# Índice

## Contenido

Política de Nomenclatura para Desarrollo en JavaScript – BeMaster
24/01/2024

## 1. Introducción

En BeMaster, la consistencia en la nomenclatura de JavaScript es clave para un desarrollo ágil y eficiente. Estas pautas aseguran uniformidad en variables, funciones y clases, reflejando la identidad de BeMaster en nuestro código.

## 2. Nomenclatura de variables y funciones

### 2.1. Variables:

- Utilizar "camelCase" para variables locales.
- Utilizar "snake_case" para variables globales.
- Ser descriptivo y alinearse con la terminología del dominio de BeMaster.

### 2.2. Funciones/Métodos:

- Utilizar "camelCase".
- Nombrar de manera descriptiva, indicando la acción que realizan.
- Evitar funciones excesivamente largas o con múltiples responsabilidades.

## 3. Nomenclatura Clases

### 3.1. Clases:

- Utilizar "PascalCase".
- Nombrar de manera descriptiva y representar un objeto o entidad claramente definida

## 4. Nomenclatura en git

### 4.1 Ramas:

- Utilizar "kebab-case" para nombres de ramas.
- Ser descriptivo y reflejar el propósito de la rama.

### 4.2 Commits:

Mensajes en ingles cortos y descriptivos.

- Ser conciso y descriptivo.
- Incluir el ID del ticket de seguimiento si es aplicable de jira, azure, etc.

### 4.3 Ganchos de Commit:

- Utilizar "camelCase" para nombres de funciones en ganchos de commit.
- Ser descriptivo y reflejar la acción del gancho

## 5. Ejemplo practico

```
Diego Alvarez - Bemaster
// Ejemplo 1: Variables
let cantidadProductos = 0;  // Variable local en camelCase
let stock_global = 100;     // Variable global en snake_case

// Ejemplo 2: Funciones
function calcularTotalProductos(productos) {
    // Función que calcula el total de productos
    return productos.reduce((sum, producto) => sum + producto, 0);
}

// Ejemplo 3: Clases
class Producto {
    // Clase en PascalCase que representa un producto
    constructor(nombre, precio) {
        this.nombre = nombre;
        this.precio = precio;
    }

    calcularDescuento() {
        // Método para calcular el descuento del producto
        // ...
    }
}

// Ejemplo 4: Git Branch y Commit
// feature/nueva-funcionalidad
// git commit -m "BM-123: Add function to calculate total products"

// Ejemplo 5: Git Hook
function preCommitHook() {
    // Gancho de commit en camelCase para verificar estilos antes de hacer commit
    // Implementar lógica de verificación de estilos aquí
}

// Ejemplo 6: Mensajes de Commit
// git commit -m "BM-456: Fix error in stock validation"

// Ejemplo 7: Ramas de Git
// fix/issue-123
// feature/add-new-feature

// Ejemplo 8: Constantes
const MAX_DESCUENTO = 0.2;  // Descuento máximo permitido
```