SOBRE ESTE DOCUMENTO

Este documento expõe, em linhas gerais, como se utilizar os modelos prontos de códigos HTML e de "*scripts para servidor*" do Projeto Caracol, que estão incluídos na pasta "Modelos". Estes modelos foram criados buscando auxiliar na criação de telas sinópticas melhores e mais bonitas no ScadaBR, mesmo por aqueles que não têm grande experiência com HTML, CSS e Javascript.

COMO USAR OS MODELOS HTML

Os modelos HTML apresentam trechos de códigos HTML que podem ser usados para controlar algumas funções do ScadaBR. Para usá-los corretamente, é bom observar os comentários do código HTML (um comentário em HTML começa com **<!--** e termina com **-->**). Na seção "MODELOS DE HTML", deste documento, será apresentada uma descrição detalhada do funcionamento de cada modelo.

COMO USAR OS MODELOS DE SCRIPT PARA SERVIDOR

Os modelos de *scripts para servidor* foram desenvolvidos para possuírem duas partes distintas. A primeira possui apenas variáveis de configuração, pelas quais é possível controlar a aparência e comportamento do elemento gerado. A segunda parte é o código que gera o elemento em si, a partir daquilo que foi configurado na primeira parte. Assim, tem-se um código facilmente reutilizável e que não necessita de grandes conhecimentos de Javascript para sua utilização.

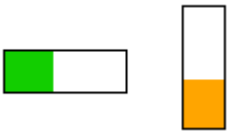
As duas partes de um script são separadas por uma linha de comentário contendo o texto:

/ / NÃO ALTERE A PARTIR DAQUI

Na próxima seção deste documento será apresentada uma breve descrição de cada modelo desenvolvido e uma tabela com suas variáveis de configuração, os valores para cada variável e uma descrição daquilo que a variável controla no script.

MODELOS DE SCRIPT PARA SERVIDOR

1. BARRA DE PROGRESSO

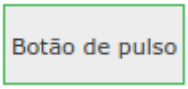


Nome do arquivo: barra.js

Descrição: Este modelo gera uma barra de progresso simples, de dimensões e cores configuráveis.

Variável de configuração	Tipo	Descrição
maximo	Numérico	Valor máximo da escala a ser representada
minimo	Numérico	Valor mínimo da escala a ser representada
altura	Numérico	Altura da barra, em pixels.
largura	Numérico	Largura da barra, em pixels.
cor_barra	String (texto)	Cor da barra (use um nome de cor em inglês ou cor hexadecimal, segundo o padrão para cores HTML)
modo	String (texto)	Dois valores possíveis: "vertical" ou "horizontal". Define a orientação da barra de progresso.
inverter_sentido	Booleano	Se **true**, faz com que o preenchimento da barra ocorra em sentido invertido (de baixo para cima/ da direita para a esquerda).

2. BOTÃO DE PULSO

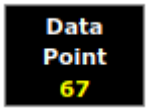


Nome do arquivo: botão de pulso.js

Descrição: Este modelo gera um botão de pulso, cujo estilo, texto e tempo de pulso podem ser configurados.

Variável de configuração	Tipo	Descrição
descricao	String (texto)	É o texto que será exibido dentro do botão pelo navegador.
altura	Numérico	Altura do botão, em pixels.
largura	Numérico	Largura do botão, em pixels.
estilo_do_scadabr	Booleano	Se **true**, faz o botão geral seguir o estilo CSS padrão usado no ScadaBR. Caso contrário, o estilo do botão irá variar conforme cada navegador.
valor_padrao	Booleano	O valor que será escrito quando o botão for pressionado. Se configurado como **false**, o botão irá se comportar como um botão físico NA (normalmente aberto). Se configurado como **true**, o botão irá se comportar como um botão físico NF (normalmente fechado).
janela_confirmacao	Booleano	Se **true**, será exibida uma janela de confirmação após pressionar o botão. Isso é útil para evitar cliques acidentais.

3. MOSTRADOR DE DATAPOINT



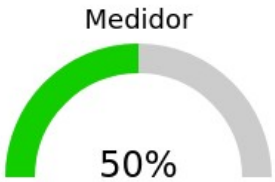
Nome do arquivo: datapoint.js

Descrição: Este modelo gera um mostrador para *datapoints* personalizável, que pode ser uma alternativa mais elegante ao componente *data point simples* do ScadaBR.

Variável de configuração	Tipo	Descrição
descricao_do_datapoint	String (texto)	Uma descrição a ser exibida acima do valor do datapoint, que pode ser livremente definido.
prefixo	String (texto)	Um prefixo, livremente configurável, que será adicionado antes do valor do datapoint.
sufixo	String (texto)	Um sufixo, livremente configurável, que será adicionado após o valor do datapoint.
casas_decimais	Numérico	Número de casas decimais a exibir no valor do datapoint. Use zero para não exibir casas decimais.
largura_minima	Numérico	Largura mínima, em pixels, que o mostrador terá. Esse valor não será respeitado caso o texto seja mais largo que o valor definido.
altura_minima	Numérico	Altura mínima, em pixels, que o mostrador terá. Esse valor não será respeitado caso o texto seja mais alto que o valor definido.

Variável de configuração	Tipo	Descrição
cor_fundo	String (texto)	Cor que será exibida como fundo do mostrador (use um nome de cor em inglês ou cor hexadecimal, segundo o padrão para cores HTML)
cor_descricao	String (texto)	Cor do texto da descrição do datapoint (use um nome de cor em inglês ou cor hexadecimal, segundo o padrão para cores HTML)
cor_valores	String (texto)	Cor do texto dos valores do datapoint (use um nome de cor em inglês ou cor hexadecimal, segundo o padrão para cores HTML)
tamanho_fonte	Numérico	Tamanho da fonte, em pixels, a ser usada tanto na descrição como nos valores.
usar_negrito	Booleano	Se **true**, a fonte da descrição e dos valores será exibida em negrito.

4. MEDIDOR CIRCULAR (GAUGE) - REQUER HTML 5



Nome do arquivo: gauge.js

Descrição: Este modelo gera um medidor em forma de semicírculo (gauge), como alternativa ao *gif analógico* "dial" do ScadaBR. Não funciona em navegadores sem suporte ao elemento HTML5 <canvas>.

Variável de configuração	Tipo	Descrição
descricao	String (texto)	Uma descrição a ser exibida acima do medidor.
prefixo	String (texto)	Um prefixo, livremente configurável, que será adicionado antes do valor exibido no medidor.
sufixo	String (texto)	Um sufixo, livremente configurável, que será adicionado após o valor exibido no medidor.
casas_decimais	Numérico	Número de casas decimais a exibir no valor do medidor. Use zero para não exibir casas decimais.
tamanho	Numérico	Define a largura do medidor. A altura e o tamanho de outros elementos são automaticamente definidos.
maximo	Numérico	Valor máximo da escala a ser representada
minimo	Numérico	Valor mínimo da escala a ser representada

Variável de configuração	Tipo	Descrição
cor_do_fundo	String (texto)	Cor a ser exibida como fundo do medidor. Se deixada como uma *string vazia* (" "), o fundo será transparente. Você também pode usar um nome de cor em inglês ou cor hexadecimal, segundo o padrão para cores HTML.
cor_do_texto	String (texto)	Cor a ser usada nos textos do medidor (use um nome de cor em inglês ou cor hexadecimal, segundo o padrão para cores HTML).
cor_padrao	String (tetxo)	Cor a ser exibida, por padrão, como o preenchimento da barra do medidor (use um nome de cor em inglês ou cor hexadecimal, segundo o padrão para cores HTML).
usar_cores_extra	Booleano	Se **true**, habilita a funcionalidade de "cores extra" para o medidor.
cores_extra	Array de strings (textos)	Define cores extras para quando o valor de leitura ultrapassar certa faixa. Esta variável é um array. Isto significa que ela armazena vários valores entre um par de colchetes. Esses valores são separados entre si por vírgulas. Use as strings no formato "cor:valor", em que *cor* é a cor HTML a ser exibida na barra e *valor* é o valor que a escala deve atingir para começar a usar tal cor.

5. SELETOR NUMÉRICO

Nome do arquivo: seletor numérico.js

Descrição: Este modelo gera uma entrada numérica pela qual podem ser escritos valores no *datapoint* associado.

Variável de configuração	Tipo	Descrição
altura	Numérico	Altura, em pixels, do seletor.
largura	Numérico	Largura, em pixels, do seletor.
cor_fundo	String (texto)	Cor a ser exibida como fundo da caixa do seletor (use um nome de cor em inglês ou cor hexadecimal, segundo o padrão para cores HTML).
cor_texto	String (texto)	Cor a ser exibida no texto do seletor (use um nome de cor em inglês ou cor hexadecimal, segundo o padrão para cores HTML).

Variável de configuração	Tipo	Descrição
restringir_valores	Booleano	Se **true**, permite apenas que sejam escritos valores numéricos que correspondam a um padrão predeterminado através das variáveis "multiplos_de", "minimo" e "maximo".
multiplos_de	Numérico	Define que só serão aceitos números que sejam múltiplos desse valor. Use 1 para aceitar qualquer número inteiro.
minimo	Numérico	Define um valor mínimo para a entrada. Valores menores serão rejeitados.
maximo	Numérico	Define um valor máximo para a entrada. Valores menores serão rejeitados.

6. LISTA DE OPÇÕES

Nome do arquivo: lista.js

Descrição: Este modelo gera uma lista de seleção no estilo *drop-down* para escrever valores predeterminados no *datapoint* associado.

Variável de configuração	Tipo	Descrição
opcoes	Array de strings (textos)	Define qual serão as opções a serem exibidas na lista, e os valores associadas a cada uma. Esta variável é um array. Isto significa que ela armazena vários valores entre um par de colchetes. Esses valores são strings separadas entre si por vírgulas. Use as strings no formato "texto:valor", em que *texto* é o texto a ser exibido na opção e *valor* é o valor associado à opção.

7. TEXTO BINÁRIO

Nome do arquivo: texto binário.js

Descrição: Este modelo gera um texto que pode ser alterado conforme o valor binário do *datapoint* associado.

Variável de configuração	Tipo	Descrição
texto_true	String (texto)	Texto, livremente configurável, que será exibido quando o valor do datapoint for **true**.

Variável de configuração	Tipo	Descrição
texto_false	String (texto)	Texto, livremente configurável, que será exibido quando o valor do datapoint for **false**.
cor_true	String (texto)	Cor que o texto terá quando o valor do datapoint for **true** (use um nome de cor em inglês ou cor hexadecimal, segundo o padrão para cores HTML).
cor_false	String (texto)	Cor que o texto terá quando o valor do datapoint for **false** (use um nome de cor em inglês ou cor hexadecimal, segundo o padrão para cores HTML).
tamanho_fonte	Numérico	Define o tamanho, em pixels, da fonte a ser usada, independentemente do valor do datapoint.
usar_negrito	Booleano	Se **true**, o texto será escrito em negrito, independentemente do valor do datapoint.
permitir_clique	Booleano	Se **true**, permitirá que o texto se comporte como um botão, alternando o valor do datapoint a ser clicado.
janela_confirmacao	Booleano	Se **true** e se a opção "permitir_clique" estiver ativada, exibirá uma janela de confirmação antes de alterar o valor do datapoint.

8. IMAGEM BINÁRIA

Nome do arquivo: imagem binária.js

Descrição: Este modelo gera duas imagens diferentes dependendo do estado binário do *datapoint* associado. Também pode ser usado para gerar um botão a partir de uma imagem personalizada.

Variável de configuração	Tipo	Descrição
imagem_true	String (texto)	Caminho da imagem a ser exibida quando o valor do datapoint for **true**. Pode ser um caminho local (ex.: "/ScadaBR/images/house.png") ou um link para uma imagem na internet.
imagem_false	String (texto)	Caminho da imagem a ser exibida quando o valor do datapoint for **false**. Pode ser um caminho local (ex.: "/ScadaBR/images/house.png") ou um link para uma imagem na internet.
permitir_clique	Booleano	Se **true**, permitirá que a imagem se comporte como um botão, alternando o valor do datapoint a ser clicado.

9. ANIMAÇÃO SIMPLES (SEQUÊNCIA DE IMAGENS)

Nome do arquivo: animação simples.js

Descrição: Este modelo gera uma sequência de imagens que podem ser utilizadas para gerar uma animação simples.

Variável de configuração	Tipo	Descrição
descricao	String (texto)	Um texto pop-up de descrição que será exibido quando o mouse for mantido sobre a imagem.
imagem_padrao	String (texto)	Caminho para a imagem a ser usada por padrão, quando nenhuma outra imagem atender às condições predeterminadas. Pode ser um caminho local (ex.: "/ScadaBR/images/house.png") ou um link para uma imagem na internet.
usar_valor_exato	Booleano	Se **true**, as imagens definidas no array "imagens" só serão selecionadas se o valor definido for exatamente igual ao valor definido no array. Caso contrário o valor será considerado como um valor mínimo a partir do qual a imagem será exibida.
imagens	Array de strings (textos)	Define qual serão as imagens a serem exibidas, e os valores associados a cada uma. Esta variável é um array. Isto significa que ela armazena vários valores entre um par de colchetes. Esses valores são strings separadas entre si por vírgulas. Use as strings no formato "imagem:valor", em que *imagem* é a imagem a ser exibida e *valor* é o valor associado à opção.

10. TOCAR MÚSICA (BINÁRIO) - REQUER HTML5

Nome do arquivo: música binária.js

Descrição: Este modelo permite reproduzir um arquivo de áudio quando o valor binário do *datapoint* mudar para o valor predefinido. Não funciona em navegadores sem suporte ao elemento HTML5 <audio>.

Nota: não se recomenda utilizar este componente como uma alternativa ao sistema de alarmes do ScadaBR, pois ele não funciona em outras páginas fora da Representação Gráfica!

Variável de configuração	Tipo	Descrição
arquivo_audio	String (texto)	Caminho para o áudio a ser reproduzido.

Variável de configuração	Tipo	Descrição
valor_disparo	Binário	O valor binário que irá disparar a reprodução do áudio.
ativar_loop	Binário	Se **true**, o áudio será repetido indefinidamente enquanto o datapoint se mantiver no valor de disparo. Se **false**, o áudio será reproduzido apenas uma vez.

11. TABELA DE USUÁRIOS ATIVOS

ID	Usuário	E-mail	É admin?	Último login
1	admin	admin@example.com	Sim	12:52 23/11/2020

Nome do arquivo: tabela usuários ativos.js

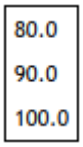
Descrição: Este modelo gera uma tabela com os usuários ativos do ScadaBR. Pode ser associado a qualquer *datapoint* ativo.

Nota: existe diferença entre usuários *ativos* e usuários *on-line*. Esta tabela mostra todos os usuários do ScadaBR que estão ativos (podem realizar *login* no sistema). Para uma solução paliativa para ver os usuários *on-line*, consulte o vídeo extra "Tabela de Usuários On-Line" do Projeto Caracol.

Variável de configuração	Tipo	Descrição
mostrar_id	Booleano	Deixe como **true** para exibir o *id* do usuário na tabela.
mostrar_email	Booleano	Deixe como **true** para exibir o e-mail do usuário na tabela.
mostrar_telefone	Booleano	Deixe como **true** para exibir o telefone do usuário na tabela.
mostrar_admin	Booleano	Deixe como **true** para exibir na tabela se o usuário é administrador.
mostrar_login	Booleano	Deixe como **true** para exibir a hora do último *login* do usuário na tabela.
cor_cabecalho	String (texto)	Define a cor de fundo que será usada no cabeçalho da tabela (use um nome de cor em inglês ou cor hexadecimal, segundo o padrão para cores HTML).
cor_texto_cabecalho	String (texto)	Define a cor que será usada no texto do cabeçalho da tabela (use um nome de cor em inglês ou cor hexadecimal, segundo o padrão para cores HTML).
tamanho_fonte	Numérico	Tamanho da fonte, em pixels, a ser usada na tabela.

Variável de configuração	Tipo	Descrição
largura	Numérico	Define a largura, em pixels, da tabela.
usar_css_externo	Booleano	Se **true**, as variáveis que definem o estilo da tabela serão ignoradas e a tabela usará regras externas de CSS para definir seu estilo.
classe_css	String (texto)	Define a classe CSS que a tabela possuirá para poder receber um estilo CSS externo. Só tem efeito caso a opção "usar_css_externo" esteja ativada.

12. TABELA DOS ÚLTIMOS VALORES DE UM PONTO – EXPERIMENTAL



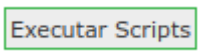
Nome do arquivo: últimos valores.js

Descrição: Este modelo gera uma tabela simples com os últimos *n* valores do *datapoint* associado. É um modelo experimental que visa apenas demonstrar como é possível usar o método *getLatestPointValues()* para ler *datapoints*.

Variável de configuração	Tipo	Descrição
numero_valores	Numérico	Número de quantos valores do datapoint serão lidos (ex.: use **10** para ler os 10 últimos valores

MODELOS DE HTML

1. EXECUTAR SCRIPTS

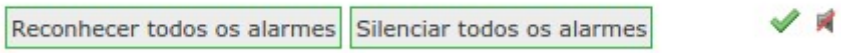


Nome do arquivo: executar scripts.html

Descrição: Este modelo possui 3 partes. A primeira é o código HTML para gerar um botão que execute a função Javascript *runScripts()*. A segunda é um código que também gera um botão para executar a mesma função. A diferença entre os dois códigos é que o primeiro gera um botão que irá assumir o estilo CSS definido por cada navegador,

enquanto o segundo gera um botão que irá utilizar o estilo CSS padrão dos botões do ScadaBR. Por fim, a terceira parte é a mais importante, nela está o código Javascript necessário para criar a função *runScripts()*, que pode ser usada para executar um ou mais scripts internos do ScadaBR (a função de *Scripting*, não confundir com os *scripts para servidor*). Neste código, você deve inserir o XID de cada script dentro do *array* "scripts".

2. GERENCIAR ALARMES (SILENCIAR/RECONHECER)



Nome do arquivo: gerenciar alarme.html

Descrição: Este modelo possui 2 partes. A primeira parte consiste de um código HTML que gera 2 botões. O primeiro tem a função de reconhecer todos os alarmes ativos no ScadaBR, enquanto o segundo tem a função de silenciar todos os alarmes ativos no ScadaBR. A segunda parte gera um código parecido e com as mesmas funções. A única diferença é que, em vez de botões, são criadas imagens que podem ser clicadas, como se fossem botões. Este código é especialmente útil por reproduzir as imagens de reconhecer/silenciar alarmes presentes na “Página de Alarmes” do ScadaBR, mas ausentes no componente “Lista de Alarmes” das Representações Gráficas.

MODELO COMPOSTO - DATAPOINT "ZUMBI"

O modelo a seguir contém duas partes de código: uma que deve ser inserida na Representação Gráfica como HTML (arquivo **zumbi.html**) e uma que deve ser inserida como *script para o servidor* (arquivo **zumbi.js**). Este modelo cria um *datapoint* "zumbi", isto é, um *datapoint* que pode ser controlado (lido e escrito) por outros elementos HTML da página, através das funções *readZombie()* e *writeZombie()*.

A função deste modelo consiste principalmente em simplificar a criação de componentes na Representação Gráfica, visto que usar um código HTML que se comunique com um *datapoint* zumbi muitas vezes pode ser mais fácil que converter elementos HTML em *strings* para componentes *script para o servidor*.

Como utilizar:

- Insira o arquivo **zumbi.html** apenas uma vez na Representação Gráfica, como componente HTML.
- Por sua vez, o arquivo **zumbi.js** deve ser inserido como *script para servidor* uma vez para cada *datapoint* a ser controlado, associando o *script para servidor* ao *datapoint* em questão. Este arquivo possui uma variável de configuração booleana chamada “invisivel”, que quando **true** deixa o componente completamente invisível na Representação Gráfica.
- Uma vez inseridos os arquivos, será possível usar as funções **readZombie(***idZumbi***)** e **writeZombie(***idZumbi* **,** *valor***)**, em que *idZumbi* é o XID do *datapoint* (ao menos que isso tenha sido alterado pela variável “id_zumbi” no arquivo **zumbi.js**) e *valor* é o valor a ser escrito no *datapoint*.

REFERÊNCIAS

O modelo de “Tabela de Usuários Ativos” foi baseado nas informações deste tópico do fórum do ScadaBR:

Fórum ScadaBR - Utilizando script-servidor para visualizar os usuários do sistema: http://forum.scadabr.com.br/t/utilizando-script-servidor-para-visualizar-os-usuarios-do-sistema/2045