JavaScript

Material feito por Eduardo Cruz Guedes (GitHub abaixo)

https://github.com/Educg550

Módulo A

Aula 1 (Conceitos Iniciais)

- Conceito Cliente X Servidor: todo usuário na internet através de dispositivos inteligentes é um cliente, que solicita informações de um ou mais servidores (exemplo: uso de navegadores)

- O JavaScript foi criado pensando no front-end (interfaces amigáveis para o uso de clientes e usuários comuns), mas também pode funcionar em servidores (back-end)

- Em um site, por exemplo, temos três conceitos básicos, o conteúdo (todas as informações, dados, arquivos, etc.), o design (layout, organização e o embelezamento gráfico) e a programação (interação dinâmica, segurança e infraestrutura), representados respectivamente nas tecnologias HTML, CSS e JavaScript

- As interações do usuário dentro de um site ficam muito mais dinâmicas e úteis quando o JavaScript é aplicado adequadamente

- Diversas empresas e sites dependem (parcial ou completamente) do JavaScript, por isso é muito importante entendê-la

Aula 2 (História do JavaScript dentro da Internet)

- JavaScript teve sua origem influenciada pelos primórdios da internet, aproximadamente em 1970, durante o período de Guerra Fria, conflito tecnológico e armamentista

- O presidente militar dos EUA decretou a criação de uma agência de pesquisa tecnológica, a DARPA, que teria como objetivo criar uma rede que armazenaria os dados confidenciais, caso um bombardeio acabasse com o computador central. Essa rede ficou conhecida como Arpanet, que aumentaria cada vez mais ao longo dos próximos anos, até que se consolidasse como a internet

- Em 1993, outras importantes tecnologias surgiram, são estas o HTML, o protocolo HTTP e a WorldWide Web (WWW), por Tim Berners-Lee e sua equipe. Além disso, para o funcionamento de todas essas tecnologias, um navegador foi desenvolvido nos EUA, o Mosaic, feito por Marc Andreesen, pertencente ao instituto de pesquisa NCSA (Centro Nacional de Aplicações de Super Computadores)

- Marc Andreesen, juntamente com James Clark, fundaram a empresa Netscape, com o intuito de aperfeiçoarem a navegação na Web, criando o navegador com o mesmo nome, baseado no Mosaic. Se tornou o navegador mais famoso da época

- Em 1995, Brendan Eich, pensando em inovar a navegação web, para complementar as tecnologias já existentes, começou a desenvolver uma linguagem que inicialmente se chamava Mocha (ou LiveScript), ao mesmo tempo em que a linguagem Java estava em ascensão na mídia. A Netscape então, decidiu seguir o sucesso feito pelo Java, nomeando sua nova linguagem como JavaScript

- Java e JavaScript, mesmo possuindo o mesmo nome como base, não tem qualquer relação direta, a não ser que ambos foram baseados na mesma linguagem, a linguagem C

- A Microsoft, no mesmo ano, começou a trabalhar no Internet Explorer, baseado também no Mosaic, assim como "copiou" a linguagem JavaScript, agora denominada por eles como JScript, para ser usada dentro do seu próprio navegador

- Para evitar isso, a Netscape decidiu padronizar a sua linguagem, em 1997, com a ajuda da empresa europeia ECMA (Associação Europeia de Fabricantes de Computadores), o que fez surgir a linguagem ECMAScript, que nada mais é do que o JavaScript padronizado

- Mesmo com as tentativas de se tentar manter no mercado, a Netscape acabou falindo em 2002. Um grupo dessa empresa, chamado de Mozilla, decidiu então continuar com o desenvolvimento de um novo navegador, o Firefox, que mesmo existindo até hoje, não possui uma grande base de usuários, apesar de ser bem conhecido

- Em 2008, nasceu o Google Chrome, que se popularizou em grande escala de maneira muito rápida, devido ao seu motor V8, que gerava códigos JIT (Just in Time). Hoje, o Chrome é o navegador mais usado do mundo, com mais da metade de todos os usuários do mundo

- O motor V8 da Google, por ser open-source, serviu de base para a criação do Node.js em 2010, um software capaz de compilar JavaScript fora dos navegadores

- A versão mais popular do ECMAScript atualmente é a ES6, mesmo não sendo a mais recente atualmente. Foi lançada no ano de 2015

- Muitas tecnologias atualmente utilizam o JavaScript como base, agindo como Frameworks ou somente como bibliotecas adicionais, usadas com distintos propósitos

Aula 3 (Motivação, Recomendações de Material e Instalação dos Programas)

- Recomendações de livros:

JavaScript – O guia definitivo (David Flanagan – Editora O'Reilly)

JavaScript – Guia do programador (Maurício Samy Silva – Editora novatec)

-Recomendações de referências e documentações:

developer.mozilla.org - Documentação oficial em português de JavaScript

ecma-international.org – Documentação oficial técnica em inglês de ECMAScript (abrir Standards > ECMA – 262 > Previous editions)

Aula 4 (Comandos Básicos)

- No Visual Studio, digitar "html:5" gera o texto inicial básico para começar a escrever

- HTML:

CSS sempre dentro do "head"

JavaScript sempre no final do "body" ou em um arquivo separado

- CSS:

Mudar a fonte e o tamanho dela: font: normal "num"pt "fonte";

Mudar a cor do elemento: color: "cor";

Mudar a cor do fundo do elemento: background-color: "cor";

Mudar a imagem de fundo do elemento: background-image: url('link');

- JavaScript:

Mostrar um aviso no site: window.alert ("mensagem")

Mostrar uma janela de confirmação no site: window.confirm("mensagem")

Mostrar uma janela de pergunta no site: window.prompt("pergunta")

Módulo B

Aula 5 (Variáveis e Data Types)

- Comentários em JavaScript: // para uma linha ou /* */ para mais de uma linha

Exemplos:

window.alert("Vsf boomber") //Esse comando vai mandar o usuário se foder

/*

Vai

Se

Foder

Boomber

*/

- Variáveis são alocações físicas na memória que delimitam um determinado dado ou informação

Exemplos:

var a1 = 10 //Define a variável a1, que recebe o valor 10

a1 = 15 //Variável a1 recebe o valor 15 e perde seu valor anterior de 10

var a2 = a1 //Define a variável a2, que recebe o valor de a1 (15)

a1 = null //Remove o valor da variável a1

OBS: No JavaScript moderno, além de utilizar a palavra "var", também podemos usar a palavra "let". Também é possível iniciar uma variável sem qualquer prefixo!

- Variáveis em formato string, podem ser informadas com algum texto cercado por aspas duplas, simples ou crases

Exemplos:

var s1 = “GG ozei”

var s2 = ‘Vsf boomber’

var s3 = `Ó o Alek gg`

- Nomes de variáveis (mais conhecidos como identificadores) podem começar com uma letra, com $ ou com _. Identificadores também não podem conter espaços ou conterem palavras reservadas da linguagem, como “alert” ou “function” por exemplo. Acentos podem ser utilizados. MAIÚSCULAS e minúsculas FAZEM DIFERENÇA, JAVASCRIPT É CASE-SENSITIVE!

Exemplos:

var aléK1 //Esta variável é válida

var $gg //Esta variável também é válida

var ?alek //Esta variável é inválida

var function // Esta variável também é inválida

var 550Educg //Esta variável é inválida (no entanto, ela respira, foda-se)

DICA: para limpar o terminal do Node.js, basta apertar Ctrl + L

- Em JavaScript, todos os números, sejam eles inteiros ou reais, são tratados como “number”, cadeias de caracteres (números e/ou letras) são “string” e verdadeiro ou falso (true/false) são “boolean”. Todos esses dados são do tipo primitivo

Exemplos:

var n = 10 //number

var s = ‘JavaScript’ //string

var b = false //boolean

OBS: existem outros tipos de variáveis, como “null” (variável vazia ou que deve receber um valor), “undefined” (variável declarada, mas sem valor), “object” (um exemplo de objeto é o array ou vetor) ou “function”. O tipo de uma variável pode ser descoberto com o prefixo “typeof” antes da variável. Diferentemente de outras linguagens, no JavaScript uma variável pode mudar seu tipo de dado, se um tipo de dado for atribuído a ela enquanto ela possuía outro tipo.

OBS2: Quando existe um erro dentro do JavaScript, o código inteiro deixa de ser executado dentro do site, por isso é importante sempre verificar bem!

Aula 6 (Comentários e Tratamento de Dados)

- Os “data types” primitivos utilizados com maior frequência, são o number e o string

- Comentários em HTML: <!—Isso é um comentário -->

- Comentários em CSS: /*Isso é um comentário*/

- Comentários em JavaScript: //Isso é um comentário

Ou

/*Isso

É

Um

Comentário

*/

- Concatenação: Unir dois ou mais conjuntos de caracteres em um só

Exemplos:

alert('GG' + 'boy')

var nome = 'Eduardo'

alert('Duas noites com' + nome + '2')

CUIDADO! O SÍMBOLO DE + PODE SER FACILMENTE CONFUNDIDO ENTRE SOMA E CONCATENAÇÃO DE VALORES. O COMANDO PROMPT SEMPRE RETORNA STRINGS, O VALOR DEVE SER CONVERTIDO PARA NUMBER!

DICA: A seguir algumas funções que mudam o valor de string para number:

Number.parseInt(n)

Number.parseFloat(n)

Number(n)

DICA 2: A seguir algumas funções que mudam o valor de number para string:

n.toString()

String(n)

OBS: Booleanos convertidos para number, se tornam 0 ou 1 (falso e verdadeiro respectivamente). Null convertido em number se torna 0.

- Interpolação: Inserir o valor de uma variável dentro de uma string, de forma semelhante ao printf em Java. A variável deve ser inserida dentro de um template string: ${variável}. Em certos casos, acaba sendo muito mais prático!

Exemplos:

var n1 = 49

alert('Bom dia senhores, eu caguei um total de ${n1} quilogramas dentro das minhas calças!')

OBS: Para que o template string funcione adequadamente, a string toda deve ser infelizmente demarcada por crases ` (;-;)

- Strings podem ser facilmente definidas com a ajuda do comando prompt:

Exemplos:

var nome = prompt('Qual é o seu nome?')

alert(nome) //Vai mostrar o nome do usuário na tela

- Formatação de strings:

string.length: retorna o número de caracteres em uma string

string.toUpperCase(): transforma todos os caracteres da string em maiúsculos

string.toLowerCase(): transforma todos os caracteres da string em minúsculos

- Formatação de numbers:

number.toFixed(n): retorna o número informado com n números após a vírgula

number.toFixed(n).replace('.', ','): retorna o número informado com o ponto (padrão das casas decimais) substituído por vírgula, além de ajustar o número para n casas decimais

number.toLocaleString('pt-BR', {style: 'currency', currency: 'BRL'}): converte automaticamente o número em reais brasileiros

- document.write('texto'): escreve algo dentro do HTML (E APAGA TODO O CONTEÚDO ANTERIOR DO SITE!)

- document.writeln('texto'): escreve algo dentro do HTML e pula uma linha

OBS: Quando document.write ou writeln é usado, podemos inserir tags dentro dele, que funcionam igual ao HTML, como <h2></h2> ou
 por exemplo.

Aula 7 (Operadores Aritméticos e de Atribuição)

- O JavaScript possui vários tipos de operadores. Os que mais vamos utilizar, são os: aritméticos, de atribuição, relacionais, lógicos e ternários

- Os operadores aritméticos são aqueles que auxiliam na realização de expressões matemáticas. São eles: + (soma), - (subtração), * (multiplicação), / (divisão), % (resto da divisão inteira) e ** (potenciação)

Exemplos:

```
alert(5 + 2) // = 7

alert(5 – 2) // = 3

alert(5 * 2) // = 10

alert(5 / 2) // = 2.5

alert(5 % 2) // = 1

alert(5 ** 2) // = 25
```

OBS: Os operadores aritméticos, obviamente, funcionam somente com dados do tipo number. O operador (**) é novo e surgiu nas versões recentes do ECMAScript, por isso, um pouco de cuidado, pois alguns navegadores podem não ser compatíveis.

ATENÇÃO! A ORDEM DE PRECEDÊNCIA DAS OPERAÇÕES ARITMÉTICAS EM JAVASCRIPT E EM TODAS AS OUTRAS LINGUAGENS DE PROGRAMAÇÃO É DIFERENTE DO USUAL, REPRESENTADA A SEGUIR:

OBS: Os operadores que estão na mesma linha tem a mesma precedência de solução. No caso de dois ou mais operadores de mesma precedência em uma mesma expressão, ela será resolvida sempre da esquerda para a direita.

- Os operadores de atribuição são aqueles que auxiliam no armazenamento de certa informação dentro de variáveis, atribuem valor a elas. É representada pelo símbolo = (atribuição simples) ou pelos símbolos +=, -=, *=, /=, **= e %= (auto-atribuição). Na auto-atribuição, vemos que o próprio valor da variável faz parte da operação de sua nova atribuição, por isso, dizer que **a = a + 6** e **a += 6** são expressões equivalentes

Exemplos:

var a = 3 * 9 // 27

var b = 3 % 2 + 4 / 2 // 3

a += 3 // 27 + 3 = 30

b **= 2 // 3 ** 2 = 9

OBS: Existem versões "reduzidas" das expressões representadas por a += 1 ou a -= 1. São elas, respectivamente: **a++** e **a--**. Também são conhecidos como operadores de **Incremento** e **Decremento**. Se o símbolo de incremento ou decremento for colocado antes da variável, temos agora um **Pré-Incremento** ou **Pré-Decremento**, que primeiro realiza a soma ou a subtração na variável e depois a retorna, diferentemente do que acontece com o normal.

Aula 8 (Operadores Relacionais, Lógicos e Ternários)

- Os operadores relacionais são aqueles que auxiliam na comparação entre duas expressões ou valores. Cada expressão que possui um operador relacional em seu meio, retorna um valor **booleano** (true/false). São eles: > (maior), < (menor), >= (maior ou igual), <= (menor ou igual), == (igual), === (identidade ou igualdade estrita), != (diferente) e !== (desigualdade estrita)

Exemplos:

7 > 4 // true

8 > 10 // false

'seu penis' == '30cm' // false

9 === 6 + 3 // true

9 === '9' // false

9 == '9' // true

OBS: Não confundir o símbolo de atribuição '=' com o símbolo relacional de igualdade '=='. O símbolo de identidade (===) sempre verifica, além do valor em si, se ambas expressões são do mesmo data type.

Operadores relacionais são resolvidos depois dos aritméticos

- Os operadores lógicos são aqueles que auxiliam na lógica das expressões, geralmente acompanhados de operadores relacionais. São eles: ! (negação), && (conjunção: e) e || (disjunção: ou) (OBS: Se todos esses operadores estiverem em uma única operação, a ordem de precedência é a mesma que está representada aqui atrás: negação > conjunção > disjunção). Uma negação sempre inverte o valor booleano de uma expressão, a disjunção só a torna verdadeira se uma das expressões for verdadeira e a conjunção exige que ambas sejam verdadeiras para torná-la verdadeira

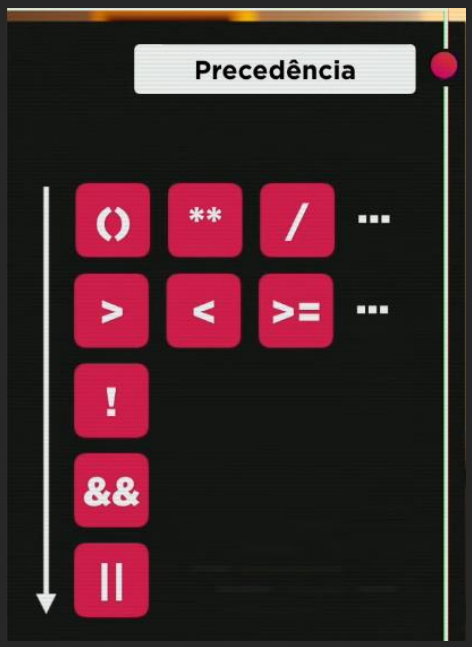
Exemplos:

var b1 = 4 == 4 && 5 > 3 // true

var b2 = 8 >= 4 || 4 == 9 // true

var b3 = !(9 >= 2.5) // false

Operadores lógicos são resolvidos depois dos relacionais



- Os operadores ternários são aqueles que unem três operações, um teste, um bloco true e outro false, onde, dependendo do resultado lógico do teste, um dos outros dois blocos será retornado. São representados pelos símbolos '?' e ':', que separam a expressão dessa

forma: “teste ? true : false”. Pode ser considerado como uma espécie de “if/else abreviado”.

Exemplos:

var media = 8

media >= 6 ? alert(‘Aprovado’) : alert(‘Reprovador’) // Aprovado

media = 3

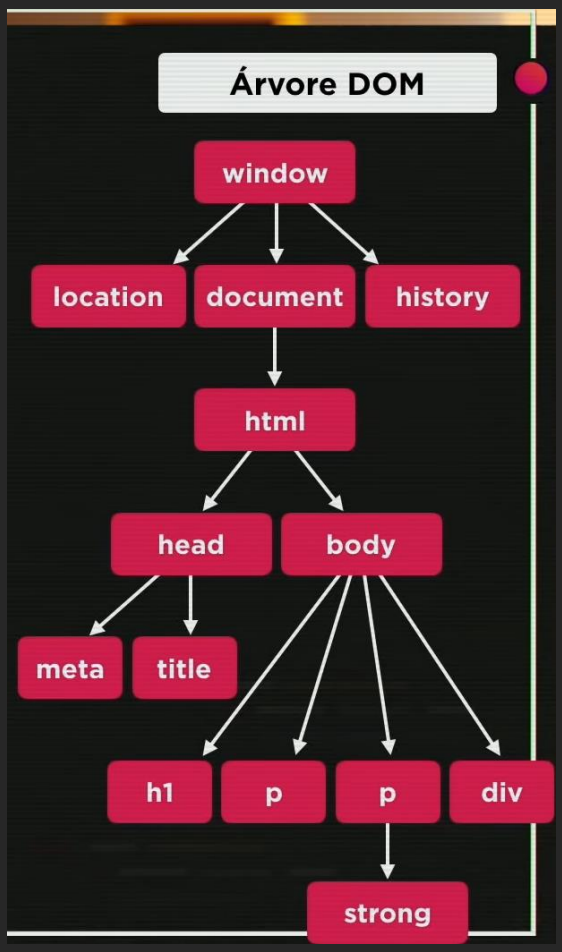
media >= 6 ? alert(‘Aprovado’) : alert(‘Reprovador’) // Reprovador

Módulo C

Aula 9 (Introdução ao DOM)

- O DOM (Document Object Model) é um conjunto de objetos dentro de um navegador que dá acesso aos componentes internos dentro de um site. Para entendermos melhor esse tipo de esquema, é essencial construir primeiro uma árvore DOM, partindo dos conceitos mais básicos (raiz) até os menores detalhes (cada “folha” da sua árvore)

- A árvore DOM possui como sua raiz principal o objeto window (basicamente, todo o site), que abriga outros elementos importantes para a divisão do website, alguns deles são: o location (endereço atual do site), o document (toda a documentação e conteúdo da página) e o history (histórico de navegação dentro do site)



- As posições hierárquicas acima na árvore são chamadas de parentes (pais, em inglês), e as que ficam abaixo dos parentes, são suas respectivas childs (crianças ou filhos, em inglês)

- Cada retângulo dentro da árvore DOM é um elemento, que pode ser selecionado por diversas maneiras, tais como: por Marca, por ID, por Nome, por Classe ou por Seletor (método mais recente)

- Quando usamos o comando window.prompt(), por exemplo, estamos chamando um elemento função chamado prompt, um dos childs do elemento window. A navegação dentro da árvore DOM é muito importante e eficaz para uma boa integração entre HTML e JavaScript

- Comando para escrever dentro do HTML com JavaScript (com o HTML não formatado)

window.document.write('texto'.innerText)

- Comando para escrever dentro do HTML com JavaScript (com o HTML formatado)

window.document.write('texto'.innerHTML)

- Comandos para selecionar elementos:

Por Marca: document.getElementsByTagName('tag')[número do elemento (Ex: 0, 1, 2, etc.)]

Por ID: document.getElementById('id')

Por Nome: document.getElementByName('name')[número do elemento (Ex: 0, 1, 2, etc.)]

Por Classe: document.getElementsByClassName('class')[número do elemento (Ex: 0, 1, 2, etc.)]

Por Seletor: document.querySelector('tag.class' OU 'tag#id')

Aula 10 (Eventos DOM)

- Como visto anteriormente, entender o DOM é essencial para selecionar os itens da sua página adequadamente e manipulá-los de maneira correta, para a criação de uma boa integração entre HTML/CSS/JavaScript

- Os eventos são coisas que podem vir a acontecer com os elementos em HTML, detectados e tratados com JavaScript.

- Alguns exemplos de eventos são os que verificam ações do seu cursor: mouseenter (cursor entrou na área de um dos elementos), mousemove (cursor de moveu dentro dessa área), mousedown (cursor está segurando o botão esquerdo no elemento), mouseup (cursor soltou o botão esquerdo), click (cursor clicou no elemento) e mouseout (cursor saiu da área do elemento)

OBS: Os eventos são extremamente importantes, pois são eles que ajudam em grande parte da integração de todas as tecnologias do site

- Esses eventos são uma espécie de **função**, linhas de comandos com um objetivo específico que, neste caso, tem como objetivo disparar quando um evento é realizado dentro de um determinado elemento

- As linhas de código de uma função em JavaScript são contidas dentro de um bloco, cercadas por duas chaves '{ }' que são precedidas pela palavra function, da seguinte forma:

Exemplo:

```
function nomeDaFunção(parâmetro)
{
        alert('Alek') // Exibe 'Alek'
}
```

- Os eventos podem ser inseridos dentro do HTML, adicionando o prefixo "on" + evento dentro das tags.

Exemplos:

HTML:

```
<div id="area" onclick="clicar()" onmouseenter="entrar()" onmouseout="sair()">
      <h1>Ganhafoto...🐑</h1>
      <p>Long time no see, please subscribe to my channel and click this 👽✌</p>
      <p id="thanks"></p>
      <p id="cuidado"></p>
   </div>
```

JavaScript:

```
var a = document.querySelector('p#cuidado')
var b = document.querySelector('p#thanks')

      function clicar()
      {
         alert('Sensei Guabanára... é.. você?? 🦁✌')
```

```
        b.innerText = 'THANKS, NOW YOU WILL SUFFER WITH MY BRAND-NEW SPONSORSHIP, GOOGLE 👹🤦‍♂️💀😈'
        a.remove()
    }
    function entrar()
    {
        a.innerText = '😈'
    }
    function sair()
    {
        a.innerText = ' '
    }
```

- Ou podemos simplesmente criar listeners dentro do JavaScript, de forma muito mais prática e menos poluída, causando o mesmo efeito na página:

Exemplo:

```
var div = document.querySelector('div#area')
div.addEventListener('click', clicar)
div.addEventListener('mouseenter', entrar)
div.addEventListener('mouseout', sair)
...
```

DICA: Para encontrar erros mais facilmente no seu código, através do navegador, basta abrir o console da página e o erro será exibido

- Para receber um valor inserido dentro de um input de HTML, é necessário que uma variável receba o value da caixa de texto:

Exemplos:

```
var txt1 = document.getElementById('n1')
```

var txt2 = document.getElementById('n2')

var n1 = Number(txt1.value)

var n2 = Number(txt2.value)

OBS: É importante resgatar o elemento value contido pelas variáveis txt, pois elas na verdade são objetos. Sendo assim, o valor delas é uma das muitas características e atributos que elas possuem.

Módulo D

Aula 11 (Condições Simples e Compostas)

- Quando uma condição é posta dentro de um código JavaScript, a sequencialidade é, de certa forma, dividida, pois as linhas que antes eram executadas de forma deliberada, agora podem ou não serem executadas, criando duas ou até mais possibilidades. Esse tipo de "bifurcação" é denominado como um **Desvio Condicional**

- Para declarar ambos os caminhos, utilizamos os comandos **if/else**, que realizam um bloco de linhas de código quando executados:

Exemplos:

```
if (10 >= 4) // A condição se aplica (true), o bloco de comando if será executado
{
        alert('GG boy')
}
else // Como a condição if já teve sua condição atendida, esse bloco será ignorado (ele só seria executado se a condição acima fosse false)
{
```

```
        alert('GGn't boy')
}
```

OBS: Comandos if podem também ser colocados sozinhos, são chamadas de **condições simples**, e quando são acompanhados de um else, são chamadas de **condições compostas**.

- Comando para escrever na tela do console ou no Node.js:

```
console.log('texto')
```

Aula 12 (Condições Aninhadas e Múltiplas)

- As condições aninhadas são aquelas compostas por dois ou mais blocos if/else if, ou seja, se a primeira condição for falsa, o programa tenta validar a próxima, até encontrar uma condição aplicável, ou simplesmente não atende a nenhum bloco, se todas forem falsas. Também é possível terminar um conjunto de condições aninhadas com um else simples, o que implica que se nenhuma das condições anteriores for verdadeira, o útlimo bloco será executado

Exemplo:

```
if (10 < 4)
{
        //bla bla bla
}
else if (7 < 4)
{
        //bla bla bla
}
```

```
else if (9 == 3)
{
        //bla bla bla
}
else //apenas este bloco será executado
{
        alert ('gg man')
}
```

OBS: Outras linguagens também utilizam condições aninhadas, mas com sintaxes diferentes, como em Phyton por exemplo, onde ao invés de else if, é utilizada a expressão elif

- Comando para resgatar o horário atual da máquina:

```
var agora = new Date()
var hora = agora.getHours()
```

OBS: Existem outras funções parecidas, que resgatam outros dados, como por exemplo o dia, ano, minutos, etc.

- As condições múltiplas são aquelas compostas pelos blocos switch/case, onde é verificada uma única expressão entre diversas possibilidades para ela mesma. Ela pode verificar basicamente todos os data types primitivos.

Exemplo:

```
var num = 12
switch (num)
{
        case 14:
```

```
            //bla bla bla
            break
        case 15:
            //bla bla bla
            break
        case 19:
            //bla bla bla
            break
        default:
            //esta parte será executada, pois nenhuma das outras condições foi atendida
            break
}
```

OBS: É importante que o comando break esteja presente no final de cada case, caso contrário, a estrutura executará comandos de outros blocos que não deveria

Aula 13 (Repetições)

- Repetições são definidas como instruções específicas que se repetem durante a validez verdadeira de uma expressão lógica. Esse tipo de repetição simples é definida pelo comando while. Cada repetição também pode ser chamada de iteração:

Exemplo:

```
var i = 0
while (i <= 10)
{
```

```
        alert(i) // Vai apresentar todos os números entre 0 e 10
        i++;
}
```

- Existe outra estrutura de repetição que primeiro realiza uma iteração, e somente após verifica a validez do teste lógico. É representada pelos comandos do/while.

Exemplo:

```
var i = 0;
do
{
        alert(i) // Apresentará a variável i (0)
} while (i != 0) // Falso, o programa segue em frente
```

Aula 14 (Repetições For)

- O comando de repetições for é construído a partir de três expressões e um bloco de códigos. A primeira expressão é realizada na primeira iteração, sendo ignorada depois. A segunda expressão representa a condição para a repetição do laço, e a terceira representa uma linha de código que será executada sempre que uma iteração chega ao seu fim.

Exemplo:

```
for (var i = 0; i < 10; i++)
{
        console.log(i) // Exibirá todos os números entre 0 e 9
}
```

OBS: Variáveis inicializadas dentro do escopo de um laço de repetição é inutilizada quando o laço se rompe.

Aula 15 (Variáveis Compostas – Arrays/Vetores)

- As variáveis compostas nada mais são do que tipos especiais de variável que podem receber mais de um valor, diferentemente das variáveis simples, que recebem apenas um valor por vez

- Essas variáveis compostas são mais conhecidas como Vetores (ou Arrays, em inglês)

- Seus valores internos podem ser acessados através da identificação da chave de cada valor, ou índice (OBS: começa em 0, não em 1)

Exemplo:

```
var vetorgg = [7, 28, 91, 03]

alert(vetorgg[2]) // Exibirá o número 91, armazenado na Terceira posição
```

- Os arrays em JavaScript são dinâmicos, o tamanho de um array pode mudar dependendo das necessidades do programa. Novos valores além do tamanho atual podem ser definidos manualmente (vetorgg[4] = 39) ou de forma automatizada, usando a função interna push(), que acrescenta o valor declarado na próxima casa do vetor (vetorgg.push(28))

- Comando para adicionar um valor automaticamente na próxima casa do array: vetor.push(num)

- Comando para organizar os elementos do array em ordem alfabética/crescente: vetor.sort()

- Comando para encontrar o índice de um certo valor dentro de um array (Se não existir um índice que possua esse valor, a função retornará o número -1. Se existir mais de um índice possuindo esse valor, a função retornará o primeiro índice encontrado, em ordem crescente dos índices): vetor.indexOf(num)

- Além de poder percorrer o array utilizando o for ou algum outro laço comum, em JavaScript, existe uma forma simplificada de percorrer um vetor, utilizando o comando for in, onde podemos apenas declarar uma variável que percorrerá automaticamente o vetor e também o vetor a ser percorrido (for (var i in vetor))

Exemplo:

```
for (var I in vetor)
{
        console.log(vetor[i]) // Exibirá todos os números do array
}
```

Aula 16 (Funções)

- As funções são conjuntos de código que são executados com o objetivo de realizar uma ação específica dentro de um programa. Elas são executadas através das chamadas, podem ou não receber dados através dos parâmetros (expressões dentro de parênteses. Ex: alert ('gg boy') // A função alert recebe essa frase e exibe como um aviso na tela)

- As funções também são conhecidas como métodos

- A execução de uma função também é conhecida como ação

- O retorno ocorre no fim de um programa, é o ato de finalizar a função e retornar o valor ao local onde foi chamado (ou pode

simplesmente finalizar, sem retornar um resultado específico). Ele é declarado pelo comando return

OBS: Na declaração de uma função, nos parâmetros, diferentemente de outras linguagens, não é utilizado o comando var, let ou const antes do nome da variável inicializada. Diversas variáveis podem ser inicializadas dentro de parâmetros, através do uso de vírgula entre as variáveis. Se por acaso menos parâmetros forem definidos na chamada do que existem a serem declarados na função, as variáveis que não são preenchidas ficam como undefined.

Curiosidade: É possível declarar uma variável que recebe uma função, de forma que para chamar a função é utilizado o nome da variável: var gg = function () {//código}

- Uma função recursiva ocorre quando uma função chama ela mesma, de forma que consegue substituir a necessidade de um laço de repetição como o while, do while ou for. Abaixo um exemplo, calculando um número fatorial:

Exemplo:

```
function fatorial (n)
{
	if (n == 1)
	{
		return 1
	}
	else
	{
		return n * fatorial(n – 1) // A menos que o valor inicial seja 1, entrará em um loop, até que o valor seja 1, retornando cada valor testado na ação
	}
```

}

console.log (fatorial(5)) // Retorna 120, o fatorial de 5

Aula 17 (Próximos Passos)

- Coisas importantes para estudar futuramente:

functions (call-backs, funções anônimas, iife functions, etc.),

objects,

modularização (encapsulamento de dados, separação de linguagens por arquivo, organização de ficheiros, etc.),

RegEx (expressões regulares),

JSON,

AJAX (um bom exemplo do uso disso é quando a página é carregada dinamicamente, enquanto o usuário navega),

NodeJS

- Os objetos, por vezes, são muito melhores para armazenamento de dados, substituindo assim os arrays em certas situações. Usamos ' : ' para declarar os atributos do objeto:

Exemplo:

let muie = {

nome: 'Maria Claudiar',

sexo: 'F',

peso: 352.82,

engordar(p) { this.peso += p }

```
}
```

OBS: o comando “this” é utilizado para se referir ao atributo do próprio objeto onde o “this” está.

Espero que tenha gostado deste material, até a próxima amigor 😉