

# JEX / JS Sandbox

A partir da versão **12.25**, foi disponibilizado no DOX uma alternativa ao PEX, possibilitando a codificação e a execução dos eventos de formulário (ao sair de um campo de formulário, ao sair da atividade, ao entrar na atividade, ao sair de um campo da grade) existentes em processos de negócio (BPM) com a linguagem **JavaScript**.

**OBS: Essa nova funcionalidade só está disponível para ambientes PostgreSQL e Oracle.**

E para isso, foi criado um novo servidor desenvolvido na tecnologia **Node.js** (JavaScript a nível de servidor), com o nome **JS Sandbox**, que possui diversos recursos disponíveis.

O nome **JS Sandbox** foi escolhido porque '**JS**' é a sigla para **JavaScript**, e o termo 'Sandbox' é geralmente utilizado na programação para se referir a um ambiente de execução de código fonte. Os desenvolvedores DOX utilizam o termo "JEX" para se referenciar ao novo recurso também, mantendo o padrão de três letras (PEX, EMA).

## Alterações no Ema DOX Estúdio

Segue as alterações feitas no Ema DOX Estúdio para possibilitar a codificação em JavaScript nos eventos de formulário.

Tela de edição de código fonte dos eventos de formulário.

Nessa tela, foi incluído um novo combo na parte superior com o nome 'Linguagem' e as opções 'Object Pascal' e 'JavaScript Node.js'. A opção 'Object Pascal', vai manter as funcionalidades que existiam anteriormente na tela, e continuará executando o evento no Servidor PEX. Mas com a opção 'JavaScript', a tela irá ser reajustada, modificando o cabeçalho, a aba lateral 'Assistente', o design do editor, e os atalhos do teclado.

- **Editor**

Foi aplicado o mesmo padrão de cores utilizado nas expressões, o Solarized. A codificação será feita dentro da assinatura da função, que foi incluída apenas por questões de indentação, e para deixar claro que se trata de uma função assíncrona. E o nome da função é fixo, variando de acordo com o evento, podendo ser '**aoSairCampoFormulario**', '**aoSairCampoGrade**', '**aoEntrarAtividade**' e '**aoSairAtividade**'.

A imagem a seguir mostra como ficará o editor ao entrar em um evento ao sair de um campo do formulário.

image1640633546996.png

- **Aba 'Assistente'**

Esse assistente possui o acesso facilitado a muitos dos recursos disponibilizados, no mesmo padrão do 'Assistente do processo' na linguagem 'Object Pascal'.

Ao clicar em algum dos assistentes, irá vincular no código fonte uma base inicial da funcionalidade em questão. O item 'Documentações' do assistente possui links para o acesso a documentação das tecnologias utilizadas, e a esse post do fórum.

[image-1640636061218.png](#)

Image not found or type unknown

- **Atalhos do teclado**

1. Ctrl + G → Vai focar no campo 'Ir para a linha' setando a linha focada.
2. Ctrl + ';' → Comentar ou descomentar.
3. Tab + texto sendo selecionado em múltiplas linhas → Indentação (2 caracteres).
4. F5 → Validar código fonte.
5. Ctrl + S → Salvar alterações.
6. Ctrl + Barra de espaço → Atalho do editor para completção de código e menu com acesso aos recursos disponibilizados no JS Sandbox.

Segue uma imagem que mostra como ficou a tela de edição de código fonte de eventos com a linguagem 'JavaScript Node.js')'

[image-1640636594671.png](#)

Image not found or type unknown

*Parâmetro geral para definir linguagem padrão dos eventos.*

## Principais Recursos

Para os eventos de formulário, o JS Sandbox oferece acesso a diversos recursos, incluindo recursos nativos da linguagem JavaScript, acesso a várias informações da atividade, acesso a arquivos e pastas, acesso a WebServices, acesso a banco de dados, inserção de log no processo, e assim por diante.

Acesso a informações da atividade.

- **Variáveis (variáveis)**

Possibilita o acesso a todas as variáveis de processo e de sistema. Exemplo:

image-1640638250923.png

Image not found or type unknown

Observação: Não é possível alterar o valor de variáveis de sistema.

- **Campos do formulário (formulário)**

Possibilita o acesso a todos os campos cadastrados no formulário da atividade.

*Observação: Está disponível em todos os eventos de formulário, com exceção do evento ao sair de um campo da grade.*

Acessando propriedades de um formulário

Acessadas no padrão → `formulario.IDENTIFICADOR.propriedade`.

Exemplo:

image-1640638503845.png

Image not found or type unknown

- **Valores do registro atual da grade (valoresRegistroAtual)**

Possibilita o acesso direto aos valores de cada coluna do registro atual da grade, no padrão → `valoresRegistroAtual.NOMECAMPO`.

Exemplo:

image-1640638578688.png

Image not found or type unknown

*Observação: Disponível apenas para o evento ao sair de um campo da grade.*

- **Objeto de mensagem (mensagem)**

Possibilita definir uma mensagem para aparecer na tela de colaboração da atividade.

Propriedades:

- **texto:** Define o texto da mensagem.
- **tipo:** Define o tipo da mensagem, que pode ser:
  - **Informação** TipoMensagem.INFO ;
  - **Erro** TipoMensagem.ERRO ;
  - **Alerta** (TipoMensagem.ALERTA);
- **timeout:** Define o timeout da mensagem em milissegundos.

Exemplo:

image-1640895135348.png

Image not found or type unknown

*Observação: As propriedades dos objetos da atividade (no padrão → recurso.IDENTIFICADOR.propriedade) foram padronizadas para serem acessadas em **CamelCase**.*

- Outras informações disponíveis: **idProcesso, idAtividade, idElementoFoco, dadosFK**, etc.
- **Acesso a arquivos e pastas (fsPromises):** Possibilita o acesso aos métodos para manipulação de arquivos e pastas do módulo **fs/promises** do Node.js.
- **Acesso a web services (axios):** Possibilita o acesso aos métodos disponíveis para requisições HTTP da biblioteca **Axios**.
- **Acesso a banco de dados (conexaoBD):** Possibilita a execução de um SQL no banco de dados local, ou em bancos de terceiros via conector.
  - Banco local utilizando os métodos **conexaoBD.executarComandoSQL** e **conexaoBD.executarConsultaSQL**.
    - **Bancos: Postgres e Oracle.**
  - Bancos de terceiros via conector utilizando os métodos **conexaoBD.executarComandoSQLPorConector** e **conexaoBD.executarConsultaSQLPorConector**.
    - **Bancos: Postgres, Oracle, MySQL e Firebird.**
  - **Observações:** Para o acesso a bancos Firebird via conector é necessário ter o ODBC instalado na máquina. Disponível em **ODBC Driver**. Em consultas SQL em qualquer banco local ou de terceiros via conector: Se a consulta resultar em apenas um registro, a função vai retornar um objeto, se resultar em mais do que um registro, o retorno será um array de objetos. E as propriedades dos objetos retornados precisam ser acessadas em minúsculo.
- **Inserir log no processo (logProcesso):** Possibilita a inserção de um log na instância do processo, podendo ser de informação (**logProcesso.info**) ou de erro (**logProcesso.erro**), ambos métodos possuem apenas o parâmetro 'mensagem'.

## Outros detalhes importantes

- Necessário habilitar o serviço 'Ema JS Sandbox' no Ema Configurador.
- O código JavaScript vai ser executado a nível de servidor (Node.js).
- A versão do Node.js mínima exigida é a **14**.
- JavaScript é **Case-sensitive**.
- O servidor JS Sandbox funciona de forma assíncrona, então os métodos de acesso a vários recursos são funções assíncronas, que quando executadas retornam Promises, necessitando então que seja colocado 'await' antes da chamada, ou tratar com '.then' depois da chamada. Recursos onde seus métodos retornam Promises:
  - **fsPromises**

- conexaoBD
- axios
- logProcesso
- Para qualquer erro gerado dentro do JS Sandbox, será inserido um log de erro no processo.
- O serviço tem um log interno, encontrado na pasta de instalação → 'js sandbox' → 'nest.log'.

## Links para mais informações:

[Usando Promises - JavaScript](#)

[Promise - JavaScript](#)

[Funções assíncronas - JavaScript](#)

[JavaScript Assíncrono](#)

---

Revisão #8

Criado 27 December 2021 16:26:17 por Nicolly Andrielly

Atualizado 1 September 2022 16:45:59 por Nicolly Andrielly