

# РАБОТА С HTML-ФОРМАМИ



МИХАИЛ КУЗНЕЦОВ / ING



# МИХАИЛ КУЗНЕЦОВ

Developer ING



[@mkuznetcov](https://www.instagram.com/mkuznetcov)



# ПЛАН ЗАНЯТИЯ

1. Текстовые поля
  - Веб-формы
  - Получаем значение поля
  - Особенности поля для ввода текста
  - События
2. Выбор одного или нескольких значений
3. Итоги



# ТЕКСТОВЫЕ ПОЛЯ



# ОБЩАЯ МОДЕЛЬ ОРГАНИЗАЦИИ ВЕБ-ПРИЛОЖЕНИЯ

Что мы знаем о взаимодействии с пользователем на текущий момент:

1. Пользователь может совершать какие-то действия на странице, кликать, набирать текст.
2. Браузер на эти действия генерирует события.
3. Мы можем повесить обработчик на нужные нам события.
4. Получить дополнительную информацию о возникшем событии.
5. И выполнить какие-то полезные действия.

# УСТАРЕВШИЕ ФУНКЦИИ

Функции `alert`, `prompt`, `confirm` позволяют проинформировать пользователя и получить какую-то информацию от него через диалоговое окно. Но они устарели и имеют большое число недостатков:

- Пользователь не может продолжить взаимодействовать со страницей, пока не отреагирует на диалог.
- Функции синхронные и блокируют основной поток JavaScript. Веб-приложение встает на паузу.
- Политика браузеров мотивирует не использовать диалоговые окна в своих веб-приложениях. Например, [Google Chromium](#) в марте 2017.
- Пользователь может заблокировать показ таких окон для страницы, сайта или вообще для всех сайтов.

# ВЕБ-ФОРМЫ – ЛУЧШАЯ АЛЬТЕРНАТИВА

Используя поля HTML-формы, мы можем решить те же задачи и предоставить пользователю лучший опыт использования нашего приложения. При этом модель организации приложения не отличается от той, которую мы уже активно используем:

1. Пользователь может совершать какие-то действия с полями формы, кликать, набирать текст.
2. Браузер на эти действия генерирует события.
3. Мы можем повесить обработчик на нужные нам события.
4. Получить дополнительную информацию о возникшем событии.
5. И выполнить какие-то полезные действия.

# ПРИВЕТСТВУЕМ ПОЛЬЗОВАТЕЛЯ

Давайте запросим имя пользователя и выведем сообщение

Привет, Иван! , когда он его введет.

Разметка формы будет выглядеть так:

```
1 <fieldset>
2   <label for="name">Укажите имя:</label>
3   <input id="name" type="text">
4   <button id="set">Войти</button>
5 </fieldset>
6 <output id="message"></output>
```



# ПОЛУЧАЕМ ЗНАЧЕНИЕ ПОЛЯ

После клика на кнопку мы можем обратиться к тегу

`<input type="text">` и запросить текст, который пользователь в него ввел.

Практически у всех элементов формы есть атрибут `value` и одноименное свойство `value`, в котором хранится актуальное значение. Обратимся к нему при клике на кнопку:

```
1  const setButton = document.getElementById('set');
2  const nameField = document.getElementById('name');
3  function showMessage() {
4      console.log(nameField.value);
5  }
6  setButton.addEventListener('click', showMessage);
```

## ВЫВОДИМ СООБЩЕНИЕ

На самом деле, мы уже знаем хороший способ вывести сообщение — поместить его в свойство `innerHTML` (или `innerText`, если не требуется форматирование).

Но предлагаю рассмотреть еще один вариант — использовать новый HTML5 тег `<output>`, который так же, как и текстовые поля, имеет атрибут и свойство `value`, через которое можно задать результат.

```
1  const result = document.getElementById('message');
2  function showMessage() {
3      result.value = `Привет, ${nameField.value}!`;
4  }
```

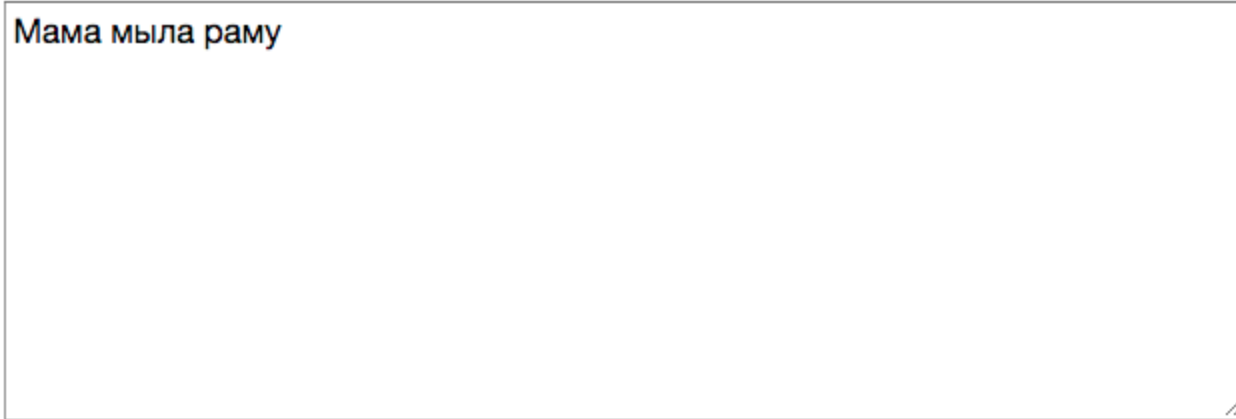
[Полный пример](#)

# ОСОБЕННОСТИ ПОЛЯ ДЛЯ ВВОДА ТЕКСТА

Кроме тега `<input type="text">`, есть еще тег `<textarea>`:

У тега `<textarea>` нет атрибута `value`. Это парный тег, содержимое поля записывается между открывающим и закрывающим тегом:

```
<textarea id="message">Мама мыла раму</textarea>  
<output id="result"></output>
```



Мама мыла раму

## ПОЛУЧАЕМ ЗНАЧЕНИЕ ТЕКСТОВОГО ПОЛЯ

Но у объекта, который представляет тег `textarea` в JavaScript, свойство `value` все равно есть и работает точно так же:

```
1  const textField = document.getElementById('message');  
2  const output = document.getElementById('result');  
3  
4  output.value = textField.value;
```

# ПЕРЕЗАПИСЫВАЕМ ЗНАЧЕНИЕ ТЕКСТОВОГО ПОЛЯ

Свойство `value` текстового поля также работает и на запись, аналогично тегу `<output>`:

```
1 | const textField = document.getElementById('message');  
2 |  
3 | textField.value = 'Папа мыл машину';
```

Поэтому в JavaScript эти поля работают идентично.

# СОБЫТИЯ

Пока мы только читали текущее значение поля ввода. А что, если мы хотим внимательно следить за взаимодействием пользователя с ним?

Кроме событий мыши `click` и клавиатуры `keydown`, `keyup` и `keypress`, браузер генерирует дополнительные события на поле ввода:

- `focus` и `focusin` – когда поле ввода становится активным (*в фокусе*) с помощью клика мышки по полю или по метке поля либо при переключении фокуса клавишей `Tab`.
- `input` – значение поля изменилось.
- `change` – значение поля изменилось и было зафиксировано; происходит при потере фокуса.
- `blur` и `focusout` – когда поле ввода теряет фокус.

## ЗАДАЧА «ПОДСКАЗКА»

У поля ввода есть атрибут `placeholder`, позволяющий задать текст, который будет отображаться в поле, пока оно пустое. Этот текст позволяет добавить краткую подсказку по заполнению поля. Давайте добавим более подробную подсказку, пока пользователь набирает текст:

```
1 <textarea id="message" placeholder="Сообщение">  
2 </textarea>  
3 <div id="hint" class="hidden">Текст подсказки!</div>
```

# ИСПОЛЬЗУЕМ СОБЫТИЕ ФОКУСА

При активации поля покажем слой с подсказкой. При потере фокуса скроем его:

```
1  const textField = document.getElementById('message');
2  const hintBox = document.getElementById('hint');
3  function showHint() {
4      hintBox.classList.remove('hidden');
5  }
6  function hideHint() {
7      hintBox.classList.add('hidden');
8  }
9  textField.addEventListener('focus', showHint);
10 textField.addEventListener('blur', hideHint);
```

[Полный пример](#)



## ЗАДАЧА «ПРИВЕТСТВИЕ ПОЛЬЗОВАТЕЛЯ»: ДОРАБАТЫВАЕМ СКРИПТ

Давайте доработаем наш скрипт приветствия так, чтобы он работал без кнопки:

```
1 <fieldset>
2   <label for="name">Укажите имя:</label>
3   <input id="name" type="text">
4 </fieldset>
5 <output id="message"></output>
```

# ОБНОВЛЯЕМ ПРИВЕТСТВИЕ ПРИ ВВОДЕ

Воспользуемся для этого событием `input` на поле ввода:

```
1  const nameField = document.getElementById('name');
2  const result = document.getElementById('message');
3
4  function showMessage() {
5      result.value = `Привет, ${nameField.value}!`;
6  }
7
8  nameField.addEventListener('input', showMessage);
```

[Полный пример](#)

## input ПРОТИВ change

Перепишем наш пример, используя событие `change`:

```
1  const nameField = document.getElementById('name');
2  const result = document.getElementById('message');
3
4  function showMessage() {
5      result.value = `Привет, ${nameField.value}!`;
6  }
7
8  nameField.addEventListener('change', showMessage);
```

[Полный пример](#). На первый взгляд может показаться, что он не работает. Но попробуйте переключить фокус клавишей `Tab`.

## РАЗЛИЧИЕ `input` И `change`


- Событие `input` срабатывает при любом изменении значения поля.
- Событие `change` срабатывает, когда поле изменено и **его изменение закончилось**.

## ОСОБЕННОСТЬ `change`

Как браузер поймет, что мы закончили вводить своё имя? Например, мы переключились к следующему полю или начали взаимодействие с другим элементом страницы.

Поэтому на текстовом поле событие `change` срабатывает только непосредственно перед потерей фокуса.

Если поле одно или оно последнее, то высока вероятность, что событие `change` вовсе не наступит. Пользователь введет текст и останется в нем.



# **ВЫБОР ОДНОГО ИЛИ НЕСКОЛЬКИХ ЗНАЧЕНИЙ**

# ЭЛЕМЕНТЫ ФОРМЫ

В HTML-формах есть следующие элементы:

- `<input type="radio">` – радиокнопка,
- `<input type="checkbox">` – флажок,
- `<select>` – выпадающий список,
- `<select multiple>` – выпадающий список с возможностью выбора нескольких вариантов.

# РАДИОГРУППА

Несколько радиокнопок, имеющих одинаковый атрибут `name`, объединяются в радиогруппу. Если в группе несколько кнопок, то выбрать можно только одну из них.

```
1 <fieldset>
2   <legend>Укажите пол:</legend>
3   <input type="radio" id="female" name="gender"
4     value="женский">
5   <label for="female">Женский</label>
6
7   <input type="radio" id="male" name="gender"
8     value="мужской">
9   <label for="male">Мужской</label>
10 </fieldset>
```



# СОСТОЯНИЕ И ЗНАЧЕНИЕ

У полей типа `radio` и `checkbox` есть два свойства:

- `checked` – состояние. Если флажок или радиокнопка выбраны, то будет равен `true`, иначе `false`.
- `value` – значение, аналогично текстовым полям.

## ОТЛИЧИЕ ОТ ТЕКСТОВЫХ ПОЛЕЙ

Ключевое отличие полей типа `radio` и `checkbox` от текстовых в том, как браузер будет использовать их при отправке формы на сервер:

- Значения текстовых полей отправляются «как есть», без каких-либо условий.
- Отправляются значения только тех флажков и радиокнопок, которые находятся в состоянии **выбран** (`checked`).



# ИТОГИ

# ТЕКСТОВЫЕ ПОЛЯ

- Функции `prompt`, `alert`, `confirm` – устаревший способ взаимодействия с пользователем. Лучшая альтернатива – использование веб-форм.
- Для большинства типов полей ввода значение получают и записывают через свойство `value`.
- Однострочные текстовые поля: `<input type="text">`.
- Многострочные поля ввода: `<textarea>`.
- Для группировки элементов формы используют HTML-тег `<fieldset>`.
- Для вывода значения можно либо поместить его в свойство `innerHTML` / `innerText`, либо воспользоваться HTML5-тегом `<output>` (также имеющим атрибут и свойство `value`).

# СОБЫТИЯ

- `focus`, `focusin` – срабатывает, когда поле ввода становится в фокусе (активным).
- `input` – срабатывает при любом изменении значения поля ввода.
- `change` – срабатывает при завершении изменения значения поля ввода (когда изменение зафиксировано).
- `blur`, `focusout` – срабатывает при потере фокуса.

# ВЫБОР ОДНОГО ИЛИ НЕСКОЛЬКИХ ЗНАЧЕНИЙ

- Радиокнопки: `<input type="radio">`. Атрибут `name` позволяет объединять радиокнопки в группы.
- Чекбоксы: `<input type="checkbox">`.
- Свойство `checked` радиокнопок и чекбоксов содержит логическое значение, `true / false` – в зависимости от того, выбран ли элемент. При отправке формы на сервер отправляются значения только «отмеченных» элементов.
- Свойство `value` радиокнопок и чекбоксов содержит строковое значение выбранного элемента (либо пустую строку).
- Выпадающие списки: `<select>`. Наличие атрибута `multiple` позволяет использовать множественный выбор.



**НЕТОЛОГИЯ**  
университет интернет-профессий

**Задавайте вопросы и напишите отзыв о лекции!**

**МИХАИЛ КУЗНЕЦОВ**



[@mkuznetcov](https://www.instagram.com/mkuznetcov)