Работа с формами

работа с текстовыми полями и проверка их содержимого

Формы

Веб-форма представляется уже знакомым для нас элементом html - тегом <form>, а также типом HTMLFormElement в JS, который наследуется от типа HTMLElement.

Формы имеют некоторые свойства и методы по умолчанию.

Некоторые из них:

name - имя формы (эквивалент HTML-атрибута name);

submit( ) - отправляет данные формы;

reset( ) - сбрасывает все поля формы, восстанавливает значения по умолчанию;

Работа с формами

для того, чтобы получить форму, достаточно воспользоваться методом getElementById.

Помимо этого, все формы на странице хранятся в коллекции document.forms.

Можно получить форму из этой коллекции по индексу, либо имени:

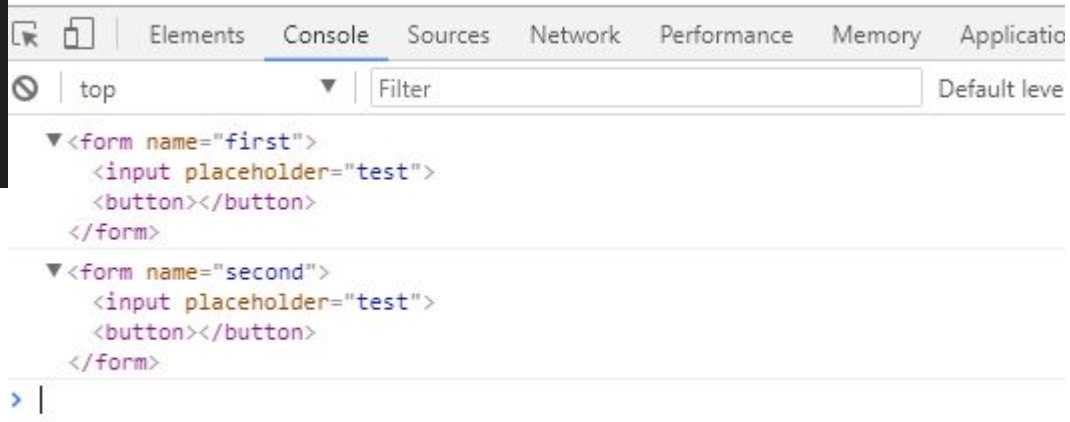
(см. след. слайд)

```html
<!--First form -->
<form name="first">
  <input placeholder="test">
  <button></button>
</form>

<!--Second form -->
<form name="second">
  <input placeholder="test">
  <button></button>
</form>
```

```js
var first = document.forms[0];
var second = document.forms['second'];

console.log(first);
console.log(second);
```



Отправка данных формы

Данные формы отправляются к серверу, когда пользователь щёлкает на кнопке отправки либо на графической кнопке. Роль кнопки отправки играет <button> , у которого атрибут type имеет значение submit (по умолчанию), а так же тег input, атрибут type которого может иметь значения submit либо image.

Пример :

```
<!--обобщённая кнопка отправки -->
<input type="submit">
<!--графическая кнопка -->
<input type="image">
<!--пользовательская форма отправки -->
<button type="submit"></button>
```

Отправка и отмена отправки

Когда данные с формы отправляются, перед отправкой в браузере генерируется событие submit. В его обработчике можно проверить введенные в форме данные, и при необходимости заблокировать их отправку, отменив для события действие, предлагаемое по умолчанию.

Помимо этого, в любой момент можно отправить данные с формы программно, для этого не требуется даже наличия самой кнопки отправки. Достаточно лишь самой формы и обработчика:

```
var form = document.getElementById('newForm');
//отправка данных с формы
form.submit ();
```

Проблема

При такой отправке данных формы событие submit не генерируется, поэтому проверять введенные данные лучше заранее.

Одна из главных проблем при работе с формами - многократная отправка. Иногда пользователи от нетерпения кликают много раз по кнопке “отправить”, что приводит к перегрузке сервера и другим проблемам.

Решения всего два - отключение кнопки отправки после того, как данные отправлены , и отмена последующих попыток отправки данных в обработчике submit.

Пользовательский сброс формы

Сбросить форму можно, нажав на кнопку сброса - элемент <input>/<button>, у которого атрибут type имеет значение reset.

```
<!--Second form -->
<form name="second">

  <input placeholder="test">
  <input type="submit">
  <input type="reset">
</form>
```

Программный сброс формы

Также сбросить форму можно программно, при помощи метода reset(), который в отличие от метода submit(), генерирует соответствующее событие.

```
var form = document.getElementById('newForm');
//сброс формы
form.reset ();
```

Поля форм

Как и другие элементы страницы, элементы форм доступны с помощью встроенных DOM-методов. Помимо этого, все элементы любой формы содержатся в ее коллекции elements - упорядоченном списке ссылок на все поля формы и все элементы <input>, <textarea>, <select>, <fieldset>.

Поля формы хранятся в коллекции elements в том же порядке, что и в разметке, и индексируются по позиции либо по имени:

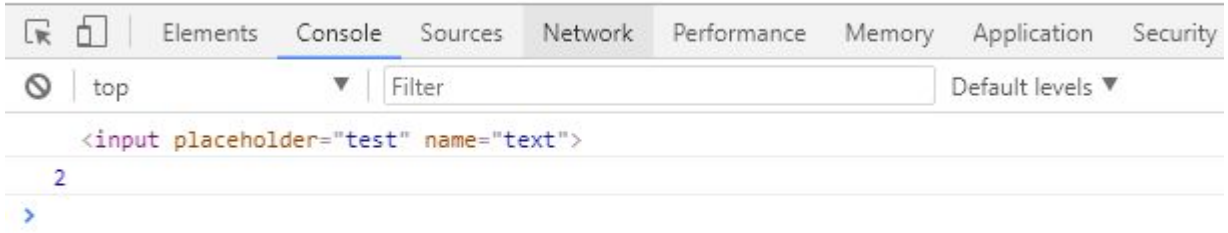
(см.след.слайд) ======>

```html
<!--First form -->
<form name="first-form">
  <input placeholder="test" name="text">
  <button></button>
</form>
```

```js
var form = document.forms[0];


var field = form.elements['text'];
var inputLenth = form.elements.length;

console.log(field);
console.log(inputLenth);
```



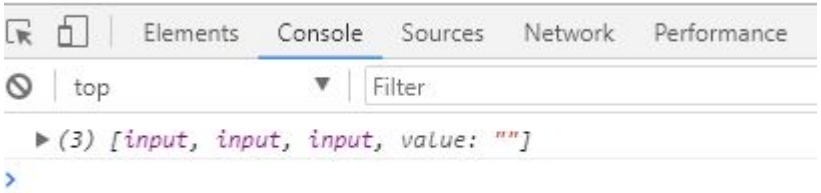
Несколько name

Если одно имя идентифицирует несколько элементов формы, то все элементы возвращаются в коллекции HTMLCollection:

```html
<!--First form -->
<form name="first-form">
  <input type="radio" name="color" value="green">
  <input type="radio" name="color" value="red">
  <input type="radio" name="color" value="blue">
  <button></button>
</form>
```

```js
var form = document.forms[0];

var colorFields = form.elements['color'];
console.log(colorFields);
```



Общие свойства полей форм

Практически все поля форм имеют несколько общих свойств. Так как многие поля представляют собой тип <input>, одни свойства используются только с полями определенного типа, другие же доступны для всех типов полей.

Некоторые из них:

disabled - логическое значение, указывающее, отключено ли поле

name - имя поля

readOnly - логическое значение, указывающее, доступно ли поле только для чтения

tabIndex - порядок перехода по нажатию клавиши табуляции

type - тип поля (checkbox/radio ...)

value - значение поля, отправляемое серверу.

form - указатель на форму, к которой относится поле

Все свойства, кроме form, можно изменять динамически. Например:

```
var form = document.forms[0];

var field = form.elements['txt'];

// изменение значения
field.value = 'Another value';
//изменение типа поля
field.type = 'checkbox';
```

Отключение формы

Возможность динамически изменять свойства полей форм позволяет в любой момент модифицировать форму самыми разными способами. Например, иногда пользователь кликает слишком много раз по кнопке отправки. Это может привести к выставлению нескольких счетов в приложениях электронной коммерции.

В связи с этим полезно отключать работу кнопки отправки после первого клика по ней, или, другими словами, при генерировании события submit.

Отключение кнопки отправки

*естественно, данный код помещается в обработчик события submit

```
var form = document.forms[0];

var btn = form.elements['submit-btn'];
// Отключение кнопки отправки
btn.disabled = 'true';
```

Свойство type

Оно есть практически у всех полей формы, у элементов input оно имеет то же значение, что содержание атрибута type в соответствующем теге.

У элементов input и button свойство type можно изменять динамически, у элемента select оно доступно только для чтения, изменять его нельзя.

Некоторые свойства полей:

список с возможностью одиночного выбора	<select></select>	‘select-one’
список с возможностью множественного выбора	<select multiple></select>	‘select-multiple’
Пользовательская кнопка	<button>	‘submit’
Пользовательская кнопка (не отправки)	<button type=”button”>	‘button’
Пользовательская кнопка сброса	<button type=”reset”>	”reset”
Пользовательская кнопка отправки	<button type=”submit”>	”submit”

Задание 1

Создать форму в HTML с 2 полями ввода и кнопкой отправки. Получить эту форму на событие отправки этой формы (т е нажатие кнопки), сделать кнопку неактивной и очистить форму

Задание 2

Создать форму в HTML с полем ввода и кнопкой отправки, на submit формы менять значение поля ввода на произвольное

Задание 3

Создать форму в HTML с полем ввода, несколькими checkbox и кнопкой отправки. На submit формы вывести значения поля ввода и имена всех отмеченных полей checkbox.

Общие методы полей форм

У каждого поля формы есть метод focus() и blur ();

Первый назначает фокус полю формы, то есть делает его активным, после чего оно начинает реагировать на события клавиатуры.

Например, в текстовом поле, получившем фокус, появляется курсор, показывающий, что оно готово принимать ввод.

При загрузке

при перезагрузке либо загрузке обычно фокус задают первому полю формы:

*при выполнении кода возникнет ошибка, если первое поле формы это input с атрибутом hidden, либо если оно скрыто при помощи css-свойств.

```
var form = document.forms[0];

var btn = form.elements['submit-btn'];

// добавление фокуса полю (при загрузке )
btn.focus();
```

Метод blur()

Этот метод отменяет полю фокус и не назначает его никакому другому полю. Раньше он часто использовался, когда не было атрибута readonly, на данный момент практически не используется.

*Когда к тегу <input> добавляется атрибут readonly, текстовое поле не может изменяться пользователем, в том числе вводиться новый текст или модифицироваться существующий. Тем не менее, состояние и содержимое поля можно менять с помощью js.

Общие события полей форм

Все поля форм поддерживают три события:

blur - генерируется при утрате фокуса полем

change - генерируется для элементов input и textarea при утрате фокуса, если свойство value было изменено, а также для элементов select при выборе другого элемента из списка

focus - генерируется при получении фокуса полем

Работа с текстовыми полями

Содержимое текстовых полей тегов input и textarea хранится в свойстве value, которое можно использовать для чтения и задания значения текстового поля, например:

```
alert('form.value');
```

Метод substring

Метод substring возвращает подстроку, начиная с позиции indexA до, но не включая indexB.

```
str.substring(indexA, [indexB])
```

Частичное выделение текста

В HTML5 разрешается выделять фрагменты текстовых полей. Для этого используется метод setSelectionRange(). Он принимает два аргумента: индекс первого выделяемого символа, и индекс конца выделения

```
<input type="text" id="mytextbox" value="my name is"/>

<script>
    let input = document.getElementById("mytextbox");
    input.focus();
    input.setSelectionRange(3,5); // выделено 'na'
    input.setSelectionRange(0,2); // выделено 'my'
    input.setSelectionRange(1,8); // выделено 'y name i'
</script>
```

У каждого поля можно назначить максимальное количество вводимых символов:

```
<input type="radio" name="color" value="green" maxlength="5">
```

Задание 4

Создать форму в HTML с полем ввода и кнопкой отправки. На submit формы, если поля ввода пустое, делать фокус на поле ввода.

Задание 5

Создать форму в HTML с полями для ввода имени, фамилии, телефона и сообщения. Для поля телефона задать максимальное количество символов 13, сообщение может быть большим. При submit формы создавать объект со всеми значениями формы и выводить его.