Como funciona o código da palavra cruzada

Primeiramente, as variáveis globais:

currentTextInput: É a variável global que vai armazenar o ID da célula do caça palavras que está selecionada. Veremos ela daqui a um tempo

puzzelArrayData: É a variável que armazena os valores da palavra cruzada.

Agora, a função que inicializa a palavra cruzada, initializeScreen(), ela já é chamada quando a página carrega, para a palavra cruzada ser criada na mesma hora.

Nessa função a gente cria a variável puzzleTable, que é a nossa tabela, onde nossa palavra cruzada vai ficar.

Após isso, nós atribuímos os valores da palavra cruzada para a variável puzzleArrayData, através da função preparePuzzleArray(). Mas como isso funciona? Pra isso, nós vamos para a função preparePuzzleArray() para ver o que tem lá.

Na função preparePuzzleArray(), começamos com uma variável items, que é o conteúdo da nossa palavra cruzada. Item é um vetor, que possúi vetores dentro de si. Cada vetor dentro de item é uma linha da palavra cruzada, e cada ocorrência dentro dessa linha é o valor da célula na palavra cruzada.

Depois desse vetor, temos o return item, onde faz com que a função retorne o valor da variável item, quando essa função for chamada lá em cima pela função initializeScreen().

Voltando para a função initializeScreen(), nós temos um for, que nós vamos usar para construir a nossa palavra cruzada em si. Este for será feito até terminar a quantidade de arrays dentro da variável puzzleArrayData.

Dentro desse for nós criamos duas variáveis, a variável row, e a variável rowData. Nós criamos uma linha para nossa tabela, utilizando puzzelTable.insertRow(-1). Além disso, atribuímos essa nova linha para a variável row. Após isso, nós damos os valores das células dessa linha para a variável rowData.

Depois disso, nós vamos para outro for, que vai inserir as células da palavra cruzada na tabela. Nós começamos com a variável cell, que recebe o comando row.insertCell(-1), que insere na linha que está uma mais nova célula.

Após criarmos a variável cell, nós checamos para ver se o valor dela correspondente na palavra cruzada é igual a 0, pois, caso for igual a 0, não é considerado como parte da palavra cruzada. Porém, caso for diferente de 0, ela faz parte da palavra cruzada.

Então, nós criamos a variável txtID, para conseguir dar uma identificação única para essa célula, baseada na posição da célula na tabela, ou seja, baseada em qual linha e coluna ela está. Primeiro nós começamos com txt, e depois adicionamos a linha em que está, para então adicionar sua posição nessa linha. Ou seja, uma célula na linha 5 e na posição 3, terá o id de txt\_5\_3 (lembrando que a contagem começa a partir do 0, então a linha 5 seria a 6ª linha, e a célula 3 seria a 4ª célula).

Após isso, nós inserimos na nossa célula o input para poder escrever na nossa célula. Nós damos o tipo como texto, damos a classe de inputbox, para se relacionar com o css, damos uma maxlength de 1, para apenas poder acomodar 1 caracter, aplicamos um style para transformar o texto em lowercase, e colocamos nosso id da caixa, que é o txtID. Além disso, nós referenciamos a função textInputFocus(txtID), que é usada para conseguir o id da célula que está focada no momento. Esta função será útil para o futuro.

Ao finalizar ambos for, nós chamamos a função addHint(), que funciona de uma maneira simples. Nós pegamos o id das células desejadas, e damos um placeholder para ela.

Agora nós temos a função textInputFocus(txtID123), que armazena qual o id da célula que está focada, dando esse valor para a variável currentTextInput.

Depois disso, nós temos a função preparePuzzleArray(), que já foi comentada anteriormente, que dá os valores das células da palavra cruzada.

Após essa função, nós temos clearAllClicked(), que reinicia o puzzle, limpando tudo que contém nele. Ela começa a remover o conteúdo da variável currenttextInput, além de limpar toda a tabela que está nossa palavra cruzada. Após ambos, ela chama a função initializeScreen(), para reconstruir nossa palavra cruzada.

Agora nós temos a função checkClicked(), que verifica se suas respostas estão corretas. Nós fazemos isso usando uma varredura das células da palavra cruzada, usando um for para saber em qual linha ela está, e outro for para saber a posição da célula nessa linha. Após isso, nós verificamos se o conteúdo de cada célula que nós estamos vendo é igual a 0, pois, caso for, nada será feito, e iremos para a próxima célula. Caso for diferente de 0, é criada uma variável chamada selectedInputTextElement, que pega o id da célula que iremos checar se está correta.

Então, vamos para nossa estrutura de decisão, que verifica se o valor da variável selectedInputTextElement for igual à ocorrência que a corresponde na matriz, dá uma cor de fundo verde, caso o contrário, fica vermelho.

Depois dessa função, temos a função da dica, que dá a dica para a célula selecionada. Ela começa verificando se existe uma célula selecionada, para conseguir funcionar. Caso não tenha uma célula selecionada, a dica não funciona. Após isso, guardamos o id da célula selecionada em uma mova variável, para podermos trabalhar com mais facilidade. Depois disso, nós usamos o método split.

O método split funciona de maneira simples. Ele divide uma string em uma lista ordenada de substrings, e coloca essas substrings em um array, e retorna o array. Então, no nosso código, esse método está dividindo o id usando \_, ou seja, um id que é txt\_5\_3, agora virou um array [txt,5,3]. Após isso, nós temos duas variáveis, a variável row, que armazena a primeira ocorrência, que vira a linha, e a variável column, que é a segunda ocorrência, que vira a coluna. Após isso, nós procuramos a célula que está em foco, e nós damos para ela o valor que está atrelado à ela na matriz.

A próxima função é a solveClicked(), que tem o objetivo de resolver a palavra da célula selecionada. Ela começa da mesma maneira que a função clueClicked, porém, com a diferença que precisamos ter mais células para serem completadas que apenas a célula selecionada. Nós começamos Nós fazemos o mesmo que a outra, dividindo o ID da célula. Com isso, nós começamos a solucionar as células da palavra, usando um laço de repetição para cada lado que a célula pode ir.

Primeiramente, nós printamos o valor da célula para cima, usando um for. Nós damos o valor inicial da linha para a variável j, falamos que j é maior ou igual a 0, para chegar até o final da palavra cruzada, e j--, pois, já que vamos subir de linha a cada repetição, o id da linha diminui.

Com isso, nós verificamos se a célula tem um valor diferente de 0, pois, se tiver, significa que faz parte da palavra que estamos solucionando, e nós procuramos na nossa matriz o valor que corresponde a célula, e damos o valor para essa célula. Caso o valor da célula for igual a 0, significa que não faz parte da palavra cruzada, e que chegou ao final, então, aplicamos um break para parar o nosso laço de repetição, e mover para o próximo.

Após isso, nós fazemos o mesmo procedimento para todos os lados, direita, embaixo, e para a esquerda, usando o mesmo processo, indo até o final, e vendo se existe um valor 0, e, caso exista, ir para o próximo laço.

Depois dessa função, nós temos a função que dá um highlight para as palavras e suas posições no grid, que funciona de uma maneira simples. Você clica na descrição das palavras, dando um highlight azul para a palavra destacada, e deixando branco o resto da palavra cruzada. Isso funciona, primeiramente, conosco chamando a função limpar, que dá uma varredura em todas as células que fazem parte da palavra cruzada, e, caso seu valor for diferente de 0, damos um fundo branco para essa célula, e, depois, nós damos um fundo azul claro para todas as células que pertencem a palavra selecionada.

E isso é tudo.