Javascript I: Fundamentos da Linguagem

Capacitar o aluno a compreender, manipular, abstrair e estruturar corretamente dados e seus respectivos tipos.

Evolução do Javascript

Apresentar resumidamente a evolução da linguagem, sua utilização e a ECMA262 e TC39

JavaScript ou JS é uma linguagem de programação que segue a especificação ECMAScript.
É uma linguagem de alto nivel, interpretada e multi-paradigma
Sintaxe de chaves, tipagem dinâmica, orientação a objetos baseada em protótipos e funções de primeira classe
Ao lado do HTML e CSS é uma das tecnologias core da Web, habilita paginas web interativas, por isso todos os navegadores possuem uma engine para executar JS.
Foi criada em 1995 pro Brendan Eich para ser incorporado no navegador da Netscape.
O primeiro nome foi Mocha, depois mudou para LiveScript e antes do lançamento foi mudado para JavaScript para aproveitar a popularidade do Java na época.
Em 1995 a Microsoft criou seu Internet Explorer dando inicio a guerra dos browsers e lançou o JScript.
Em Novembro de 1996, a Netscape enviou o JAvaScript para a ECMA International, com o intuito de padronizar o JavaScrip para todos o navegadores e tirar o monopólio da Microsoft
Em Junho de 1997 eles lançaram a primeira especificação da linguagem ECMAScript.
Em Junho de 1998 lançaram a ECMAScript 2.
Em Dezembro de 1999 a ECMAScript 3.
No entanto a popularidade do Internet Explorer chegou a 95% no inicio de 2000, fazendo que o JScript fosse o padrão de linguagem de script na Web.
Com a não colaboração da Microsoft, o ECMAScript 4 foi parado e por isso nunca foi lançado.
Em 2004 a Mozilla (sucessora da Netscape), lança o Firefox, que foi bem recebido pelo mercado.
Em 2005 eles entram na ECMA Internacional e participam do trabalho para a próxima versão do ECMAScript.
Em 2005, Jesse James Garret lança um artigo com o termo Ajax e descreve as tecnologias das quais o JavaScript era o sustentação, para criar web applications.
Algumas bibliotecas são lançadas, como jQuery. Prototype.
Em 2008 a Google lança o Chrome com a engine V8.
Em Julho de 2008, na conferencia de Oslo, esses novos integrantes da web se reunem e entram em acordo para a nova versão do ECMAScript, que foi lançada em 2009 como ECMAsScript 5.

Em 2009 foi lançado a plataforma Node.js, permitindo que o JavaScript se tornasse uma linguagem do lado do servidor também.
Em 2016 a versão ECMAScript 6 é lançada, com uma coleção extensiva de adições e refinamentos.
De 2016 até 2019, uma nova versão do padrão ECMAScript foi lançado por ano, tornando a linguagem madura.

A Engine do Javascript

Um motor JavaScript (JavaScript engine) é um programa de computador que executa um código JavaScript.

Os primeiros motores foram somente interpretadores, hoje todos o navegadores modernos utilizam a compilação just-in-time para melhorar a performance

Primeiro motor foi escrito por Brendan Eich para o navegador da Netscape, ele evoluiu para o SpiderMonkey do Firefox

O primeiro navegador moderno com o just-int-time foi o V8 do Chrome.

A apple desenvolveu o Nitro(JavaScriptCore) para o navegador safari

Em 2017 os navegadores adicionaram suporte ao WebAssembly. A JavaScript engine executa o código WebAssembly no mesmo lugar que o código JavaScript

Valores e Inferência

Em ciência da computação, o tipo de dado é uma combinação de valores e operações que uma variável pode executar, varia de acordo com a linguagem e o sistema operacional.

Sã utilizados para indicar ao compilador ou interpretador como o programador pensa em usar o dado.

Um tipo de dado limita os valores que uma expressão, variável ou função, podem pegar. Ele define as operações que podem ser feitas no dado, o significado do dado e a forma que ele será armazenado.

Tipos de dados comuns são: Inteiro, Ponto Flutuante, Caracter, String, Boolean.

Tipos de dados são utilizados dentro de sistemas de tipos. Existem diferentes sistemas de tipos porém os mais comuns são o de tipagem estática e o de tipagem dinâmica.

Tipagem estática é utilizado nas linguagens como C, C++, Java, nele o programador deve definir qual o tipo de dado que determinada variável irá armazenar.

`int numero = 45;` => numero será do tipo inteiro
`numero = “quarenta”` => vai dar erro de compilação pois a variável numero so pode receber tipos de dados inteiro

Tipagem dinâmica a responsabilidade de dizer o tipo de dado fica com o interpretador, é utilizado em linguagens como Python, JavaScript, Ruby.

`numero = 45` => numero será do tipo inteiro
`numero = “quarenta”` => numero será do tipo string

Declaração de variáveis

Antes da versão 6, o ECMAScript 2015, para se declarar uma variável no JavaScript utilizamos a palavra chave var

`var num = 45`
Um variável declarada com var ela pode ser utilizada no escopo de uma função, ou seja, no abre e fecha chaves de uma função

A ECMAScript 2015 introduziu duas novas palavras chaves para se declarar variáveis: const e let

`let num = 45;`
``const text = ‘Hello’``

A diferença de let e const com var é que os dois são do escopo de bloco, ou seja, o abre e fecha chaves de um if, for, função. Uma variável declarada com let e const não pode ser re-declarada dentro do mesmo escopo.

A diferença entre let e const é que uma variável declarada com let pode receber outro valor no decorrer do código, já const não pode ser retribuída depois da sua declaração.

Tipos de dados primitivos

Os tipos de dados primitivos do JavaScript são: Strings, Numbers, BigInts, Booleans, Null, Undefined

String

É a representação de textos, caracteres. Declarada utilizando aspas simples, aspas duplas ou crase.

Principais propriedades:
String.length => propriedade que retorna o tamanho da string

Numbers

É o tipo para números

Principais propriedades:
Number.NaN => valor especial que representa que não é um numero
Number.MAX_SAFE_INTEGER => valor do maior inteiro no JavaScript (2^53 - 1)
Number.MIN_SAFE_INTEGER => valor do menor inteiro (-(2^53 - 1))

BigInts

Tipo de dado que representa inteiros com precisão arbitrária, com ele podemos armazenar e operar inteiros gigantes que vão além do limite de Numbers, 2^53 - 1

Boolean

Tipo de dado para true e false.
Para operações booleanas possuímos outros valores considerados como false no JavaScript: 0, “”(string vazia), null, undefined, NaN(not a number)

Null

É o tipo que representa nenhum valor no JavaScript. Diferente do undefined ele deve ser atribuído a variável.

Undefined

É um valor primitivo que é atribuído automaticamente as variáveis que foram declaradas porém não receberam nenhum valor.

Tipos de dados não-primitivos

Object

A classe Object representa um dos tipos de dados do JavaScript. É utilizado para armazenar várias coleções de chave e valor e entidades mais complexas

Objetos podem ser declarados utilizando o constructor Object() ou a sintaxe literal (utilizando o abre e fecha chaves)

Todos os objetos no JavaScript são instâncias de Object, um objeto herda propriedades do Object.prototype.

```
let o = new Object()
let Person = {
        name: ‘React’
}
console.log(Person.name)

let User = {
        email: ‘user@email’,
        login: () => return ‘OK’
}
console.log(User.login())
```

Array

A classe **Array** do Javascript é um objeto que é usado na construção de arrays

Arrays são objetos tipo lista, não possuem tamanho e nem tipos fixos, podendo mudar durante a execução do programa

```
let fruits = [‘maça’, ‘banana’]
console.log(fruits.length)

let array = [123, ‘carro’, { name: ‘react’ }]
console.log(array)
console.log(array[2])
```

Funções

No JavaScript as funções são tratados como membros de primeira classe, ou seja, podem ser passadas como parâmetro para outras funções

Podemos declarar uma função utilizando a palavra chave function mais o nome da função

```
function minhaFuncao() {
        return ‘Hello’
}
minhaFuncao()
```

Ou podemos declarar uma função anônima utilizando a sintaxe da arrow function

```
const minhaFuncao = () => {  return ‘Hello’  }
minhaFuncao()
```

Podemos passar parâmetros para as nossas funções, e eles podem ser de qualquer tipo de dados

```
function calc(num1, num2) {
        return num1 + num2
}
calc(23,12)
```

```
function meuNome(name) {
```

```
        return “Ola ” + name
}
meuNome(‘React’)
```

Podemos passar funções como parâmetro para as nossas funções. Chamamos essas funções de callbacks, ou seja, funções que serão executadas após uma computação.

```
function print(num) {
        console.log(num)
}

function calc(func) {
        const num1 = 23
        const num2 = 12
        func(num1 + num2)
}

calc(print)
```

Fontes

[Documentação da ECMA]
[Javascript: Primitive Values & Object References]