**Что такое форма?**

Форма — это компонент страницы для получения информации от пользователя. Область заказа в интернет-магазине, окно для написания твита, строка поиска на сайте — всё это формы. Чтобы пользователь мог вводить и отправлять данные, в форме размещают специальные инструменты: кнопки, переключатели, флажки, поля с выпадающими списками и поля ввода текста.

В HTML-документе форма задаётся тегами form:

Скопировать кодHTML

<form></form>

**Разметка формы**

Когда форма объявлена, её наполняют полями. Поля служат для сбора информации: имени пользователя, адреса доставки, согласия на обработку персональных данных и любой другой. В следующем уроке мы начнём рассматривать эти поля. Но сначала разберёмся, где эти поля размещать.

Можно разместить поля прямо внутри формы. Но это плохая практика, поскольку все поля окажутся свалены «в кучу», и по какой логике они расположены, останется непонятным.

Чтобы по форме было понятно, что и куда вводить, поля для сбора информации объединяют в группы — филдсеты (англ. fieldset). Роль организатора выполняет тег fieldset — набор полей:

Скопировать кодHTML

<form class="form">

<fieldset class="form\_\_contact-info">

*<!-- Элементы формы для получения данных о пользователе -->*

</fieldset>

<fieldset class="form\_\_delivery-info">

*<!-- А тут — для информации о времени и адресе доставки -->*

</fieldset>

</form>

Первый блок fieldset отведён под сбор контактной информации, а второй — под информацию о доставке.

По умолчанию fieldset визуально сопровождается серой рамкой. Обычно она не нужна, поэтому её сбрасывают файлом normalize.css или CSS-свойством в таблице стилей.

**Отправка формы**

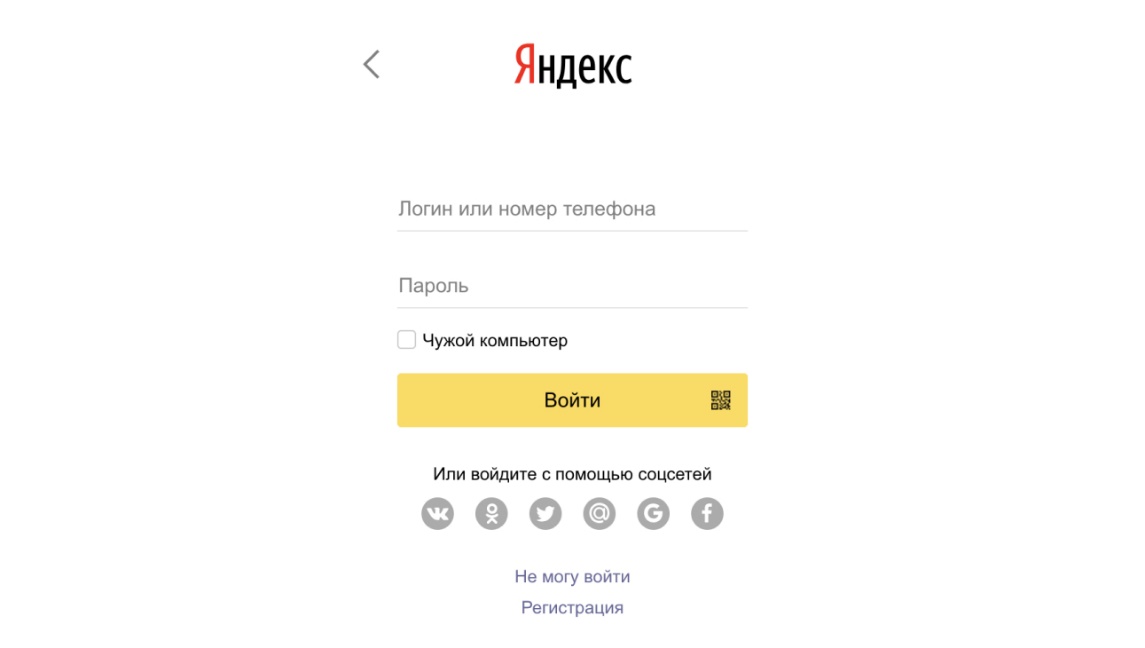
Формы отличаются от элементов, которые вам уже доводилось верстать: секций, параграфов, изображений, шапки и подвала. Форму недостаточно сверстать. Чтобы форма заработала, тег form сопровождают атрибутами. Атрибуты нужны, чтобы указать серверу, чтó ему делать с формой. Мы расскажем, как задавать атрибуты формы, но немного позже. А пока потренируйтесь размечать форму филдсетами.

**Поля ввода с тегом input**

Для получения информации от пользователя внутрь формы помещают поля ввода. Большинство из них задаётся тегом input. Причём неважно, что мы ожидаем получить — номер телефона, имя или даже файл — почти всегда подойдёт тег input. Об исключениях поговорим в пятом уроке.

Рассмотрим для примера форму авторизации электронной почты. В ней четыре поля ввода: логин, пароль, флажок «Чужой компьютер» и кнопка «Войти».

Это всё поля ввода, но выглядят и ведут они себя по-разному. Как именно — определяет атрибут type.



В нашем примере:

| **ПОЛЕ** | **ЗНАЧЕНИЕ TYPE** |
| --- | --- |
| логин или номер телефона | text (текст) |
| пароль | password (пароль) |
| флажок «Чужой компьютер» | checkbox (флажок, то есть поле для галочки) |
| кнопка «Войти» | submit (отправить, передать) |

Самые распространённые значения type:

**Код Пример**

<input type="text">

<!-- для текста -->



<input type="number">

<!-- для чисел.На мобильном

устройстве вместо стандартной

будет вызвана клавиатура с

цифрами, как на кнопочном

телефоне -->

<input type="email">

<!-- для электронной почты.

Браузер проверяет, есть ли в этом

поле точка и @ («коммерческое “эт”»)

— обязательные символы для

доменного имени. -->

<input type="password">

<!-- для паролей. Вместо символов

отображаются звездочки

или точки -->



<input type="tel">

<!-- для номеров телефонов.

Отобразится цифровая клавиатура,

как и для number -->

<input type="url">

<!-- для адреса сайта -->

<input type="range">

<!-- создает "ползунок",

который можно двигать от

максимального до минимального

значения.-->

<!-- Поля ввода для указания времени и даты: -->

<input type="date">

<!-- дата без времени:

07/03/2009 -->

<input type="datetime-local">

<!-- дата со временем:

04/10/1957 22:28 -->

<input type="month">

<!-- месяц и год: 1988-11 -->

<input type="week">

<!-- неделя и год: 2018-W45 -->

<input type="time">

<!-- время: 03:14 -->

Управляя атрибутом type, вы сконструируете форму под разные задачи.

Иногда поле нужно заблокировать. Например, пока форма отправляет данные на сервер, стоит запретить возможность редактировать поля. Или при выборе одного из пунктов формы, может стать недоступным выбор другого. Выбрав вечернюю доставку, все утренние временные интервалы следует заблокировать для выбора. Для таких случаев у полей есть атрибут disabled. Его прописывают внутри открывающего тега, не указывая значения:

Скопировать кодHTML

*<!-- Если пользователь предпочитает связь по электронной почте, а не по телефону -->*

<input type="tel" disabled>

Атрибут disabled сделает поле ввода серым: в него нельзя будет установить курсор или выбрать его клавишей Tab.

**Минимальные и максимальные значения**

В предыдущем уроке мы рассказывали об атрибуте type. Но у полей ввода есть и другие атрибуты.

Помните, мы говорили о ползунках? Это специальные регуляторы, которыми можно выставить значение из диапазона: скажем, уровень громкости.

Границы диапазона определяются атрибутами min и max. По умолчанию их значения 0 и 100. Если вам нужны другие границы, определите их вручную:

Скопировать кодHTML

<input type="range" min="-20" max="500">

Эти атрибуты работают и с числовыми полями:

Скопировать кодHTML

<input type="number" min="1970" max="2025">

Значением ползунков и числовых полей можно управлять клавишами ↑ и ↓. По умолчанию оно будет меняться на 1. Но и это можно переопределить атрибутом step:

Скопировать кодHTML

*<!-- Значение меняется с шагом 10 -->*

<input type="range" min="20" max="50" step="10">

Настраивайте атрибуты min, max и step для полей ввода, чтобы пользователь не мог ввести значений, которые не могут быть обработаны. Правильные ограничения улучшают интерфейс.

# Поля загрузки, сброса и отправки данных

Тег input часто используют для ввода строк и чисел. Но он также может загружать файлы, сбрасывать форму и отправлять её. Это определяет значение атрибута type:

Скопировать кодHTML

<input type="file"> *<!-- Поле для загрузки файла -->*

<input type="reset"> *<!-- Кнопка «Сбросить». При нажатии устанавливает все поля формы на значения по умолчанию -->*

<input type="submit"> *<!-- Кнопка ”Submit“ (переводится как отправить на рассмотрение). Отправляет данные формы на сервер -->*

<input type="image"> *<!-- То же, что и submit, но вместо кнопки будет картинка -->*

Для отправки и сброса формы мы используем тег input с атрибутами submit и reset. Но сейчас всё чаще для этого используют тег button:

Скопировать кодHTML

<button type="submit">Отправить</button> *<!-- Отправит форму -->*

<button type="reset">Сбросить</button> *<!-- Сбросит форму -->*

Во многих случаях теги button удобнее. Но сейчас это неважно: с тегом input всё тоже будет работать.

# Поля ввода с другим синтаксисом

Не все поля ввода задаются тегом input, есть исключения. Рассмотрим их.

Тег <input type="text"> подходит для ввода одной строки. Если текст может быть многострочным, лучше применять тег textarea (переводится как текстовая область):

Скопировать кодHTML

<textarea></textarea>

Второе исключение из правил — тег select (англ. выбирать). Он создаёт поле с выпадающим списком, в котором варианты заключены между тегами <option></option> (переводится как вариант):

Скопировать кодHTML

<select>

<option>Пункт 1</option>

<option>Пункт 2</option>

</select>

# Ярлыки

В этом уроке мы сообщим браузеру и пользователю, что именно вводить в каждое поле. Для этого используют тег label (переводится как ярлык).

Скопировать кодHTML

<label>Имя</label>

<input type="text">

<label>Фамилия</label>

<input type="text">

Указывая, к какому полю относится ярлык, связывают сразу два атрибута.

Первый — атрибут id (идентификатор), он задаётся полю ввода. Так элемент получает название, на которое теперь можно ссылаться. За это отвечает атрибут for тега label (от английского ”for“ в значении «для»). for указывает браузеру, к какому полю относится ярлык.

Скопировать кодHTML

<label for="first-name">Имя</label>

<input type="text" id="first-name">

<label for="last-name">Фамилия</label>

<input type="text" id="last-name">

Это один из тех редких случаев, когда в современном HTML нужен атрибут id. Если связать ярлык и поле ввода описанным образом, при клике на ярлык поле станет активным.

Не делайте значениями атрибута id цифры и спецсимволы — это может повлечь ошибки. Идентификаторы при работе используют редко, случаи когда они необходимы, можно перечислить по пальцам. Имена идентификаторам лучше задавать, разделяя слова дефисом. Это создаёт единообразие HTML-кода.

Часто используют ещё один подход к вёрстке ярлыков — поле ввода делают дочерним элементом ярлыка:

Скопировать кодHTML

<label>Имя

<input type="text">

</label>

В этом случае связывать ярлык с полем ввода необязательно, ведь они уже находятся в родстве. Клик по ярлыку будет активировать поле ввода. Такой подход встречается очень часто, так как решает типичную задачу — расположить ярлык рядом с полем ввода. Но так вы теряете возможность расположить лейбл в другом месте, если это потребуется, поэтому в заданиях этого урока мы будем связывать ярлыки с полями ввода через атрибуты.

Важно помнить, что атрибут for для ярлыка — это сигнал, что где-то есть идентификатор. Если пишете атрибут for, идентификатор нужно присвоить, иначе не заработает связка ярлыка с полем ввода. Даже при втором подходе.

# Передаваемые значения

После того как пользователь ввёл свои данные в форму, её необходимо передать на сервер. Для этого браузер собирает все данные формы в одну строчку-запрос:

Скопировать код

адрес\_сервера/form.html?firstname=Аркадий&authorized=yes&search=yandex

Знак вопроса отделяет адрес сервера от запроса. Тело запроса разделено знаками амперсанд (&) на конструкции вида:

Скопировать код

firstname=Аркадий

Слева от знака равенства — имя поля, его нужно установить атрибутом name для каждого поля ввода, иначе данные на сервер не уйдут.

Скопировать кодHTML

<input type="text" name="firstname">

Значения передаёт специальный атрибут value. В него записывается то, что пользователь ввёл в поле. Некоторым полям ввода, например пунктам выпадающих списков и кнопкам отправки и сброса данных в форме, value задают явно:

Скопировать кодHTML

<input type="text" name="firstname">

<select name="search">

<option value="yandex">Yandex</option>

<option value="some-search">Другой поисковик</option>

</select>

<input type="submit" value="Отправить на сервер">

Атрибут value можно задать и обычным текстовым полям. Эти значения и будут отправлены на сервер, если пользователь их не изменит.

Скопировать кодHTML

<input type="text" name="firstname" value="Невведённое имя">

<select name="search">

<option value="yandex">Yandex</option>

<option value="some-search">Другой поисковик</option>

</select>

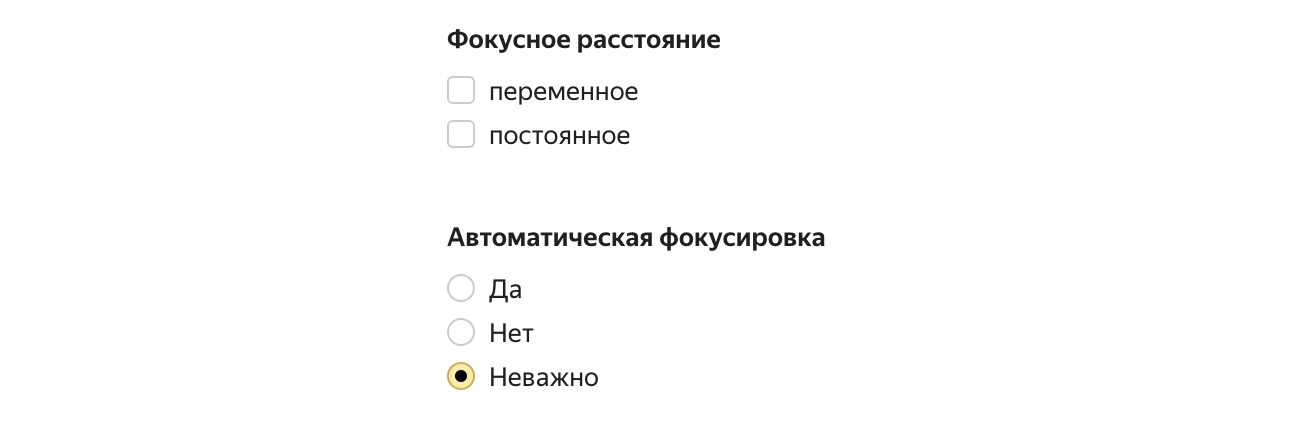
<input type="submit" value="Отправить форму сейчас">

Мы задали значения для полей, и это повлияло на их внешний вид:

# Поля множественного и единичного выбора

Поле выбора — это элемент формы со списком, где можно отметить один или несколько вариантов.

Такие поля бывают двух типов: с возможностью единичного и множественного выбора. На скриншоте «Фокусное расстояние» — поле с возможностью множественного выбора, а «Автоматическая фокусировка» — единичного.



Оба типа поля создаются тегом input, а поведение определяет атрибут type.

Скопировать кодHTML

<input type="radio"> *<!-- поле единичного выбора (радиокнопка) -->*

<input type="checkbox"> *<!-- поле множественного выбора (флажок) -->*

Начнём с флажков, или чекбоксов. Чтобы создать несколько чекбоксов, достаточно расположить их в коде один за другим.

Создадим список, где можно выбрать несколько музыкальных жанров:

Скопировать кодHTML

<h2>Выберите жанры:</h2>

<label for="rock">Рок</label>

<input type="checkbox" name="genre" id="rock" value="rock">

<label for="pop">Поп</label>

<input type="checkbox" name="genre" id="pop" value="pop">

<label for="hiphop">Хип-хоп</label>

<input type="checkbox" name="genre" id="hiphop" value="hiphop">

<label for="techno">Техно</label>

<input type="checkbox" name="genre" id="techno" value="techno">

Ярлыки связаны с флажками через идентификатор, а атрибут name у всех флажков одинаковый. Так сделано, чтобы клик по тексту ярлыка активировал флажок, а при отправке формы выбранное значение передавалось ключом genre на сервер.

В атрибуте value заданы значения, которые будут отправлены на сервер. Атрибут value обязательный для флажков, ведь в чекбокс пользователь не может ничего ввести сам, он может лишь поставить флажок или его убрать.

С радиокнопками похожая ситуация, они должны быть друг с другом связаны.

Скопировать кодHTML

<input type="radio">

<input type="radio">

Такой код создаст две кнопки, активируемые по отдельности.

Чтобы объединить радиокнопки, укажите им одинаковое значение атрибута name. Тогда выбрать больше одного варианта не получится:

Скопировать кодHTML

<label for="yes">Да</label>

<input type="radio" name="choice" id="yes" value="yes">

<label for="no">Нет</label>

<input type="radio" name="choice" id="no" value="no">

Идентификатор привязывает ярлык, а атрибут value указывает, какое значение отдавать на сервер при отправке формы.

По умолчанию радиокнопки и флажки не активны после загрузки страницы. За это отвечает атрибут checked (отмеченный). Ему не нужно указывать какое-то значение — достаточно просто написать checked внутри тега input, чтобы элемент стал активным:

Скопировать кодHTML

<input type="checkbox" name="music" checked>

<input type="radio" name="choice" checked>

**Подсказки в текстовых полях**

В предыдущих уроках мы объясняли браузеру, что нужно делать с каждым полем ввода. Теперь объясним это пользователю. Добавим к текстовым полям подсказки.

Их можно создать атрибутом value, но это неудобно по двум причинам:

* чтобы ввести своё значение, пользователю придётся удалять подсказку;
* если не заполнить поле, на сервер уйдёт значение по умолчанию.

Избежать этого позволяет атрибут placeholder (переводится как заполнитель, в значении «шаблон»).

Скопировать кодHTML

<input type="text" name="firstname" placeholder="Иванов Иван">

# Обязательные поля

Поля формы можно помечать как обязательные. Без заполнения таких полей форму не получится отправить — браузер выдаст сообщение об ошибке.

Отвечает за это атрибут required (обязательный). Он не требует значения:

Скопировать кодHTML

<input type="text" required>

**Особенности стилизации текстовых полей ввода, кнопок и ярлыков**

У текстовых полей ввода только одна функция — получить текст. В них не выбирают варианты и это не переключатели. Вёрстка таких полей довольно интуитивна, можно пользоваться уже знакомыми CSS-стилями. Вы знаете всё, что нужно для такой вёрстки. Тем не менее, есть ряд особенностей.

1. Все поля ввода и лейблы по умолчанию строчные элементы. Поэтому вам часто будет нужно свойство display, чтобы управлять их расположением.
2. Поля ввода не наследуют некоторые стили от своих родителей. Например, стили шрифтов. Какой бы шрифт ни был задан форме, в полях ввода пользователь будет всё набирать стандартным для браузера шрифтом. Поэтому всем полям ввода нужно задавать наследование явно. Для этого существует специальное значение inherit (унаследовать). Оно означает приказ «унаследовать от родителя»:

Скопировать кодCSS

input {

font-family: inherit;

*/\* Теперь поле ввода наследует шрифт от родителя \*/*

}

1. Когда пользователь выбрал поле, нужна возможность управлять его поведением. Например, задать активному полю цвет рамки. Для этого есть специальный псевдокласс :focus:

Скопировать кодCSS

input:focus {

*/\* стили для активного поля \*/*

}

По умолчанию в состоянии :focus элементы ввода окружены синей внешней линией, или обводкой:

image

Обводка полезна, когда между элементами формы пользователь перемещается не мышью, а нажатием клавиши Tab. Благодаря обводке люди с пониженным зрением лучше видят элемент в фокусе. Настраивается эта полезная опция свойствами группы outline:

Скопировать кодCSS

input:focus {

outline-color: yellow; */\* цвет обводки — жёлтый \*/*

outline-style: dashed;*/\* стиль линии — прерывистая \*/*

outline-width: 3px; */\* толщина обводки — 3px \*/*

}

Иногда outline сбрасывают, а состояние фокуса стилизуют другими средствами, например, границей border:

Скопировать кодCSS

input:focus {

outline: none;

border: 1px solid red;

}

# Селекторы атрибутов

В CSS есть селекторы, способные выбирать элементы с любым значением любого атрибута. Вы можете выбрать ссылки, ведущие на определённый адрес, изображения с определённым атрибутом alt и тому подобное. Выглядит это в коде так:

Скопировать кодCSS

любой-селектор[любое-имя-атрибута="любое-значение-атрибута"] {

...

}

Этот тип селекторов может быть полезен при работе с формами, ведь большинство полей ввода отличаются только атрибутом type и работают по-разному в зависимости от него. Например, задать стили для однотипных полей можно, записав после слова input селектор атрибута — пару атрибут="значение":

Скопировать кодCSS

input[type="submit"] {

*/\* Стили для кнопки submit \*/*

}

С такими селекторами нужно обращаться аккуратно и применять их только там, где они действительно нужны. Если в будущем решите заменить input на button, селектор тоже придётся менять. А если на странице появится поле ввода с аналогичным типом, он будет оформлен так же, как первый. Вот где селекторы классов значительно удобнее.

Методология БЭМ постулирует отказ от селекторов такого рода в пользу модификаторов. Несмотря на это, мы поработаем с подобными селекторами, ведь на собеседованиях на позицию верстальщика часто проверяют именно знания нестандартных селекторов.

# Стилизация переключателей и полей выбора

Необычные поля ввода — такие как checkbox, radio, range, select — словом, все те, куда нельзя ввести текст, стилизуются браузером. Через CSS их оформить достаточно сложно, а порой и невозможно. Вот где используют такой приём: поле визуально скрывают, а поверх создают аналог с собственным дизайном.

Мы не будем пошагово объяснять, как верстать все виды полей ввода. Это слишком долго. Объясним принцип на примере чекбокса и дадим ссылки, которые помогут вам разобраться с другими полями — ведь у каждого из них свои особенности.

Первое, что нужно — это скрыть текущее поле ввода. Но сделать это так, чтобы оно осталось на странице, ведь мы хотим, чтобы оно активировалось с клавиатуры. Для этого нужно применить много свойств, которые вам пока незнакомы.

## Делаем поле ввода дочерним для label

Мы будем скрывать основной input и менять на новый. Принципиально важно, чтобы label окружал эти элементы. Мы будем активировать новый элемент по клику на лейбл:

Скопировать кодHTML

<label>

<input type="checkbox">

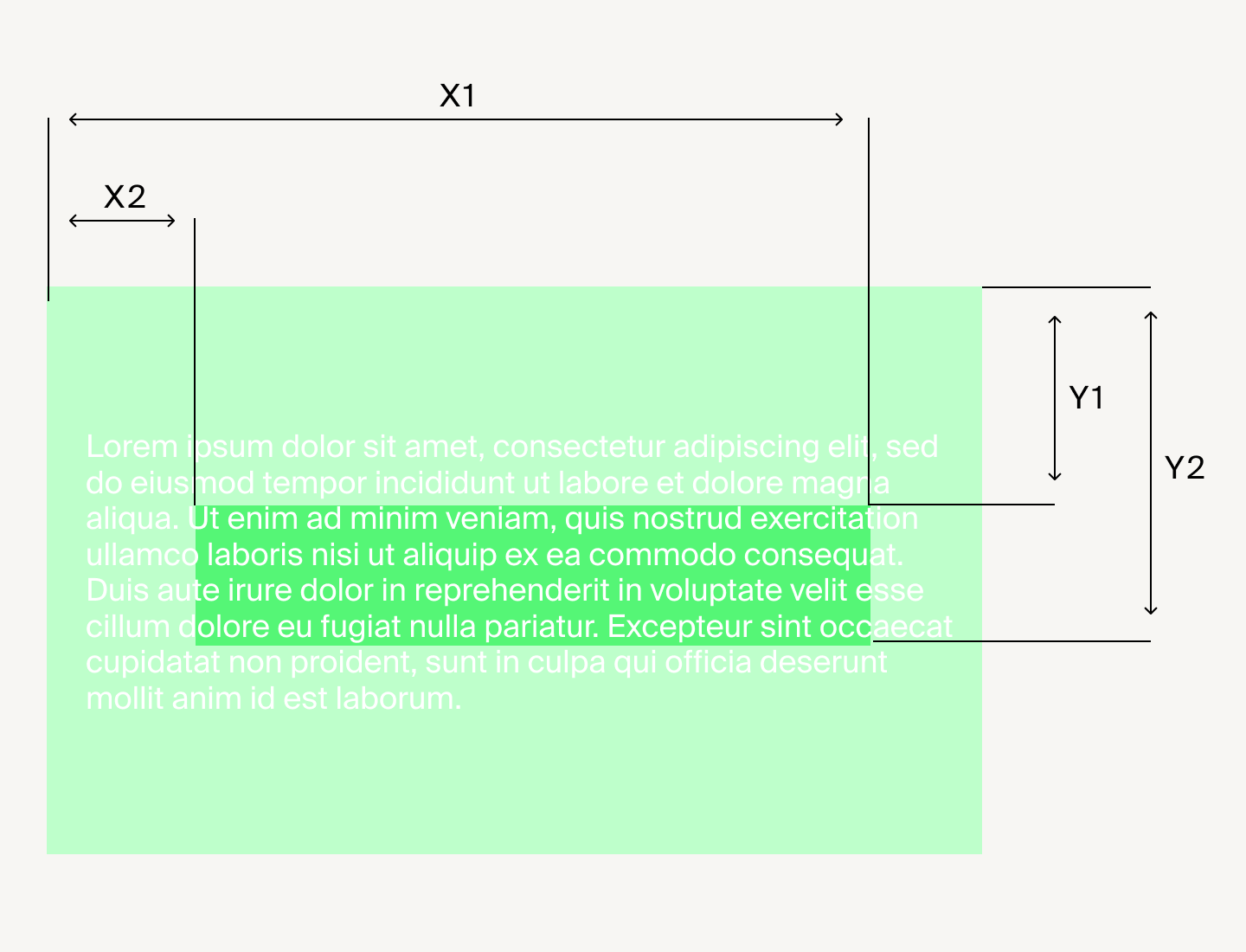
</label>

## Скрываем элемент через clip

Свойство clip работает с абсолютно спозиционированными элементами и отвечает за область, внутри которой будет показано содержимое этого элемента.

Скопировать кодCSS

clip(X1, Y1, X2, Y2);



Если все координаты нулевые, элемент не будет показан. Такой код скроет чекбокс, и это будет кроссбраузерно:

Скопировать кодCSS

.input[type="checkbox"] {

position: absolute;

width: 1px; */\* если у элемента не будет размеров, некоторые браузеры посчитают, что его вообще нет \*/*

height: 1px;

overflow: hidden; */\* без этого свойства тоже сработает, но так наверняка. Мы его ещё изучим, оно скрывает элементы, выходящие за границы родителя \*/*

clip: rect(0 0 0 0);

}

Когда чекбокс скрыт, нужно создать второй элемент рядом с ним, он будет заменять скрытый новым дизайном. Примерно так:

Скопировать кодHTML

<label>

<input type="checkbox" class="invisible-checkbox"> *<!-- этот элемент мы скрыли -->*

<span class="visible-checkbox"></span> *<!-- этот будем стилизовать -->*

</label>

Стилизация альтернативного элемента — дело техники. Немного позиционирования, отступов, граница — этого хватит. Важно, чтобы новый элемент расположился там, где вам нужно, и был похож на чекбокс. В задании мы покажем, как это сделать.

## Описать все состояния чекбокса через псевдоклассы

Состояния чекбокса:

* включённый и выключенный — определяется псевдоклассом disabled;
* выбранный и не выбранный — определяется псевдоклассом checked;
* в фокусе и не в фокусе — определяется псевдоклассом focus.

Получаются такие селекторы:

Скопировать кодCSS

input[type="checkbox"]:disabled {} */\* выключенный \*/*

input[type="checkbox"]:checked {} */\* выбранный \*/*

input[type="checkbox"]:focus {} */\* в фокусе \*/*

Есть проблема. Наш чекбокс скрыт, поэтому на состояние должен реагировать альтернативный элемент. Выбрать его и стилизовать — целая задача. Нам потребуется очередной необычный селектор — селектор первого соседа, следующего за нашим скрытым чекбоксом. Этот сосед и станет альтернативным элементом. Селектор первого соседа определяется символом +:

Скопировать кодCSS

input[type="checkbox"] + span {} */\* так для чекбокса выглядит первый сосед с именем span \*/*

Этих знаний достаточно, чтобы сверстать чекбоксы и радиокнопки. Как всё это совместить — увидим, решая задачи.

Другие элементы форм содержат другие псевдоклассы и псевдоэлементы для управления стилем. Иногда, чтобы настроить всё так, как вам нужно, не обойтись без JavaScript. Хорошая новость в том, что есть много библиотек, которые работают со стилями нестандартных полей ввода. Но принцип знать нужно: оборачивать лейблом, прятать, создавать альтернативный элемент, стилизовать альтернативный элемент в разных состояниях.

# Стилизация плейсхолдеров

Стили для плейсхолдера прописывают селектором с псевдоэлементом ::placeholder:

Скопировать кодCSS

input::placeholder {

font-family: Arial;

}

Этот код задаёт семейство шрифтов для всех плейсхолдеров. Но в браузерах компании Microsoft (Internet Explorer и Edge) псевдоэлемент ::placeholder не поддерживается, и код не сработает. Чтобы стилизовать плейсхолдеры в IE, псевдоэлемент предваряется вендорным префиксом -ms-input — ::-ms-input-placeholder:

Скопировать кодCSS

input::-ms-input-placeholder {

font-family: Arial;

}

Ранее сам стандарт CSS не позволял стилизовать плейсхолдеры. Поэтому каждый производитель браузера придумал свои вендорные префиксы, чтобы поддержать экспериментальные решения:

Скопировать кодCSS

input::-webkit-input-placeholder {color:#c0392b;} */\* для Safari и Chrome \*/*

input::-moz-placeholder {color:#c0392b;} */\* для Firefox \*/*

input::-ms-input-placeholder {color:#c0392b;} */\* для IE \*/*

Вот такой код обеспечит железобетонную надёжность работы во всех браузерах:

Скопировать кодCSS

input::-webkit-input-placeholder {color:#c0392b;}

input::-moz-placeholder {color:#c0392b;}

input::-ms-input-placeholder {color:#c0392b;}

input::placeholder {color:#c0392b;}

Выбирайте вендорные префиксы для основных браузеров:

-moz-свойство — префикс для Firefox;

-webkit-свойство — префикс для Safari и Chrome;

-ms-свойство — префикс для Internet Explorer и Edge;

-o-свойство — для поддержки браузерами Opera.

# В работу!

Сделаем так, чтобы форма заработала.

Средств HTML и CSS здесь недостаточно.

Понадобится ещё обработчик — программа, которая обработает данные формы на сервере. Такая программа написана на одном из серверных языков программирования (PHP, Java, Python).

У тега form есть два атрибута: action (переводится как действие) и method (переводится как метод).

Значение action — это ссылка на обработчик. Этим атрибутом мы сообщаем, какой программе отдавать данные.

Атрибут method принимает значения GET или POST. Метод GET (получить) нужен, когда информация уже есть на сервере и вам нужно получить её себе в браузер. Метод POST (отправить) — когда данные есть у вас в браузере и их надо обновить на сервере.

Скопировать кодHTML

<form action="ссылка на обработчик" method="POST">

В нашем случае отправлять форму на сервер необязательно — можно обработать её прямо в браузере. Для этого достаточно JavaScript.

Мы обязательно расскажем, как обрабатывать формы на стороне сервера. А сейчас подключим заранее подготовленный скрипт, чтобы форма заработала.

# Заключение

Поздравляем, вы изучили очень большую и важную тему!

Вы разобрались, как работают формы, и поняли, насколько важно реализовывать их правильно. Вы научились стилизовать элементы форм и знаете, как подружить их со всеми браузерами.

В этой теме мы лишь поверхностно затронули стилизацию форм, в списке дополнительных источников вы найдёте больше интересных решений:

[Документация элемента input](https://www.w3.org/TR/html52/sec-forms.html#the-input-element) (на английском языке).

[Стили HTML-форм](https://developer.mozilla.org/ru/docs/Learn/HTML/Forms/Styling_HTML_forms) (на английском языке).

[84 примера стилизации чекбоксов](https://freefrontend.com/css-checkboxes/) (на английском языке, код).

[Руководство по элементам HTML-формы](https://proglib.io/p/study_html_forms).

[Как стилизовать input[type=range]](https://css-tricks.com/styling-cross-browser-compatible-range-inputs-css/) (на английском языке).

[Стилизация выпадающего списка](https://24ways.org/2019/making-a-better-custom-select-element/) — тут не обойтись без JS, поэтому сохраните в закладках на будущее. На английском языке.

Основная часть спринта позади. Впереди темы, которые помогут вам доработать личный проект и подготовиться ко второму ревью. Вы узнаете, как работать с гитом на продвинутом уровне, и научитесь удобно организовывать файлы проектов.