Intrexx Professional
Intrexx Compact

RELEASE 5

Dynamische Filter und Abhängigkeiten

Inhaltsverzeichnis

Copyright

Schreibkonventionen

In diesem Handbuch werden Textstellen *kursiv* dargestellt, wenn sie sich auf Einstellungen in den abgebildeten Dialogen beziehen. Menüpunkte, die in Kontextmenüs erreichbar sind, sind immer auch über das Hauptmenü erreichbar. Hauptmenüpunkte werden nicht beschrieben, es sei denn, sie sind nicht über das Kontextmenü erreichbar. Eine Beschreibung der allgemeinen Hauptmenüpunkte finden Sie im Handbuch *Portale*. Programmiercode im Text wird in der Schriftart `Courier` dargestellt. Kontextmenüs können mit einem Klick mit der rechten Maustaste auf das beschriebene Element geöffnet werden.

<*intrexx*> bezeichnet im Folgenden Ihren Intrexx Installationspfad, unter Windows z.B. *c:\intrexx*, unter Linux z.B. */opt/intrexx/*. Folgende Symbole werden für die Kennzeichnung von speziellen Informationen verwendet:

- Informationen
- Verweise auf ein Intrexx Handbuch
- Verzeichnisse
- URLs
- Klick auf Schaltflächen

Vorkenntnisse

Für das Verständnis dieser Dokumentation sind Vorkenntnisse beim Erstellen von Applikationen und in der Programmierung mit JavaScript erforderlich.

1. Einleitung

Mit Intrexx 5.0 wurde eine Vielzahl an Anpassungen und Verbesserungen bei dynamischen Filtergruppen umgesetzt.
Dieser Umbau umfasst folgende Punkte:

- Vereinheitlichung des Filtervorgangs bei Tabellen und Auswahllisten
- Abbilden von komplexen Filterausdrücken
  (Verknüpfung mehrerer Filterelemente mit *und*/*oder*)
- Einbinden von „normalen" Eingabeelementen zum Filtern
- Mehr Transparenz, wann welches Element einen Filtermechanismus auslöst
- Freies Platzieren von Filterelementen (innerhalb einer Seite)

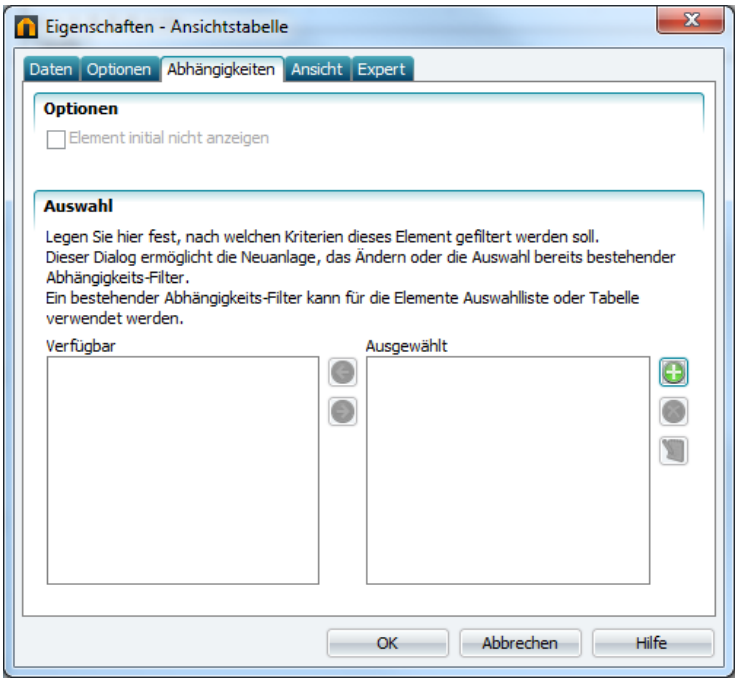
Im Folgenden soll ein Überblick darüber gegeben werden, welche Mechanismen sich verändert haben und was für neue Möglichkeiten zur Verfügung stehen. Weiter unterstützen wir Sie bei etwaigen notwendigen Anpassungen, die innerhalb von benutzerdefinierten JavaScript-Methoden notwendig sind, da solche Skripte nicht während eines Updatevorgangs verarbeitet werden können.

2. Aufbau

Die Filterkontrollen und der Filtermechanismus setzen sich aus mehreren Elementen zusammen.

2.1. Abhängigkeit (Dependency)

Eine Abhängigkeit lässt sich im Eigenschaftendialog eines zu filternden Elements, also z.B. einer Ansichtstabelle, definieren. Dies bedeutet, dass an dieser Stelle angegeben wird, nach welchen Kriterien und von welchem Element die angezeigten Daten dieser Kontrolle gefiltert werden soll.



Weitere Informationen zur Definition von Abhängigkeiten finden Sie im Intrexx Handbuch *Applikationen* im Kapitel *Abhängigkeiten*.

2.1.1. Auslösendes Ereignis (Trigger)

Bei der Definition einer Abhängigkeit muss im ersten Schritt definiert werden, durch welche Kontrolle ein Filter ausgelöst werden soll.
Hierbei kann jede Kontrolle auf einer Seite angegeben werden, die JavaScript-Events unterstützt sowie die Seite selber. Wurde eine Kontrolle ausgewählt, wird automatisch eine Auswahl der bei dieser Kontrolle zur Verfügung stehenden JavaScript-Events bereitgestellt (z.B. Events wie *onload()* oder *onchange()*).

2.1.2. Filter

Nach der Definition der auslösenden Kontrolle wird der eigentliche Filter definiert. Dies bedeutet, dass ein Datenfeld der zu filternden Datengruppe mit einem Filterelement verknüpft wird. Jedes Filterelement, als auch jede normale Eingabekontrolle, kann mit einem Datenfeld gleichen Typs verglichen werden.

Im Gegensatz zum *Dynamischen Filter* lassen sich Filter aus Eingabekontrollen in beliebiger Komplexität erstellen und schachteln.

2.1.3. Events

Optional können in Verbindung mit einem Filtervorgang benutzerdefinierte Skripte aufgerufen und ausgeführt werden, um beispielsweise vor oder nach dem Filtern weitere Bedingungen abzuprüfen.
Bei der Definition einer Abhängigkeit können auf dem Reiter *Skript* das Event und dazugehörige Skript definiert werden. Es stehen hierbei die beiden folgenden Events zur Verfügung.

Event	Beschreibung
ontrigger	Ein hinterlegtes Skript wird vor dem eigentlichen Filtern ausgeführt. Somit können z.B. für das Filtern notwendige Voraussetzungen geprüft werden.
onmatch	Ein hinterlegtes Skript wird nach dem Filtern und dem Neuladen der Elemente ausgelöst. Nutzen Sie diese Variante, um die gefilterte Datenmenge zu verarbeiten.

Für beide Events lasse sich beliebig viele Listener, also aufzurufende JavaScript-Funktionen, hinterlegen. Eine definierte Ereigniskette kann mit *return false;* abgebrochen werden.

2.2. Zielkontrolle (Dependency Target)

Bei der Zielkontrolle handelt es sich um das Element, das die zu filternden Daten enthält. Bei diesen Elementen kann es sich im Tabellen, Listenfelder und Auswahllisten handeln.

Eine definierte Abhängigkeit kann beliebig vielen Zielkontrollen zugeordnet sein. Dabei kann es sich bei den Zielkontrollen sowohl um Kontrollen des gleichen Typs (z.B. mehrere Ansichtstabellen) handeln als auch im einen Mix aus unterschiedlichen Kontrollen.

2.3. Ablauf eines Filterereignisses

Ein Filterereignis läuft in folgenden Schritten ab:

- Die Triggerkontrolle, d.h. die Kontrolle in Verbindung mit dem definierten Triggerevent, löst die Abhängigkeit aus
- Das *ontrigger*-Event wird ausgelöst und eventuell bei diesem Event hinterlegte Skripte werden ausgeführt
- Der definierte Filter wird ausgewertet und die damit verbundenen Filter-XMLs werden erstellt
- Die Zielkontrollen, die mit der ausgelösten Abhängigkeit verknüpft sind, werden abgearbeitet, d.h. der Filter wird angewandt und die Zielkontrollen werden neu geladen
- Das *onmatch*-Event wird ausgelöst und eventuell bei diesem Event hinterlegte Skripte werden ausgeführt

Sind vom Benutzer keine *onmatch*-Events definiert, so erfolgt das Neuladen der Zielkontrollen asynchron, um einen Benutzer nicht unnötig in seiner Arbeit zu behindern.

3. Dynamische Filter

Dynamische Filter sind nun eigenständige Ansichtselemente, die frei und unabhängig voneinander auf einer Seite platziert werden können.

3.1. Modul Applikationen: Expertreiter

Über den Reiter *Expert* im Eigenschaftendialog einiger dynamischer Filterelemente können Defaultwerte gesetzt werden, die beim Laden der Seite und des Elements ausgewählt werden.

Alphaindex, Auswahlliste, Operator (Typ Textfeld)
Attribut:
customdefault

Beispiel :
Setzt den Buchstaben *M* im Element *Alphaindex* als Defaultwert

customdefault	M

Setzt den Eintrag *Mustermann* als Defaultwert im Element *Auswahlliste*

customdefault	Mustermann

Auswahlliste (Mehrfachauswahl)
Attribute:
customdefault
customdefaulttype

Das Attribut customdefaulttype legt fest, ob es sich bei dem in customdefault definierten Wert um die Record-ID eines Datensatzes handelt oder ob ein tatsächlicher Wert übergeben wird.

Beispiel :
Setzt die Werte *1* und *2* im Element *Mehrfachauswahl* als Defaultwert

customdefault	1||2
customdefaulttype	value

Setzt die Werte mit der Record-Id 1 und 2 als Defaultwert im Element *Mehrfachauswahl*

customdefault	1||2
customdefaulttype	key

Wertebereich, Kalender
Attribute
customdefaultfrom
customdefaultto

Beispiel :
Grenzt den *Wertebereich* auf Werte zwischen 1 und 10 ein

customdefaultfrom	1
customdefaultto	10

Grenzt den *Wertebereich* vom Typ Datum/Datum-Uhrzeit auf den Tag *01.02.2010* ein

Ist der Wertebereichsfilter vom Typ Datum/Datum-Uhrzeit, so ist der Defaultwert im Format *<Jahr>,<Monat>,<Tag>,<Stunde>,<Minute>,<Sekunde>* anzugeben. Wird z.B. wie im unteren Beispiel das Jahr nicht angegeben, wird der zum Zeitpunkt des Aufrufs aktuelle Wert verwendet.

customdefaultfrom	,02,01,0,0,0
customdefaultto	,02,01,23,59,59

3.2. Zugriff per JavaScript

Dynamische Filter und deren Werte können mit JavaScript referenziert, ausgelesen und gesetzt werden.

3.2.1. UP-Objekt

```
var oHtml = getElement(<GUID_FILTERELEMENT>).oUp;
```

3.2.2. Auslesen von Filterelementen

```
var oHtml = getElement(<GUID_FILTERELEMENT>);
Browser.getValue(oHtml);
```

3.2.3. Setzen von Filterelementwerten

```
var oHtml = getElement(<GUID_FILTERELEMENT>);
Browser.setValue(oHtml, value);
```

3.2.4. Setzen von Defaultwerten

Auswahlliste

```
Browser.setValue(getElement(GUID), “1“, “key“);
```

selektiert den Eintrag mit der Record-Id 1 in der referenzierten Auswahlliste (wenn diese mit einem Wert aus einer Datengruppe oder einer Referenz gefüllt ist)

```
Browser.setValue(getElement(GUID), “Anton“);
```
selektiert den Eintrag *Anton* in der referenzierten Auswahlliste

Auswahlliste (Mehrfachauswahl)
```
Browser.setValue(getElement(GUID), “1||2“,“key“);
```
selektiert die Einträge mit den Record-Ids 1 und 2 (wenn die Auswahlliste mit einem Wert aus einer Datengruppe oder einer Referenzen gefüllt ist)

```
Browser.setValue(getElement(GUID), “Anton||Berta“);
```
selektiert die Einträge *Anton* und *Berta* in der referenzierten Auswahlliste

Alphaindex
```
Browser.setValue(getElement(GUID), “M“);
```
selektiert entsprechende Tab des Alphaindex mit dem Wert *M*

Wertebereich, Kalender
```
Browser.setValue(getElement(GUID), “01.01.2010“, “from“);
Browser.setValue(getElement(GUID), “01.01.2010“, “to“);
```
setzt als Defaultwert den 01.01.2010 im referenzierten Kalender- oder Wertebereich-element

Operator
```
Browser.setValue(getElement(GUID), “GUID1“, “datafield“);
Browser.setValue(getElement(GUID), “GUID1||GUID2“,
“datafield“);
Browser.setValue(getElement(GUID), “=“, “operator“);
Browser.setValue(getElement(GUID), “Anton“, “search“);
```

GUID1 und *GUID2* sind in diesem Beispiel Guids von Datenfeldern, die beim Operator-Filterelement vorselektiert werden sollen. datafield ist hierbei ein Identifier.

Mit *operator* können Sie einen Operator aus der Auswahl angeben, der in der Liste Operatoren vorselektiert werden soll.

Im letzten Beispiel wird das Suchfeld mit dem Wert *Anton* vorbelegt.

RadioControlGroup
```
Browser.setValue(getElement(GUID), “Anton“)
```
wählt als Defaultwert aus einer RadioControlGroup den Radio-Button mit dem Wert *Anton* aus.

3.2.5. Unterdrücken des onchange-Events

Beim Setzen eines Filterelementwertes über die JavaScript-Funktion *Browser.setValue()* wird automatisch das onchange-Ereignis der Kontrolle ausgelöst.

Dieses automatische Auslösen beim Setzen von Filterwerten kann wie folgt unterbunden werden.

```
Browser.setValue(<Kontrolle>, <Wert>, <Zielinfo>, <Onchange >)
```

wobei:

Kontrolle: Zielkontrolle, deren Wert gesetzt werden soll
Wert: der zu setzende Wert
Zielinfo: Zusätzliche Infos über den zu setzenden Wert bzw. Zielkontrolle
Onchange: Boolscher Wert, ob das onchange Event ausgelöst werden soll oder nicht. Der Defaultwert ist hierbei true.

Beispiele:
Von-Eingabefeld eines Wertebereich-Filters befüllen

```
Browser.setValue(getElement(GUID-WERTEBEREICH), "11.11.2011",
"from", false);
```

Ein Wert in einer Auswahlliste selektieren:

```
Browser.setValue(getElement(GUID-DROPDOWN), "Option Eins",
null, false);
```

Dies könnte nützlich sein wenn über JS mehrere Filterelemente zunächst mit Werten gefüllt werden sollen und erst im Anschluss der Filtervorgang ausgelöst werden soll. (Alle Elemente sind in diesem Fall als Trigger eingetragen).

3.2.6. Weitere Funktionalitäten

Zurücksetzen (Reset) des Filterelements

```
var oHtml = getElement(<GUID_FILTERELEMENT>);
oHtml.reset()
```

Deaktivierung des Filters

```
oHtml.disable()
```

Aktivierung des Filters

```
oHtml.enable()
```

Gibt `true` zurück, wenn ein Filter aktiviert ist, sonst `false`.

```
oHtml.isEnabled()
```

4. Update auf Intrexx 5.0

Bei einem Update einer bestehenden Intrexx Installation auf Version 4.5 werden Filtergruppen von Applikationen auf die neuen Abhängigkeiten gepatcht.

Da sich das zugrundliegende JavaScript-API geändert hat, kann dies bei manuellen Erweiterungen im Bereich der dynamischen Filter zu notwendigen Anpassungen führen. Der Updatevorgang sowie der Import von Applikationen sorgen dafür, dass bereits bestehende JavaScript Programmierungen weiterhin funktionsfähig bleiben, da sie im Laufe des Updates an das neue API angepasst werden.

Angepasste Stellen im JavaScript werden von Intrexx mit einem speziellen Kommentar (*TASK: Deprecated API*) versehen. Es empfiehlt sich die markierten Stellen im Laufe der Zeit auf die neuen Zugriffsmethoden von Intrexx 5 zu überführen.
Sie finden die Hinweise bei einer im Modul Applikationen geöffneten Applikation auf dem Reiter Hinweise/Suche oder bei geöffnetem Intrexx JavaScript-Editor.

4.1. Element Filtergruppe

Das Element *Filtergruppe* ist in Intrexx 5.0 nicht mehr vorhanden. Aus diesem Grund ist auch ein Zugriff auf das UP-Objekt *upFilterGroup* nicht mehr möglich.

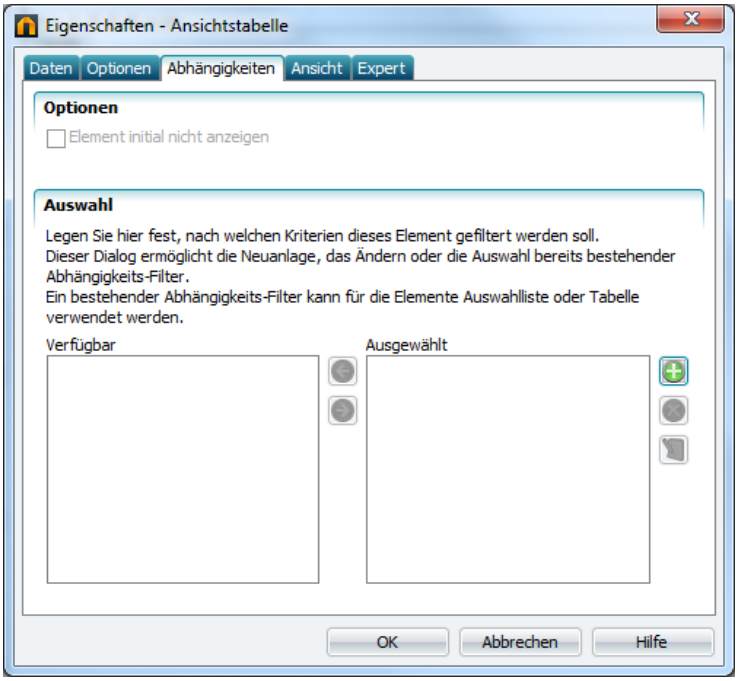
4.1.1. Auslösen des Filtervorgangs

Häufig wurde in bisherigen Versionen innerhalb von JavaScript auf die Filtergruppe zugegriffen, um den Filtervorgang per JavaScript auszulösen. Verwendet wurde dafür die Methode *oUp.submitForm()*, z.B.:

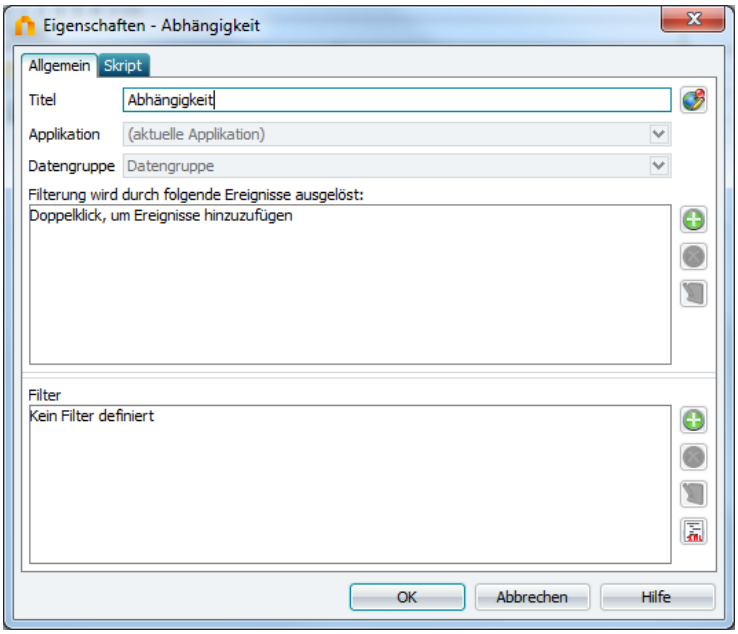
```
getElement(GUID_FILTERGRUPPE).oUp.submitForm()
```

Dieser Aufruf ist mit Intrexx 5.0 deprecated und sollte manuell angepasst werden, da der Aufruf lediglich noch in einem Kompatibilitätsmodus läuft.
Gehen Sie hierzu wie folgt vor:

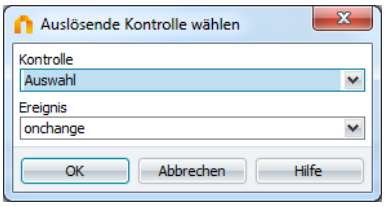
Jede Kontrolle kann nun zu einem beliebigen Zeitpunkt den Filtervorgang auslösen. Sowohl die Trigger (also die auslösende Kontrollen) als auch der Zeitpunkt (Event) lassen sich im Abhängigkeitsdialog des zu filternden Elements (z.B. Ansichtstabelle) definieren. Öffnen Sie den Eigenschaftendialog der Tabelle und wechseln Sie auf den Reiter *Abhängigkeiten*.



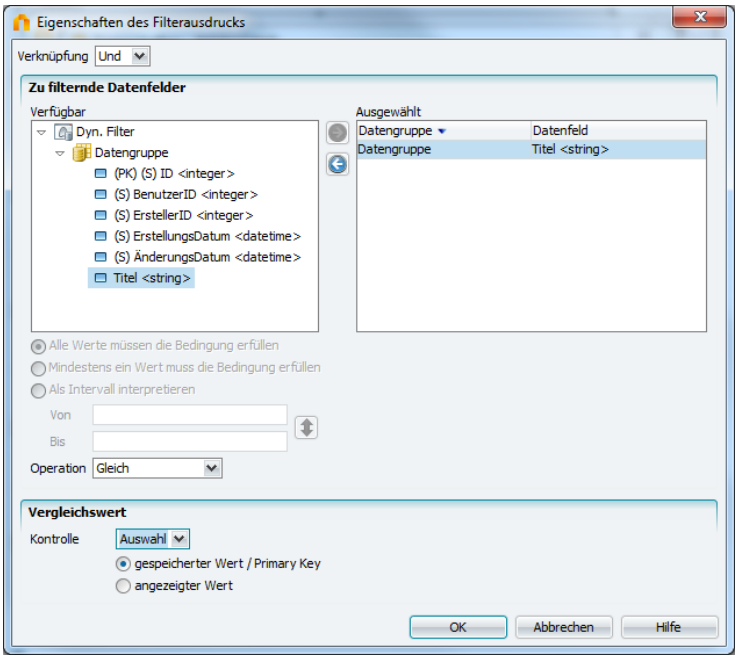
Über  legen Sie eine neue Abhängigkeit an.



Wählen Sie zunächst die auslösende Kontrolle.



Anschließend definieren Sie den Filter, der angewandt werden soll, wenn das zuvor gewählte Ereignis der auslösenden Kontrolle stattfindet.



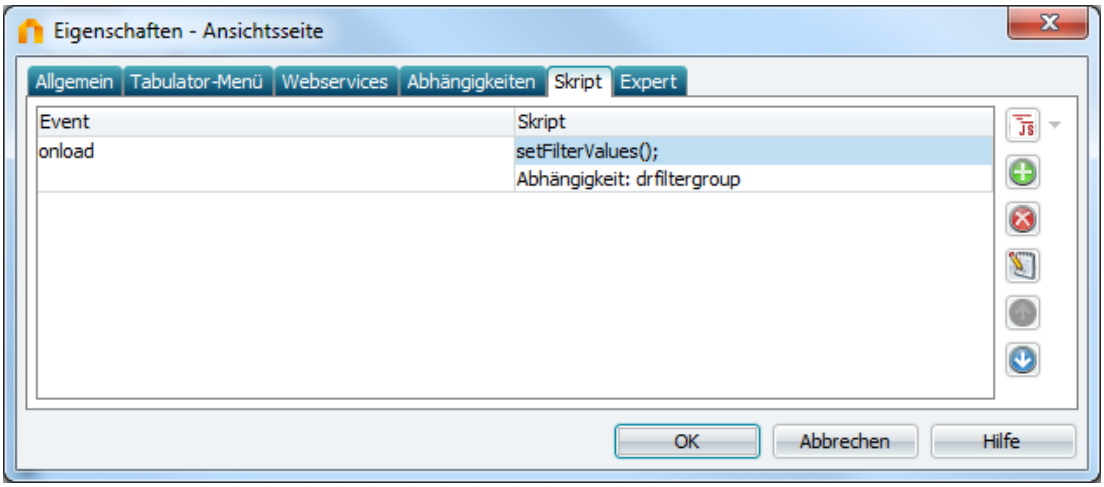
Schließen Sie die Dialoge mit *OK*.

4.1.2. Filtervorgang nach JavaScript-Aufruf ausführen

Ein weiterer Anwendungsfall besteht darin, z.B. beim Laden einer Seite vorhandene Werte als Defaultwerte eines Filters zu setzen und erst dann anschließend den Filtervorgang mit diesen Werten auszuführen.

Gehen Sie in diesem Fall wie folgt vor:

Als Trigger stellt man zunächst im Abhängigkeitsdialog des zu filternden Elements die aktuelle Seite und als Event *onload* ein. Wechseln Sie anschließend in den Eigenschaftendialog der Seite und fügen Sie dort auf dem Reiter *Skript* einen neuen Skriptaufruf für das onload-Ereignis hinzu. Mit den Pfeiltasten kann die Reihenfolge der Skripte verändert werden. Ändern Sie die Reihenfolge so ab, dass sich der JavaScript-Aufruf (hier im Beispiel *setFilterValues()*) oberhalb der bereits definierten Abhängigkeit *drfiltergroup* befindet.



Ist dies definiert wird bei dem auslösenden Event (in diesem Beispiel beim Laden der Seite) die hinterlegte JavaScript-Funktion *setFilterValues()* ausgeführt. Wird diese Funktion korrekt ausgeführt, d.h. die Filterwerte wurden korrekt gesetzt und es wurde true zurückgeliefert, wird anschließend der eigentliche Filtervorgang mit den über JavaScript gesetzten Werten ausgeführt.

4.1.3. Zugriff auf Filterelemente

Der JavaScript-Zugriff auf Filter und deren Unterelemente zum Auslesen und Setzen von Werten erfolgt nun immer über die Funktion `Browser.setValue()` bzw. `Browser.getValue()`.

Beispiele für das Auslesen und Setzen von Werten von Filterelementen finden Sie im Kapitel 3.2 *Zugriff per JavaScript*.

Ein direkter Zugriff auf Unterelemente eines Filters ist nicht mehr erforderlich und teilweise auch nicht mehr möglich.

Folgende bisher gültige Zugriffe auf Elemente oder Unterelemente in JavaScript sind mit Intrexx 5.0 nicht mehr einsetzbar und müssen angepasst werden.

Beispiel 1:

```
getElement(GUID-FILTER).oUp.fldInputfrom
```

Ersetzen Sie diesen Aufruf durch folgende Zeile:

```
Browser.getValue(getElement(GUID-FILTER), “from”)
```

Beispiel 2:

```
getElement(GUID-FILTER).oUp.fldInputto
```

Ersetzen Sie diesen Aufruf durch folgende Zeile:

```
Browser.getValue(getElement(GUID-FILTER), “to”)
```

Beispiel 3:

```
document.getElementById(“ID_from” + filtergruppe + “_” + filterelement)
```

Ersetzen Sie diesen Aufruf durch folgende Zeile:

```
Browser.getValue(getElement(GUID-FILTER), “from”)
```