[Custom Page] Quais são as funções nativas do AgilityFlow disponíveis no JavaScript para utilizar em uma Custom Page? customPageContext

Introdução

Este script fornece um conjunto de funções para manipulação de contexto de página, carregamento dinâmico de componentes e arquivos, manipulação de mensagens, conversão de números e datas, e interação com URLs base.

Constantes

emptyGuid

Uma string que representa um GUID vazio:

```
var emptyGuid = "00000000-0000-0000-0000-000000000000";
```

Estrutura Principal

customPageContext

Objeto que gerencia funções relacionadas ao contexto da página.

parentFrame

Identifica e armazena a referência do iframe pai.

```
customPageContext.parentFrame
```

isMobile()

Retorna `true` se o dispositivo for móvel.

```
customPageContext.isMobile();
```

redirectTo(url)

Redireciona a página para a URL informada.

```
customPageContext.redirectTo('https://example.com');
```

openLightBox(link, tamanholightbox, titulo)

Abre um lightbox com a URL especificada.

```
customPageContext.openLightBox('/pagina.html', 'm', 'Exemplo');
```

Carregamento de Componentes CSHTML

Funções para carregar componentes de forma dinâmica:

```
customPageContext.loadCsHtmlComponent(componentName, options);
customPageContext.loadHtmlComponent(componentName, options);
customPageContext.loadPartialView(componentName, options);
customPageContext.loadViewComponent(componentName, options);
customPageContext.loadAjax(componentName, options);
customPageContext.loadAjaxComponent(componentName, options);
```

Veja um exemplo de uso para Inserir, Atualizar ou deletar um registro em um Formulário, clicando aqui

Exemplo passando parâmetros para o cshtml:

```
function loadPartialToUpdateRegister() {

    var cshtmlPartialName = "[coloque aqui o nome do partial cshtml que vc fez as regras de atualizar]";
    customPageContext.loadCsHtmlComponent(cshtmlPartialName,
        {
            placeholderId: 'result_cshtml', //parametro opcional, se o cshtml retornar algum html e vc quiser, vc pode colocar o retorno dentro de algum placeholder, div, etc..
            extraData: {
                idRegistro: $('#guidRegistro').val(),
                nome: $('#nome').val(),
                email: $('#email').val()
            },
            onSuccess: function (result) {
                customPageContext.msg.success("Executado com sucesso", "Confirmação");
            },
            onError: function (error) {
                customPageContext.msg.error("Execução não realizada", "Erro");
                console.log('error', error)
            }

        }
    )
}
```

Carregamento de Arquivos JavaScript

addJavascriptFile(options)

Adiciona um arquivo JavaScript à página de forma assíncrona e executa um callback após o carregamento.

- **Parâmetros:**
  - `options` (Object): Objeto com as propriedades:
    - `file_url` (String): URL do arquivo JavaScript.
    - `callback_onload` (Function): Função a ser executada após o carregamento do arquivo.
- **Exemplo:**

```
customPageContext.addJavascriptFile({
    file_url: "https://www.exemplo.com/script.js",
    callback_onload: function() {
        console.log("Script carregado!");
    }
});
```

Manipulação de URLs

url.getBaseUrl()

Retorna a URL base do sistema.

- **Retorno:**
  - String com a URL base.
- **Exemplo:**

```
console.log("URL base:", customPageContext.url.getBaseUrl());
```

url.getStaticFileUrl()

Retorna a URL base para arquivos estáticos.

- **Retorno:**
  - String com a URL de arquivos estáticos.
- **Exemplo:**

```
console.log("URL de arquivos estáticos:", customPageContext.url.getStaticFileUrl());
```

Idioma Atual

currentLanguage.get()

Retorna o idioma atual do sistema.

- **Retorno:**
  - String com o código do idioma (ex: `'pt-BR'`, `'en-US'`).
- **Exemplo:**

```
console.log("Idioma atual:", customPageContext.currentLanguage.get());
```

---

currentLanguage.isEnglish()

Verifica se o idioma atual é inglês.

- **Retorno:**
  - `true` se for inglês.
  - `false` caso contrário.
- **Exemplo:**

```
if (customPageContext.currentLanguage.isEnglish()) {
    console.log("Idioma atual é inglês.");
}
```

---

currentLanguage.getIIF_TextFromCurrentLanguage(text_ptBR, text_ENG, text_ESP)

Retorna o texto correspondente ao idioma atual.

- **Parâmetros:**
  - `text_ptBR` (String): Texto em português.
  - `text_ENG` (String): Texto em inglês.
  - `text_ESP` (String): Texto em espanhol.
- **Retorno:**
  - String com o texto no idioma atual.
- **Exemplo:**

```
const texto = customPageContext.currentLanguage.getIIF_TextFromCurrentLanguage("Olá", "Hello", "Hola");
console.log(texto); // Retorna "Olá" se o idioma for português.
```

Mensagens

O módulo `customPageContext.msg` fornece funções para exibir e ocultar mensagens de alerta, erro, sucesso e aviso no sistema. Abaixo estão as funcionalidades disponíveis.

Ocultar Mensagens

hide(forcarFechar)

Oculta todas as mensagens exibidas.

- **Parâmetros:**
  - `forcarFechar` (Boolean): Força o fechamento das mensagens.
- **Exemplo:**

```
customPageContext.msg.hide(true); // Oculta todas as mensagens, forçando o fechamento.
```

Exibir Mensagens

show(type, msg, title, options)

Exibe uma mensagem de alerta, erro, sucesso ou aviso.

- **Parâmetros:**
  - `type` (String): Tipo da mensagem. Valores possíveis:
    - `'warning'`: Mensagem de aviso.
    - `'error'`: Mensagem de erro.
    - `'success'`: Mensagem de sucesso.
  - `msg` (String): Texto da mensagem.
  - `title` (String): Título da mensagem.
  - `options` (Object): Opções adicionais para personalização da mensagem.
- **Exemplo:**

```
customPageContext.msg.show("success", "Operação realizada com sucesso!", "Sucesso", { timeout: 5000 })
```

---

warning(msg, title, options)

Exibe uma mensagem de aviso.

- **Parâmetros:**
  - `msg` (String): Texto da mensagem.
  - `title` (String): Título da mensagem.
  - `options` (Object): Opções adicionais.
- **Exemplo:**

```
customPageContext.msg.warning("Atenção: Este campo é obrigatório.", "Aviso");
```

---

error(msg, title)

Exibe uma mensagem de erro.

- **Parâmetros:**
  - `msg` (String): Texto da mensagem.
  - `title` (String): Título da mensagem.

- **Exemplo:**

```
customPageContext.msg.error("Ocorreu um erro ao salvar o formulário.", "Erro");
```

---

success(msg, title, options)

Exibe uma mensagem de sucesso.

- **Parâmetros:**
  - `msg` (String): Texto da mensagem.
  - `title` (String): Título da mensagem.
  - `options` (Object): Opções adicionais.
- **Exemplo:**

```
customPageContext.msg.success("Formulário salvo com sucesso!", "Sucesso", { timeout: 3000 });
```

Funções Legadas de Mensagens

As funções abaixo são mantidas para compatibilidade com versões anteriores, mas é recomendado utilizar as funções acima (`show`, `warning`, `error`, `success`).

hideMsgs()

Oculta todas as mensagens exibidas.

- **Exemplo:**

```
customPageContext.msg.hideMsgs();
```

---

showMsgWarning(msg, title, options)

Exibe uma mensagem de aviso (legado).

- **Parâmetros:**
  - `msg` (String): Texto da mensagem.
  - `title` (String): Título da mensagem.

- `options` (Object): Opções adicionais.
- **Exemplo:**

```
customPageContext.msg.showMsgWarning("Atenção: Este campo é obrigatório.", "Aviso");
```

---

showMsgError(msg, title)

Exibe uma mensagem de erro (legado).

- **Parâmetros:**
  - `msg` (String): Texto da mensagem.
  - `title` (String): Título da mensagem.
- **Exemplo:**

```
customPageContext.msg.showMsgError("Ocorreu um erro ao salvar o formulário.", "Erro");
```

---

showMsgSuccess(msg, title, options)

Exibe uma mensagem de sucesso (legado).

- **Parâmetros:**
  - `msg` (String): Texto da mensagem.
  - `title` (String): Título da mensagem.
  - `options` (Object): Opções adicionais.
- **Exemplo:**

```
customPageContext.msg.showMsgSuccess("Formulário salvo com sucesso!", "Sucesso", { timeout: 3000 });
```

Exemplos de Uso de Mensagem

Exibindo uma mensagem de sucesso:

```
customPageContext.msg.success("Dados salvos com sucesso!", "Sucesso", { timeout: 5000 });
```

Exibindo uma mensagem de erro:

```
customPageContext.msg.error("Erro ao processar a solicitação.", "Erro");
```

Ocultando todas as mensagens:

```
customPageContext.msg.hide(true);
```

Manipulação de Números

number.convertString_toNumber(strValueToConvert, qtdCasasDecimais)

Converte uma string formatada em número.

- **Parâmetros:**
  - `strValueToConvert` (String): Valor em formato de string.
  - `qtdCasasDecimais` (Number): Quantidade de casas decimais.
- **Retorno:**
  - Número convertido.
- **Exemplo:**

```
const numero = customPageContext.number.convertString_toNumber("1.000,50", 2);
console.log(numero); // Retorna 1000.50
```

---

number.convertNumber_toStringFormatted(numberToConvert, qtdCasasDecimais)

Converte um número em uma string formatada de acordo com o idioma.

- **Parâmetros:**
  - `numberToConvert` (Number): Número a ser convertido.
  - `qtdCasasDecimais` (Number): Quantidade de casas decimais.
- **Retorno:**
  - String formatada.
- **Exemplo:**

```
const texto = customPageContext.number.convertNumber_toStringFormatted(1000.50, 2);
console.log(texto); // Retorna "1.000,50" em português.
```

Manipulação de Datas

isCurrentMonthAndYear(data)

Verifica se a data fornecida pertence ao mês e ano atuais.

- **Parâmetros:**
  - `data` (String): Data no formato `DD/MM/YYYY`.
- **Retorno:**
  - `true` se a data for do mês e ano atuais.
  - `false` caso contrário.
- **Exceções:**
  - Lança um `TypeError` se a data estiver em um formato inválido.
- **Exemplo:**

```
customPageContext.datetime.validation.isCurrentMonthAndYear("25/10/2023"); // Retorna true se outubro de
```

---

compare(date1, compare, date2)

Compara duas datas com base no operador especificado.

- **Parâmetros:**
  - `date1` (String): Primeira data no formato `DD/MM/YYYY`.
  - `compare` (String): Operador de comparação. Valores possíveis:
    - `'greater'` ou `'>'`
    - `'greater-or-equal'` ou `'>='`
    - `'less'` ou `'<'`
    - `'less-or-equal'` ou `'<='`
    - `'equal'` ou `'=='`
  - `date2` (String): Segunda data no formato `DD/MM/YYYY`.
- **Retorno:**
  - `true` ou `false`, dependendo da comparação.

- **Exceções:**
  - Lança um `TypeError` se as datas estiverem em formato inválido ou se o operador de comparação não for reconhecido.
- **Exemplo:**

```
customPageContext.datetime.validation.compare("25/10/2023", "greater", "20/10/2023"); // Retorna true.
```

`compareDatetime(datetime1, compare, datetime2)`

Compara duas datas e horários com base no operador especificado.

- **Parâmetros:**
  - `datetime1` (String): Primeira data e horário no formato `DD/MM/YYYY HH:mm`.
  - `compare` (String): Operador de comparação. Valores possíveis:
    - `'greater'` ou `'>'`
    - `'greater-or-equal'` ou `'>='`
    - `'less'` ou `'<'`
    - `'less-or-equal'` ou `'<='`
    - `'equal'` ou `'=='`
  - `datetime2` (String): Segunda data e horário no formato `DD/MM/YYYY HH:mm`.
- **Retorno:**
  - `true` ou `false`, dependendo da comparação.
- **Exceções:**
  - Lança um `TypeError` se os dados estiverem em formato inválido ou se o operador de comparação não for reconhecido.
- **Exemplo:**

```
customPageContext.datetime.validation.compareDatetime("25/10/2023 14:30", "greater", "25/10/2023 12:00
```

`compareToday(compare, date2)`

Compara a data atual com uma data fornecida.

- **Parâmetros:**
  - `compare` (String): Operador de comparação. Valores possíveis:
    - `'greater'` ou `'>'`
    - `'greater-or-equal'` ou `'>='`
    - `'less'` ou `'<'`

    - `'less-or-equal'` ou `'<='`
    - `'equal'` ou `'=='`
  - `date2` (String): Data no formato `DD/MM/YYYY`.
- **Retorno:**
  - `true` ou `false`, dependendo da comparação.
- **Exemplo:**

```
customPageContext.datetime.validation.compareToday("less", "30/10/2023"); // Retorna true se a data atual
```

Adição de Tempo

day(data, days)

Adiciona dias a uma data.

- **Parâmetros:**
  - `data` (String): Data no formato `DD/MM/YYYY` ou `DD/MM/YYYY HH:mm`.
  - `days` (Number): Número de dias a serem adicionados.
- **Retorno:**
  - Data resultante no mesmo formato da entrada.
- **Exemplo:**

```
customPageContext.datetime.add.day("25/10/2023", 5); // Retorna "30/10/2023".
```

---

month(data, months)

Adiciona meses a uma data.

- **Parâmetros:**
  - `data` (String): Data no formato `DD/MM/YYYY` ou `DD/MM/YYYY HH:mm`.
  - `months` (Number): Número de meses a serem adicionados.
- **Retorno:**
  - Data resultante no mesmo formato da entrada.
- **Exemplo:**

```
customPageContext.datetime.add.month("25/10/2023", 2); // Retorna "25/12/2023".
```

year(data, years)

Adiciona anos a uma data.

- **Parâmetros:**
  - `data` (String): Data no formato `DD/MM/YYYY` ou `DD/MM/YYYY HH:mm`.
  - `years` (Number): Número de anos a serem adicionados.
- **Retorno:**
  - Data resultante no mesmo

---

hour(data, hours)

Adiciona horas a uma data e horário.

- **Parâmetros:**
  - `data` (String): Data e horário no formato `DD/MM/YYYY HH:mm`.
  - `hours` (Number): Número de horas a serem adicionadas.
- **Retorno:**
  - Data e horário resultante no formato `DD/MM/YYYY HH:mm`.
- **Exemplo:**

```
customPageContext.datetime.add.hour("25/10/2023 14:30", 3); // Retorna "25/10/2023 17:30".
```

---

minute(data, minutes)

Adiciona minutos a uma data e horário.

- **Parâmetros:**
  - `data` (String): Data e horário no formato `DD/MM/YYYY HH:mm`.
  - `minutes` (Number): Número de minutos a serem adicionados.
- **Retorno:**
  - Data e horário resultante no formato `DD/MM/YYYY HH:mm`.
- **Exemplo:**

```
customPageContext.datetime.add.minute("25/10/2023 14:30", 15); // Retorna "25/10/2023 14:45".
```

Obtenção de Datas

getLastDayOfCurrentMonth()

Retorna o último dia do mês atual.

- **Retorno:**
  - Data no formato `DD/MM/YYYY`.
- **Exemplo:**

```
customPageContext.datetime.getLastDayOfCurrentMonth(); // Retorna "31/10/2023" se outubro for o mês atu
```

---

getLastDayOfMonth(data)

Retorna o último dia do mês da data fornecida.

- **Parâmetros:**
  - `data` (String): Data no formato `DD/MM/YYYY`.
- **Retorno:**
  - Data no formato `DD/MM/YYYY`.
- **Exemplo:**

```
customPageContext.datetime.getLastDayOfMonth("25/10/2023"); // Retorna "31/10/2023".
```

Data e Horário Atuais

getDateNow()

Retorna a data e horário atuais.

- **Retorno:**
  - Objeto `Date` representando a data e horário atuais.
- **Exemplo:**

```
customPageContext.datetime.getDateNow(); // Retorna a data e horário atuais.
```

---

getFormattedDateTime()

Retorna a data e horário atuais formatados.

- **Retorno:**
  - String no formato `DD/MM/YYYY HH:mm`.
- **Exemplo:**

```
customPageContext.datetime.getFormattedDateTime(); // Retorna "25/10/2023 14:30".
```

---

getFormattedDate()

Retorna a data atual formatada.

- **Retorno:**
  - String no formato `DD/MM/YYYY`.
- **Exemplo:**

```
customPageContext.datetime.getFormattedDate(); // Retorna "25/10/2023".
```

Revision #11
Created 5 March 2025 12:32:20 by agilityflow

Updated 6 March 2025 11:39:20 by agilityflow