**Лабораторная работа № 8**

**«Современные формы с помощью CSS3 и валидации HTML5»**

**Ход выполнения работы:**

1. ***Формирование представления о функциональности***

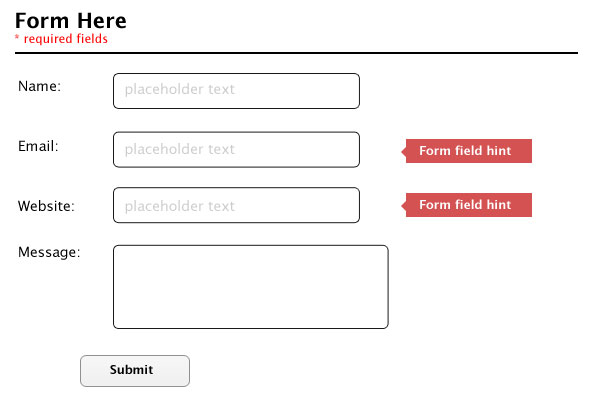
Составим концепцию того, как будет выглядеть наша форма, и как она будет функционировать. Для примера давайте создадим простую контактную форму, запрашивающую у пользователя следующую информацию:

* Имя
* Адрес электронной почты
* Вебсайт
* Сообщение

Нам нужно убедиться, что пользователь вводит информацию правильно. Чтобы этого добиться, применим новые техники валидации на стороне клиента HTML5. А как насчет пользователей, у которых нет возможности использовать HTML5? Можно просто применить валидацию со стороны сервера.

***2. Разработка формы***

Определим, как следует выглядеть нашей форме, создав грубую модель.



Нашу форму составляют следующие элементы:

* Заголовок

*Обозначение обязательных для заполнения полей*

* Названия полей ввода

* Поля ввода данных

*Текст-заполнитель*

* Подсказки к полям

* Кнопка «Отправить» (Submit)

Теперь, когда вы известно, какие элементы составляют форму, можно создадим разметку HTML.

***3. Первичный код HTML***

01.<!DOCTYPE html>

02.<html>

03.<head>

04.<meta charset="utf-8">

05.<title>HTML5 Contact Form</title>

06.<link rel="stylesheet" media="screen" href="styles.css" >

07.</head>

08.<body>

09.</body>

10.</html>

Вплоть до этого момента HTML-файл в браузере пока пуст. Это – просто начальный код для страницы HTML5.

***4. Форма HTML***

Давайте создадим форму HTML (оставим метод action пока пустым, так как проверка данных на стороне сервера в этой лабораторной работе не раскрывается):

1.<form class="contact\_form" action="" method="post" name="contact\_form">

2.</form>

***5. Элементы формы HTML***

Для получения организованного и структурированного контента своей формы, обернем ее элементы (label, input и т.д.) в список. Создадим заголовок формы и первого элемента input:

01.<ul>

02.<li>

03.<h2>Contact Us</h2>

04.<span class="required\_notification">\* Denotes Required Field</span>

05.</li>

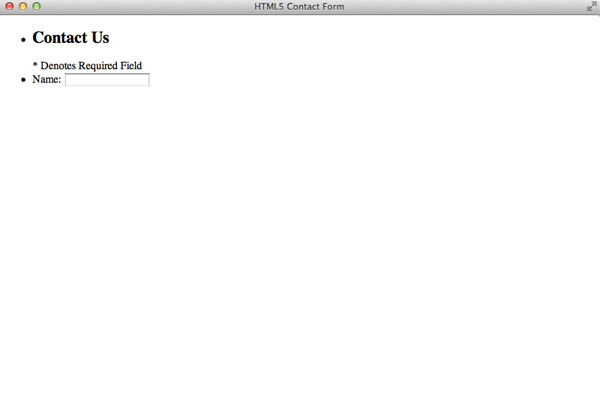
06.<li>

07.<label for="name">Name:</label>

08.<input type="text" name="name" />

09.</li>

10.</ul>



***Подсказки для полей формы***

Как видно из макета, необходимо сделать форматированные подсказки для полей ввода электронного адреса “email” и вебсайта “website”. Поэтому добавим свои подсказки под поля ввода, где это нужно, и назначим им класс, чтобы можно было позже определить им стили.

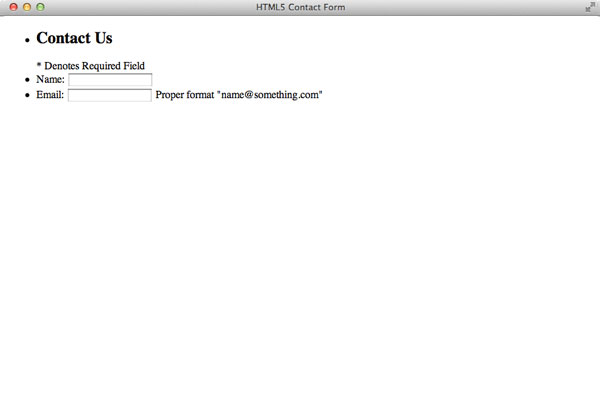
1.<li>

2.<label for="email">Email:</label>

3.<input type="text" name="email" />

4.<span class="form\_hint">Proper format "name@something.com"</span>

5.</li>



***Остальные элементы input***

Создадим остальные элементы формы, не забывая о том, что нужно обернуть каждый раздел в элемент списка.

01. <li>

02. <label for="website">Website:</label>

03. <input type="text" name="website" />

04. <span class="form\_hint">Proper format "http://someaddress.com"</span>

05. </li>

06. <li>

07. <label for="message">Message:</label>

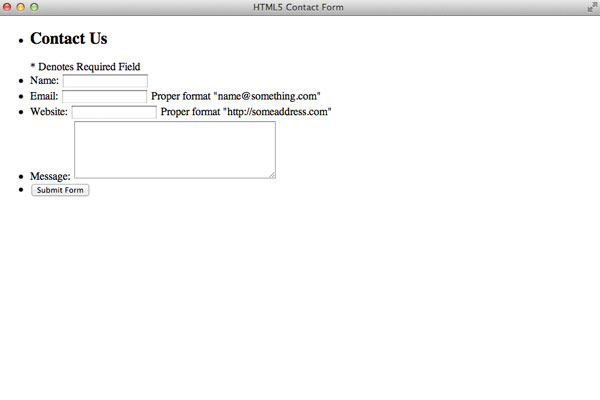
08. <textarea name="message" cols="40" rows="6" >

09. </li>

10. <li>

11. <button class="submit" type="submit">Submit Form</button>

12. </li>



***6. Добавляем атрибут placeholder***

Одно из первых усовершенствований, которые предлагает HTML5 для веб-форм (с которым вы, возможно, уже знакомы) – это способность установить текст-подсказку. Он показывается, когда поле ввода либо пустое, либо находится не в фокусе.

Добавим атрибут placeholder для всех текстовых тегов input. Это поможет пользователю понять, что нужно ввести в каждое поле.

1. <input type="text" name="name" placeholder="John Doe" />

2. <input type="text" name="email" placeholder="john\_doe@example.com" />

3. <input type="text" name="website" placeholder="http://johndoe.com/" required/>

***Подсказка: Назначьте placeholder’у стили***

Вот вам подсказка: если нужно определить стили тексту-подсказке, к вашим услугам имеются префиксы браузеров:

1. :-moz-placeholder {

2. color: blue;

3. }

4. ::-webkit-input-placeholder {

5. color: blue;

6. }

В современных браузерах поддержка атрибута placeholder налажена довольно хорошо (кроме IE9, к сожалению). Если вам реально требуется поддерживать его во всех браузерах, можно посмотреть решение проблемы в javascript.

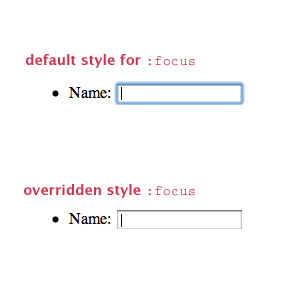
***7. Основной CSS***

Чтобы создать для формы некую структуру, добавим немного основного CSS.

***Удалите стиль : focus***

Webkit автоматически добавляет к input-ам стили, когда те находятся в фокусе. Так как мы будем добавлять собственные стили, то стили по умолчанию нужно отменить:

1.\*:focus {outline: none;}



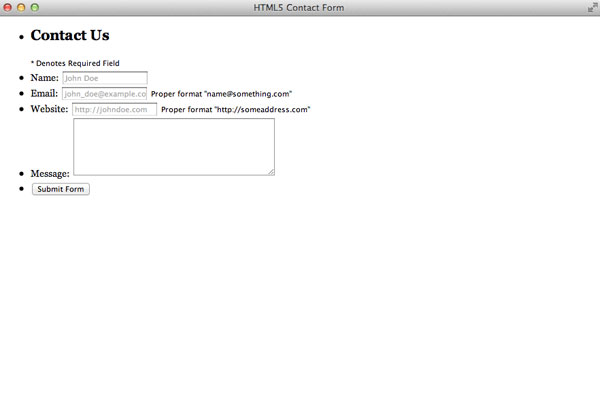
***Типографские стили***

Давайте определим элементам своей формы типографские стили:

1. body {font: 14px/21px "Lucida Sans", "Lucida Grande", "Lucida Sans Unicode", sans-serif;}

2. .contact\_form h2, .contact\_form label {font-family:Georgia, Times, "Times New Roman", serif;}

3. .form\_hint, .required\_notification {font-size: 11px;}



***Стили списка***

Давайте назначим стили элементам списка, чтобы придать форме структуру:

1. .contact\_form ul {

02. width:750px;

03. list-style-type:none;

04. list-style-position:outside;

05. margin:0px;

06. padding:0px;

07. }

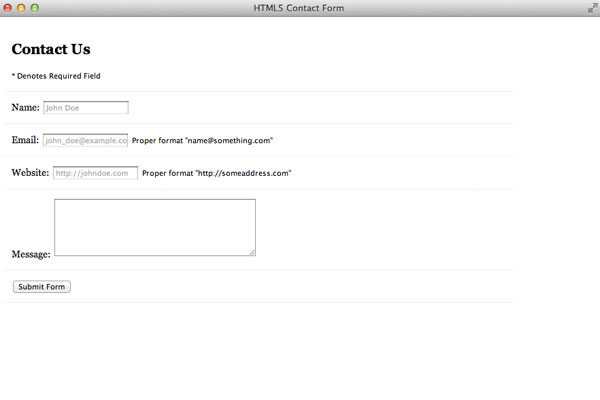
08. .contact\_form li{

09. padding:12px;

10. border-bottom:1px solid #eee;

11. position:relative;

12. }



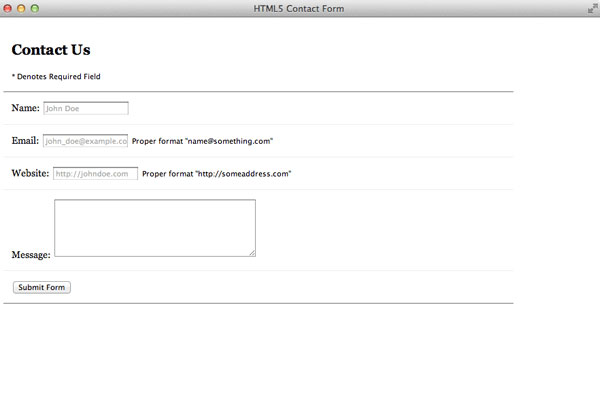
Кроме того, добавим тонкую рамку к верхнему и нижнему разделам формы. Выполнить это можно, применив селекторы :first-child и :last-child. Они выбирают, как подразумевается в их названиях, первый и последний элементы списка ul.

Так создается визуальное разделение формы. Помните, что эти селекторы CSS не поддерживаются старыми браузерами. Так как для основной функциональности это не жизненно важно, то в выгодном положении окажутся те, кто пользуется современными браузерами.

1. .contact\_form li:first-child, .contact\_form li:last-child {

2. border-bottom:1px solid #777;

3. }



***Заголовок формы***

Назначим стили разделу заголовка своей формы. Он включает тэг заголовка и уведомление, которое информирует пользователей о том, что звездочка (\*) обозначает поля, обязательные для заполнения.

01. .contact\_form h2 {

02. margin:0;

03. display: inline;

04. }

05. .required\_notification {

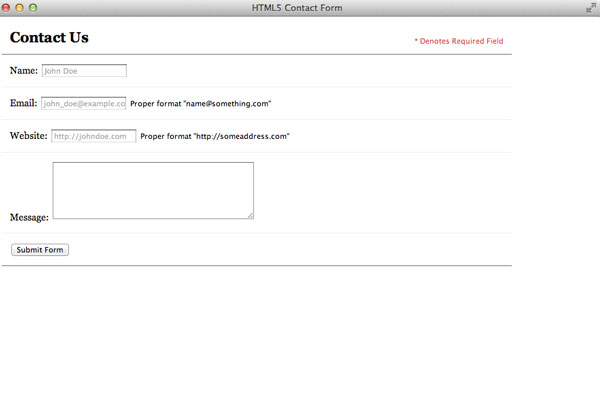
06. color:#d45252;

07. margin:5px 0 0 0;

08. display:inline;

09. float:right;

10. }



***Элементы input***

Назначим стили основным элементам формы, тем, которые предназначены для сбора пользовательской информации.

01..contact\_form label {

02.width:150px;

03.margin-top: 3px;

04.display:inline-block;

05.float:left;

06.padding:3px;

07.}

08..contact\_form input {

09.height:20px;

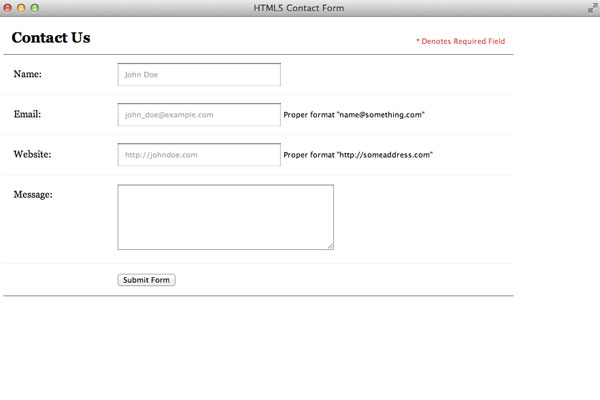
10.width:220px;

11.padding:5px 8px;

12.}

13..contact\_form textarea {padding:8px; width:300px;}

14..contact\_form button {margin-left:156px;}



Теперь добавим несколько дополнительных визуальных стилей CSS. Некоторые из них видны только пользователям современных браузеров.

01..contact\_form input, .contact\_form textarea {

02.border:1px solid #aaa;

03.box-shadow: 0px 0px 3px #ccc, 0 10px 15px #eee inset;

04.border-radius:2px;

05.}

06..contact\_form input:focus, .contact\_form textarea:focus {

07.background: #fff;

08.border:1px solid #555;

09.box-shadow: 0 0 3px #aaa;

10.}

11./\* Button Style \*/

12.button.submit {

13.background-color: #68b12f;

14.background: -webkit-gradient(linear, left top, left bottom, from(#68b12f), to(#50911e));

15.background: -webkit-linear-gradient(top, #68b12f, #50911e);

16.background: -moz-linear-gradient(top, #68b12f, #50911e);

17.background: -ms-linear-gradient(top, #68b12f, #50911e);

18.background: -o-linear-gradient(top, #68b12f, #50911e);

19.background: linear-gradient(top, #68b12f, #50911e);

20.border: 1px solid #509111;

21.border-bottom: 1px solid #5b992b;

22.border-radius: 3px;

23.-webkit-border-radius: 3px;

24.-moz-border-radius: 3px;

25.-ms-border-radius: 3px;

26.-o-border-radius: 3px;

27.box-shadow: inset 0 1px 0 0 #9fd574;

28.-webkit-box-shadow: 0 1px 0 0 #9fd574 inset ;

29.-moz-box-shadow: 0 1px 0 0 #9fd574 inset;

30.-ms-box-shadow: 0 1px 0 0 #9fd574 inset;

31.-o-box-shadow: 0 1px 0 0 #9fd574 inset;

32.color: white;

33.font-weight: bold;

34.padding: 6px 20px;

35.text-align: center;

36.text-shadow: 0 -1px 0 #396715;

37.}

38.button.submit:hover {

39.opacity:.85;

40.cursor: pointer;

41.}

42.button.submit:active {

43.border: 1px solid #20911e;

44.box-shadow: 0 0 10px 5px #356b0b inset;

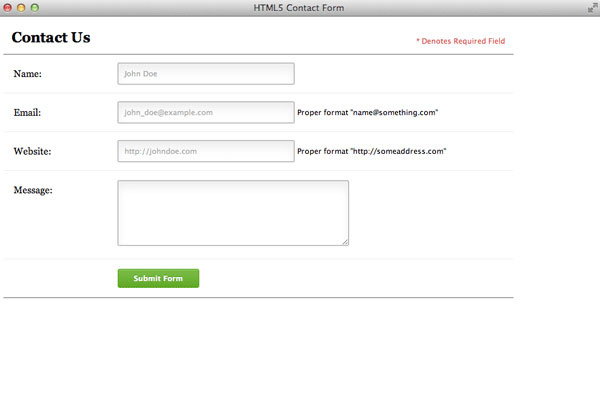
45.-webkit-box-shadow:0 0 10px 5px #356b0b inset ;

46.-moz-box-shadow: 0 0 10px 5px #356b0b inset;

47.-ms-box-shadow: 0 0 10px 5px #356b0b inset;

48.-o-box-shadow: 0 0 10px 5px #356b0b inset;

49.}



***8. Добавляем интерактивность с помощью CSS3***

Давайте добавим немного интерактивности. Заставим выбранное в данный момент поле увеличиваться путем добавления отступа.

1. .contact\_form input:focus, .contact\_form textarea:focus { /\* add this to the already existing style \*/

2. padding-right:70px;

3. }

Теперь с помощью CSS3 для поддерживающих браузеров сделаем увеличение поля гладким переходом.

1..contact\_form input, .contact\_form textarea { /\* add this to the already existing style \*/

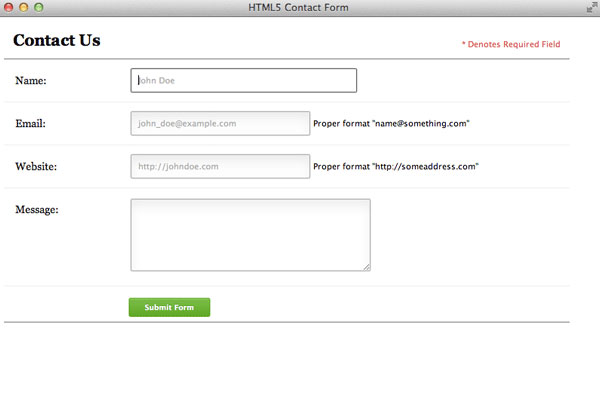
2.-moz-transition: padding .25s;

3.-webkit-transition: padding .25s;

4.-o-transition: padding .25s;

5.transition: padding .25s;

6.}



***9. Атрибут required в HTML5***

Теперь пора заняться инструментами управления формой HTML5.

Добавление атрибута required в любой элемент области ввода текста скажет браузеру, что перед отправкой формы требуется ввести значение. Таким образом, форму нельзя отправить, если нужное поле осталось незаполненным.

Итак, продолжим и добавим атрибут required во все элементы своей формы (потому что нам нужно, чтобы они все были заполнены).

1.<input type="text" name="name" required />

2.<input type="text" name="email" required />

3.<input type="text" name="website" required />

4.<textarea name="message" cols="40" rows="6" required ></textarea>

***10. Стили обязательных для заполнения полей***

Вы, возможно, заметите, что в визуальном плане от добавления атрибута required ничего не изменилось. Мы собираемся назначить стили обязательных для заполнения полей с помощью CSS. В этом примере каждое поле в качестве фонового изображения получит красную звездочку. Чтобы это сделать, нам нужно сначала добавить padding справа input-а, где будет находиться фоновое изображение (так будет предотвращено наложение текста, если запись окажется длинной строкой):

1..contact\_form input, .contact\_form textarea {

2.padding-right:30px;

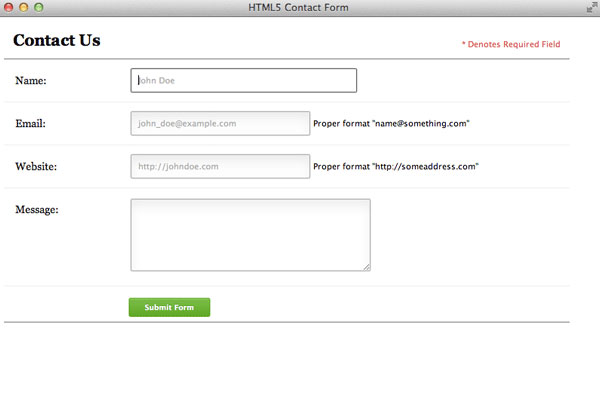
3.}

Теперь воспользуемся псевдоселектором CSS :required для того, чтобы выбрать все элементы формы с нужным атрибутом. В папке ***images*** есть иконка с красной звездочкой размером 16×16 px, которая послужит визуальным индикатором поля, обязательного к заполнению.

1. input:required, textarea:required {

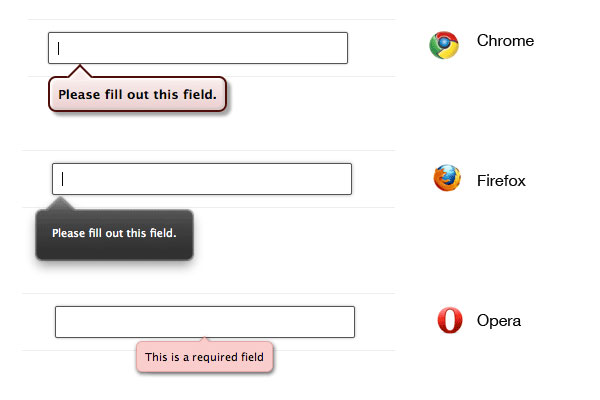
2. background: #fff url(images/red\_asterisk.png) no-repeat 98% center;

3. }



***Что происходит при отправке формы?***

Прямо сейчас при отправке формы с использованием элементов HTML5 различные браузеры действуют по-разному. Большая часть браузеров воспрепятствует отправке и покажет пользователю подсказку, отмечающую первое обязательное к заполнению поле, в которое не введено значение. Визуальные стили и поддержка таких полей довольно широки. К счастью, в будущем такое поведение будет стандартизировано.



***Подсказка:***

На самом деле в webkit назначить стили сообщению в поле-«пузыре» можно с помощью следующего:

1. ::-webkit-validation-bubble-message {

2. padding: 1em;

3. }

***11. Понимание новых атрибутов type и валидации на стороне клиента в HTML5***

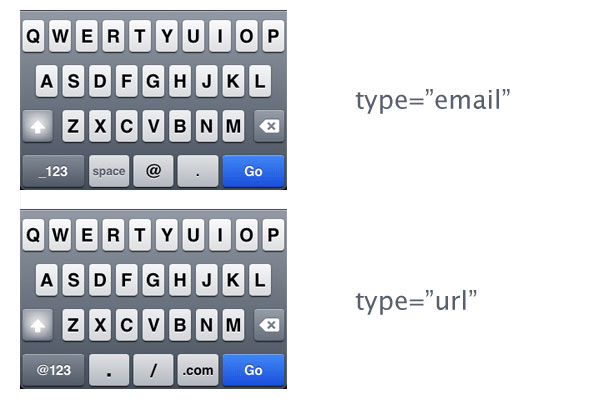
Валидация HTML5 работает в соответствии с атрибутом type, установленном в полях формы. Годами HTML поддерживал лишь несколько атрибутов type, таких, как type= «text», но у HTML5 имеется более дюжины новых типов ввода данных, включая электронную почту и url, которые мы и собираемся использовать в своей форме.

Сочетая атрибуты ввода type с новым атрибутом required, теперь браузер способен делать валидацию данных пользователя на клиентской стороне. Если браузер пользователя не поддерживаем новые атрибуты type, такие как type= «email», он просто по умолчанию вернется к type= «text».

Так что происходит, если браузер на самом деле поддерживает новые атрибуты type? Для браузеров десктопов визуально нет никакой разницы (кроме определенной пользовательскими правилами CSS). Поле type= «text» выглядит точно так, как поле type= «email». Однако для браузеров мобильных устройств в отношении пользовательского интерфейса разница имеется.

***Пример: iPhone***

iPhone от Apple’а распознает типы формы и динамично изменяет экранную клавиатуру, предоставляя требуемые по контексту символы. Например, все электронные адреса требуют следующих символов: “@” и “.” И iPhone предоставляет эти символы, когда у input-a задан соответствующий тип.



***12. Изменение атрибутов type***

Поля нашей формы уже установлены по умолчанию на type= «text». Но теперь требуется изменить атрибут типа в полях, предназначенных для электронной почты и вебсайта на соответствующий тип HTML5.

1. <input type="email" name="email" placeholder="john\_doe@example.com" required />

2. <input type="url" name="website" placeholder="http://johndoe.com" required/>

***13. Валидация HTML5***

Как уже говорилось, валидация HTML5 основана на атрибутах type и включена по умолчанию. Для активации валидации формы никакой особенной разметки не требуется. Если хотите ее выключить, можете употребить атрибут novalidate, как здесь:

1. <form novalidate>

2. <-- do not validate this form -->

3. <input type="text" />

4. </form>

***Поле имени***

Взгляните на первое поле, которое запрашивает у пользователя его имя. Как уже описывалось ранее, мы добавили атрибут type= «text» и атрибут required. Они информируют веб-браузер о том, что это обязательное поле и тот должен делать его валидацию как простого текста. Так что, если пользователь введет в поле по меньшей мере один символ, оно будет достоверным.

Теперь мы создадим свой собственный CSS для назначения стилей полям ввода, считающимся браузером достоверными и недостоверными. Если помните, мы использовали :required в своем CSS для определения стилей всем элементам ввода с обязательным атрибутом. Теперь нам можно назначить стили обязательным к заполнению полям, достоверным или нет, добавив в правила CSS :valid или :invalid.

Во-первых, давайте назначим стили недостоверным полям. В этом примере нам нужно определить стили формы как невалидные, когда та находится в фокусе. Мы добавим красную окантовку, красную тень и созданную красную иконку (images/invalid.png) для обозначения невалидного поля.

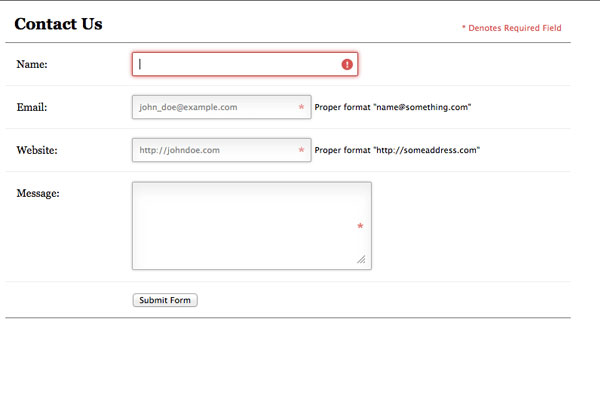
1. .contact\_form input:focus:invalid, .contact\_form textarea:focus:invalid { /\* when a field is considered invalid by the browser \*/

2. background: #fff url(images/invalid.png) no-repeat 98% center;

3. box-shadow: 0 0 5px #d45252;

4. border-color: #b03535

5. }



Теперь создадим правила, которые обозначат достоверное поле. Добавим зеленую окантовку, зеленую тень и зеленую иконку с «галочкой» (images/valid.png). Они будут применены ко всем валидным полям, независимо от того, находятся те в фокусе или нет.

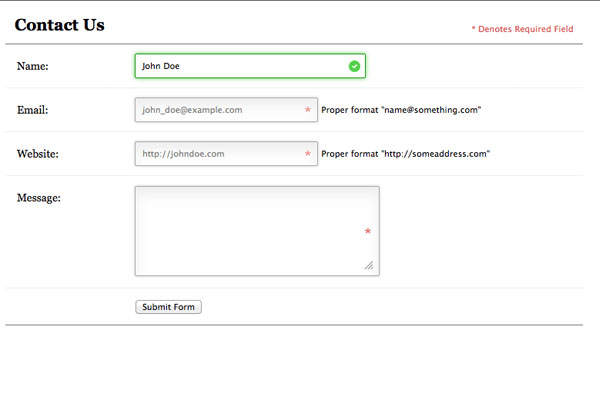
1. .contact\_form input:required:valid, .contact\_form textarea:required:valid { /\* when a field is considered valid by the browser \*/

2. background: #fff url(images/valid.png) no-repeat 98% center;

3. box-shadow: 0 0 5px #5cd053;

4. border-color: #28921f;

5. }



Сейчас, когда мы активизируем поле формы, видны стили в красном цвете, означающие невалидность. Как только в поле вводится хотя бы один символ, оно становится валидным, и отражают этот факт появляющиеся зеленые стили CSS.

***Поля электронного адреса и URL’а***

Стили CSS и правила валидации уже применены к полю электронного адреса, так как ранее мы уже установили атрибуты type и required.

***14. Представляем атрибут HTML5 pattern***

Если для примера использовать атрибут type= «email», то похоже, что большинство браузеров делают валидацию этого поля как \*@\* (любой символ + символ “@” + любой символ). Это, конечно, не очень ограничивает, однако предотвращает введение пользователем пробелов или совершенно неправильных значений.

В примере атрибута type= «url», похоже, что минимальные требования большинства браузеров – это символ, за которым следует двоеточие. Поэтому, если вы ввели “h:”, то поле считается валидным. Это не очень-то помогает, но действительно предотвращает введение пользователями неподходящей информации, такой как их электронный или домашний адрес. Рассмотрим, как это сделать в HTML5.

***Атрибут pattern***

Атрибут pattern принимает регулярное выражение javascript’а. Это выражение, а не выражение браузера по умолчанию, применяется для валидации значения поля. Так что теперь наш HTML выглядит так:

1. <input type="url" name="website" placeholder="http://johndoe.com" required pattern="(http|https)://.+" />

Теперь наше поле будет принимать только значения, начинающиеся с “http://” или “https://” и один дополнительный символ.

***15. Подсказки к полям формы (CSS)***

Теперь давайте определим стили подсказок к своей форме, говорящих пользователю формат, который следует использовать при вводе информации.

01. .form\_hint {

02. background: #d45252;

03. border-radius: 3px 3px 3px 3px;

04. color: white;

05. margin-left:8px;

06. padding: 1px 6px;

07. z-index: 999; /\* hints stay above all other elements \*/

08. position: absolute; /\* allows proper formatting if hint is two lines \*/

09. display: none;

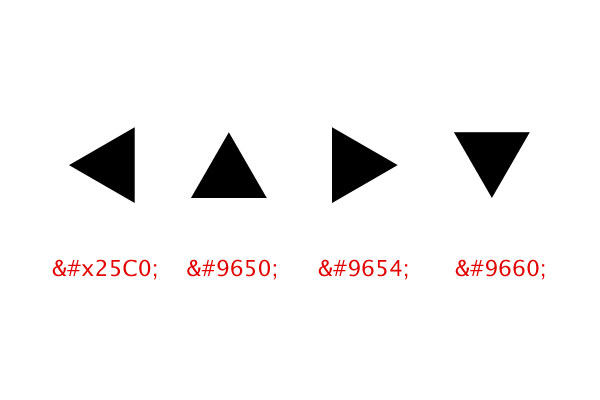
10. }

Мы устанавливаем display:none, потому что нам нужно показывать подсказки только тогда, когда пользователь фокусируется на поле ввода. Мы также по умолчанию устанавливаем всплывающие подсказки на красный цвет недостоверности, так как они всегда считаются невалидными, пока в них не будет введена правильная информация.

***Применение селектора ::before***

Теперь нам нужно добавить к блокам подсказок маленький треугольник, который поможет направлять взгляд и руководить им. Это можно делать с использованием изображений, но в данном случае мы сделаем все с помощью одного лишь CSS.

Так как это – чисто презентационный элемент, который не является жизненно важным для функциональности страницы, то добавим маленький треугольник, указывающий влево, при помощи псевдоселектора ::before. Это можно сделать с помощью одной из геометрических фигур unicode’а.



Обычно мы бы применили формат HTML Unicode, чтобы отобразить их в своем HTML (как на изображении вверху). Однако из-за использования селектора CSS ::before нам придется взять соответствующий треугольнику escaped unicode с применением правила content: «». Затем просто применим позиционирование, чтобы поместить его в нужное место.

1. .form\_hint::before {

2. content: "\25C0"; /\* треугольник, указующий влево, в escaped unicode \*/

3. color:#d45252;

4. position: absolute;

5. top:1px;

6. left:-6px;

7. }

**Применение смежного селектора +**

Наконец, мы используем смежный селектор CSS для показа и скрытия подсказок к нашим полям формы. Смежный селектор (x + y) выбирает элемент, которому непосредственно предшествует предыдущий элемент. Так как подсказки для полей идут в нашем HTML сразу после полей ввода, можно применять этот селектор для показа/скрытия всплывающих подсказок.

1. .contact\_form input:focus + .form\_hint {display: inline;}

2. .contact\_form input:required:valid + .form\_hint {background: #28921f;} /\* change form hint color when valid \*/

3. .contact\_form input:required:valid + .form\_hint::before {color:#28921f;} /\* change form hint arrow color when valid \*/

Как видно из CSS, мы также установили подсказки к форме, чтобы те меняли цвет наряду с рамкой ввода, когда поле валидно или невалидно.

***16. Откиньтесь в кресле и любуйтесь своей великолепной формой HTML5***

Давайте, наконец, рассмотрите свой полученный продукт!

