Яндекс

SVARX:
фреймворк для семантической валидации форм

Макс Ширшин
Руководитель группы разработки интерфейсов
Рекламных Технологий

Я.Субботник, Киев, 28 мая 2011

О клиентской валидации


Клиентская
валидация
Серверная
валидация
Общие правила

Серверная валидация

- защита от **случайных** и **намеренных** ошибок
- **не пропустить** ошибку
- нет цели **пояснить суть** ошибки
- риск **потери данных** или **контекста**
- **плюс** перезагрузка — **минус** конверсия :-(

Клиентская валидация

- защита от **случайных** ошибок
- **не перегружать** страницу
- пояснить **суть** ошибки
- указать **способ устранения**
- повышаем **конверсию** :-)

Клиентская валидация
часто делается «на коленках»

```
<form onsubmit="if (!this.elements.email.value) return
false; if (this.elements.firstname.value == '' ||
this.elements.secondname.value == '') {alert('Укажите имя
и фамилию'); return false}">
```

```
<script>
$(function(){$('form').eq(1).submit(function() {
  var eml = $(this).find('input[name=email]').val();
  eml = $.trim(eml);
  var emlRe = /[\w\d]+[2,20]\@[\w\d]+[2,20]\.\w+[2,10]/;
  if (!emlRe.test(eml)) return false;
});});
</script>
```

Нужно найти
более универсальное решение

Решений для client-side валидации не так много

HTML 5 Forms

- не кроссбраузерно
- только элементарные валидации
- правила — в вёрстке или в js
- сообщения об ошибках неясно где
- `#fail :-(`

Решений для client-side валидации не так много

JavaScript libraries

- не предлагают отдельного формата описания правил
- не отделяют валидацию от отображения ошибок
- не умеют важного: препроцессинг, группы правил, условные правила, инверсия проверок, одноимённые элементы...
- jQuery Validation — хорошая альтернатива

По́том и кровью выстраданные принципы:

- проверка может включать **более одного поля**
- валидация может зависеть **от внешних условий**
- тексты ошибок **отделяем** от самих правил
- правила **неудобно** хранить в вёрстке
- для валидации нужен **препроцессинг**
- нужен атомарный **code reuse**

Что такое SVARX?

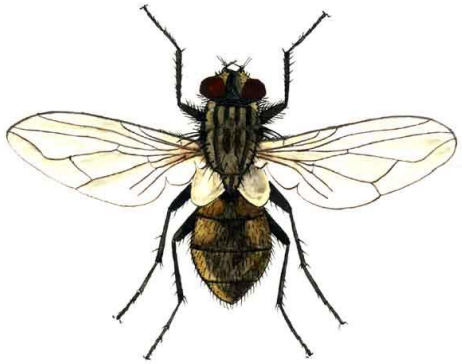
SVARX — это...
Semantical **VA**lidation **R**ulesets in **X**ML

Идеология работы с правилами валидации.

Мы описываем отдельно:

- Смысловые (**семантические**) правила
- Реализацию проверок
- Тексты ошибок (если нужно)
- Способ реакции на эти ошибки

SVARX — это...
Semantical **VA**lidation **R**ulesets in **X**ML



Мухи — отдельно

Котлеты — отдельно

SVARX

- **human-readable** XML-формат
- связки **and**-**or**-**not**, условия **if**-**then**-**else**
- **любые** правила (built-in + user-defined)
- семантически описанный **препроцессинг**

И ещё...

...есть плагин для ***jQuery***, который умеет:

- **читать** (парсить) SVARX-файлы
- **валидировать** веб-формы в соответствии со SVARX-правилами
- интегрировать любые способы **показа ошибок** *
- добавлять **user-defined правила** *

* *плагины второго уровня на основе API*

Как устроен SVARX-формат

Общая структура

```
<svarx>
   <preprocess>
      <!-- правила препроцессинга -->
   </preprocess>

   <validate>
      <!-- правила валидации -->
   </validate>
</svarx>
```

Каркас типичного правила

```
<svarx>
   <preprocess>
      <!-- правила препроцессинга -->
   </preprocess>

   <validate>
      <rule
         for="имя поля формы"
         type="тип валидации"
         onerror="идентификатор ошибки"
         ... доп. атрибуты, специфичные для правила

      />

   </validate>
</svarx>
```

Простой пример

```
<preprocess>
    <rule for="age" type="parseint" />
</preprocess>
<validate>
    <rule
        for="age" type="required"
        onerror="age_not_specified" />
    <rule
        for="age" type="range"
            min="14"
            max="99"
        onerror="age_incorrect" />
</validate>
```

Кумулятивное правило

```
<block logic="and" onerror="epic_fail">

    <rule for="user_email" type="email" />
    <rule for="age" type="required" />
    <rule for="age" type="range"
          min="14"
          max="99" />

</block>
```

Многоэлементное правило

```
<rule type="eq" comparison="string"
      onerror="fail">
    <!-- сравниваем 2 значения -->
    <el name="password1" />
    <el name="password2" />

    <!-- опционально можем переопределять
         элемент, который будет target'ом
         svarx-ошибки  -->
    <errtarget name="password1" />
</rule>
```

Условные проверки

```
<block logic="if">
    <!-- IF-условие -->
    <rule for="agreed" type="checked" />
    <!-- THEN-условие -->
    <block onerror="need_email">
        <rule for="email" type="required" />
        <rule for="email" type="email" />
    </block>
    <!-- ELSE-условие (опционально) -->
    <block onerror="tel_number_required">
        <rule for="tel" type="required" />
    </block>
</block>
```

Ещё вкусности

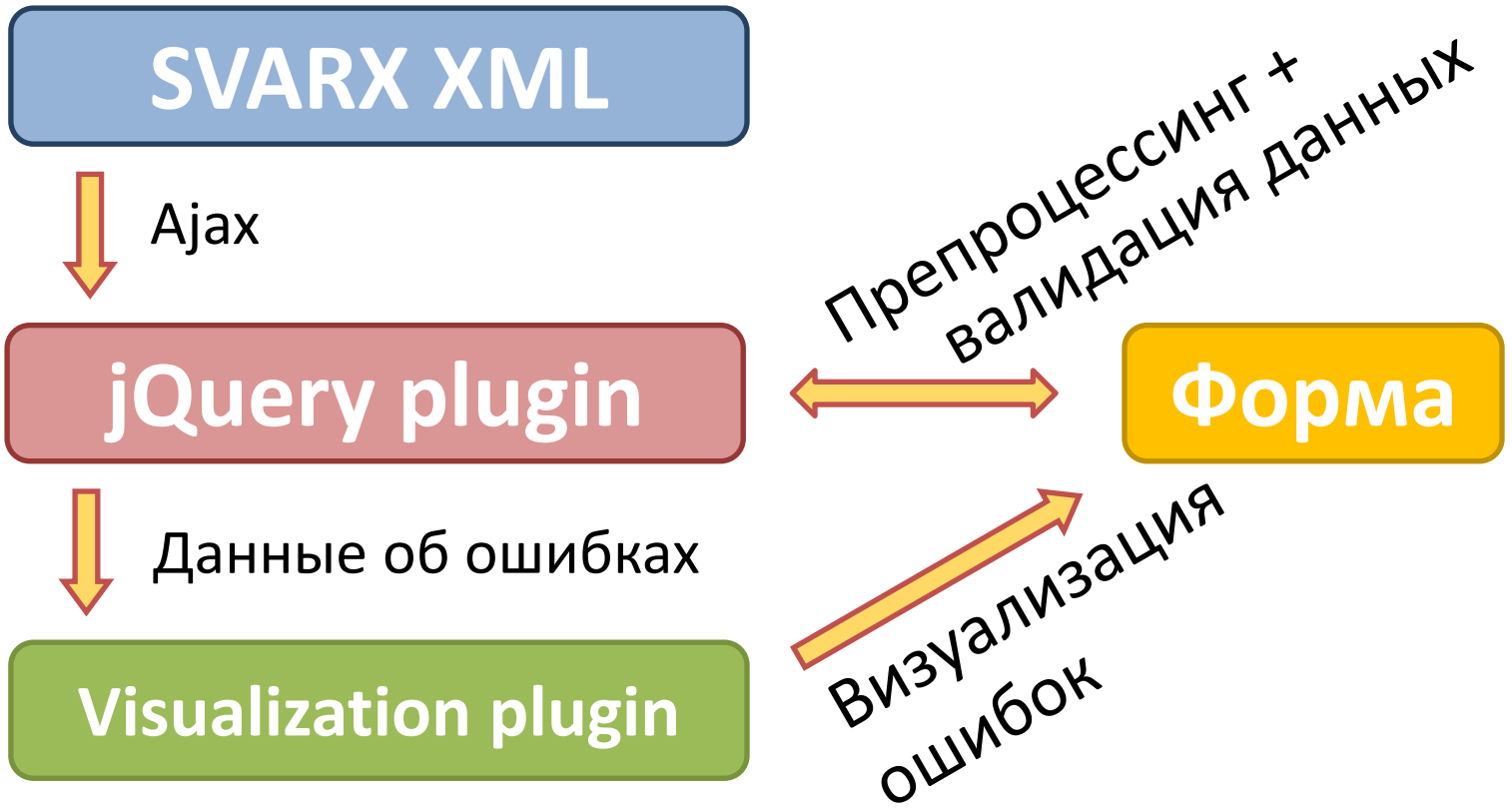
```
<!-- отрицание NOT -->
<rule for="address" type="regexp"
      match="[A-Z]"
      inverted="yes"
      onerror="cant_have_CAPS" />

<!-- выборка из одноимённых элементов -->
<rule for="address" type="email"
      item="0"
      onerror="fail" />
```

Важные договорённости

- пустые элементы дают TRUE на любой проверке...
- ...кроме правила required
- несуществующие элементы дают TRUE на любой проверке
- условные проверки от несуществующих элементов пропускаются целиком (IF-THEN-ELSE)
- ...кроме случая, когда мы явно укажем обратное failifnull=“yes”

SVARX и JavaScript

Архитектура SVARX на клиенте



Подключение на проект

```
<script type="text/javascript"
  src="jquery.svarx.js"></script>
<script type="text/javascript"
  src="svarx.methods.messages.js"></script>
<script type="text/javascript">
$(function() {
  $('form').svarx({
     svarxURL: '/validation/svarx.xml',
     bindTo: 'submit',
     immutable: true
  });
});
</script>
```

Что такое «плагин визуализации»?

```
$.fn.svarx.methods.messages = {
    before: function(e) {},
    after:  function(e, result, eType) {
        if (!result && eType == 'submit') {
            e.preventDefault();
        }
    },
    error: function(e, id) {
        alert('Error '
            + id
            + ' at element '
            + e.target.name);
    }
};
```

Что мы с этого имеем?

Профит

- set it and forget it
- перестаём писать JS-валидацию на коленках
- порою — вообще перестаём писать JS-валидацию
- все знания о валидации в одном файле
- правим отображение ошибок независимо от правил

Где попробовать

Документация + последние версии:

https://github.com/ingdir/svarx

Использование в боевых условиях :

http://sprav.yandex.ru/

http://passport.yandex.ru/

To Do

- нет асинхронных проверок
- нет условий, не зависящих от состояния формы
- XML — not the way to go?
- скорость
- нет «компиляции» SVARX XML в JS-код
- мало syntax sugar внутри самого формата

Вопросы?

Макс Ширшин
Руководитель группы разработки
интерфейсов Рекламных Технологий

ingdir@yandex-team.ru