INSTITUTO FEDERAL
Catarinense
Campus Camboriú

Desenvolvimento Web I

Professor: Euclides Paim

euclides.paim@ifc.edu.br

INSTITUTO FEDERAL
Catarinense
Campus Camboriú

Instruções e Sintaxe *JavaScript*

Professor: Euclides Paim

euclides.paim@ifc.edu.br

Programas *JavaScript*

Sumário

Sumário

- Definição Ampla
- Instruções *JavaScript*
- Modelos Cognitivos
- Sintaxe
- Variáveis
- Constantes

INSTITUTO FEDERAL
Catarinense
Campus Camboriú

Programas *JavaScript*

Um programa de computador é uma lista de "regras" a serem "executadas" por um computador. Em uma linguagem de programação, essas regras de programação são chamadas de **instruções** (statements). Um programa JavaScript é uma lista de **instruções** de programação. Em HTML, os programas JavaScript são executados pelo navegador da web.

Instruções *JavaScript*

As instruções JavaScript são compostas por Valores, operadores, expressões, palavras-chave e comentários. A declaração a seguir diz ao navegador para escrever "Hello World". dentro de um elemento HTML com id = "demo":

```
document.getElementById("demo").innerHTML = "Hello World.";
```

A maioria dos programas JavaScript contém muitas instruções JavaScript. As instruções são executadas, uma a uma, na mesma ordem em que são escritas. Os programas JavaScript (e instruções JavaScript) geralmente são chamados de código JavaScript.

Instruções *JavaScript*

Modelos Cognitivos

Observe este código:

```
var a = 10;
var b = a;
a = 0;
```

Quais são os valores de "a" e "b" após sua execução? Resolva isso em sua cabeça antes de irmos adiante...

O objetivo deste exercício não é apresentar variáveis. Sabemos que vocês já estão familiarizados com elas. Em vez disso, esperamos fazer você notar e refletir sobre seu *modelo cognitivo*.

Instruções *JavaScript*

Modelos Cognitivos

Leia o código anterior novamente com a intenção de realmente ter certeza qual é o resultado. (Veremos por que essa intenção é importante um pouco mais tarde.) Enquanto você lê pela segunda vez, preste muita atenção ao que está acontecendo em sua cabeça, passo a passo. Você pode notar um monólogo como este:

- `var a = 10;`
  - Declare uma variável chamada "a". Atribua `10` para "a".

- `var b = a;`
  - Declare uma variável chamada "b". Atribua "a" para "b".

- `a = 0;`
  - Atribua `0` para "a".

Instruções *JavaScript*

Modelos Cognitivos

Talvez o seu monólogo seja um pouco diferente. Talvez você diga: *"a" recebe 10,* ou talvez você o leia em uma ordem ligeiramente diferente. Talvez você tenha chegado a um resultado diferente. Preste atenção em como exatamente era diferente. Observe como esse monólogo não captura o que realmente está acontecendo em sua cabeça. Você pode dizer *"defina b como a"*, mas o que significa definir uma variável?

Instruções *JavaScript*

Modelos Cognitivos

Você vai notar que, para todo conceito fundamental de programação (como uma variável) e operações (como definir seu valor), há um conjunto de **analogias profundas** que você associou a eles. Algumas delas podem vir do mundo real. Outros podem ser redirecionados de outros campos que você aprendeu primeiro, como números de **matemática**. Essas analogias podem se sobrepor e até se contradizer, mas ainda ajudam a entender o que está acontecendo no código.

Instruções *JavaScript*

Modelos Cognitivos

Por exemplo, muitas pessoas aprenderam sobre variáveis como **“caixas”** nas quais você pode **colocar** coisas. Mesmo que você não use mais as caixas quando vê uma variável, elas ainda podem se comportar "quadradas" em sua imaginação. Essas aproximações de como algo funciona na sua cabeça são conhecidas como "**modelos cognitivos**". Pode ser difícil se você estiver programando há muito tempo, mas tente observar e introspectar seus modelos mentais. Eles provavelmente são uma combinação de atalhos mentais visuais, espaciais e mecânicos.

Instruções *JavaScript*

Modelos Cognitivos

Essas analogias (como "caixas" de variáveis) **influenciam** como lemos o código por toda a vida. Mas, às vezes, nossos modelos mentais estão errados. Talvez um tutorial que lemos no início tenha sacrificado a correção pela facilidade de explicação. Talvez tenhamos transferido incorretamente uma analogia sobre um determinado recurso de outra linguagem que aprendemos anteriormente. Talvez tenhamos inferido um modelo mental de algum trecho de código e nunca realmente verificado se ele era preciso.

Identificar e corrigir esses problemas é o objetivo deste exercício. Um bom modelo mental o ajudará a encontrar e corrigir erros mais rapidamente, a entender melhor o código de outras pessoas e a se sentir confiante no código que você escreve.

Instruções *JavaScript*

Sintaxe

O **ponto-e-vírgula** separa as instruções JavaScript. Adicionamos um ponto-e-vírgula no final de cada instrução executável:

```
var a, b, c;      // Declara 3 variáveis
a = 5;            // Atribui o valor 5 para a
b = 6;            // Atribui o valor 6 para b
c = a + b;        // Atribui a soma de a e b para c
```

Obs1.: Quando separadas por ponto e vírgula, são permitidas várias instruções em uma linha.

Obs2.: Finalizar uma instrução com ponto e vírgula não é obrigatório, mas é altamente recomendado.

Instruções *JavaScript*

Sintaxe

JavaScript ignora múltiplos **espaços em branco**. Você pode adicionar espaço em branco ao seu script para torná-lo mais legível. As seguintes linhas são equivalentes:

```
var person = "Hege";
var person="Hege";
```

Uma boa prática é colocar espaços ao redor dos operadores (= + - * /):

```
var x = y + z;
```

Instruções *JavaScript*

Sintaxe

Para melhor legibilidade, os programadores geralmente gostam de evitar linhas de código com mais de 80 caracteres. Se uma instrução JavaScript não couber em uma linha, o melhor lugar para quebrá-la é após um operador, exemplo:

```
document.getElementById("demo").innerHTML =
"Hello World!";
```

INSTITUTO FEDERAL
Catarinense
Campus Camboriú

Instruções *JavaScript*

Sintaxe

As instruções JavaScript podem ser agrupadas em blocos de código, dentro de colchetes **{.....}**. O objetivo dos blocos de código é definir instruções a serem executadas juntas. Um lugar em que você encontrará instruções agrupadas em blocos é nas funções JavaScript, exemplo:

```
function myFunction() {
  document.getElementById("demo1").innerHTML = "Hello World!";
  document.getElementById("demo2").innerHTML = "How are you?";
}
```

INSTITUTO FEDERAL
Catarinense
Campus Camboriú

Instruções *JavaScript*

Sintaxe

As instruções JavaScript geralmente começam com uma **keyword** (palavra-chave) para identificar a ação JavaScript a ser executada. Acesse a referência de palavras reservadas para ver uma lista completa de palavras-chave JavaScript.

Keyword	Descrição
break	Encerra um switch ou um loop.
continue	Sai de um loop e recomeça em um novo bloco.
for	Marca um bloco de instruções a serem executadas, desde que uma condição seja verdadeira.
function	Declara uma função.
if ... else	Marca um bloco de instruções a serem executadas, dependendo de uma condição.
return	Sai de uma função.

Obs.: Palavras-chave são palavras reservadas JavaScript e não podem ser usadas como nomes para variáveis.

Sintaxe *JavaScript*

Sintaxe

JavaScript pega emprestado a maior parte de sua sintaxe do Java, mas também é influenciado por Awk, Perl e Python. JavaScript é **case-sensitive** e usa o conjunto de caracteres **Unicode.** Por exemplo, a palavra Früh (que significa "cedo" em Alemão) pode ser usada como nome de variável.

```
var Früh = "foobar";
```

Mas a variável früh não é a mesma que Früh porque JavaScript é case sensitive.

Sintaxe *JavaScript*

Sintaxe

A sintaxe dos comentários em JavaScript é semelhante a do C++ e muitas outras linguagens:

```
1. // comentário de uma linha
2. …
3. /* isto é um comentário longo
4.          de múltiplas linhas. */
5. …
6. /* Você não pode, porém, /* aninhar comentários */ SyntaxError */
7. …
```

Sintaxe *JavaScript*

Variáveis

Variáveis são nomes simbólicos para os valores em sua aplicação. O nome das variáveis, chamados de identificadores, obedecem determinadas regras:

Um identificador JavaScript deve começar com uma letra, *underline* (_), ou cifrão ($); os caracteres subsequentes podem também ser números (0-9). Devido JavaScript ser ***case sensitive***, letras incluem caracteres de "A" a "Z" (maiúsculos) e caracteres de "a" a "z" (minúsculos).

Você pode usar a ISO 8859-1 ou caracteres Unicode tal como os identificadores å e ü. Você pode também usar as sequências de escape Unicode como caracteres e identificadores.

Obs.: Hifens não são permitidos no JavaScript. Eles são reservados para subtrações.

INSTITUTO FEDERAL
Catarinense
Campus Camboriú

Sintaxe *JavaScript*

Declarando Variáveis

Existem três tipos de declarações em JavaScript:

`var` - Declara uma variável, opcionalmente, inicializando-a com um valor.

`let` - Declara uma variável local de escopo do bloco, opcionalmente, inicializando-a com um valor.

`const` - Declara uma constante de escopo de bloco, apenas de leitura.

Fonte: **var**, **let** e **const**. (leitura complementar recomendada)

Sintaxe *JavaScript*

Declarando variáveis

Você pode declarar uma variável de três formas:

- Com a palavra-chave **`var`**. Por exemplo, `var x = 42`. Esta sintaxe pode ser usada para declarar tanto variáveis locais como variáveis globais.
- Por simples adição de valor. Por exemplo, `x = 42`. Isso declara uma variável global. Essa declaração gera um aviso de advertência no JavaScript. Você não deve usar essa variante.
- Com a palavra chave **`let`**. Por exemplo, `let y = 13`. Essa sintaxe pode ser usada para declarar uma variável local de escopo de bloco.

Sintaxe *JavaScript*

Escopo de variável

Quando você declara uma variável fora de qualquer função, ela é chamada de **variável *global***, porque está disponível para qualquer outro código no documento atual. Quando você declara uma variável dentro de uma função, é chamada de **variável *local***, pois ela está disponível somente dentro dessa função.

JavaScript antes do ECMAScript 6 não possuía escopo de declaração de bloco; pelo contrário, uma variável declarada dentro de um bloco de uma *função* é uma variável local (ou contexto *global*) do bloco que está inserido a função.

Ecma International: http://www.ecma-international.org/
ECMAScript 6: https://www.w3schools.com/js/js_es6.asp

Sintaxe *JavaScript*

Escopo de variável

Por exemplo o código a seguir exibirá 5, porque o escopo de `x` está na função (ou contexto global) no qual `x` é declarado, não o bloco, que neste caso é a declaração `if`.

```
if (true) {
    var x = 5;
}
console.log(x); // 5
```

Esse comportamento é alterado, quando usado a declaração `let` introduzida pelo ECMAScript 6.

```
if (true) {
    let y = 5;
}
console.log(y); // ReferenceError: y não está definido
```

Sintaxe *JavaScript*

Constantes

Você pode criar uma constante apenas de leitura por meio da palavra-chave `const`. A sintaxe de um identificador de uma constante é semelhante ao identificador de uma variável: deve começar com uma letra, sublinhado ou cifrão e pode conter caractere alfabético, numérico ou sublinhado.

```
const PI = 3.14;
```

Uma constante não pode alterar seu valor por meio de uma atribuição ou ser declarada novamente enquanto o script está em execução. Deve ser inicializada com um valor.
As regras de escopo para as constantes são as mesmas para as variáveis ***let*** de escopo de bloco. Se a palavra-chave `const` for omitida, presume-se que o identificador represente uma variável.

Sintaxe *JavaScript*

Wrap-up

Wrap-up...

- Definição Ampla
- Instruções *JavaScript*
- Modelos Cognitivos
- Sintaxe
  - Ponto e vírgula, espaços em branco, quebras de linha
  - Blocos de códigos, palavras reservadas, conjunto de caracteres, comentários
- Variáveis
  - Declarando Variáveis
  - Escopo de Variáveis
- Constantes

INSTITUTO FEDERAL
Catarinense
Campus Camboriú

Referências

Referências Básicas

Livro NIEDERAUER, Juliano. Desenvolvendo websites com PHP: aprenda a criar websites dinâmicos e interativos com PHP e bancos de dados. 2. ed. rev. e atual. São Paulo: Novatec, 2011.

Livro SILVA, Maurício Samy. HTML 5: a linguagem de marcação que revolucionou a web. São Paulo: Novatec, 2011.

Livro SILVA, Maurício Samy. CSS3: desenvolva aplicações web profissionais com uso dos poderosos recursos de estilização das CSS3. São Paulo: Novatec, 2012.

Referências Complementares

Livro DEITEL, Paul J. Ajax,. Rich Internet applications e desenvolvimento Web para programadores. . Pearson Prentice Hall. 2009

Livro DALL'OGLIO, Pablo. PHP: programando com orientação a objetos. . Novatec. 2009

Livro SOARES, Walace. PHP 5: conceitos, programação e integração com banco de dados. . Érica. 2010

Livro SILVA, Maurício Samy. Criando sites com HTML: sites de alta qualidade com HTML e CSS. . Novatec. 2010

Livro FLANAGAN, David. o guia definitivo. . O Really. 2012

Referências na Internet

https://www.w3schools.com

https://developer.mozilla.org/pt-BR/docs/Web

https://illustrated.dev/advancedjs