Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
Филиал
«Минский радиотехнический колледж»

Учебная дисциплина «Программные средства создания Internet-приложений»

Инструкция

по выполнению лабораторной работы
«Проверка корректности информации, введенной пользователем. Обработка данных формы»

Минск
2021

Лабораторная работа № 25

Тема работы: Проверка корректности информации, введенной пользователем. Обработка данных формы

1. Цель работы

Формирование умений реализации проверки корректности информации, введенной пользователем в элементы формы, организовывать обработку данных форм средствами JavaScript.

2. Задание

Выполнить задания в соответствии с порядком выполнения лабораторной работы.

3. Оснащение работы

ПК, редактор исходного кода, браузер.

4. Основные теоретические сведения

При работе с формами в JavaScript используется объект **Form,** который также является очередным свойством объекта Document.

Доступ к форме осуществляется по её имени или по индексу.

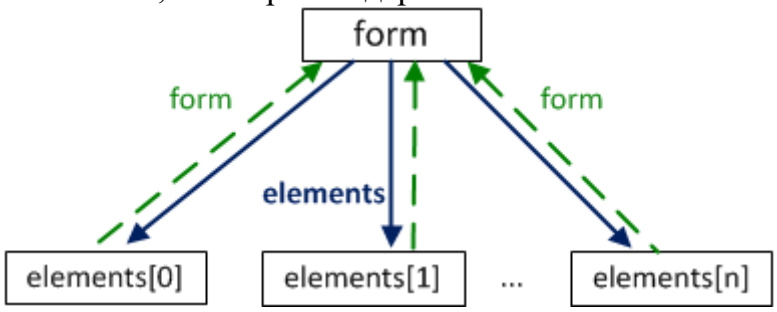
Пример:

```
document.forms.my -- форма с именем 'my'
document.forms[0] -- первая форма в документе
```

Общий вид получения доступа к полю формы по ее имени следующий:

```
document.имя_формы.имя_поля.value;
```

Доступ к элементу формы осуществляется аналогично свойству document.forms. Каждая форма имеет свойство **elements**, в котором содержится список всех элементов формы.




```
<html>
<head>
  <title>DOM интерфейс</title>
</head>
<body>
  <form name="search">
    <input name="query">
  </form>
</body>
</html>

<script>
  let form = document.forms.search;
  let input = form.elements.query; // или
  let input = form.elements['query']; // или
```

```
  let input = form.elements[0];
</script>
```

Если в форме используется несколько элементов с одинаковым именем, то обращение к элементу по имени вернёт список всех элементов с этим именем.

```
<html>
<head>
<title>DOM интерфейс</title>
</head>
<body>
<form name="search">
<input type="radio" name="query" value="1">
<input type="radio" name="query" value="2">
</form>
</body>
</html>

<script>
let form = document.forms.search;
let input = form.elements.query;
alert(input[0].value); /* 1 */
alert(input[1].value); /* 2 */
</script>
```

Свойство elements также есть у элементов <**fieldset**>.

```
<body>
  <form>
    <fieldset name="set">
      <legend>fieldset</legend>
      <input name="text" type="text">
    </fieldset>
  </form>

  <script>
    let form = document.forms[0];

    alert( form.elements.text ); // INPUT
    alert( form.elements.set.elements.text ); // INPUT
  </script>
</body>
```

По элементу можно получить его форму, используя свойство element.form.

```
<body>
<form>
  <input type="text" name="surname">
</form>

<script>
let form = document.forms[0];

let elem = form.elements.surname;
elem;             //  <input type="text" name="surname">
elem.form;        // <form>…</form>
</script>
</body>
```

Кнопки (BUTTON, RESET, SUBMIT)	
Свойства	
Name	Имя объекта
Value	Надпись на кнопке

Метод	
click( )	Вызов этого метода тождественен щелчку мышкой по кнопке
Флажок (CHECKBOX)	
Свойства	
name	Имя объекта
value	Надпись на кнопке
checked	Состояние флажка: true - флажок установлен, false - флажок не установлен
defaultChecked	Отражает наличие атрибута CHECKED: true - есть, false – нет
Метод	
click( )	Вызов этого метода меняет состояние флажка
Переключатель (RADIO)	
Свойства	
name	Имя объекта
value	Надпись на кнопке
length	Количество переключателей в группе
checked	Состояние переключателя: true - переключатель включен, false – выключен
defaultChecked	Отражает наличие атрибута CHECKED: true - есть, false – нет
Метод	
click( )	Вызов этого метода включает переключатель
Так как группа переключателей имеет одно имя NAME, то к переключателям надо адресоваться как к элементам массива.	
Список (SELECT)	
Свойства	
name	Имя объекта
selectedIndex	Номер выбранного элемента или первого среди выбранных (если указан атрибут MULTIPLE)
length	Количество элементов (строк) в списке
options	Массив элементов списка, заданных тегами OPTION
Методы	
focus( )	Передает списку фокус ввода
blur( )	Отбирает у списка фокус ввода
*Каждый элемент массива **options** является объектом со следующими свойствами:*	
value	Значение атрибута VALUE
text	Текст, указанный после тега OPTION
index	Индекс элемента списка
selected	Присвоив этому свойству значение true, можно выбрать данный элемент
defaultSelected	Отражает наличие атрибута SELECTED: true - есть, false – нет
Поле ввода (TEXT)	
Свойства	
name	Имя объекта
defaultValue	Начальное содержимое поля
value	Текущее содержимое поля
Методы	
focus( )	Передает полю фокус ввода
blur( )	Отбирает у поля фокус ввода
select( )	Выделяет содержимое поля
Текстовая область (TEXTAREA)	

Свойства	
name	Имя объекта
defaultValue	Начальное содержимое области
value	Текущее содержимое области
Методы	
focus( )	Передает области фокус ввода
blur( )	Отбирает у области фокус ввода
select( )	Выделяет содержимое области

Для установки курсора в определенное место textarea-области можно воспользоваться следующей кроссбраузерной функцией:

```
function setCaretPosition(o, pos) {
      o=document.getElementById(o);
      if(o.setSelectionRange) {
         o.focus();
         o.setSelectionRange(pos,pos);
      }else if (o.createTextRange) { // IE
         var range = o.createTextRange();
         range.collapse(true);
         range.moveEnd('character', pos);
         range.moveStart('character', pos);
         range.select();
      }
   }
```

Свойства	
name	Имя объекта
defaultValue	Начальное содержимое поля
value	Текущее содержимое поля
Методы	
focus( )	Передает полю фокус ввода
blur( )	Отбирает у поля фокус ввода
select( )	Выделяет содержимое поля

Свойство **value** содержит текущее введённое пользователем значение. До начала ввода пользователем нового значения данное свойство определяется атрибутом value, если он указан.

```
let form = document.forms.user_form;
let input = form.elements.user_input;

function showValue() {
alert(input.value); /* всегда выводит текущее значение */
}
```

У элементов с типом **checkbox** и **radio** свойство **value** не может изменяться пользователем. При их использовании необходимо знать, выбраны ли они пользователем. Для этого используется логическое свойство **checked**. Оно отражает текущее состояние элемента (отмечен или не отмечен) и позволяет его изменить.

```
<body>
<form name="search" autocomplete="off">
<input type="radio" name="query" value="1" checked>
<input type="radio" name="query" value="2" >
</form>
</body>
</html>

<script>
let form = document.forms.search;
let input = form.elements.query;
```

```
if (input[0].checked) {
let a=input[0].value
alert(a);}
else {
alert(input[1].value); }
</script>
```

Метод **select()** используется для выделения введённого пользователем текста. При этом поле становится активным.

```
<body>
  <form name="search">
    <label>Введите что-нибудь: <input name="text_in" oninput="show()"></label>
    <br>
    <label>Вы ввели: <input name="text_out"></label>
  </form>
</body>
</html>

<script>
  let form = document.forms.search;
  let text_in = form.elements.text_in;
  let text_out = form.elements.text_out;
  text_in.focus();

function show() {
  text_out.value = text_in.value;
  text_out.select(); /* выделяет содержимое */
}
</script>
```

Элемент **<textarea>** аналогичен полю <input>. Для него так же доступны методы **focus()**, **blur()**, **select()**. Их значения и действия полностью совпадают с теми же методами элемента <input>.

Элемент <textarea> не имеет атрибута value, но свойство value всё равно имеет. Его значением является содержимое тега.

Элемент **<select>** не имеет атрибута value. Однако, свойство value всё равно присутствует. Его значение берётся из той опции (из её свойства value), которая выбрана в данный момент. Если выбрано несколько опций, то значение берётся из первой.

```
<body>
<form name="test" autocomplete="off">
<select name="user_select" multiple>
<option value="1" selected>Число 1</option>
<option value="2" selected>Число 2</option>
</select>
</form>
</body>
<script>
let form = document.forms.test;
let select = form.elements.user_select;
alert(select.value); /* 1 */
</script>
```

Селект в JavaScript можно установить двумя путями: поставив значение select.value, либо установив свойство select.selectedIndex в номер нужной опции.

Свойство **selectedIndex** содержит индекс той опции, которая выбрана в данный момент (если используется атрибут multiple, то свойство содержит индекс первой из выбранных опций). Изменяя данное свойство, можно изменить текущее значение в поле<select>. Нумерация начинается с нуля. Значение”-1”устанавливается для очистки поля.

```
<body>
<form name="test" autocomplete="off">
<select name="user_select">
```

```
<option value="1">Число 1</option>
<option value="2" selected>Число 2</option>
<option value="3">Число 3</option>
</select>
</form>
</body>
<script>
let form = document.forms.test;
let select = form.elements.user_select;
alert(select.selectedIndex); /* 1 */
select.selectedIndex = -1;
alert(select.value); /* пустая строка */
</script>
```

Свойство **options** содержит список всех тегов <option> внутри данного элемента <select>.

```
<body>
<form name="user_form" onsubmit="showNum(); return false" autocomplete="off">
<select name="user_select">
<option value="1">Число 1</option>
<option value="2">Число 2</option>
<option value="3">Число 3</option>
<option value="4">Число 4</option>
<option value="5">Число 5</option>
<option value="6">Число 6</option>
</select>
<input type="submit" value="Отправить">
</form>
</body>
<script>
let form = document.forms.user_form;
let select = form.elements.user_select;


function showNum() {
alert('Доступно опций: ' + select.options.length);
}
</script>
```

Список элементов-опций доступен через select.options. Свойство **selectedOptions** содержит список тех тегов <option>, которые в данный момент выбраны.

Если select допускает множественный выбор (атрибут multiple), то значения можно получить/установить, сделав цикл по select.options. При этом выбранные опции будут иметь свойство option.selected = true.

```
<form name="form">
  <select name="genre" multiple>
    <option value="blues" selected>Мягкий блюз</option>
    <option value="rock" selected>Жёсткий рок</option>
    <option value="classic">Классика</option>
  </select>
</form>

<script>
let form = document.forms[0];
let select = form.elements.genre;

for (let i = 0; i < select.options.length; i++) {
  let option = select.options[i];
  if(option.selected) {
    alert( option.value );
  }
}
</script>
```

Элементы **<option>**имеют булевое свойство selected. Оно характеризует состояние опции: выбрана или не выбрана.

```
<body>
<form name="test" autocomplete="off">
<select name="user_select">
<option value="1">Число 1</option>
<option value="2" selected>Число 2</option>
</select>
</form>
</body>
<script>
let form = document.forms.test;
let select = form.elements.user_select;
alert(select.options[0].selected); /* false */
alert(select.options[1].selected); /* true */
</script>
```

В стандарте **the option element** есть короткий синтаксис для создания элемента с тегом option:

```
option = new Option(text, value, defaultSelected, selected);
```

Параметры:

text – содержимое;

value – значение;

defaultSelected и **selected** необходимо поставить в true, чтобы сделать элемент выбранным.

Описанный способ можно использовать вместо document.createElement('option'), *например:*

```
let option = new Option("Текст", "value");
// создаст <option value="value">Текст</option>
Такой же элемент, но выбранный:
let option = new Option("Текст", "value", true, true);
```

5. Порядок выполнения работы

1. Создать html-документ, содержащий форму. На форме должны присутствовать текстовое поле для ввода, зависимый и независимые переключатели, выпадающий список, кнопки <button>, <input type=’submit’>, <input type=’reset’> и многострочное текстовое поле.

2. Напишите сценарий, который при первом нажатии на кнопку <button>, добавит в текстовое поле текст «текстовое поле». При последующих нажатиях на указанную кнопку реализуйте переключение цвета текста между двумя произвольными цветами (второй клик установит цвет, а последующие – будут переключать).

3. Напишите сценарий, который, при нажатии на кнопку <input type=’reset’> будет получать текст из многострочного текстового поля, запрашивать у пользователя искомую строку, сравнивать содержимое поля со значением, введенным пользователем и выдавать соответствующий результат.

4. Напишите сценарий, который по двойному клику на кнопке <input type=’submit’> выведет на веб-страницу контент вида: переключатель: выбранный элемент radio, флажки: список отмеченных флажков, выпадающий список: выбранный элемент/элементы списка. При этом, если хоть один из элементов не выбран (radio, checkbox, select), необходимо вывести сообщение, содержащее элементы, которые необходимо заполнить. Выполнить стилистическое оформление выводимого результата.

5. Напишите сценарий, который по нажатию на кнопку <button> будет создавать

и вызывать событие для элемента <input type=’submit’> ‘click’.

6.* Добавьте на веб-страницу блок, содержащий тест, в соответствии с вариантом. Тест должен иметь заголовок. Необходимо предусмотреть различные варианты ответов: единственный верный из предложенных, несколько верных из предложенных, ответ, введенный пользователем. После ответа на все вопросы, по нажатию кнопки «Результат», необходимо вывести результат теста с указанием вопросов, на которые даны неверные ответы.

Вариант	Тема теста
1	Явления природы
2	Автомобили
3	Растения
4	Животные
5	Графические редакторы
6	Операционные системы
7	Языки программирования
8	Базы данных
9	Язык гипертекстовой разметки HTML
10	История Беларуси
11	Персонажи сказок
12	Известные люди
13	Чудеса Света
14	Рекорды Гиннеса
15	Бытовая техника

6. Форма отчета о работе

Лабораторная работа № ___

Номер учебной группы _____

Фамилия, инициалы учащегося _____

Дата выполнения работы _____

Тема работы: _____

Цель работы: _____

Оснащение работы: _____

Результат выполнения работы: _____

_____

_____

7. Контрольные вопросы и задания

1. Для чего в JavaScript используется объект Form?
2. Опишите способы организации доступа к форме.
3. Каким образом осуществляется доступ к элементам формы?
4. В чем заключается валидация формы?

8. Рекомендуемая литература

1. **JAVASCRIPT.RU** [Электронный ресурс] / Современный учебник JavaScript – 2007—2020 Илья Кантор. – Режим доступа: https://learn.javascript.ru. – Дата доступа: 04.03.2020.

2. **Никсон, Р.** Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 / Р. Никсон. 4-е изд. – СПб.: Питер, 2018.

3. **Симпсон, К.** ES6 и не только / К. Симпсон. – СПб.: Питер, 2017.

4. **Хавербеке, М.** Выразительный JavaScript. Современное веб-программирование / М. Хавербеке – СПб.: Питер, 2019.