DataTables: Executar uma função depois que a tabela for carregada

Algumas vezes você pode precisar executar uma função depois que a tabela for carregada. Se você tentar fazer uma modificação na tabela fora do escopo do DataTables, via jQuery por exemplo, isso não irá funcionar. Assim, para executar funções depois que a tabela for carregada, precisamos usar a propriedade `drawCallback`.

Por exemplo, digamos que desejemos alterar o campo "buscar". Observe como podemos fazer isso no DataTables com a propriedade drawCallback:

```
// tabela interativa
let tabela  = $('#minha-tabela').DataTable({
  // configurações do datatables *****

  "drawCallback":()=>{ // depois que a tabela for carregada
    $('#licitantes_filter').html('<label><small>Buscar por CNPJ ou razão social:</small><input type="search" class="" placeholder="" aria-controls="licitantes"></label>');
  }


}) //$('#licitantes').DataTable({
```

Isso fará com que o campo busca apareça assim:

IMPORTANTE: após testes, percebi que alterar o campo de busca causa erros ao usar essa funcionalidade. Vou deixar esse código aqui apenas como ilustração de como usar o drawCallback, mas não faça isso em um exemplo real.

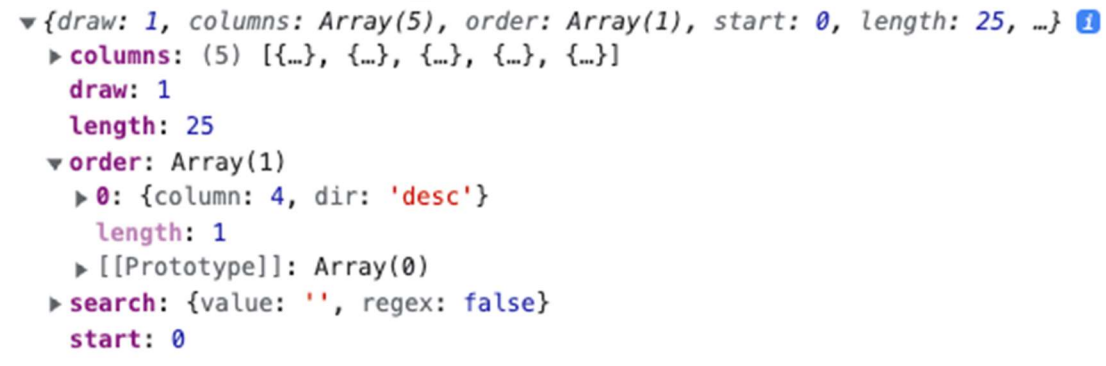
DataTables: Processamento via server-side com Ajax

Por padrão, DataTables aplica paginação via front-end. Entretanto, algumas vezes podemos ter a necessidade de exibir tabelas gigantescas (requisições maiores do que 10 MB já são consideradas grandes). Muitas vezes é inviável esperar o carregamento desses dados antes de exibi-los, podemos implementar uma paginação via back-end. Nesse caso, o processo de paginação é mais complexo, mas você ganha bastante em performance.

Em Datatables, podemos fazer isso usando a propriedade `ajax` combinada com as propriedades `serverSide` e `processing`. Note que `ajax` recebe como entrada uma função com dois parâmetros. O primeiro deles armazena os valores atuais: e o segundo é uma função de *callback* que deverá ser chamada para atualizar a tabela.

```
// requisições assíncronas via ajax + fetch API
ajax: function(d, cb){
  // d => recebe os dados atuais do datatables (págin
  // cb => função de callback (deve ser chamada pra a
```

Observe o que tem nesse objeto `d`:



Caso deseje alterar os valores da tabela, chame a função cb passando como argumento essas propriedades acima alteradas. Por exemplo, para mudar para a página 2, mande como valor da propriedade d.*start* = 26 (note que length está valendo 25, logo são 25 itens por página, então a página 2 deve começar em 26). Você verá um exemplo prático a seguir.
No exemplo a seguir faremos a implementação de uma tabela interativa que pega dados do back-end usando fetch API. Há muitas formas de fazer isso, como por exemplo, usando PHP. Entretanto, faremos usando apenas JavaScript por uma escolha pessoal.

Note que para funcionar, você precisará da aplicação back-end funcionando e pelo menos dois endpoints: (1) retorna os dados requisitados e (2) retorna a quantidade de itens.

Neste exemplo, os endpoints realizam requisições via GET e podem receber parâmetros como nome, ordem, busca, cnpj (se trata da uma busca por empresas). Nossa API está armazenada na variável `backend` (não vamos exibi-la aqui, mas imagine que é algo como http://localhost:8000 ou https://site-exemplo.com). Há ainda algumas funções e classes presentes que não tem utilidade para compreender o funcionamento do código, por isso, detalhes foram ocultados (como formata_cnpj).
Observe como podemos implementar paginação via back-end:

```
// cabeçalho ------------------------------------------------------------
-------------
async function total_itens(busca=''){
  // retorna a quantidade de itens (importante para o datatables)
  let url = backend + '/api/quantidade/listar?itens=true'

  if((busca != '')&&(busca.length >= 2)){
    // não ativo se campo busca rápida estiver com menos do que 2
caracteres
    if(eh_numero(busca.slice(0,1))){
      url += '&cnpj='+limpa_cnpj(busca);
    }
    else{
      url += '&nome='+busca.toUpperCase();
    }
  }
```

```
  return await fetch(url).then(response => response.json())
}
// FIM cabeçalho ------------------------------------------------------------
--------------

async function lista_itens(itens=25, inicio=0,
ordenar_por='nome',ordem='desc',busca=''){
  // função que recebe os dados da API

  const pagina = parseInt(inicio / itens)
  let url = backend + '/api/listar?'+
    'pagina='+pagina+
    '&itens='+itens+
    '&ordenar_por='+ordenar_por+
    '&ordem='+ordem;

  if((busca != '')&&(busca.length >= 2)){
    // não ativo se campo busca rápida estiver com menos do que 2
caracteres
    if(eh_numero(busca.slice(0,1))){
      url += '&cnpj='+limpa_cnpj(busca);
    }
    else{
      url += '&nome='+busca.toUpperCase();
    }
  }
  // console.log(url)
  return await fetch(url)
    .then(response => response.json())
}

// tabela interativa
let tabela  = $('#itens').DataTable({
  // configurações do datatables *****
  serverSide: true,
  processing: true,
  pageLength: 25,
  "lengthMenu": [[10, 25, 50, 100, -1], [10, 25, 50, 100, "Tudo"]],
  language: { url: frontend + "/data/datatables_br.json" },
  columnDefs: [
    { targets: [2],    className: 'text-center' },
    { targets: [3, 4], className: 'dt-body-right' },
    { targets: [0],    className: 'w130px' }
  ],

  order: [4, 'desc'], // ordena por alarme => desc

  "drawCallback":()=>{ // depois que a tabela for carregada
    $('#itens_filter').html('<label><small>Buscar por CNPJ ou razão
social:</small><input type="search" class="" placeholder="" aria-
controls="licitantes"></label>');
  },

  // requisições assíncronas via ajax + fetch API
  ajax: function(d, cb){
    // d => recebe os dados atuais do datatables (página atual, ordem,
etc.)
```

```
    // cb => função de callback (deve ser chamada pra atualizar a página)
    let inicio = d.start;
    let itens = d.length;
    if(itens == -1){ itens = 10 }

    let ordem = d.order[0].dir;
    let ordenar_por = d.order[0].column;

    ordenar_por = ordenar_por == 0 ? 'cnpj' :
                  ordenar_por == 1 ? 'nome_empresa' : 'nome_empresa' //
default

    let busca = d.search.value;
    // console.log(d)
    total_itens().then(total => {

      lista_itens(itens, inicio, ordenar_por, ordem, busca)
        .then(dados => {
          return cb({
            recordsTotal:total,
            recordsFiltered: total,//(busca != '') ? total : itens,
            start: inicio,
            length: itens,

            data:dados.map(
              i=>{
                return [
                  // colunas são exibidas aqui
                  i.id, // id
                  i.cnpj, //  cnpj
                  i.nome_empresa // nome da empresa
                ]
              }
            )
          })
        });

    });

  }

}) //$('#itens').DataTable({
```