# Contexto isolado

## Entenda os dois contextos da mesma janela

### O que está isolado

Com contextIsolation, o preload e os scripts da página usam contextos JavaScript distintos dentro do renderer. Cada lado tem seu próprio objeto global window. Atribuir uma propriedade a window no preload só a torna visível naquele contexto.

### Como atravessar a fronteira

Use contextBridge.exposeInMainWorld para publicar os dados ou as funções que a página pode acessar. A ponte copia e congela valores expostos e cria funções intermediárias para chamadas. Uma propriedade de dados serve como fotografia do valor publicado. Consulte uma função específica quando precisar de dados atualizados.

### Configurações complementares

Mantenha contextIsolation e sandbox ativos e nodeIntegration desabilitado. O isolamento separa contextos JavaScript. O sandbox restringe o processo e a configuração de Node limita os recursos da página. Controle também quais páginas a janela pode carregar.

# Contexto isolado

## Publique somente a informação da build

### preload.cjs

```
const { contextBridge } = require("electron");

const diagnostico = { build: "homologacao" };
window.marcadorDoPreload = "privado";

contextBridge.exposeInMainWorld("agendaInfo", {
  nomeDaBuild: diagnostico.build,
});
```

### Ligue à janela

Use a janela local do guia Electron e informe o caminho absoluto para este preload em webPreferences.preload. A página pode ler window.agendaInfo.nomeDaBuild e mostrar o ambiente ao lado do título.

### Exponha dados apropriados para a interface

O valor homologacao foi escolhido para aparecer na tela. Separe dados de diagnóstico públicos de informações que a página não deve acessar. Quando publicar funções, limite cada uma à ação necessária para a tela.

# Contexto isolado

## Exercício resolvido

### Cenário

O renderer consegue ler o nome da build, mas window.marcadorDoPreload é undefined. Onde está o marcador? Qual valor a página deve usar para exibir o ambiente?

### Resolução

O marcador está no objeto global do preload. A página lê a propriedade publicada em window.agendaInfo.nomeDaBuild. Execute as verificações abaixo na página local, usando o contexto padrão do renderer no DevTools.

### Verificações no renderer

```
console.assert(window.marcadorDoPreload === undefined);
console.assert(window.agendaInfo.nomeDaBuild === "homologacao");
console.assert(Object.isFrozen(window.agendaInfo));
```

### Referências primárias

Electron — Context Isolation

Electron — contextBridge

# Contexto isolado

## Publique valores deliberadamente através da ponte

### Distinga uma fotografia de uma consulta

Valores publicados pela ponte são copiados e congelados. Use um valor fixo para a versão da build. Para retornar o status atual da agenda, exponha uma operação específica que busque o valor quando chamada.

### Exercício adicional

O nome do ambiente pode ser fixo, mas o estado da sincronização muda. Como expor cada um?

### Resolução

Publique o nome como valor configurado e exponha consultarSincronizacao como função. Verifique na janela que a função acompanha as mudanças entre duas leituras.

### Ao assistir ao vídeo

Observe onde o preload usa contextBridge para publicar métodos.

### Vídeo complementar

Electron & Context Bridge - IPC Deep Dive (em inglês)