variable.
* Variable scope – Global, Local, Functional, Block, Lexical
* In Javascript all the variable declarations code will be moved to top automatically. So if you assign a value to a variable first and then declare it. It’s perfectly alright in javascript. This is called Variable Hoisting.
* Type Coercion – Converting from one data type to another data type. [Implicit (automatic) '3' \* '2', 2/’5', 123 + '' & Explicit (Manual) Number('3'), String (123), Boolean(2)]
* Functions – Block of code with a name or without a name. – used to perform some task on parameters or arguments.
* Types of Functions – Normal Functions/ Anonymous or Functional Expressions / IIFE (Immediately invoked Function Expressions) or Self Invoked Functions / Asynchronous or callback functions
* Generally functions will have a name and normal brackets. For Ex : displayResult() //It’s a simple function
* IIFE or Self Invoked Function : syntax --- (function(){ code goes here...})();
* Callback functions – Function passed as an argument to another function. Usually used in async functions.
* Closure – Is a function which remembers it’s arguments and parameters of it’s parent function even after the function return.
* Encapsulation – hiding information or data (In JS Encapsulation is done using closure)

**Example:**

const Book = function(t, a) {

let title = t;

let author = a;

return {

summary : function() {

console.log(`${title} written by ${author}.`);

}

}

}

const book1 = new Book('Hippie', 'Paulo Coelho');

book1.summary(); // Returns Hippie written by Paulo Coelho.

* In JS, Objects are prototype based and they are not Class based.
* In JS, inheritance is achieved using prototype Chains

Example :

let animal = {

eats: true

walk() {

alert("Animal walk");

}

};

let rabbit = {

jumps: true

\_\_proto\_\_: animal // sets animal to be a prototype of rabbit.

};

// we can find both properties in rabbit now:

alert( rabbit.eats ); // true

alert( rabbit.jumps ); // true

// walk is taken from the prototype

rabbit.walk(); // Animal walk

* In JS, we don’t have access modifiers & behaviour (Non-access modifiers)
* Array Types – Literal Array (using [] ) Object Array (Using new keyword)

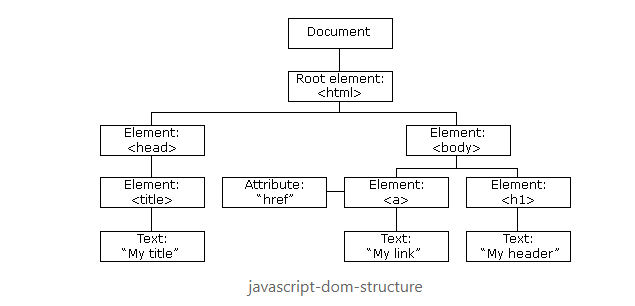
// array literal

let cheeses = ['bleu', 'cheddar', 'parmesan', 'brie'];

// new keyword

let primes = new Array(2, 3, 5, 7, 11, 13);

* In JS array, there is a property called “length” which gives the size of the array
* In JS array, there are few functions called push(), pop(),splice(), shift(), forEach(), sort(), filter(), map() & reduce()
* Timing Events – setTimeOut(), setInterval()
* Error Handling in JS – Runtime Error & User-Defined or Custom Error
* Try .. catch block, throw keyword
* In ES6, class, extends, static keywords are available
* JS was mainly used for client side validation, DOM manipulation
* JS can change HTML attributes dynamically (Changes at run time)
* JS can change html elements dynamically (Adds/Removes HTML elements)
* JS can change CSS attributes dynamically
* Generally JS gives life to the Web Page/ Web site
* JS is used for User interaction.
* DOM – Document Object Model
* DOM – It’s a tree like representation of each HTML file (document)
* DOM tree has one root element called html and two child branches called head & body.
* Both head & body are siblings to each other because they belongs to same parent



* DOM Manipulation Methods – cloneNode(), appendChild(), removeChild(), replaceChild(),createElement(), innerHTML,innerText, setAttribute(attribute\_name, attribute\_value), getAttribute(attribute\_name)
* DOM element selection methods – getElementById(), getElementsByName(), getElementsByTagName(), getElementsByClassName()
* This (alone – refers to global object) (in function refers to Global object -Window) (in class refers to current object)
* Event refers to user interaction
* Types of Events – Keyboard Events, Mouse Events, Page Events, Form Events
* Bubbling – Event propagation from bottom to top
* Capturing – Event Propagation from top to bottom (Needs to be added to an eventListener)
* JSON – JavaScript Object Notation [Representing the data in JS Object form]
* In JSON, {} – represents objects, [] – represents array,
* In JSON, the data will be stored in Key, value pairs

AJAX = Asynchronous Javascript And XML

AJAX is the combination of two or more technologies.

Why AJAX?

Normally during data/form submission client is completely blocked. During this blocked time can’t perform any operation in the client side.

When client sends data to the server, server sends a response back to the client.

During this time page reload will happen.

If you like to share the partial data to the server without reloading the page & also blocking the client opeations, then AJAX is the solution for this.

<https://www.javatpoint.com/ajax-tutorial>

JSP = Java Code embedded inside the HTML

Servlet = HTML code embedded inside the JAVA code.

INSERT INTO `ers`.`user\_table` (`username`, `password`, `role`) VALUES ('user123', 'user123', 'Employee');

INSERT INTO `ers`.`user\_table` (`username`, `password`, `role`) VALUES ('superuser', 'superuser', 'Finance Manager');

CREATE TABLE `ers`.`profile` (

`id` INT NOT NULL AUTO\_INCREMENT,

`userid` INT NOT NULL,

`first\_name` VARCHAR(75) NULL,

`last\_name` VARCHAR(75) NOT NULL,

`email` VARCHAR(155) NOT NULL,

`phone` BIGINT(10) NULL,

`address` VARCHAR(255) NULL,

PRIMARY KEY (`id`),

INDEX `profile\_fk1\_idx` (`userid` ASC) VISIBLE,

CONSTRAINT `profile\_fk1`

FOREIGN KEY (`userid`)

REFERENCES `ers`.`user\_table` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION);

INSERT INTO `ers`.`profile` (`userid`, `first\_name`, `last\_name`, `email`, `phone`, `address`) VALUES ('1', 'sivakumar', 'os', 'sivakumar.os@gmail.com', '9944159844', 'Madurai, TN');

INSERT INTO `ers`.`profile` (`userid`, `first\_name`, `last\_name`, `email`, `phone`, `address`) VALUES ('2', 'AdminUser', 'ERS', 'ersadmin@gmail.com', '9878675676', 'Delhi, IN');

INSERT INTO `ers`.`profile` (`userid`, `first\_name`, `last\_name`, `email`, `phone`, `address`) VALUES ('3', 'normal', 'user', 'normalers@gmail.com', '8978675645', 'Mumbai,IN');

INSERT INTO `ers`.`profile` (`userid`, `first\_name`, `last\_name`, `email`, `phone`, `address`) VALUES ('4', 'superuser', 'ers', 'superuser@ers.com', '7898897898', 'Goa,IN');

CREATE TABLE `ers`.`reimbursement` (

`id` INT NOT NULL AUTO\_INCREMENT,

`amount` DECIMAL(10,2) NOT NULL,

`author` INT NOT NULL,

`resolver` INT NULL,

`status` VARCHAR(45) NULL,

`description` VARCHAR(255) NULL,

`creation\_date` DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP,

`resolution\_date` DATETIME NULL,

`receipt\_image` BLOB NULL,

PRIMARY KEY (`id`),

INDEX `reim\_fk1\_idx` (`author` ASC) VISIBLE,

INDEX `reim\_fk2\_idx` (`resolver` ASC) VISIBLE,

CONSTRAINT `reim\_fk1`

FOREIGN KEY (`author`)

REFERENCES `ers`.`user\_table` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `reim\_fk2`

FOREIGN KEY (`resolver`)

REFERENCES `ers`.`user\_table` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION);

INSERT INTO `ers`.`reimbursement` (`amount`, `author`, `status`, `description`) VALUES ('123.45', '1', 'pending', 'Travel Request');

INSERT INTO `ers`.`reimbursement` (`amount`, `author`, `status`, `description`) VALUES ('76.89', '3', 'pending', 'Team Lunch');

INSERT INTO `ers`.`reimbursement` (`amount`, `author`, `status`, `description`) VALUES ('250', '1', 'pending', 'Furniture');

INSERT INTO `ers`.`reimbursement` (`amount`, `author`, `status`, `description`) VALUES ('750', '1', 'pending', 'Internet');

MIME Type = Multi Purpose Internet Mail Extension

Files = File Extension (.txt, .doc, xls, .rtf, .mp3, .jpeg, .gif)

Synchronous & Asynchronous

Let’s assume the following tasks

1. Need to print a data 1000 times (50ms)
2. Need to fetch 5000 records from database table (700 ms)
3. Need to open a file and search for content in it (200 ms)
4. Need to get input from the user and find whether it is a palindrome or not. (500 ms to 2000ms)

Total time spent for all 4 operations in Synchronous programming = 50+700+200+1500 = 2450 ms

Synchronous operations – Executing each operation in sequence (One after Another)

Asynchronous Operations – Executing each operation in parallel using multiple cores of the processor or multiple thread.

Synchronous operations is also called as Blocking Operation. Bcos it blocks the program for a particular time period. No other operations are allowed during blocked time.

Promises, Observables & Arrow functions supports Async programming

Javascript is a Multi paradigm programming language because it supports Object Oriented Programming, Procedure Oriented Programming, Functional Programming, Asynchronous programming etc.,

Asynchronous operations is also called non-blocking operation which means the program continuously runs without any block.