Aula 15.

Estrutura de dados:

Estrutura de dados é um um conjunto de regras e/ou critérios para organizar e administrar os dados.

Aula 18.

Como JavaScript é organizado ?

R: É organizado em sentenças de código, podendo ou não terminar em ponto e virgula, e também é organizado em blocos de códigos, que são delimitados por chaves.

Aula 29.

Arrays em JS:

Em JS um Array ele é heterogeneo e flexivel, ou seja você pode colocar variáveis de tipos diferenetes e alterar o tamanho do array.

Aula 30.

Json = JavaScript Object Notation:

Um Json e um Objeto em JS não são a mesma coisa.

Aula 35/36

Definições Var:

A variável var, dentro de um bloco de código (QUE NÃO SEJA UMA FUNÇÂO), está visível também para o código fora do bloco. Ou seja, quando você cria uma variável com var, fora de uma função ela será global.

Vantagem: Pode-se acessar a variável de qualquer lugar código.

Desvantagem: É perigo pois se não tomar cuidado você vai acabar sobrescrevendo a variável.

Aula 37.

Definições de let:

Diferentemente do var, o let só está visível dentro do ‘escopo’ que foi declarado, ou seja, se você declarou dentro de um bloco de código, mesmo que este não seja de uma função, a variável let não estará visível fora do bloco.

Aula 42.

Hoisting:

Variáveis com declaradas com var sofrem de Hoisting, ou seja “içamento” você pode usá-las antes mesmo de ter declarado elas, com let isso não ocorre.

Aula 44.

Par nome/valor:

Objetos são grupos aninhados de nome/valor.

Aula 71.

Operadores ternários:

Os operadores ternários são divididos em 3 partes, a primeira parte uma expressão lógica, então o operador “?” depois o valor se for verdadeiro e por ultimo valor se for falso.

Aula 81.

While:

Você usa o while quando for fazer uma estrutura de repetição com número Indeterminado de vezes. Ou seja um valor que você não possui. Enquanto o Do while você usa quando quer que o código repita pelo menos uma vez.

Aula 85.

Break/Continue:

Break e Continue são estruturas de desvio, eles agem em cima de estruturas de repetição( menos o if), os dois agem em cima do bloco de repetição mais próximo, caso você queira usar em um bloco que não seja o mais interno você precisará de um “rótuto”(nome da repetição).

Break, encerra as repetições.

Continue, encerra a repetição atual, e passa para a próxima.

Aula 86.

First Class Function:

É quando a linguagem trata função como se fosse um dado, ou seja você pode passar funções por parâmetro.

Aula 87.

Parametros:

Em js parâmetros são opcionais

Aula 90.

This:

Em JS this é sempre o objeto que está sendo referenciado naquele contexto de função, ou seja o ‘dono’ da função. Quando utilizado uma função normal o this pode variar dependendo do contexto em que foi chamado, mas isso muda se for uma arrow function, nesse caso o this será sempre o mesmo do contexto em que ele foi definido. (This Lexico).

Aula 98.

Callback:

Callback são funções que ativam quando acontece determinado evento.

Aula 102.

Tipos de Declaração:

A ‘function declaration’, pode ser chamada até mesmo antes de ter sido declarada, pois o compilador do JS da preferencia em executar as funções primeiro sempre, já a ‘function expression’, e a ‘name function expression’, só podem ser chamadas depois de declaradas no código.

Aula 104.

Closure:

Closure é o escopo criando uma função é declara. Esse escopo permite a função acessar e manipular variáveis externas a função.

Aula 109.

IIFE:

Imadiatly Invoked Function Expression, uma função que se auto invoca, é mais utilizada no browser para fugir do escopo global.

Aula 111;

Comparação de números em string:

Se fizermos a comparação entre nossas strings de número até o nove, ela mantém um comportamento normal. Como se estivéssemos comparando números.

Agora se compararmos os valores depois do nove, a comparação irá validar o primeiro número da cadeia numeral, validando sua ordem alfabética.

Aula 113.

O.O:

Linguagens não estruturadas -> linguagens antigas, geralmente com go to

Linguagens estruturadas -> linguagens atuais, normalmente com um fluxo de código mais limpo, e separada em blocos.

O foco no paradigma OO, é o objeto, ou seja antigamente funções levavam dados, já com o OO os dados podem carregar funções.

Princípios importantes:

Abstração -> Pegar um objeto do mundo real e conseguir traduzir para o sistema de maneira simplificada e otimizada para o sistema.

Encapsulamento -> Os detalhes de implementação escondidos, e mostrar somente o necessário.

Herança (prototype) -> Reutilização de código, herdar determinado atributo, princípio “é um”, mas é preferível usar o “tem um”.

Aula 117.

Notação Literal:

Na nova versão do EcmaScript 2015, foi – se retirado a duplicidade na hora de criar variáveis, e reduzida a maneira de criar funções, ou seja você pode definir a função direto:

Const obj = {

Variável,

Nome\_func() {}

}

Aula 120.

Herença 1:

Um objeto em JS tem uma referência ao seu protótipo (pai), e essa é a forma que o JS define herança, á partir da referência para um protótipo, ele vai apontar para um outro objeto e você pode acessar os atributos deles. O atributo prototype só está presente em funções.

Aula 127.

JsonVsObject:

Json é um arquivo textual que pode ser convertido em objeto, porem o mesmo só suporta dados, assim não podendo carregar consigo funções, enquanto um objeto também pode ser convertido em JSON, porém sem carregar suas funções caso tenha.

Aula 130.

Array:

Em JS array é uma estrutura de tamanho dinâmico e heterogênea, ou seja aceita diversos tipos de dados, mas a boa prática diz que é interessante trabalhar com arrays homogêneos. Em JS o tipo de dado do array é tratado como um objeto.

Aula 135.

Map:

O map serve para transformar os dados de um array, por exemplo se um array tem 6 elementos, o map vai gerar outro array de 6 elementos porém com os dados transformados. O map não transforma o arry atual e sim gera um novo array.

Aula 138.

Filter:

Percorre o array e retorna um subArray baseado em algum critério de filtro.

Aula 140.

Reduce:

O reduce, agrega os valores do array para um único valor, tendo 2 parâmetros o acumulador (resultado da iteração anterior) e o próximo valor.

Aula 145.

FlatMap:

Uma estrutura map juntamente com concat.

Aula 148.

NodeJS:

NodeJS é um ambiente de execução de JavaScript do lado do servidor.

I/O é sempre que você precisa ler algo de um arquivo, quando você vai fazer uma conexão com alguma rede.

O node é um RuntimeJS, que é formado por 2 partes, o V8 que é um interpretador JS de código aberto do google, e o LIB UV que trata de forma assíncrona tudo que diz respeito a I/O.

Aula 151.

Require:

Sistema de módulos para acessar dados exportados de outros módulos, evitar usar o caminho absoluto, pois pode dar erro quando for implementar em outro lugar. Se você não passar a extensão do arquivo, ele interpreta como .js

Aula 152

Exports:

O this e o exports são só uma mesma referência para o objeto que module.exports aponta, ou seja. Você não pode atribuir um novo objeto para exportar através do this e do exports.

Aula 153.

Package.json:

NPM – nome package manager

I – instal

--save – para salvar na pasta package.json

Aula 155.

Cache:

O require, guarda cache do objeto do node, ou seja se você estiver chamando um objeto que já tenha sido criado, ele retornará sempre a mesma instancia, mas isso pode ser driblado usando FactoryFunction.

Aula 157.

Entendendo o this:

Cuidado ao utilizar o this dentro de uma função, ele apontará para o escopo global this === global, porém fora dela ele apontará para module.exports this === module.exports, ou seja o valor de this muda de acordo com o lugar que ele é chamado.

Aula 162.

Framework:

É um arcabouço, é algo que estrutura sua aplicação, diferente de uma biblioteca. Um framework é algo que você constrói sua aplicação em cima, ele te da uma estrutura base e a partir dela você continua.

Aula 166.

API com Express:

Toda vez que estiver mexendo com um projeto que meche com conexão, é necessário declarar uma porta. Uma porta é como se fosse um processo dentro do computador, então cada aplicação que precisa fazer uma comunicação por rede, ela precisa abrir uma porta pata encontrar o comunicador, e cada porta é única por processo.

A porta 80 é a porta padrão quando se faz uma requisição http

Aula 182.

Promise:

Promise é uma estrutura utilizada para realizar operações assíncronas, apartir dela você pode obter uma resposta de sucesso ou de erro.

Aula 197.

Conceitos sobre a web:

A base da internet é o protocolo TCP/IP – um protocolo orientado a conexão e confiável, e em contrapartida vc tem o UDP, sendo um protocolo não orientado a conexão, porém mais leve mas não com tanta confiabilidade.

TCP/IP é baseado no padrão de layers:

* Camada Física -> Ethernet, Wifi / MAC Addres / Física (Hardware) e Enlace(Software)
* Internet -> Endereço IP / Conexão entre redes / Transferência de pacotes
* Transporte -> TCP (confiável), UDP (melhor esforço) / Comunicação Host a Host
* Aplicação -> HTTP, FTP, SMPTP / Comunicação processo a processo / Número da porta.

A camada de cima usa os serviços da camada de baixo, mas não tem uma dependência cíclica.

Conceito de porta -> um número associado á um processo.

Aula 198.

Protocolo HTTP:

Em uma requisição do tipo GET os parâmetros são enviados na própria url, e em outros métodos tipo o post os parâmetros são enviados no corpo da requisição, ou seja quando se tratar de senhas, login o POST se torna mais interessante, mas mesmo assim ainda não é seguro, para garantir a segurança você precisa da sua requisição executando em cima de um protocolo https que faz com que a comunicação entre a máquina cliente e a servidora seja criptografada.

Aula 201.

Server-Side vs Client Side:

A Aplicação Server-Side o conteúdo é gerado a partir do servidor, não apenas dados são enviados do servidor pro browser, mas o próprio HTML no caso é gerado no servidor e assim é renderizada no browser. Já em uma aplicação front-end (cliente side), é muito comum você ter o conceito de SPA(Single Page Aplicattion, uma aplicação de uma única página que quando renderizada, ele pega os JS e toda a parte visual, o html que é construído é gerado dinamicamente pelo próprio código JS, e não mais pelo servidor.), e a partir daí o servidor passa a ter serviços web (WEB Services) que geram dados, podendo ser XML ou JSON, assim toda a sua aplicação é gerada dinamicamente do lado do cliente, e os dados sendo consumidos do lado do servidor.

Aula 202.

Conteúdo Estático vs Conteúdo Dinâmico:

No ponto de vista server-side, um conteúdo é estático quando o servidor para renderizar uma página ele simplesmente lê os arquivos do disco, arquivos estáticos, que estão gravados lá, ele simplesmente serviu a página. E um conteúdo dinâmico é aquele conteúdo gerado a partir de um código, por exemplo usando informações do banco de dados para gerar um filtro personalizado, etc.

Aula 203.

HTML, CSS, JS:

Html – Estrutura

Css – Estilo

Js - Comprotamento

Aula 207.

Nuvem:

O conceito de nuvem veio a partir da necessidade de alocação de recurso de acordo com a demanda por exemplo, um aloja que tem seu pico de acessos no natal, compraria uma máquina preparada para aguentar os acessos nesse período do ano, mas nos demais períodos em que a loja não tivesse tanta necessidade de recursos de processamento, muitos recursos dessa máquina ficariam subutilizados, e ociosos, no sistema em nuvem, você paga pelo que você usa, podendo flexibilizar a quantidade de recursos quando necessário.

Tipos clássicos de Nuvens:

Nuvens de Infraestrutura ou Infra as a Service (IAAS)

Nuvens de Plataforma ou Platform as a Service

Nuvens de Software ou Software as a Service

Nuvens de Função ou Function as a Service

E tudo isso só é viável graças a virtualização