**🔍 Diferença entre os dois padrões**

**✅ 1. Padrão atual do seu TelaRecebimento.js: objeto literal global exposto diretamente**

js

CopiarEditar

(function () {

window.FiberGuardian = window.FiberGuardian || {};

FiberGuardian.TelaRecebimento = {

init: function () { /\*...\*/ },

updateCalculations: function () { /\*...\*/ },

voltarParaInicio: function () { /\*...\*/ }

};

})();

🧠 O que acontece aqui:

* Você cria um **objeto global diretamente com três funções visíveis publicamente**.
* Não há funções internas privadas — tudo que está dentro do objeto é acessível de fora.
* O IIFE **garante apenas que a definição do objeto não vaze variáveis auxiliares para o escopo global**.

**✅ 2. Padrão "Revealing Module Pattern" (como você fez em Utils e TelaLogin):**

js

CopiarEditar

(function () {

window.FiberGuardian = window.FiberGuardian || {};

FiberGuardian.TelaLogin = (function () {

function autenticar() { /\* função privada \*/ }

function configurarEventos() { /\* função privada \*/ }

return {

init: configurarEventos // apenas init é público

};

})();

})();

🧠 O que acontece aqui:

* Você define **funções privadas no escopo fechado do módulo**.
* Só **expõe explicitamente o que precisa ser acessado externamente** via return.
* Isso **protege a superfície de ataque, reduz acoplamento** e favorece SRP e KISS.

**⚖️ Comparação técnica direta**

| **Critério** | **Objeto Literal Global** | **Revealing Module Pattern (RMP)** |
| --- | --- | --- |
| Encapsulamento | ❌ Nenhum (tudo é público) | ✅ Forte (funções privadas protegidas) |
| Controle de visibilidade | ❌ Tudo visível | ✅ Exposição seletiva via return |
| Prevenção contra vazamento | ✅ Sim (IIFE ajuda) | ✅ Sim |
| Extensibilidade futura | ⚠️ Pode ficar caótico | ✅ Mais fácil de refatorar |
| Alinhamento com SRP e KISS | ⚠️ Nem sempre claro | ✅ Naturalmente modular |
| Segurança (ex: esconder helpers) | ❌ Não protege funções sensíveis | ✅ Helpers privados não são acessíveis |
| Testabilidade isolada | ⚠️ Dificulta isolamento | ✅ Foco em unidade |

**📌 Qual é o melhor padrão para o seu projeto?**

**✅ Recomendo firmemente adotar o Revealing Module Pattern (RMP) como padrão oficial do projeto.**

Justificativas:

* 🧩 **Homogeneidade** entre telas (ex: TelaLogin, TelaRecebimento, TelaRelatorio etc.)
* 🔐 **Segurança**: reduz a superfície exposta do sistema no JS
* 🛠️ **Manutenibilidade**: fácil refatorar, dividir e evoluir sem quebrar contratos globais
* 👥 **Colaboração**: facilita onboarding de novos devs, pois o padrão é claro
* 🔄 **Testabilidade**: isola melhor cada função e permite mocks mais fáceis

**🛠️ Refatoração sugerida do seu TelaRecebimento.js**

js

CopiarEditar

(function () {

window.FiberGuardian = window.FiberGuardian || {};

FiberGuardian.TelaRecebimento = (function () {

function init() {

// código de inicialização

}

function updateCalculations() {

// lógica de atualização de cálculo

}

function voltarParaInicio() {

// redirecionamento ou limpeza

}

// Revealing: expõe apenas as funções públicas

return {

init: init,

updateCalculations: updateCalculations,

voltarParaInicio: voltarParaInicio

};

})();

})();

💡 Isso permite inclusive que você **torne init() o único ponto público** no futuro, se decidir esconder updateCalculations() e voltarParaInicio() e chamá-las internamente.