

HyperText Markup Language

HTML&CSS



Lesson 10

Forms

Contents

Forms	3
Homework.....	36

Материалы урока прикреплены к данному PDF-файлу. Для доступа к материалам, урок необходимо открыть в программе Adobe Acrobat Reader.

Forms

Nowadays, people spend lots of time on the Internet. They go to various websites, buy things, search for information. And users often use a tool named form to complete their tasks. This word may sound unfamiliar to you, but you have already had some experience with forms.

For instance, forms help you search for information. Typical examples of search forms:

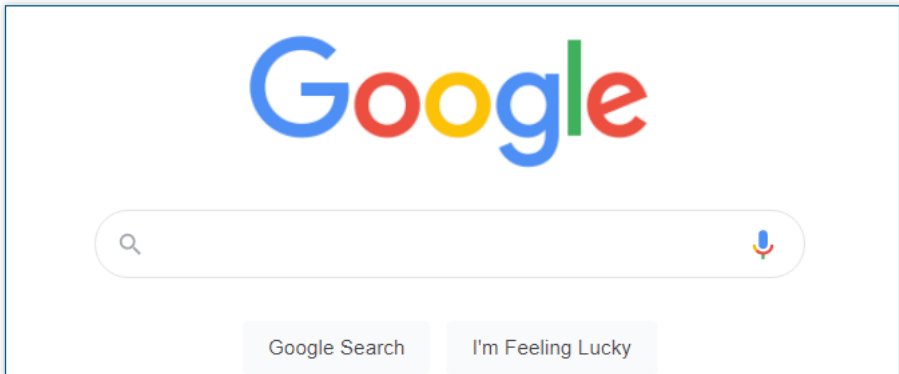


Figure 1

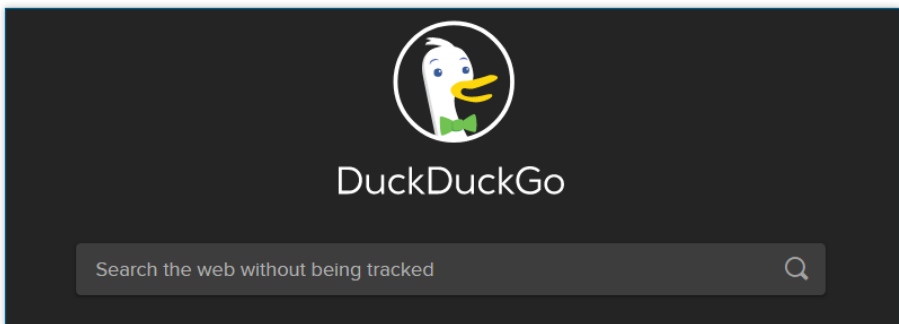
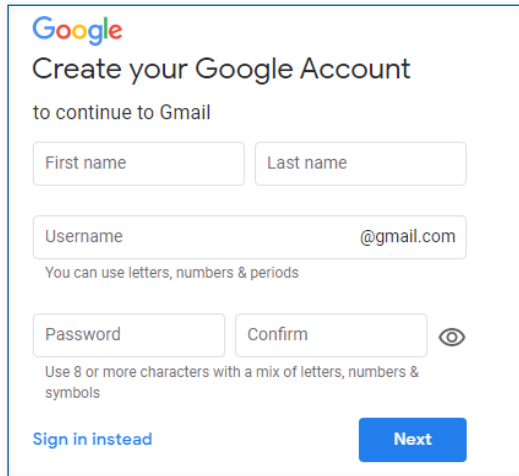


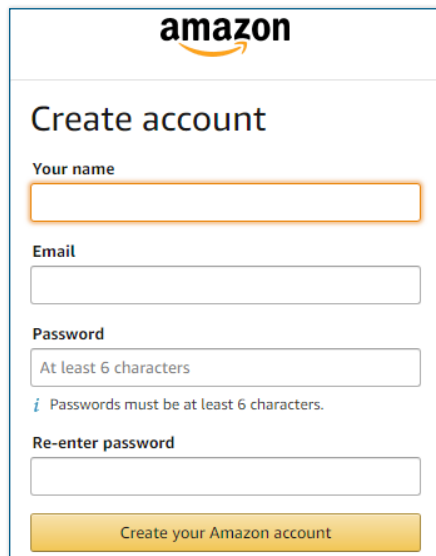
Figure 2

Forms are used for more than just finding information. When registering on sites, you might see forms like these:



The image shows the Google Account creation form. At the top is the Google logo. Below it is the heading "Create your Google Account" followed by the subtext "to continue to Gmail". The form contains several input fields: "First name" and "Last name" (two separate boxes), "Username" (with a placeholder "@gmail.com" and a note "You can use letters, numbers & periods"), "Password" (with a note "Use 8 or more characters with a mix of letters, numbers & symbols"), and "Confirm" (with an eye icon for toggling visibility). At the bottom left is a link "Sign in instead" and at the bottom right is a blue "Next" button.

Figure 3



The image shows the Amazon account creation form. At the top is the Amazon logo. Below it is the heading "Create account". The form contains several input fields: "Your name", "Email", "Password" (with a note "At least 6 characters" and a tip "i Passwords must be at least 6 characters."), and "Re-enter password". At the bottom is a yellow button labeled "Create your Amazon account".

Figure 4

So, what is a form? It is a set of controls that allows a web app to interact with the user. For example, if we are looking for information, the form provides the ability to enter a search word. If we want to create an account on a web store, the registration form allows us to provide information about ourselves.

So, how to embed the form inside an HTML page? The `<form>` tag is used for this. When using `form`, you need to specify both opening and closing tags. The controls must be between the opening and closing tags `<form>`. General structure:

```
<form>
    controls
</form>
```

The `form` tag has a set of attributes. We will get to know them a little later.

The `<input>` tag helps create a large number of controls. The `type` attribute of this tag specifies which control needs to be created. There are also controls created with other tags. We will discuss them later as well.

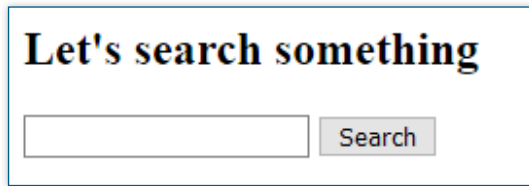
Let's create the first example of working with forms. Our page will display a search form with a text box and a button to send data to the server.

```
<!DOCTYPE html>

<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>First form</title>
  </head>
```

```
<body>
  <h2>Let's search something</h2>
  <form>
    <input type="text" />
    <input type="submit" value="Search" />
  </form>
</body>
</html>
```

Browser display:



Let's search something

Figure 5

We use the `<form>` tag to create a form.

```
<form>
  <input type="text" />
  <input type="submit" value="Search" />
</form>
```

We create two controls in the code: a text box and a button to send information. The text box is created with the `<input>` tag. The `type` attribute set to `text` indicates that we want to create exactly the text box. The button to send data to the server is also created with `input`, but its `type` is set to `submit`. We used the `value` attribute to create text on the button.

Buttons such as `submit` are always used to send data to the server.

Try to enter a value into the text box and click the [Search](#) button. What do you get? The page has reloaded, and the value from the text box has disappeared. Why? Let's try to figure it out.

How does the standard mechanism of sending data upon clicking on the [submit](#) button work?

1. The browser sends the data received from the user to the server.
2. The server receives data, generates a response, and sends it to the client (the browser in our case).
3. The browser receives the server's response and displays it on the user's screen.

There is no interaction with the server in our example. We have never indicated where the data needs to be sent. This is why our page simply refreshes. Our project code is in the folder named *First form*.

Let's complicate our example a little by adding another button and setting up some [form](#) attributes.

```
<!DOCTYPE html>

<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>Simple search form</title>
  </head>

  <body>
    <h2>Let's search something</h2>
    <form action="/search" method="POST">
      <input type="text" /><br />
```

```
<input type="submit" value="Search" />
<input type="reset" value="Clear" />
</form>
</body>
</html>
```

The result is as follows:

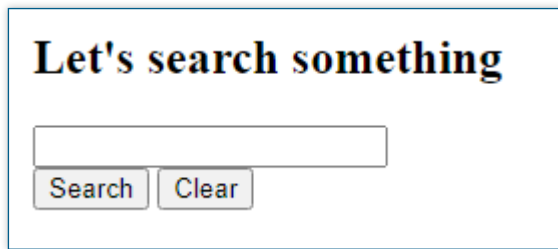
A screenshot of a web browser displaying a simple search form. The form has a title "Let's search something" in a large, bold, black serif font. Below the title is a single-line text input field. Underneath the input field are two buttons: "Search" and "Clear", both in a smaller, black serif font. The entire form is enclosed in a thin blue rectangular border.

Figure 6

Try to enter a value in the text box and click the [Clear](#) button. The value from the text box will be erased. When performing the cleanup, the page will not be sent to the server.

Let's analyze new things in the code. The [form](#) tag has attributes.

```
<form action="/search" method="POST">
```

The [action](#) tag requires you to specify the path on the server that will process the form data. If the [action](#) attribute is not specified, the current page is selected as a handler on the server. The value to the [action](#) attribute is usually assigned by a programmer who handles the back end of the web app.

In the [method](#) attribute, specify which method to use to send data to the server. There are two options: [GET](#) and

POST. Let's consider each option for sending data. And we start with **GET**.

If the form has **GET** specified as a sending method, the entered data is displayed in the address bar of the browser. For example:

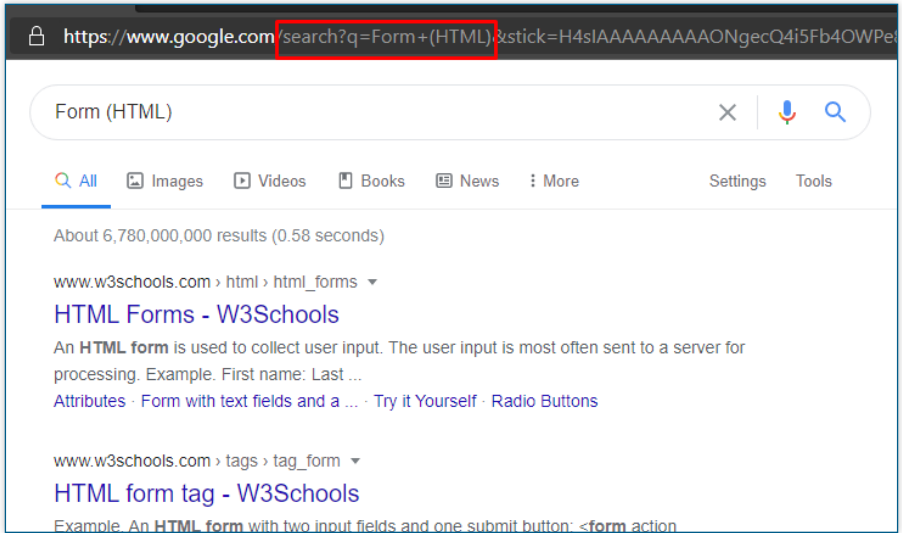


Figure 7

GET is usually specified as a sending method for search engines. When you specify **POST** as a **method**, data is sent in the body of the HTTP request and is not displayed in the address bar. It can be concluded that you should not use **GET** to send sensitive data because they will be in the URL, and anyone who comes to your computer can see it.

Important notes on **GET**:

- The URL length in your browser is limited (to about 3000 characters). This means that we have a limit on data length sent with **GET**.

- If your user wants to bookmark the URL, better use **GET**. For instance, to bookmark a specific search query.
- **GET** is best used for nonclassified data.
- Important notes on **POST**:
- No restrictions on data length.
- If a form uses **POST**, the user cannot bookmark the result.

The decision on a specific method must be made after carefully analyzing all the factors.

In our example, we specified the value of **POST** for **method**. If you do not specify a value for **method**, it defaults to **GET**.

In addition, we introduced a new kind of control in our example.

```
<input type="reset" value="Clear" />
```

The **reset** button is used to reset values entered in the form controls.

The project code is in the folder named *Simple search form*.

Let's expand the range of controls we know. For this, create a registration form with a large number of fields. This is how our example looks:



The image shows a web form with a blue border. At the top, it says "We need your opinion!" in bold blue text. Below this, there are three input fields: "Name:" followed by a single-line text box, "Email:" followed by a single-line text box, and "Message:" followed by a larger multi-line text area. At the bottom of the form is a button labeled "Send my message".

Figure 8

We ask the user to enter: name, email, message (multi-line, text box). We have not formatted the form in this example to keep it simple. In the following examples, we will get to the appearance of our forms.

Pay attention, if you left-click on the label to the left of the control, the control receives focus. Let's try to click on **Name**:



The image shows a web form with a blue border. At the top, it has a heading **We need your opinion!**. Below the heading are three input fields: a single-line text box for 'Name:', a single-line text box for 'Email:', and a multi-line text area for 'Message:'. At the bottom of the form is a button labeled 'Send my message'.

Figure 9

The text box to the left of the **Name** has the focus. Our page code:

```
<!DOCTYPE html>

<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>Send message to us</title>
  </head>

  <body>
    <h3>We need your opinion!</h3>
    <form action="/search" method="POST">
      <div>
        <label for="uname">Name:</label>
        <input type="text" id="uname" name="userName" />
```

```

    </div>
    <div>
      <label for="email">Email:</label>
      <input type="text" id="email" name="userEmail" />
    </div>
    <div>
      <label for="message">Message:</label>
      <textarea id="message" name="userMessage">
      </textarea>
    </div>
    <div>
      <input type="submit" value="Send my message" />
    </div>
  </form>
</body>
</html>

```

What new elements are there in our form?

The first element is the label. We create it with the `label` tag. It is used to create a label text. We use the `for` attribute to specify the element with which the label is associated. We specify `id` of the associated element in `for`. Thanks to this attribute, the focus moves to the desired element when the label is clicked.

```

<label for="uname">Name:</label>
<input type="text" id="uname" name="userName" />

```

In this code snippet, the label is associated with the text box, as the `for` attribute has the `uname` value. This is an `id` of the text box for entering a name. You may have a question: why do we set `id` and `name` for a control? The answer is:

- `id` is a unique identifier on the page (the `id` values must not repeat). It is used to access an element from the client-side of the app (e.g., from the JavaScript code).

- **name** should be specified to transmit data to the server. If you do not specify **name**, the server-side will not be able to access the value of the element. **name** has no requirements for uniqueness. Different forms may contain elements with the same name on the same page.
- The **id** and **name** values can be the same.

The second element is a multi-line text box. We used the **textarea** tag for its creation.

```
<textarea id="message" name="userMessage"></textarea>
```

We did not use any specific attributes for this element. In the page code, we used the **div** tags for basic formatting. The project code is in the folder named **Contact form**.

Let's add some design to our code with the registration form using CSS.

The image shows a web form with a title "We need your opinion!". Below the title is a rounded rectangular container. Inside this container, there are four labels with corresponding input fields: "Nick:" followed by a single-line text input, "Email:" followed by a single-line text input, "Phone:" followed by a single-line text input, and "Message:" followed by a multi-line text area. At the bottom of the rounded container is a button labeled "Send my message".

Figure 10

New page code:

```
<!DOCTYPE html>

<html lang="en">
  <head>
    <meta charset="utf-8" />
    <style>
      ul {
        list-style: none;
        padding: 0;
        margin: 0;
      }

      label {
        display: inline-block;
        width: 70px;
        text-align: right;
      }

      form li + li {
        margin-top: 0.5em;
      }

      /*
        Define settings for the form width, border,
        and paddings
      */

      form {
        width: 300px;
        padding: 1em;
        border: 1px solid rgba(23, 4, 56, 0.829);
        border-radius: 2em;
      }
```

```

    textarea {
      vertical-align: top;
    }

    input, textarea {
      /*
        Set the same font for input and textarea.
        The textarea uses monospace by default
      */
      font: 1em sans-serif;
      width: 200px;
      border: 1px solid #777;
    }

    input:focus, textarea:focus {
      /* Highlight the element that gets focus */
      border-color: #000;
    }

    /* center our button */
    input[id="submitBtn"] {
      margin-left: auto;
      margin-right: auto;
      display: block;
    }
  </style>
  <title>Send message to us</title>
</head>

<body>
  <h3>We need your opinion!</h3>
  <hr />
  <form>
    <ul>
      <li>
        <label for="nick">Nick:</label>

```

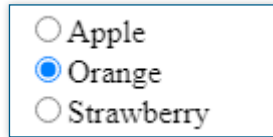
```

        <input type="text" id="nick" name="userNick" />
    </li>
    <li>
        <label for="email">Email:</label>
        <input type="email" id="email"
            name="userEmail" />
    </li>
    <li>
        <label for="phone">Phone:</label>
        <input type="text" id="phone"
            name="userPhone" />
    </li>
    <li>
        <label for="message">Message:</label>
        <textarea id="message" name="userMessage">
        </textarea>
    </li>
    <li>
        <input id="submitBtn" type="submit"
            value="Send my message" />
    </li>
</ul>
</form>
<hr />
</body>
</html>

```

The changes did not affect the form code. We added a set of styles you are already familiar with for neater design. Work on the form code to achieve the design you want. The project code is in the folder named *Contact form2*.

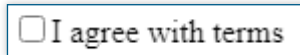
Let's continue our acquaintance with the controls. Next go two new elements: a **radio button** (**radio**) and **checkbox**. These elements are familiar to you. We are sure that you have already used them.



☐ Apple
☒ Orange
☐ Strawberry

Figure 11

This is an example of a set of radio buttons. You can choose only one button from a group of radio buttons.



☐ I agree with terms

Figure 12

This is an example of a checkbox. You can check or uncheck the box.

Let's create a new registration form using radio buttons and a checkbox. This is how our page looks:



Please provide information about you

First Name:

Last Name:

Email:

Password:

Subscribe?: ☐

Gender

☐ Male ☐ Female ☐ Other

Figure 13

This is the code:

```
<!DOCTYPE html>

<html lang="en">
  <head>
    <meta charset="utf-8" />
    <style>

      ul {
        list-style: none;
        margin: 0;
        padding: 0;
      }

      li {
        padding: 0.2em;
      }

      form {
        width: 300px;
        font-family: Verdana, sans-serif;
        font-size: 15px;
        padding: 1em;
        border: 1px solid #000;
        border-radius: 1em;
      }

      input {
        font: 1em sans-serif;
        width: 150px;
        box-sizing: border-box;
        border: 1px solid #777;
      }

      input:focus {
        border-color: rgb(25, 11, 100);
      }
    </style>
  </head>
  <body>
    <ul>
      <li><input type="text" value="Name" /></li>
      <li><input type="text" value="Email" /></li>
      <li><input type="text" value="Phone" /></li>
      <li><input type="text" value="Address" /></li>
      <li><input type="text" value="City" /></li>
      <li><input type="text" value="State" /></li>
      <li><input type="text" value="Zip" /></li>
      <li><input type="text" value="Country" /></li>
      <li><input type="text" value="Website" /></li>
      <li><input type="text" value="Comments" /></li>
    </ul>
  </body>
</html>
```

```

    input[type="checkbox"],
    input[type="radio"] {
        width: auto;
    }

    input[id="submitBtn"] {
        display: block;
        width: 120px;
        margin-left: auto;
        margin-right: auto;
        margin-top: 10px;
    }

    label span {
        width: 90px;
        text-align: right;
        display: inline-block;
    }
</style>
<title>Moonlight news</title>
</head>

<body>
    <h3>Please provide information about you</h3>
    <hr />
    <form>
        <ul>
            <li>
                <label>
                    <span>First Name:</span>
                    <input type="text" id="fname"
                        name="firstName" />
                </label>
            </li>
            <li>
                <label>
                    <span>Last Name:</span>

```

```

        <input type="text" id="lName"
              name="lastName" />
    </label>
</li>
<li>
    <label>
        <span>Email:</span>
        <input type="email" id="email"
              name="userEmail" />
    </label>
</li>
<li>
    <label>
        <span>Password:</span>
        <input type="password" id="password"
              name="userPassword" />
    </label>
</li>
<li>
    <label for="news">
        <span>Subscribe?:</span>
        <input type="checkbox" id="news"
              name="newsCheck" />
    </label>
</li>
<li>
    <p>Gender</p>
    <input type="radio" value="male"
          id="gender1" name="sex" />
    <label for="gender1">Male</label>
    <input type="radio" value="female"
          id="gender2" name="sex" />
    <label for="gender2">Female</label>
    <input type="radio" value="other"
          id="gender3" name="sex" />
    <label for="gender3">Other</label>
</li>

```

```

        <li id="buttons">
            <input id="submitBtn" type="submit"
                value="Register" />
        </li>
    </ul>
</form>
<hr />
</body>
</html>

```

We used several new techniques in this example. Our checkbox is created in this piece of code:

```

<label for="news">
    <span>Subscribe?:</span>
    <input type="checkbox" id="news" name="newsCheck" />
</label>

```

There is the `label` already familiar to us; and when it is clicked, the checkbox is checked. This happens because we set `label` in the `for` attribute to `news` (this is the `id` of our checkbox). Below you will learn that we could have skipped `for`. And it would have the same effect. The checkbox was created using the `input` tag the `type` of which is set to `checkbox`.

Now let's look at the code for creating a group of radio buttons.

```

<li>
    <p>Gender</p>
    <input type="radio" value="male" id="gender1" name="sex"/>
    <label for="gender1">Male</label>
    <input type="radio" value="female" id="gender2"
        name="sex" />

```

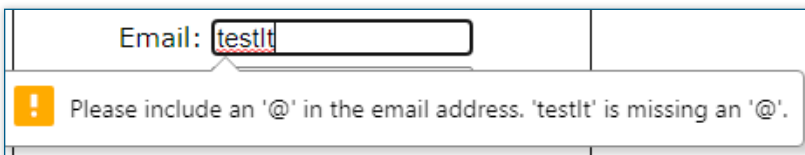
```
<label for="gender2">Female</label>
<input type="radio" value="other" id="gender3"
      name="sex" />
<label for="gender3">Other</label>
</li>
```

To create a radio button, we used the `input` tag set to `type` equal to `radio`. We created three radio buttons in our code and specified the same `name` value for each radio button. We did it so that the user could choose only one radio button from the group. Please note that we set for each of the buttons its own value in the `value` attribute. This is done so that the server-side of the app can determine which button is selected.

We created a text box where the user enters his or her `email` using a new tag:

```
<input type="email" id="email" name="userEmail" />
```

This code creates a text box that will automatically verify the `email` when sending data to the server.



The image shows a web form with a label 'Email:' followed by a text input field containing the text 'testlt'. Below the input field, there is a yellow warning icon and a message: 'Please include an '@' in the email address. 'testlt' is missing an '@'.'

Figure 14

We also used a special element for entering a password.

```
<input type="password" id="password"
      name="userPassword"/>
```

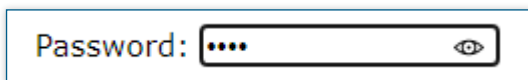

 A form element consisting of a label 'Password:' followed by a text input field. The input field contains four dots, indicating a password, and has a small eye icon to its right for toggling visibility.

Figure 15

We recommend you use these elements rather than ordinary text boxes.

And one more important aspect of our example.

```
<label>
  <span>First Name:</span>
  <input type="text" id="fname" name="firstName" />
</label>
```

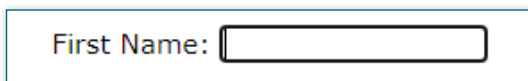

 A form element consisting of a label 'First Name:' followed by a standard text input field.

Figure 16

If you click on the [First Name](#), the focus goes to the text box next to the label. This effect is familiar to us, but we did not specify the [for](#) attribute for the [label](#) tag. In this example, we have used an alternative. We have placed a text box and [span](#) inside [label](#). This mechanism leads to the same result as [for](#).

The full code is in the folder named *Registration form*.

Let's continue on controls. We have a new element — [combo box](#). You know the appearance of the element.

An expanded combo box:

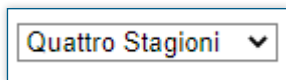

 A form element consisting of a dropdown menu (combo box) with the text 'Quattro Stagioni' and a downward-pointing arrow icon.

Figure 17

A collapsed combo box:

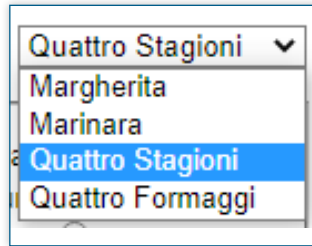


Figure 18

A list is usually used to select one specific item.

To get acquainted with it, we will create a pizza order form.
The appearance of our form:

Fill details for pizza order:

Name:

Phone:

Address:

Choose pizza:

Select size:

Small ☐

Medium ☒

Large ☐

Figure 19

The source code of the page:

```
<html lang="en">
<head>
  <style>
    h4 {
      font-family: Verdana, sans-serif;
    }

    form {
      width: 250px;
      font-family: Verdana, sans-serif;
      font-size: 12px;
    }

    input {
      font: 1em Verdana, sans-serif;
      width: 130px;
      box-sizing: border-box;
      border: 1px solid #777;
    }

    input:focus {
      border-color: rgb(255, 255, 0);
    }

    div {
      padding: 0.2em;
    }

    select {
      width: 130px;
      box-sizing: border-box;
      border: 1px solid #777;
    }

    ul {
      list-style: none;
      padding: 0;
```

```
        margin: 0;
    }

    fieldset {
        margin: 0.5em;
        width: 200px;
    }

    label span {
        width: 90px;
        text-align: right;
        display: inline-block;
    }

    input[type="radio"] {
        width: auto;
    }

    button[type="submit"] {
        width: 100px;
        height: 30px;
        font-family: Verdana, sans-serif;
        font-size: 12px;
        margin-left: 80px;
    }
</style>

<meta charset="utf-8" />
<title>Order pizza</title>
</head>

<body>
    <h4>Fill details for pizza order:</h4>

    <form>
        <div>
            <label>
```

```

        <span>Name:</span>
        <input type="text" id="name" name="userName" />
    </label>
</div>
<div>
    <label>
        <span>Phone:</span>
        <input type="text" id="phone" name="userPhone"/>
    </label>
</div>
<div>
    <label>
        <span>Address:</span>
        <input type="text" id="address"
            name="userAddress" />
    </label>
</div>
<div>
    <label>
        <span>Choose pizza:</span>
        <select id="nameOfPizza" name="nameOfPizza">
            <option>Margherita</option>
            <option>Marinara</option>
            <option>Quattro Stagioni</option>
            <option>Quattro Formaggi</option>
        </select>
    </label>
</div>
<fieldset>
    <legend>Select size:</legend>
    <ul>
        <li>
            <label>
                <span>Small</span>
                <input type="radio" id="small"
                    name="size" value="small" />
            </label>

```

```

        </li>
        <li>
            <label>
                <span>Medium</span>
                <input
                    type="radio"
                    id="medium"
                    name="size"
                    value="medium"
                    checked
                />
            </label>
        </li>
        <li>
            <label>
                <span>Large </span>
                <input type="radio" id="medium"
                    name="size" value="medium" />
            </label>
        </li>
    </ul>
</fieldset>
<div>
    <button type="submit">Order</button>
</div>
</form>

</body>
</html>

```

The `<select>` tag is used to create the combo box. The list items are described inside it. An individual list item is created with the `<option>` tag. The list creation code:

```

<select id="nameOfPizza" name="nameOfPizza">
    <option>Margherita</option>

```

```
<option>Marinara</option>
<option>Quattro Stagioni</option>
<option>Quattro Formaggi</option>
</select>
```

The number of `<option>` is equal to the number of pizza options. The page code has one more new tag — `<fieldset>`. It groups the form elements. The result of its work is as follows:

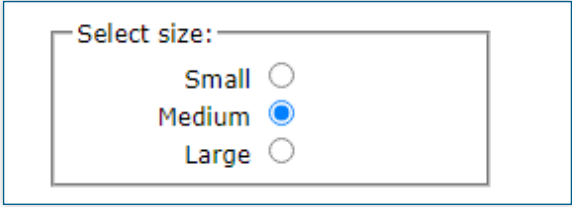


Figure 20

The browser displays a frame around elements grouped by `<fieldset>`. The element code is contained inside `<fieldset>`.

```
<fieldset>
  tags
</fieldset>
```

The `legend` tag is used for the `fieldset` header.

```
<fieldset>
  <legend>Select size:</legend>
```

Also note that we selected one of the radio buttons (pizza size `Medium`). We used the `checked` attribute to achieve this result.

```
<input
  type="radio"
  id="medium"
  name="size"
  value="medium"
  checked
/>
```


The project code is in the folder named *Pizza form*.

And one more example. We will create a registration form for a cinema website. The purpose of this page is to show form elements that we have not used before.

The appearance of the page:

Fill information about you and your cinema experience:

Name:

Birthday: 

Email:

Blog:

Genre:

How you rate yourself:

Figure 21

We used the following new features on this page: date picker, [URL](#) input, auto-complete text box, [spin control](#).

Let's start with the page code. We will only provide the form code. The full code of the page can be found in the *Cinema form* folder.

```
<form>

  <div>
    <label>
      <span>Name:</span>
      <input type="text" id="name"
              name="userName" />
    </label>
  </div>

  <div>
    <label>
      <span>Birthday:</span>
      <input type="date" name="birthday"
              id="userBirthday">
    </label>
  </div>

  <div>
    <label>
      <span>Email:</span>
      <input type="email" id="email"
              name="userEmail" />
    </label>
  </div>

  <div>
    <label>
      <span>Blog:</span>
      <input type="url" id="url" name="userUrl">
    </label>
  </div>

  <div>
    <label>
      <span>Genre:</span>
```

```

        <input type="text" name="genre" id="userGenre"
              list="genreSuggestions">
        <datalist id="genreSuggestions">
          <option>Comedy</option>
          <option>Horror</option>
          <option>Drama</option>
          <option>Thriller</option>
          <option>Science Fiction</option>
          <option>Fantasy</option>
        </datalist>
      </label>
    </div>

    <div>
      <label>
        <span>How you rate yourself:</span>
        <input type="number" name="userRating"
              id="rating" min="1" max="12"
              step="1">
      </label>
    </div>

    <div>
      <button type="submit">Send my data</button>
    </div>
  </form>

```

Let's analyze it step by step. Begin with the date picker. Dates have always been a headache for HTML developers. With the advent of HTML 5, the situation has improved due to the appearance of controls that solve this problem.

```
<input type="date" name="birthday" id="userBirthday">
```

To create a date control, we use `input` with `type = date`.

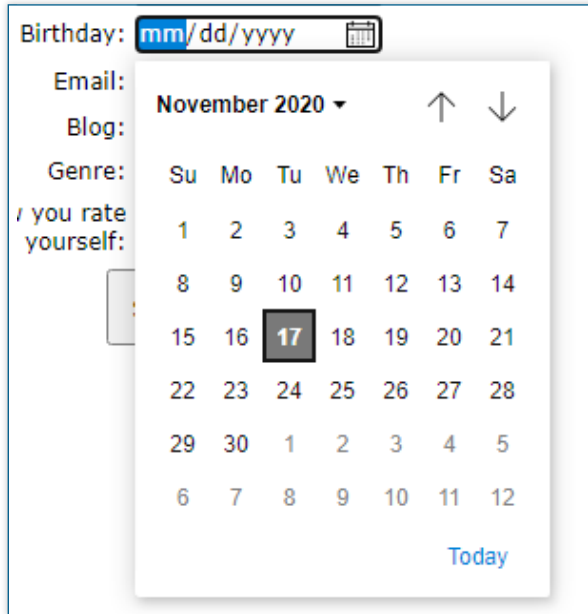


Figure 22

Unfortunately, not all browsers support this element as of yet. Be sure to check the support of a tag in the browser. Let's say you need to check if the [date](https://caniuse.com/input-datetime) tag is supported. You can do it by going to <https://caniuse.com/input-datetime>.

The second new element is the [URL](#) input field. It verifies the address. Please note that this element does not check if your [URL](#) exists. Its purpose is to verify the data format. For example:

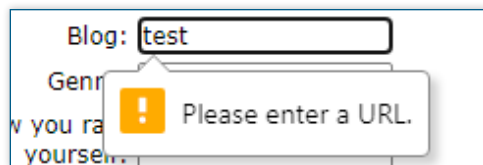
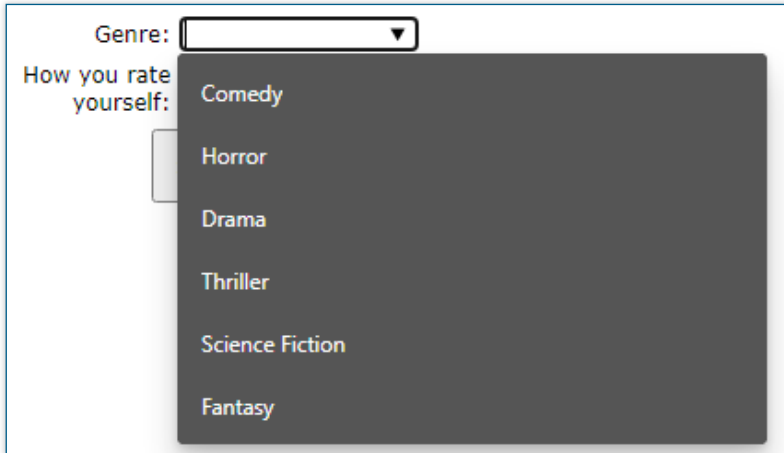


Figure 23

We used the following code to create this field:

```
<input type="url" id="url" name="userUrl">
```

The third new element is an auto-complete text box. This is how it looks before entering text:



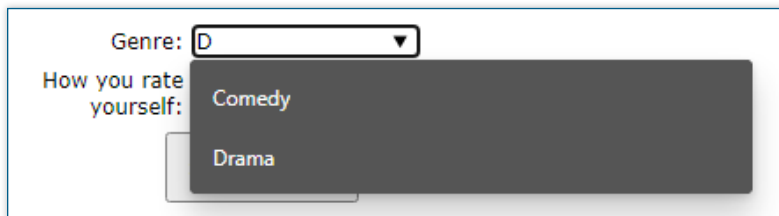
Genre:

How you rate yourself:

- Comedy
- Horror
- Drama
- Thriller
- Science Fiction
- Fantasy

Figure 24

Let's enter some text:



Genre:

How you rate yourself:

- Comedy
- Drama

Figure 25

The code for creating the element:

```
<input type="text" name="genre" id="userGenre"
      list="genreSuggestions" />
```

```
<datalist id="genreSuggestions">
  <option>Comedy</option>
  <option>Horror</option>
  <option>Drama</option>
  <option>Thriller</option>
  <option>Science Fiction</option>
  <option>Fantasy</option>
</datalist>
```

We added one **list** attribute in the text box. It specifies the list of auto completion values. The **list** has a list **id**.

The list is defined with the **datalist** below. The **option** tag creates a specific list item.

And the last new element is **spin control**. Its appearance:

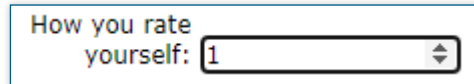


Figure 26

Its code:

```
<input type="number" name="userRating" id="rating"
  min="1" max="12" step="1">
```

The spin control is created with **type = number**. We have indicated that the minimum possible value is 1, the maximum value is 12. By clicking on the arrows to the right, the value will increase or decrease by one.

The project code is in the folder named *Cinema form*.

So, we have discussed forms. Be sure to experiment with the examples and consolidate new material by doing your homework.

Homework

1. Create a sushi ordering page. The following information is required:
 - Name;
 - Address;
 - Phone number;
 - Email;
 - Number of chopsticks;
 - Type of chopsticks:
 - ▷ Regular,
 - ▷ Training,
 - List of sushi;
 - Do you want napkins?
 - Comment.
2. Create a registration page for an online game. The following information is required:
 - Nickname;
 - Email;
 - Date of birth;
 - Gender;
 - Country (auto-complete list);
 - Character type;
 - Gaming experience, in years (from 0 to 60).



Lesson 10.

Forms

© STEP IT Academy, www.itstep.org.

All rights to protected pictures, audio, and video belong to their authors or legal owners.

Fragments of works are used exclusively in illustration purposes to the extent justified by the purpose as part of an educational process and for educational purposes in accordance with Article 1273 Sec. 4 of the Civil Code of the Russian Federation and Articles 21 and 23 of the Law of Ukraine "On Copyright and Related Rights". The extent and method of cited works are in conformity with the standards, do not conflict with a normal exploitation of the work, and do not prejudice the legitimate interests of the authors and rightholders. Cited fragments of works can be replaced with alternative, non-protected analogs, and as such correspond the criteria of fair use.

All rights reserved. Any reproduction, in whole or in part, is prohibited. Agreement of the use of works and their fragments is carried out with the authors and other right owners. Materials from this document can be used only with resource link.

Liability for unauthorized copying and commercial use of materials is defined according to the current legislation of Ukraine.