COESIONE ITALIA 21-27 PIEMONTE

Cofinanziato dall'Unione europea

REGIONE PIEMONTE

FinTech Software Developer

Programmazione WEB - HTML | CSS | Javascript

Docente: Shadi Lahham

Intervento realizzato da ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

HTML Forms

User input

Shadi Lahham - Web development

The form element

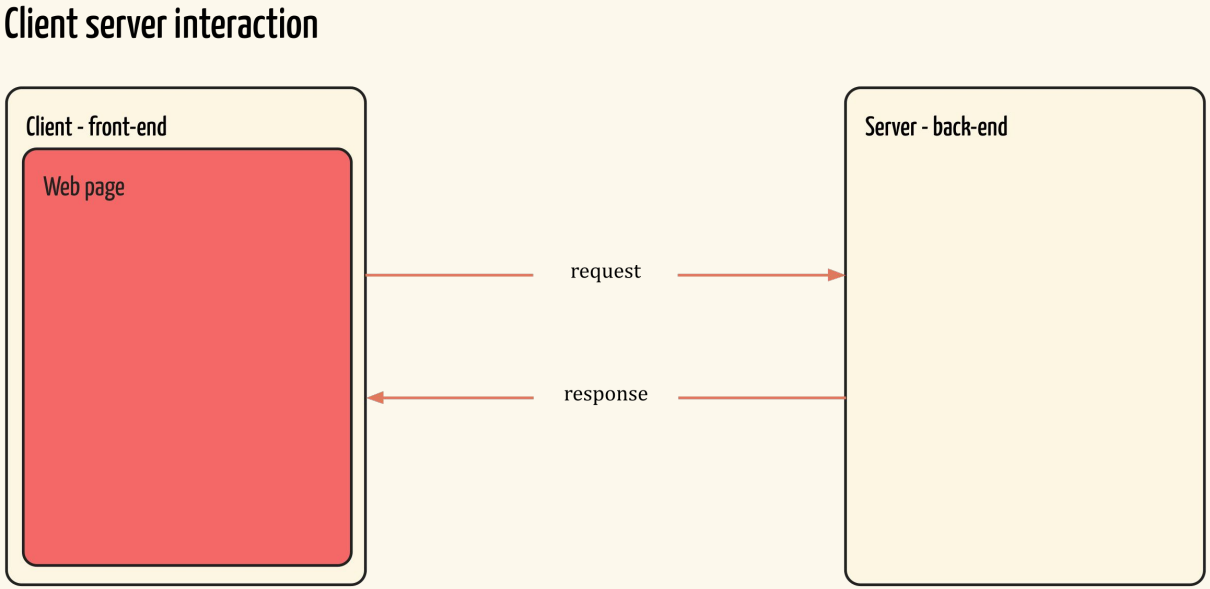
An example of a form

```
<form action="./processing-page.php" method="post">
 <ul>
  <li>
    <label for="name">Name:</label>
    <input type="text" id="name" name="user_name">
  </li>
  <li>
    <label for="password">Password:</label>
    <input type="password" id="password" name="pswd">
  </li>
 </ul>
</form>
```

What are forms

- Forms allow us to collect data from the user
  - signing up and logging in to websites
  - entering personal information (name, address, credit card details...)
  - filtering content (by using dropdowns, checkboxes...)
  - performing a search
  - uploading files
- Forms contain elements called controls
  - Text inputs , checkboxes, radio buttons, submit buttons, etc
- When users complete a form the data is usually submitted to a web server (back-end) for processing

Sending and receiving requests



Element: form

```
<form action="..." method="...">

<!-- All form elements go here -->

</form>
```

Element: form

```
<form action="./processing-page.php" target="_blank" method="post">

<!-- All form elements go here -->

</form>
```

Target:

_blank
The submitted result will open in a new browser tab

_self
The submitted result will open in the same page (this is default)

Element: form

```
<form action="./processing-page.php" target="_blank" method="post">

<!-- All form elements go here -->

</form>
```

method:

get
Data sent via get method is visible in the browser's address bar

post
Data sent via post is not visible to the user

Element: form - get vs post

Advantages and disadvantages of the GET method

- Data sent by the GET method is displayed in the URL
- It is possible to bookmark the page with specific query string values
- Not suitable for passing sensitive information such as the username and password
- The length of the URL is limited

Advantages and disadvantages of the POST method

- More secure than GET; information is never visible in the URL query string or in the server logs
- Has a much larger limit on the amount of data that can be sent
- Can send text data as well as binary data (uploading a file)
- Not possible to bookmark the page with the query

Basic form elements

Text Field

```
<form>
    <label for="username">Username:</label>
    <input type="text" name="username" id="username">
</form>
```

One line areas that allow the user to input text
The <label> tag is used to define the labels for <input> elements

placeholder:
Text inputs can display a placeholder text that will disappear as soon as some text is entered

```
<input type="text" placeholder="Enter your name">
```

Password Field

```
<form>
    <label for="user-pwd">Password:</label>
    <input type="password" name="user-password" id="user-pwd">
</form>
```

Similar to text fields except that characters are masked

Radio Buttons

```
<form action="..." method="post" target="_blank">
    <input type="radio" name="gender" id="male" value="man">
    <label for="male">Male</label>
    <input type="radio" name="gender" id="female" value="woman" checked="checked">
    <label for="female">Female</label>
    <button type="submit">Send</button>
</form>
```

Select exactly one option from a set of options
name: assigns name to the form control; used by the browser, screen readers and sometimes javascript
It also gets send to the back-end.

Value: the value will be send to the back-end when the option is selected. Must be unique

Checked: initially selected or not

CheckBoxes

```
<form action="..." method="get" target="_blank">
    <input type="checkbox" name="sports" id="soccer" value="soccer-sport">
    <label for="soccer">Soccer</label>
    <input type="checkbox" name="sports" id="baseball" value="baseball-sport" checked="checked">
    <label for="baseball">Baseball</label>
    <button type="submit">Send</button>
</form>
```

Select one or more options from a set of options
name: assigns name to the form control; used by the browser, screen readers and sometimes javascript
It also gets send to the back-end.

Value: the value will be send to the back-end when the option is selected

Checked: initially selected or not

note: value must be set otherwise a default is sent

Dropdown menus

```
<form>
    <label for="city">City:</label>
    <select name="city" id="city">
        <option value="sydney">Sydney</option>
        <option value="melbourne">Melbourne</option>
        <option value="cromwell" selected>Cromwell</option>
    </select>
</form>
```

Use <select> rather than radio buttons when the number of options to choose from is large

selected is used rather than checked

Dropdown menus

```
<form>
    <label for="city">City:</label>
    <select name="city" id="city" multiple>
        <option value="sydney">Sydney</option>
        <option value="melbourne">Melbourne</option>
        <option value="cromwell" selected>Cromwell</option>
    </select>
</form>
```

multiple
provides the ability to select multiple options. Conceptually, <select> becomes more similar to checkboxes

File Select

```
<form>
    <label for="file-select">Upload:</label>
    <input type="file" name="upload" id="file-select">
</form>
```

Upload a local file as an attachment

Textarea

multi-line text input

```
<form>
    <label for="address">Address:</label>
    <textarea rows="4" cols="20" name="address" id="address"></textarea>
</form>
```

Can start with existing text

```
<form>
    <textarea name="message" rows="2" cols="40">This text can be edited
    </textarea>
</form>
```

Submit and Reset

```
<form action="processor.php" method="post">
    <label for="name">Name:</label>
    <input type="text" name="name" id="user-name">
    <input type="submit" value="Submit">
    <input type="reset" value="Reset">
</form>
```

submit
sends the form data to the location specified in the action attribute

reset
resets all forms controls to the default values

Fieldset

```
<fieldset>
  <legend>Color</legend>
  <input type="radio" name="colour" value="red" id="colour_red">
  <label for="colour_red">Red</label>
  <input type="radio" name="colour" value="green" id="colour_green">
  <label for="colour_green">Green</label>
  <input type="radio" name="colour" value="blue" id="colour_blue">
  <label for="colour_blue">Red</label>
</fieldset>
```

Group controls into categories. Particularly important for screen readers for example

HTML5 Input types

Smarter input types

Newer input types are useful for

- validation
- restricting user input
- Using custom dialogs

Downsides

- most are not supported by very old browsers
- each browser has a different implementation so the user experience is not consistent

Email field

```
<form>
  <label for="user-email">Email:</label>
  <input type="email" name="user-email" id="form-email">
  <button type="submit">Send</button>
</form>
```

Used to receive a valid e-mail address from the user
Most browsers can validate this without needing javascript

Note: older browsers don’t support this input type

More input types

Some of the more useful ones:

```
<input type="email" id="email" name="email">
<input type="url" id="url" name="url">
<input type="number" name="age" id="age" min="1" max="10" step="2">
<input type="search" id="mysearch" name="search-keyword">
```

Complete list here:
The HTML5 input types
HTML Input Types

note: you will need some of these for the exercises

Form validation

What is validation

Validation is a mechanism to ensure the correctness of user input

- Validation can be used to:
  - Make sure that all required information has been entered
  - Limit the information to certain types (e.g. only numbers)
  - Make sure that the information follows a standard (e.g. email, credit card number)
  - Limit the information to a certain length
  - Other validation required by the application or the back-end services

Important details here:

Client-side form validation

Points of validation

Validation should be performed by the front-end as well as the back-end

Front-end

- The application should validate all information to make sure that it is complete, free of errors and conforms to the specifications required by the back-end
- It should contain mechanisms to warn users if input is not complete or correct
- It should avoid to send ‘bad’ data to the back-end

Points of validation

The back-end service should perform its own validation of any data it receives

Back-end

- It should never trust that the front-end has done validation
- Some clever users can bypass the front-end mechanisms easily
- Back-end services can receive data from other services, not necessarily front-end, that don’t perform validation

Front-end validation

Built-in validation

Some browsers have built-in validation systems.

- Not all browsers validate in the same way and some follow the specs partially
- Some browsers don’t have validation at all (older desktop browsers, some mobile browsers)
- Apart from declaring validation intention with HTML5 developers don’t have much control over what the browser actually does
- Before using build-in validation make sure that it’s supported by the target browsers. Always check the specs!

Front-end validation

Javascript validation

Most of the time, validation is done manually with Javascript

- Gives the developer more control
- The developer can make sure it works on all target browsers
- Requires a lot of custom coding, or using a library *(common practice)*

Study: form validation

You should study all links in the reference section, but especially the following page which will be very useful for the following exercises:
Client-side form validation

Your turn

1. Astronaut application

- Build a form to collect the following information from astronaut candidates
  - First, middle (optional) and last names
  - Desired mission (NASA has limited future missions Missions)
  - Age, gender, hair and eye color (color picker or choose from a list)
  - Contact information: email, phone numbers, address, etc
  - Weight (max 100kg - sorry in space weight is limited)
  - A shot biography (max 255 characters)
  - Any other information that you want
-

… continues on next slide

1. Astronaut application

- Send the data to an endpoint that tests requests such as:
  - RequestBin
  - Beeceptor
- Make sure to
  - Group inputs logically
  - Use inputs that are appropriate to the type of information
  - Allow the user to clear the form
  - Submit the form to a site that shows the results
  - Some HTML5 inputs won't work in all browsers, especially older ones, so experiment and choose the inputs wisely

2. Validation

- Validate the user input from the previous exercise
  - Validate as much user input as you can.
    - Checking before sending the data is always a good practice
  - Check if your validation works in Chrome, Firefox, Edge, Android
    - Testing that validation works is always mandatory
  - Try different techniques to make sure your validation works on the browsers above
    - Make a list of which built-in validators don't work on which browsers(useful in future)
  - If some HTML5 inputs from the previous exercise don't work on a particular browser, either replace them, or find other ways to validate
  - **Bonus:** test your form and validation on Safari (using a Mac)
- Important
  - Make sure you read Client-side form validation before starting this exercise
  - Always check from element compatibility on "Can I use"
  - See the links in references for more help

Bonus

3. Astronaut application processor

- This exercise is **optional** but very recomended
- Build a small server to process the astronaut applications that are sent
  - You can use any back-end language that you prefer (php, python, nodeJS, etc)
  - You can use any webserver to host your application (locally or remotely)
  - You can save the data in a database or a local file
  - **Bonus:** add a page to your back-end application to show a list of all applications received
- Notes
  - This exercise may be done in pairs
  - This exercise may be submitted at a future date
  - Pair up with someone that has different skills than you

References

Testing POST and GET requests

RequestBin

Beeceptor

References

Useful form references

Your first form

Sending form data

HTML Forms

HTML Form Elements

References

Validation

Differentiate between client side validation and server side validation