Neocase Framework JS

The new JavaScript framework now shows objects and is divided under namespaces corresponding to Neocase entities.

Previous methods will still be displayed for compatibility reasons with former clients.

Entities

Neocase entities are displayed under the three following **namespaces**
Everything that is displayed in the namespace `neocase` is **commun** to the Back Office and Front Office (Portail).

Form

- **neocase.form**
    - `neocase.form.section`
    - `neocase.form.field`
    - `neocase.form.iframe`
    - `neocase.form.attachment`
    - `neocase.form.event`

Request

- **neocase.request**
    - `neocase.request.document`

Session (BO only)

- **neocase.session**
    - `neocase.session.agent.id`

Form

`neocase.form` is the form of the user interface. It contains objects and methods allowing to manage the interface.

The form object is defined as follows:

```
var neocase.form = {
    section: section(name), // return section object
    field: field(name), // return field object
    iframe: iframe(name), // return iframe object
    attachment: {},
    event: {},
    getMandatoryFields: getMandatoryFields()
}
```

Summary

Section

`neocase.form.section(sectionName).hide()` allows to hide a section and delete the mandatory charater(s) present within the fields.

```
neocase.form.section('section100').hide();
// => {section object}
```

`neocase.form.section(sectionName).show()` allows to display a section and put back the mandatory charater(s) present within the fields.

```
neocase.form.section('section100').show();
// => {section object}
```

Field

The field object is defined as follows:

```
var field = {
    name: string,
    elementHTML: object,

    label: label(),
    hide: hide(),
    show: show(),
    getValue: getValue(),
    setValue: setValue(fieldValue),
    getText: getText(),
    getNumber: getNumber(),
    getDate: getDate(),
    setDate: setDate(date),
    copyValueTo: copyValueTo(targetFieldName),
    mandatory: mandatory(message),
    getFieldType: getFieldType(),
    hasOptions: hasOptions(),
    isVisible: isVisible(),
    isMandatory: isMandatory(),
    isInitiallyMandatory: isInitiallyMandatory(),
    isManuallyChanged: isManuallyChanged(),
    isReadOnly: isReadOnly(),
    isDisabled: isDisabled()
}
```

Get the field object or HTML Element

`neocase.form.field(fieldName)` allows to get the object corresponding to the field of the form which name is passed as a parameter.

```
var field = neocase.form.field('INTERVENTIONS_EN_COURS$VALEUR1');
// => {field object}
```

`neocase.form.field(fieldName).elementHTML` allows to get the HTML element.

```
var element = neocase.form.field('INTERVENTIONS_EN_COURS$VALEUR1').elementHTML;
// => {HTMLElement object}
```

Get the label text

`neocase.form.field(fieldName).label()` allows to get the label text or `null` if no label was found.

```
neocase.form.field('INTERVENTIONS_EN_COURS$VALEUR1').label();
// => "Label" or null
```

Hide/Show a field

`neocase.form.field(fieldName).hide()` allows to hide a field and his label.

Works:

- in the Front Office
- in the Back Office
- for all field types
- in update or read only

```
neocase.form.field('INTERVENTIONS_EN_COURS$VALEUR1').hide();
// => {field object}
```

`neocase.form.field(fieldName).show()` allows to display a field and his label.

Works:

- in the Front Office

- in the Back Office
- for all field types
- in update or read only

```
neocase.form.field('INTERVENTIONS_EN_COURS$VALEUR1').show();
// => {field object}
```

Get the field value

`neocase.form.field(fieldName).getValue()` allows to get the field value.

- text : text
- list : text
- checkbox : check/uncheck
- file : file name
- readonly : text

```
var fieldValue = neocase.form.field('INTERVENTIONS_EN_COURS$VALEUR1').getValue();
```

Update the field value

`neocase.form.field(fieldName).setValue(fieldValue)` Update the field value.

```
// Set fieldName value to 'test'
neocase.form.field('INTERVENTIONS_EN_COURS$VALEUR1').setValue('test');
// Empty fieldName value
neocase.form.field('INTERVENTIONS_EN_COURS$VALEUR1').setValue(null);
```

Get the field text

`neocase.form.field(fieldName).getText()` allows to get the text of a list type field.

```
var text = neocase.form.field('INTERVENTIONS_EN_COURS$VALEUR1').getText();
// => "example"
```

Get the field number

`neocase.form.field(fieldName).getNumber()` allows to get the field value as a floating point number.
If the value cannot be converted to a number, `NaN` is returned. Leading and trailing spaces are ignored.

```
neocase.form.field('INTERVENTIONS_EN_COURS$VALEUR1').getNumber();
// '123'    => 123
// '45.678' => 45.678
// '45,678' => 45
// '45.6Mo' => 45.6
// ' 45.6 ' => 45.6
// 'Ab45.6' => NaN
// 'test'   => NaN
// true     => NaN
```

Get the date of a date type field

`neocase.form.field(fieldName).getDate()` allows to get the date of a date type field.

```
var date = neocase.form.field('INTERVENTIONS_EN_COURS$VALEUR1').getDate();
```

Update the date of a date type field

`neocase.form.field(fieldName).setDate(date)` allows to update the date of a date type field.

```
neocase.form.field('INTERVENTIONS_EN_COURS$VALEUR1').setDate('01/12/2017');
```

Copy/Paste a value from a field to another field

`neocase.form.field(fieldName).copyValueTo(targetFieldName)` allows to copy a value from a field to another field.

```
neocase.form.field('INTERVENTIONS_EN_COURS$VALEUR1').copyValueTo('INTERVENTIONS_EN_COURS$VALEUR2');
// => {field object}
```

Enable the mandatory property of a field

`neocase.form.field(fieldName).mandatory(message)` allows to make a field mandatory, with a alert message.

```
// Set the fieldName 'INTERVENTIONS_EN_COURS$VALEUR1' to be required
neocase.form.field('INTERVENTIONS_EN_COURS$VALEUR1').mandatory("Please enter a value");
```

Get the type of the field

`neocase.form.field(fieldName).getFieldType()` allows to get the type of the field.

```
var fieldType = neocase.form.field('INTERVENTIONS_EN_COURS$VALEUR1').getFieldType();
// => 'readonly' if the field is read only
// => 'open list' if the HTML class contains 'combobox'
// => 'list' if the field is a SELECT element
// => 'file' if the field ID ends with '_file'
// => 'text area' if the field is a TEXTAREA element
// => 'decimal' if datatype attribute is 'double'
// => else HTML datatype attribute or HTML type attribute
```

Check if a field has options (in the case of a SELECT element)

`neocase.form.field(fieldName).hasOptions()` returns 'true' if a field is a SELECT element and has at least one option else returns 'false'.

```
neocase.form.field('INTERVENTIONS_EN_COURS$VALEUR1').hasOptions();
// => true or false
```

Check if a field is visible

`neocase.form.field(fieldName).isVisible()` returns 'true' if a field is visible and 'false' if the field is not visible.

```
neocase.form.field('INTERVENTIONS_EN_COURS$VALEUR1').isVisible();
// => true or false
```

Check if a field is mandatory

`neocase.form.field(fieldName).isMandatory()` returns 'true' if a field is mandatory and 'false' if the field is not mandatory.

```
neocase.form.field('INTERVENTIONS_EN_COURS$VALEUR1').isMandatory();
// => true or false
```

Check if a field was initially mandatory

`neocase.form.field(fieldName).isInitiallyMandatory()` returns 'true' if a field is initially mandatory and 'false' if the field is initially not mandatory.

```
neocase.form.field('INTERVENTIONS_EN_COURS$VALEUR1').isInitiallyMandatory();
// => true or false
```

Detect a manual change for a field

`neocase.form.field(fieldName).isManuallyChanged()` returns 'true' if the field was manually updated and 'false' if the field was not manually updated.

```
neocase.form.field('INTERVENTIONS_EN_COURS$VALEUR1').isManuallyChanged();
// => true or false
```

Check if a field is read only

`neocase.form.field(fieldName).isReadOnly()` returns 'true' if a field is read only and 'false' if the field is not read only.

```
neocase.form.field('INTERVENTIONS_EN_COURS$VALEUR1').isReadOnly();
// => true or false
```

Check if a field is disabled

`neocase.form.field(fieldName).isDisabled()` returns 'true' if a field is disabled and 'false' if the field is not disabled.

```
neocase.form.field('INTERVENTIONS_EN_COURS$VALEUR1').isDisabled();
// => true or false
```

Iframe

```
var neocase.form.iframe = {
    hide: hide(),
    show: show(),
    navigate: navigate(url)
}
```

`neocase.form.iframe.hide()` allows to hide a iframe.

`neocase.form.iframe.show()` allows to show a iframe.

`neocase.form.iframe.navigate()` allows to change the `location.href` of the iframe.

Attachment

```
var neocase.form.attachment = {
    count: count(),
    isAttached: isAttached(regfilename)
}
```

`neocase.form.attachment.count()` allows to get the number of files attached to the form.

`neocase.form.attachment.isAttached(expressionReguliere)` allows to know if a file is attached to the form. Takes the regular search expression parameter.

```
if(neocase.form.attachment.count() > 0){
    if(!neocase.form.attachment.isAttached("^.*\.(jpg|gif|doc|pdf)$"))
        alert('Accepted file type: jpg, gif, doc, pdf');
}
```

Event

```
var neocase.form.event = {
    bind: bind(event, func, group),
    unbind: unbind(event, group),
    trigger: trigger(event)
}
```

`neocase.form.event.bind(event, func, group)` allows to execute a function during the event trigger specified in the parameter. The `group` argument is optional.

Events examples:

- `neocase.form.event.bind('init', function)`
- `neocase.form.event.bind('loadcomplete', function)`
- `neocase.form.event.bind('submit', function)`

```
// Exemple with anonymous function
neocase.form.event.bind('loadcomplete', function(){ ... });
// Exemple with existing custom function
neocase.form.event.bind('loadcomplete', functionCustom);
```

`neocase.form.event.unbind(event, group)` allows to remove all functions related to the specified event and group (optional).

```
neocase.form.event.unbind('loadcomplete');
```

`neocase.form.event.trigger(event)` allows to manually run all the functions related to the specified event.

If one of the function returns `false`, all the following functions are ignored.

```
neocase.form.event.trigger('event');
// => true
// => false if one of the function returned false
```

Get list of mandatory fields

`neocase.form.getMandatoryFields()` returns an mandatory 'field' objects array.

```
var fields = neocase.form.getMandatoryFields();
// => [{field object}, {field object}, {field object}]
```

Request

`neocase.request` contains the properties and methods that allows to get information about the request or to do actions on the request.

The request object is defined as follows:

```
var neocase.request = {
    executeRules: executeRules(arguments),
    save: save(),
    saveAndClose: saveAndClose(),
    close: close(),
    cancel: cancel(),
    transfer: transfer(),
    transferBySkill: transferBySkill(),
    transferByLocalisation: transferByLocalisation(),
    subcontract: subcontract(),
    delegateToContact: delegateToContact(),
    resolve: resolve(),

    // Background mode
    backgroundInit: backgroundInit(),
    backgroundExecute: backgroundExecute(funcToExecute),
    backgroundEnd: backgroundEnd(callback),

    getCreationDate: getCreationDate(),
    getDeadline: getDeadline(),
    setDeadline: setDeadline(date),
    calculateDeadline: calculateDeadline(startDate, period, periodType, codeGrille),

    document: document(name) // return neocase.document object
}
```

Execute rules

You can add multiple rules.
Be careful, if you run too many rules, this can affect performance.

```
neocase.request.executeRules('REGLE ENVOI MAIL CLOTURE', 'ruleName1', 'ruleName2');
```

Execute background actions

Allows to execute functions without reloading the form.
Initially planned to use the save function, we can also call the custom function.

`neocase.request.backgroundInit();` Initialize the background execution mode.

`neocase.request.backgroundExecute(funcToExecute);` Execute an action in background mode, `funcToExecute` is the function that is running in the background.

`neocase.request.backgroundEnd(callback)` Stop the background mode and call the callback function passed in argument.

```
var funcToExecute = new Function();
var callback = new Function();

neocase.request.backgroundInit();
neocase.request.backgroundExecute(funcToExecute);
neocase.request.backgroundExecute(enregistreronly);
neocase.request.backgroundEnd(callback);
```

Deadline

`neocase.request.getDeadline()` allows to get the request's due date.

```
//Get deadline local date - return a date object
var deadline = neocase.request.getDeadline();
```

`neocase.request.setDeadline(date)` allows to update the request's due date.

```
//Update the request deadline based on the client timezone
var date = new Date();
var date = new Date("December 17, 1995 03:24:00");
var date = new Date("1995-12-17T03:24:00");
var date = new Date(1995, 11, 17);
var date = new Date(1995, 11, 17, 3, 24, 0);

neocase.request.setDeadline(date);
```

`neocase.request.calculateDeadline(startDate, period, periodType, codeGrille)` makes it possible to calculate the due date of the request according to the delay and the identifier of team schedule ID (0 by default).

- `startDate`: The start date.
- `period`: The value of the period.
- `periodType`: The type of value for the period ('j' for day and 's' for second).
- `codeGrille`: Team schedule ID (0 by default).

Document

`neocase.request.document.create(templateName)` allows to create a document from a template then you can attach this document to the request.

```
neocase.request.document.create('FICHE REMUNERATION - EMBAUCHE');
neocase.request.document.create('FICHE REMUNERATION - EMBAUCHE').attachToCase();
```

Session

`neocase.session` is the session opened by the agent from the Back Office.
It contains the properties and methods that allows to get the session's informations.

The session object is defined as follows:

```
var neocase.session = {
    agent: {
        id: SessionCodeIntervenant
    }
}
```

Migration

In case of migration of an former client, here is the list of former methods which have an equivalent.

It is still advised to **rethink** the different algorithm than to try to mechanically apply new functions in place of the formers.

Request (ThisCase => neocase.request)

- `ThisCase.IsClosed` => `neocase.request.isClosed`
- `ThisCase.IsNewCase` => `neocase.request.isNew`
- `ThisCase.IsSubContracted` => `neocase.request.isSubContracted`
- `ThisCase.IsVisibleByContact` => `neocase.request.isVisibleByContact`
- `ThisCase.ExecuteRules` => `neocase.request.executeRules`
- `ThisCase.BackgroundMode.Begin` => `neocase.request.backgroundInit`
- `ThisCase.BackgroundMode.Execute` => `neocase.request.backgroundExecute`
- `ThisCase.BackgroundMode.End` and `ThisCase.BackgroundMode.Stop` => `neocase.request.backgroundEnd`
- `ThisCase.GetDeadline` => `neocase.request.getDeadline`
- `ThisCase.SetDeadline` => `neocase.request.setDeadline`
- `ThisCase.GetNewDeadline` => `neocase.request.calculateDeadline`
- `ThisCase.Save` => `neocase.request.save`
- `ThisCase.SaveAndClose` => `neocase.request.saveAndClose`
- `ThisCase.Transfer` => `neocase.request.transfer`
- `ThisCase.TransferBySkill` => `neocase.request.transferBySkill`
- `ThisCase.TransferByLocalisation` => `neocase.request.transferByLocalisation`
- `ThisCase.SubContract` => `neocase.request.subcontract`
- `ThisCase.DelegateToContact` => `neocase.request.delegateToContact`
- `ThisCase.Resolve` => `neocase.request.resolve`
- `ThisCase.Close` => `neocase.request.close`
- `ThisCase.Cancel` => `neocase.request.cancel`
- `ThisCase.Attachments.Count` => `neocase.form.attachment.count`

- ThisCase.Attachments.Exists => neocase.form.attachment.isAttached

Form (ThisForm => neocase.form)

- ThisForm.GetElement => neocase.form.field(fieldName).elementHTML
- ThisForm.GetProperty => neocase.form.field(fieldName).getValue();
- ThisForm.SetProperty => neocase.form.field(fieldName).setValue(value);
- ThisForm.HideSection => neocase.form.section(sectionName).hide
- ThisForm.ShowSection => neocase.form.section(sectionName).show
- ThisForm.Attachments
    - Count => neocase.form.attachment.count
    - Exists => neocase.form.attachment.isAttached
- ThisForm.Documents
    - Count => neocase.form.document.count
    - Exists => neocase.form.document.isAttached
    - Open => neocase.form.document.open
- ThisForm.Bind => neocase.form.event.bind
- ThisForm.Unbind => neocase.form.event.unbind
- ThisForm.Trigger => neocase.form.event.trigger

neocase.form.field

- [+] neocase.form.field(fieldName).getFieldType
- [+] neocase.form.field(fieldName).label
- [+] neocase.form.field(fieldName).hide
- [+] neocase.form.field(fieldName).show
- [+] neocase.form.field(fieldName).getText
- [+] neocase.form.field(fieldName).getDate
- [+] neocase.form.field(fieldName).setDate
- [+] neocase.form.field(fieldName).getNumber
- [+] neocase.form.field(fieldName).isInitiallyMandatory
- [+] neocase.form.field(fieldName).isMandatory
- [+] neocase.form.field(fieldName).mandatory
- [+] neocase.form.field(fieldName).copyValueTo
- [+] neocase.form.field(fieldName).hasOptions
- [+] neocase.form.field(fieldName).isVisible
- [+] neocase.form.field(fieldName).isDisabled
- [+] neocase.form.field(fieldName).isManuallyChanged

Session (ThisSession => neocase.session)

- ThisSession.Agent.Id => neocase.session.agent.id