Padronização do Projeto

→ Comentários

- html

```
<!--
        Grupo: <nome_grupo>
        Data de modificação: <dia>/<mes>/<ano>
        Autor: <nome><sobrenome>
        Objetivo da modificação: {objetivos}
-->
```

- CSS, JavaScript, PHP e Java

```
/*
        Grupo: <nome_grupo>
        Data de modificação: <dia>/<mes>/<ano>
        Autor: <nome><sobrenome>
        Objetivo da modificação: {objetivos}
*/
```

→ Observações

- Evitar comentários a frente da linha de código;
- Não deixar de comentar pois facilita o estudo do código.
- Separar os scripts dos stylesheets e do html
    - Arquivos css separados de html
    - Arquivos javascript separados de html
- **Nunca** usar underlines, hífens, acentuação, ‘ç’
- Usem document.querySelector(".example"); em vez de *getElementById()* ou *getElementsByClassName()*, no caso example pode ser uma classe ou id. Exemplo.
-

Tipagem de Variáveis

A estrutura padrão para escrita de variáveis a ser adotada será:

- Todas as variáveis devem ser seguidas do prefixo referente aos seus devidos conteúdos **(NO CASO DE LINGUAGENS NÃO TIPADAS)** (<tipo><NomeVar>), por exemplo:
    - intFuncaodavariavel
    - strFuncaodavariavel
    - doubleFuncaodoarray:arr

- Variáveis temporárias a serem utilizadas em repetições e possíveis testes, seguirão o modelo (<tipo><Temp><Func>), por exemplo:
    - intTemp1FuncaoDaVariavel
    - doubleTemp2FuncaoDaVariavel
- Note que a primeira letra é minúscula!

Case sensitive

No caso usaremos o padrão de minúsculo para elementos que podem receber tanto letras maiúsculas ou minúsculas, para facilitar a leitura e padronização.

→ Isso não se aplicam **MySQL** que terá seus padrões de comandos em **letra maiúscula**

Usar constantes!

O uso de constantes é importante para que a alteração do código seja mais fácil. Constantes sempre em caixa alta, para diferenciar das variáveis.

// JavaScript

```
// não existem constantes em javascript, então por convenção adotaremos:
// <NOMECONSTANTE>
TAMANHOVETOR = valor;
// esse valor não será alterado no restante do código!
```

// PHP

```
define (NOMECONSTANTE, valor);
```

// JAVA

```
public final int TAMANHOVETOR;
```

Métodos

Os métodos deverão ser nomeados da seguinte forma:

- Primeira letra **sempre minúscula**, para diferenciar métodos de classes
- A descrição deve ser a melhor possível
    - Não misturar dois idiomas na nomenclatura!!

Classes

O padrão da escrita de classes será:

- Primeira letra **sempre maiúscula**, para diferenciar métodos de classes
- A descrição deve ser a melhor possível
    - Não misturar dois idiomas na nomenclatura!!

Indentar é importante!!!

// HTML

```
<html>
	<head>
		<!-- code -->
	</head>
	<body>
		<!-- code -->
	</body>
</html>
```

// CSS

```
classe
{
	color : blue;
}
```

// PHP

```
<?php
	// code
?>
```

// JAVA

```
public ClasseExemplo
{
	public int variavelExemplo;

	public void metodoExemplo ()
	{
		variavelExemplo = 0;
	}
}
```

Estruturas de repetição

Em linguagens como PHP, JavaScript e Java usar sempre o for que não utiliza variáveis para evitar erros bobos, no caso o for “melhorado”. Apenas utilize os outros fors quando realmente precisar usar o elemento de iteração na estrutura de repetição.

- PHP → foreach (nome-do-array as $valor);
- JavaScript → for (variavel of iteravel);
- Java → for (<tipo> valor: array)

Espaçamento

- Antes e depois de sinais de comparação ou atribuição
- Antes e depois de **+**, **-**, **.** , **+=**, **-=**, **.=**
- Antes da abertura de parênteses em estruturas de repetição ou condicional

```
var intContador = 0;
var intAdiciona = 5;
intContador += 1;
intContador = intContador + intAdiciona;
for (intContador = 0; intContador < TAMANHOVETOR; intContador++) {}
```

E não

```
var intContador=0;
var intAdiciona=5;
intContador+=1;
intContador=intContador+intAdiciona;
for(intContador=0;intContador<TAMANHOVETOR;intContador++){}
```

O espaçamento facilita a compreensão do código!
Os espaços não aumentam o tempo de execução, então não tem por que evitá-los!

A leitura do código deve ser fácil para todos!

Nomeando os arquivos

O arquivo de cada grupo deverá ser nomeado pelas iniciais dos integrantes do grupo mais a especificação java/web mais a funcionalidade (objetivo do código).

O padrão de iniciais dos grupos será da seguinte forma:

- Pedro Lucas => PHJL;
- Augusto => AMP;
- Matheus Quintão => MAE;
- Joice => JHJ;
- Carlos => CJF;
- Felipe Gonçalves => FAGE;
- Leonardo => LJL;
- Bryann => BLT;

Exemplo:

PHJL-Web-Menu.html
PHJL-Web-Menu.css
PHJL-Web-Menu.js

// Pasta com arquivos java
FAGE-Java-CadastroAluno

Observações:

- Não deixar espaços nem underline nem “ / ” nem “ ç ” nem acentuação pois podem ocorrer conflitos no momento de juntar os arquivos;

É DEVER dos LÍDERES garantir que TODAS as padronizações estabelecidas neste arquivo sejam cumpridas!