JAVASCRIPT

Index

Basic JS

* JavaScript vs EcmaScript
* Install VS code and first program
* Hello World program
* Declare variable using var
* More about variable
* Let
* Const
* String Indexing
* Useful String Methods
* Template Strings
* Null, Undefined, Bigint and Typeof
* Booleans and Comparison Operatores
* Truthy and Falsy Values
* If Else statement
* Ternary Operator
* && || Operator
* Nested If else
* If elseif else
* Switch Statement
* While Loop & examples
* For loop & examples
* Break and Continue keyword
* Do while loop

Arrays in Javascript :-

* Intro to arrays
* Push pop shift unshift
* Primitive vs Reference Datatypes
* Clone Array & Spread Operator
* For loops
* Use const for creating arrays
* While loop in array
* For of loop
* For in loop
* Array Destructuring

Objects in JavaScript

* Intro to Objects
* Dot vs Bracket Notation
* Iterate Objects
* Computed Properties
* Spread operator in Objects
* Object Destructuring
* Object Inside Array
* Nested Destructring

Functions in Javascript

* function Declaration
* Function Expression
* Arrow Functions
* Function declarations are covered in great detail …later in the course
* Functions inside Functions
* Lexical Scop
* Block Scop vs Function Scop
* Default Parameters
* Rest Parameters
* Parameter Destructuring
* Very brief intro to call back functions … covered in great detail later in the course
* Functions returning Functions

Very Important Array Methods

* forEach method
* map method
* filter
* reduce
* sort
* find
* every
* some
* fill method
* splice method

More Useful Things

* Iterables
* Sets
* Maps
* Object Assign
* Optional Chaining

Object Oriented JavaScript / Prototypal Inheritance

* Methods
* This keyword, Window object
* Call , Apply and Bind Method
* Some Warnings
* This Inside Arrow Functions
* Short Syntax for methods
* Factory functions& discuss some memory related problems
* First Solution to that Problem
* Why that solution is not that great
* Create functions to create multiple objects
* Store methods in different object
* Solution using object.create
* What is \_\_proto\_\_ ([prototype])
* What is Prototype
* Use prototype
* New Keyword
* Constructor function with new keyword
* More discussion about proto and prototype
* Class keyword & it’s example
* Extends keyword
* Super Keyword
* Method Overriding
* Getters and Setters
* Static Methods and Properties

How JavaScript Works

* Global Execution Context
* This and window in global execution context
* Hoisting
* Are let and const are hoisted? What is the reference error?
* Function Execution Context
* Scop Chain and Lexical Environment
* Intro to Closures
* Closure Examples – 1, 2, 3, 4

DOM Tutorial

* HTML nad CSS crash course
* Async vs defer
* Select elements using id
* querySelector
* textContent & innerText
* change the styles of elements using js
* get and set attributes
* select multiple elements and loop through them
* innerHTML
* Deeply understand the dom tree, root node, element nodes, text nodes
* classList
* Add new elements to page
* Create elements
* Insert adjacent elements
* Clone nodes
* More methods to add elements on page
* How to get the dimensions of the element
* Intro to Events
* This keyword inside eventListener callback
* Add events on multiple elements
* Event Object
* How event listener works
* Practice with events & create demo project
* More events
* Event Bubbling
* Event Capturing
* Event Delegation
* Create project using Event Delegation

Asynchronous JavaScript

* Is javascript is synchronous or asynchronous programming language?
* setTimeout()
* setimeout() with 0 miliseconds
* callback Queue
* setInterval and create small project with setinterval
* understand callbacks in general
* Callbacks in Asynchronous Programming
* Callback hell and Pyramid of doom
* Intro to Promises
* Microtask Queue
* Function that return promise
* Promise and settimeout
* Promise.resolve and more about their methods
* Convert nested callbacks to flat code using promises
* Intro to AJAX, HTTP request
* XHR requests
* Error hsndling in XHR requests
* XHR request chaining
* Promisfying XHR requests and caining using then method
* Fetch API
* Error handling in fetch API
* Continue promises with async and await
* Split code into multiple files using ES6 modules
* How JS works?

So first of all 2 trh ki cheeze ho skti he code ke sath.

Interpreter :- jo line by line code ko dekhta he nd line by line execute kr deta he.

Compiler:- Jo ek sath pehle code ko dekh leta he uske bad execute krta he.

To in JS code is

1st compiled, then 2nd code is executed.

Compilation phase me 3 stages hote he.

1. Tokening /Lexing :- isme code ko chote chote tukdo me toda jata he jisko bolte he tokens
2. Parsing :- vo tokens jo bne he usko smj ke Abstract Syntax Tree (AST) bnta he. Or AST se vo code executable format me convert hota he
3. Code Generation

* ES (EcmaScript) ke official documentation me khi pe bhi aesa nhi likha he ki JS ka code pehle compile ona chahiye. But usme aeas likha he ki code run hone se pehle usme 1st Early Error Checking honi chahiye jaise ki Syntax error, logical error etc. 2nd is Jitne bhi variables declared he uske scopes defined hone chahiye . To ye dono point se itna clear he ki code execute hone se pehle usko parse to krna hi pdega. Yes certainly different browser providers are using different methods for parsing nd then execution of code.
* So main conclusion ye he ki code ke execution se pehle compilation is needed. For fulfilling 2 purposes. 1) Early Error Checking and most important is 2) Determining Appropriate Scope for Variables. Remember ye compilation phase se JS me variable create nhi hua he abhi tk. Bt JS ko itni information jarus mil gyi he ki firstName variable globle scope me he to usko kha tk access kr skte he.

2nd Phase) Code Execution Phase :-

In JS code executes inside execution context. To code execute krne ke liye execution context create hota he. Aur first jo execution context context create hota he usko Globale execution context bolte he.

Watch that video for further and more information.

Note :- JS is Synchronous Programming language. Js is a single threaded language. It means pehle 1st line ka execution khtm ho jayega uske bad hi next line ka execution start hoga. And aese line by line execution hoga.

* Methods to link JS file to the html file. And which is the best method? (this is the M IMP question in JS interview).

Method – 1 )



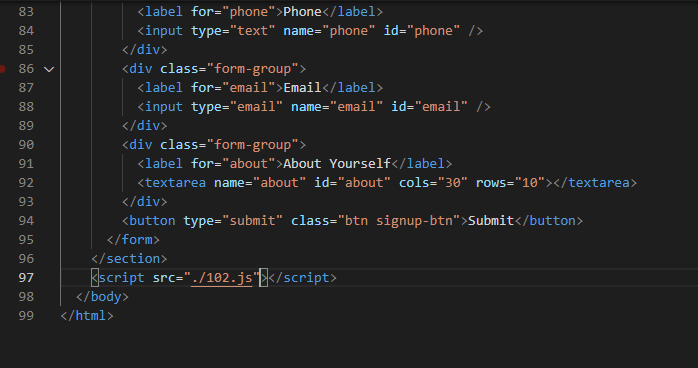
1st method is to write src = “file name” as the attribute of script tag inside the <head> tag as shown in above photo.

Backend operation (or what is the problem of using this method) :-

So ye html file bnne ke bad browser ke pas aayegi. As we know first of all browser will start parsing the file. To vo line by line parse krna start krega. Now parsing is going well till line 8. But jb browser line-8 pe aayega tb vha pe it came to know ki it has to load this JS file. To browser ne load krna start kr diya. Or load hote hi it will execute that JS file. To now JS file ka execution start ho gya. Bt hmne JS vali 102.js file ke liye <div>,<ul> etc etc sara code likha he jo line -9 ke bad likha he. Bt browser to parsing ke time me line-8 pe hi stop ho gya tha. To it doesn’t know about all these tags and code. Bt JS file ke execution me vo he. To yha pe execution me error aa jayegi and execution stop ho jayega.

Note :- agr parsing ke time koi ceeze read nhi hui he or vo execution ke time samne aati he then system(here browser) gives error nd execution will stop.

Method – 2)



2nd method is to write src=’filename’ as an attribute of script tag before just end of the <body> tag as shown in above image.

* Isme bhi same upr explain kiya us trh browser first parse whole HTML file line by line, and come to line – 97. It loads that JS file and execute JS file. Which will execute properly as browser already parsed all the code which came for execution later in JS file.
* But there is one problem in this method also.

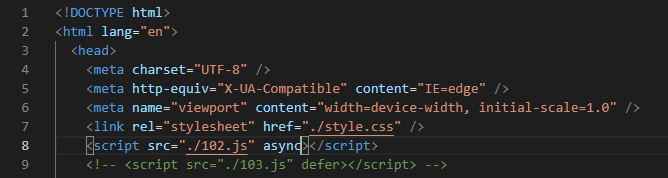
First browser 🡪parse HTML file 🡪 took some time

Then browser 🡪 Load JS files 🡪 took some time for loading

Then browser 🡪 execute JS file 🡪 took some time for execution.

* To here it looks like synchronization of process like one by one it’s done and taking time. So we can optimize performance.

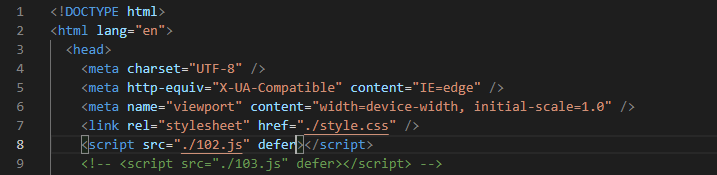
Method – 3)



3rd method is to jst add async attribute in 1st method as shown in above image. But what it will do?

* So same operation is performed. File started parsing. At line 8 browser will see file name and async. So async tells browser to not to stop parsing of next code from next line.
* So in this 3rd method browser will start loading that JS file and simultaneously continue to parsing for HTML file.
* But man lo ki parsing jb line no. 37 pe phochi tb tk JS file load ho gyi. To as file is loaded then execution of that file will start. And parsing of HTML file will stop here. To yha pe browser has parsed only half of the HTML code. And man lo ki line 37 ke bad vali lines ka koi code component JS file ke execution me aaya then browser will give error nd stop execution.

Method – 4)



4th method is to jst replace async word with defer word in 3rd method as shown in above image.

* To isme bhi same procedure hogi as above explained.
* First file parsing is started. When browser came to line – 8 , it will see file name and defer word.
* To defer word also tells browser to not to stop parsing and start loading of that JS file.

So now parsing and loading is simultaneously going on. Now suppose at line-37 loading is completed. But defer already told browser that after loading also don’t stop parsing.

* So JS file ka loading complete ho jane ke bad bhi execution start nhi hoga. Pehle HTML file ka parsing khtm ho jayega. Uske bad execution start hoga.
* And note here that is method se Loading ka separate time (as explained in 1st and 2nd method) bhi save ho jayega.
* Hence this method is better than other methods and provide optimization. So we will use this method.
* What is DOM ?

It stands for Document Object Model.

* Jb browser HTML file ko read krega tb background me ek tree jesa structure bnayega. Jisko ek object ke form me store krega. ( in key value pair. Kyuki object to key value pair hi store krta he n so). Isko ek document object bolte he.
* Abhi as we know ki JS file ke pas GEC ( Global Execution Context ) creation ke doran global memory environment me window object create hota he. And “this” keyword bhi vha pe hota he jo GEC creation time pe ye window object ko hi represent krta he.
* To browser upper bna hua Document Object, Window object ke ander put kr deta he. And abhi window object ke pas ye Document Object bhi he jisko hm log as a property fetch kr skte he or use kr skte he.