Mm

Manual de técnicas de programação

*“Boas práticas de programação; versionamento; breakpoints”*

***DESENVOLVIDO POR:***

Vinícius da Silva Ribeiro

Matheus Arthur Alves da Silva

Thiago Aquila Summo de Sá

* BOAS PRÁTICAS DE PROGRAMAÇÃO

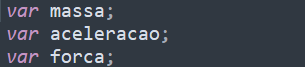
As boas práticas de programação servem para auxiliar o programador em seu desenvolvimento de um software, ou no próprio código. É por meio delas que um desenvolvedor prova ser um bom programador.

O fato de não usar elas em um código não interferem no software, ou que o programador não vai conseguir programar sem elas, mas usando elas um programador demonstra interesse pelo código, codificação, a qualidade do seu código, e que seu programa cumpre com todas as funções para que o usuário tenha um ótimo aproveitamento para com o software.

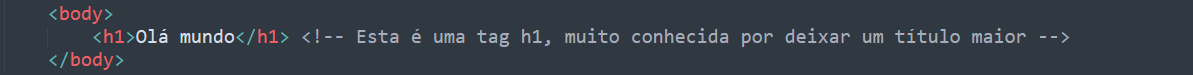
É essencial que o programador crie um código legível de maneira que o programador quando for ler o código, ou outra pessoa (sabendo sobre programação, ou não), entenda aquilo que está se passando no código e a função dele.

**VARIÁVEIS COM NOMES SUGESTIVOS**

O programador em seu código quando for criar variáveis, deve criar variáveis sugestivas, para quando alguém ler (sendo o programador, ou não) entenda melhor a variável e o que ela armazena.

***exemplo de variáveis com nome sugestivos.***

**COMENTÁRIOS**

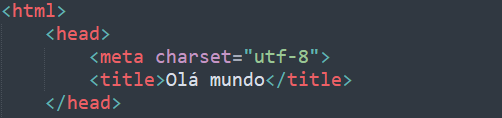
Quando o desenvolvedor estiver programando seu código, é adequado que coloque comentários (seguindo a regra de sintaxe de cada linguagem), porque pode ocorrer de o programador abrir um código e não entender aquilo que está se passando no código, e com os comentários o programador consegue entender o que está acontecendo no código e qual era o objetivo quando ele estava programando. 

***exemplo de comentário no HTML (linguagem de marcação de texto)***

**IDENTAÇÃO**

Uma das “boas práticas” que auxilia no desenvolvimento de um software é a identação que é posicionar um comando de uma forma a entender que aquele comando pertence a outro comando. Existem algumas linguagens de programação que não aceitam o código sem estar identado, exemplo a linguagem Python.

Essas são as boas práticas de programação, onde um programador consegue ter um auxílio excelente em seu software e aprender a desenvolver melhor.

***exemplo de código identado. Neste código dá para perceber que a tag “meta” e a tag “title” está dentro da tag “head”, e essas tags estão dentro da tag “html”.***

* VERSIONAMENTO

O versionamento de software é o processo de atribuir um nome único ou uma numeração única para indicar o estado de um programa de computador. Esses números são geralmente atribuídos de forma crescente e indicam o desenvolvimento de melhorias ou correção de falhas no software.

**Como Funciona o Versionamento?**

O controle de versão consiste em duas partes: o repositório e a área de trabalho. O repositório armazena todo o histórico de evolução do projeto, registrando todas e quaisquer alterações feitas em cada projeto com versão.

Os desenvolvedores não podem trabalhar diretamente nos arquivos do repositório. Em vez disso, o espaço de trabalho que ele usa contém cópias de arquivos de projeto e é monitorado para identificar mudanças. Esta área é separada e isolada das demais áreas de trabalho.

A sincronização entre o desktop e o repositório é feita através dos comandos commit e update. Commit envia um pacote contendo uma ou mais modificações feitas na área de trabalho (origem) para o repositório (destino). O processo de atualização é revertido, ou seja, envia as alterações contidas no repositório (origem) para a área de trabalho (destino).

Cada confirmação irá gerar uma nova revisão no repositório, que contém a modificação, data e autor. A revisão é como a "imagem" de todos os arquivos e diretórios a qualquer momento durante o desenvolvimento do projeto. As "fotos" antigas serão mantidas e podem ser recuperadas e analisadas a qualquer momento quando necessário. Esses conjuntos de revisão são a história do projeto.

Tanto o controle de versão centralizado quanto o distribuído têm repositórios e áreas de trabalho. A diferença está na disposição de cada peça.

**Versionamento Centralizado**

O controle de versão centralizado segue uma topologia em estrela, com apenas um repositório central, mas com várias cópias de trabalho, uma para cada desenvolvedor. A comunicação entre uma área de trabalho e outra deve passar pelo repositório central.

**Versionamento Distribuído**

Existem vários repositórios autônomos e independentes, um para cada desenvolvedor. Cada repositório possui uma área de trabalho adicional e as operações de confirmação e atualização ocorrem localmente entre os dois.

O repositório pode se comunicar com qualquer outro repositório por meio de operações básicas de pull e push:

Puxar (pull): Atualize o repositório local (destino) com quaisquer alterações feitas em outro repositório (origem).

Empurrar (push): Envie as alterações do repositório local (origem) para outro repositório (destino).

A sincronização entre os desenvolvedores ocorre entre os repositórios. Em princípio, um repositório é mais importante do que o outro, embora a função do repositório central possa ser usada para reunir fluxos de trabalho.

* BREAKPOINTS

*Sobre Breakpoints*

Breakpoints é uma das partes mais importantes de uma ferramenta de depuração, basicamente são pontos de interrupção (ou quebra) que servem para pausar a execução do depurador podendo ser em uma linha especifica do código com ou sem condições para ativação.

Os breakpoints possuem diferentes tipos, sendo eles:

| **Tipo de ponto de interrupção** | **Use isso quando quiser pausar...** |
| --- | --- |
| Linha de código | Em uma região exata do código. |
| Linha de código condicional | Em uma região exata do código, mas somente quando alguma outra condição for verdadeira. |
| DOM | No código que altera ou remove um nó DOM específico ou os filhos. |
| XHR | Quando uma URL XHR contém um padrão de cadeia de caracteres. |
| Ouvinte de eventos | No código que é executado após um evento, como click, é executado. |
| Exceção | Na linha de código que está lançando uma exceção capturada ou não capturada. |
| Função | Sempre que um comando, função ou método específico é executado. |

*Adicionando Breakpoints*

Para adicionar um ponto de interrupção basta ir no código-fonte, clicar na margem à extrema esquerda ao lado de uma linha de código.

Você também pode selecionar a linha e pressionar F9, selecionar Depurar Alternar Ponto de Interrupção ou clicar com o botão direito do mouse e selecionar Ponto de Interrupção > Inserir ponto > de interrupção. O ponto de interrupção aparece como um ponto vermelho na margem esquerda.

Um exemplo simples do breakpoint de linha de código que permite analisar o código ainda em execução, verificar e alterar o valor de suas variáveis em um determinado ponto específico:

