ΕΠΛ425

Τεχνολογίες Διαδικτύου

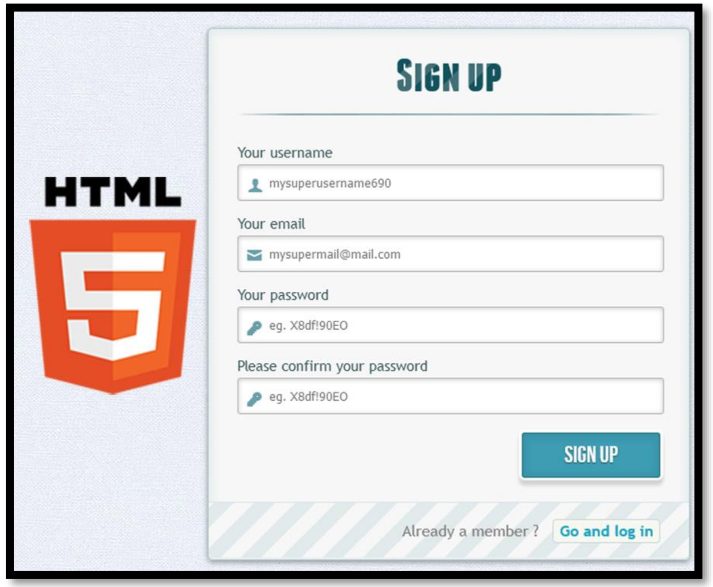
(Internet Technologies)

HTML Forms & Default Submission

Διδάσκων
Δρ. Χριστόφορος Χριστοφόρου
christophoros@cs.ucy.ac.cy

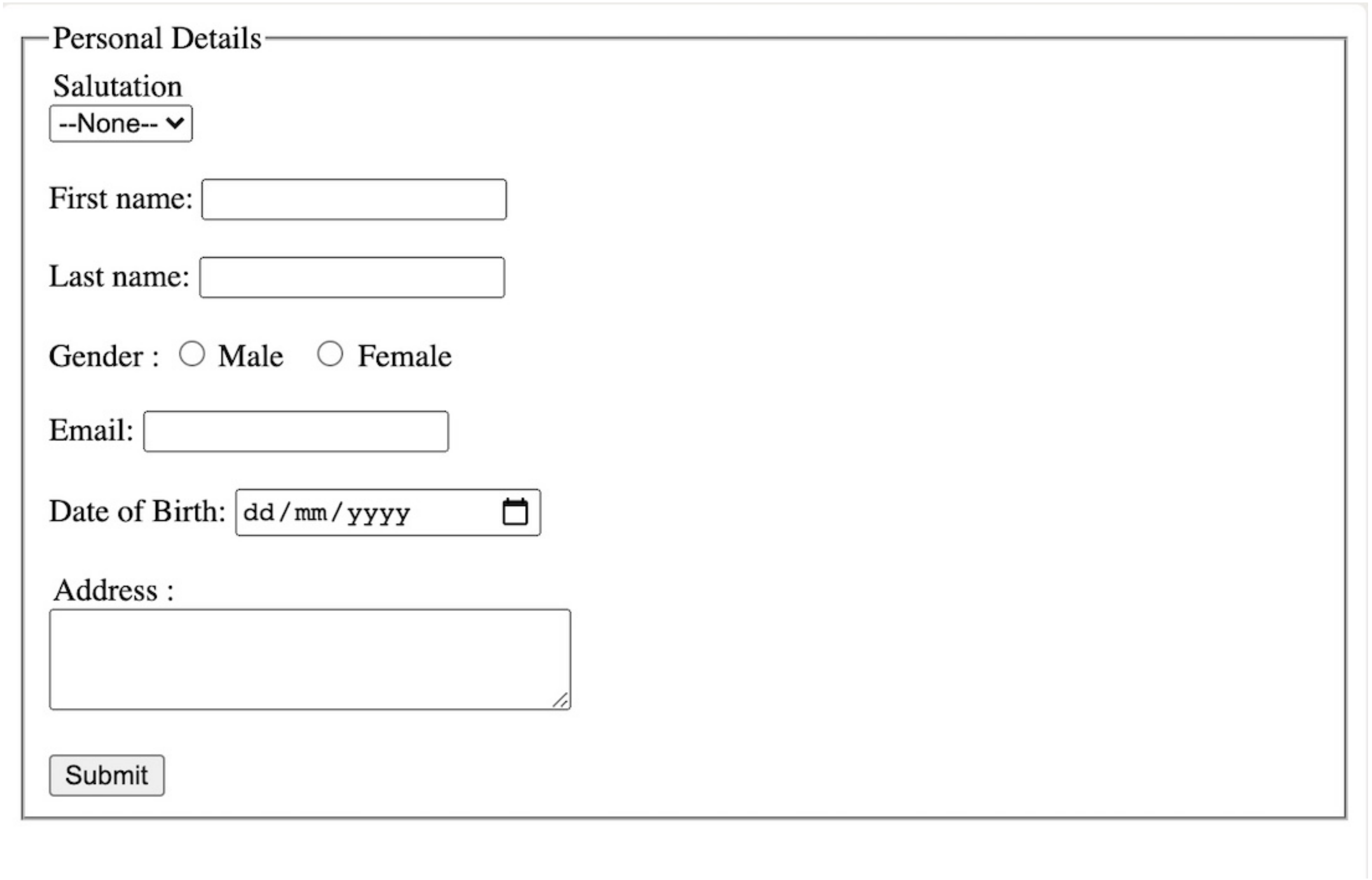
Goals

- **HTML Forms** to collect user input.



HTML Forms

- An **HTML form** is used to **collect user input**.
- However, note that HTML is a **stateless protocol** that means it **CANNOT STORE anything**, and you will **lose the data** on a **page refresh**.
- The **user input** is most often **sent** to a **server for processing, and storing upon a button click**.



Personal Details

Salutation

--None--

First name:

Last name:

Gender : Male Female

Email:

Date of Birth: dd/mm/yyyy

Address :

Submit

HMTL Forms

- The HTML **<form>** element is used to **create** an **HTML form** for user input.

```
<form>
        Form elements - different types input elements such
        as text fields, checkboxes, radio buttons, submit
        buttons, and more.
<form>
```

The <form> Element

- This element can **contain many other elements** as well, including the below:
  - **<input>**
  - **<label>**
  - **<select>**
  - **<button>**
  - **<option>**
  - **<textarea>**
  - **<fieldset>**
  - **<legend>**
  - **<datalist>**
  - **<optgroup>**
  - **……**

A Simple HTML Form Example

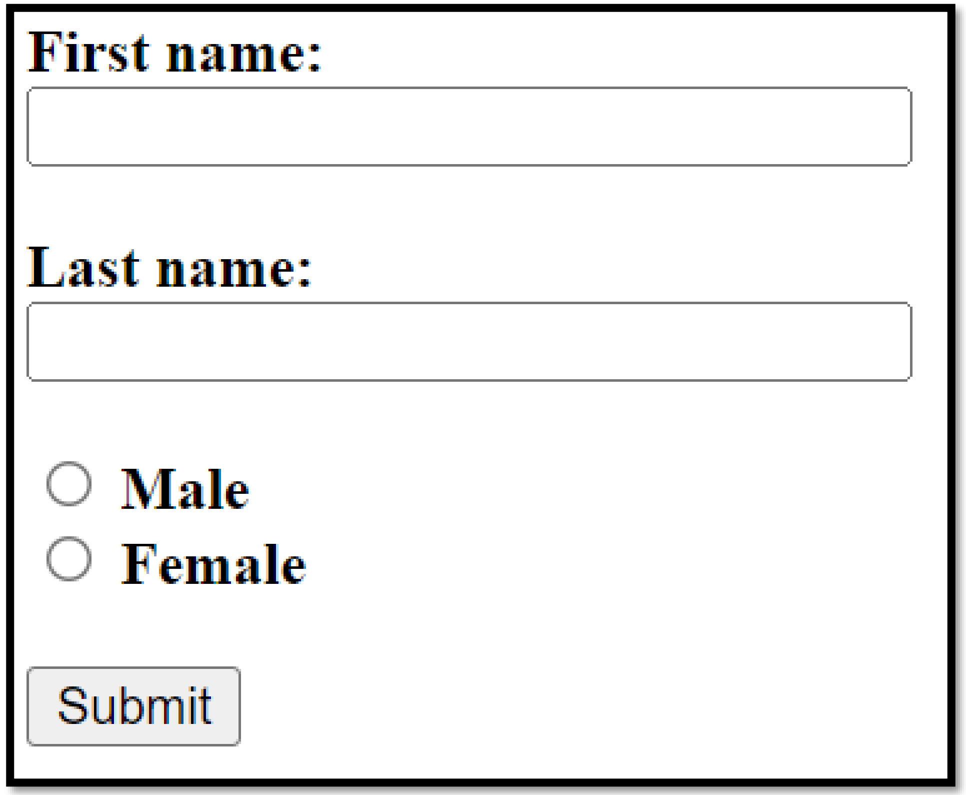
The following HTML code…..

```
<!DOCTYPE html>
<html>
<head>
    <title>The first Input Form</title>
</head>

<body>
    <form action="action_page.php" method="post">
        <b>First name: </b> <br>
        <input type="text" name="firstname" size="30" maxlength="30"> <br><br>
        <b>Last name: </b> <br>
        <input type="text" name="lastname" size="30" maxlength="30"> <br><br>

        <input type="radio" name="sex" value="male"> <b>Male</b> <br>
        <input type="radio" name="sex" value="female"> <b>Female</b> <br><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

Note: A **radio button**, allows a **single value** to be **selected out of multiple choices** when **they have the same name value**.

A Simple HTML Form Example

….will look like this in a browser!



Note that the **default size** of a **text field** is **20 characters**

The action attribute

- The **action attribute** defines the action to be performed when the form is submitted. Usually (i.e., the default way), the form data is sent to a file on the server (referred as the **form-handler**) when the user clicks on the **Submit** button.

```
<form action="action_page.php" method="post">
```

- The **form-handler** is typically a **server page** with a **script** for **processing** the **form's input data**.

- In the example above, the **form data** is **sent** to a file called "**action_page.php**".

Note: The **action** attribute **can be omitted**. Also the **method** attribute can be **omitted**. In this case we will handle submission of the form's data using JavaScript code. In our case we will use **AJAX using JSON format!!!**

The method attribute

- The **method attribute** specifies the **HTTP method** to be used when submitting the form data.
- The form-data can be sent as **URL variables** (with method="get") or as **HTTP post transaction** (method="post")

```
<form action="action_page.php" method="post">
```

Note: Always use **POST** if the form data contains sensitive or personal information! Also the method attribute can be omitted!

GET vs POST

- **Notes on GET:**
  - **Appends** the **form data** to the URL **after** ? in name=value **pairs separated** with & → NEVER use GET to send sensitive data (e.g., passwords)! The submitted form data is visible in the URL!
  - The **length** of a URL is **limited** (2048 characters)
  - GET is **good** for **non-secure data**, like query strings

- **Notes on POST:**
  - **Appends** the **form data** **inside the body** of the **HTTP request** (the submitted form data is **not shown in the URL**)
  - POST has **no size limitations**, and can be used to **send large amounts** of data.

The autocomplete and novalidate attributes

- The **autocomplete attribute** specifies whether a form should have **autocomplete** on or off. When autocomplete="on", the **browser automatically complete** values based on values that the user has **entered before**.

- The **novalidate attribute,** when present, it specifies that the form-data (input) **should not be validated when submitted**.

```
<form action="action_page.php" method="post" autocomplete="on" novalidate>
```

The name attribute (IMPORTANT!!!)

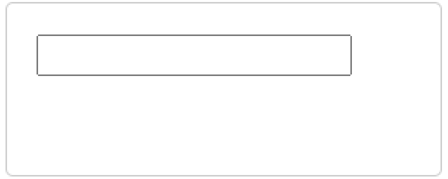
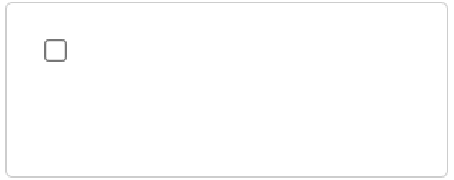
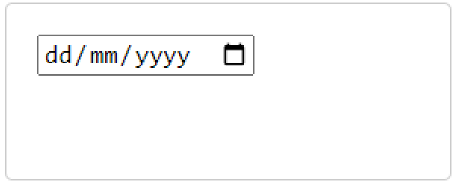
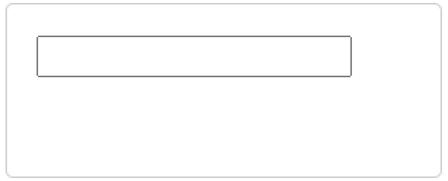
- The **name** attribute specifies the **name** of an **<input> element**.
- The **name attribute** is mainly used to **reference form data after** a **form is submitted to the form handler (which in our case is the php file.**

Note: **Only form elements with** a **name attribute** will **have their values passed** when **submitting a form**.

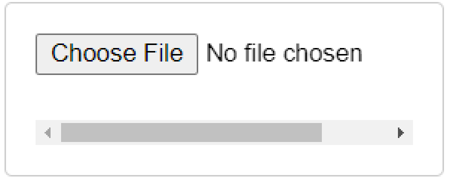
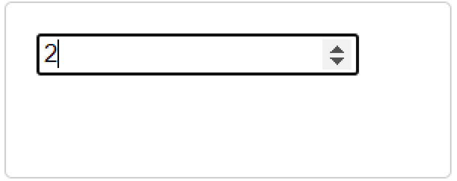
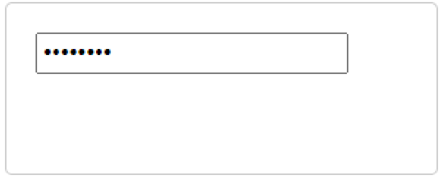
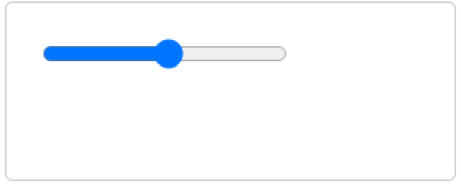
The <input> Element

- The <input> HTML element is used to **create interactive controls** for web-based forms in order to **accept data from the user**;
- How an **<input>** element work varies considerably depending on the value of its **type attribute**.
- Some of the **available types** are described in the next slides.

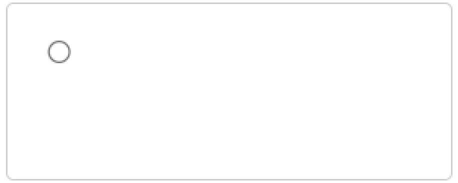
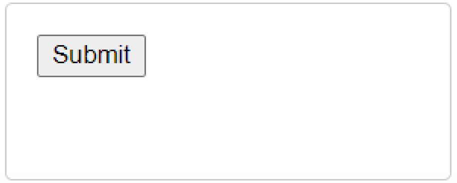
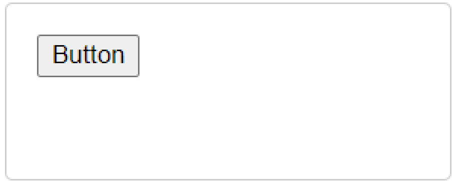
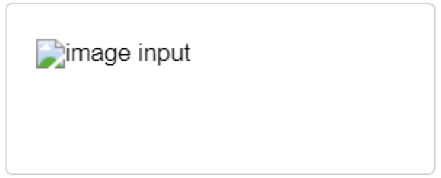
The different types of <input> Element

Type	Description	Basic Examples
text	The default value. A **single-line text field**. Line-breaks are automatically removed from the input value. The **default size** is **20 characters**.	
checkbox	A **check box** allowing single values to be selected/deselected.	
date	A **control** for **entering a date** (year, month, and day, with no time). Opens a date picker or numeric wheels for year, month, day when active in supporting browsers.	
email	A **field for editing an email address**. Looks like a text input, but has validation parameters and relevant keyboard in supporting browsers and devices with dynamic keyboards.	

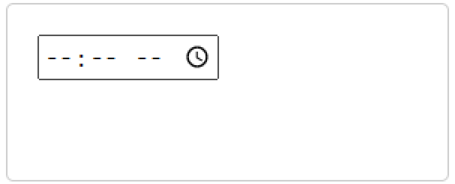
The <input> Element

Type	Description	Basic Examples
file	A **control** that lets the user **select a file or files**. Use the **accept** attribute (e.g., accept="video/*, image/png, image/jpeg") to define the types of files that the control can select.	
number	A **control** for **entering a number**. Displays a **spinner** and adds default validation. Displays a numeric keypad in some devices with dynamic keypads.	
password	A **single-line text field** whose **value** is **obscured**. Will alert user if site is not secure.	
range	A **control** for **entering** a number **whose exact value** is **not important**. Displays as a range widget defaulting to the middle value. Used in conjunction min and max to define the range of acceptable values.	

The <input> Element

Type	Description	Basic Examples
radio	A **radio button**, allowing a **single value** to be **selected out of multiple choices** with **the same name value**.	
submit	A **button** that **submits the form** (this is its **default behavior**).	
button	A **push button** with **no default behavior** **(e.g., we can invoke a JavaScript function to submit the form).**	
image	A **graphical submit button**. Displays an image defined by the **src** attribute. The **alt** attribute displays, if the image **src** is missing.	

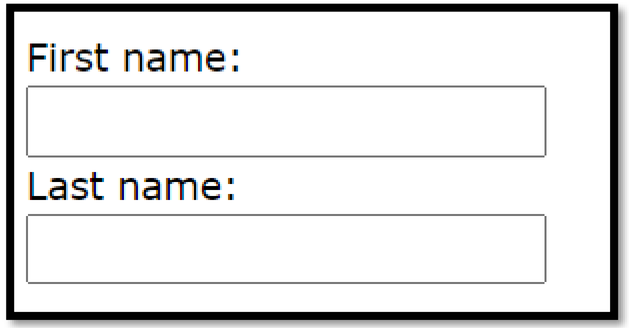
The <input> Element

Type	Description	Basic Examples
time	A control for **entering a time value** with no time zone.	
url	A **field for entering a URL**. Looks like a text input, but has validation parameters and relevant keyboard in supporting browsers and devices with dynamic keyboards.	

Text Fields

- The **<input type="text">** defines a **single-line** input field for text input.

```
<!DOCTYPE html>
<html>

<head>
    <title>HTML Forms</title>
</head>

<body>
    <form action="action_page.php" method="post">
        <label for="fname">First name:</label><br>
        <input type="text" id="fname" name="fname"><br>
        <label for="lname">Last name:</label><br>
        <input type="text" id="lname" name="lname">
    </form>
</body>

</html>
```



This is how the HTML code will be displayed in a browser:

Text Fields – The <label> element

- The **<label>** tag **defines** a **label** for **many HTML form elements** and is **useful** for **screen-reader users** (i.e., people who are blind or have very limited vision); the **screen-reader** will **read out loud the label**.

- The **<label>** element also **help users** who have **difficulty clicking** on **very small regions** (such as radio buttons or checkboxes); For example, when the **user clicks** the **text within the <label> element**, it **toggles** the **radio button/checkbox**.

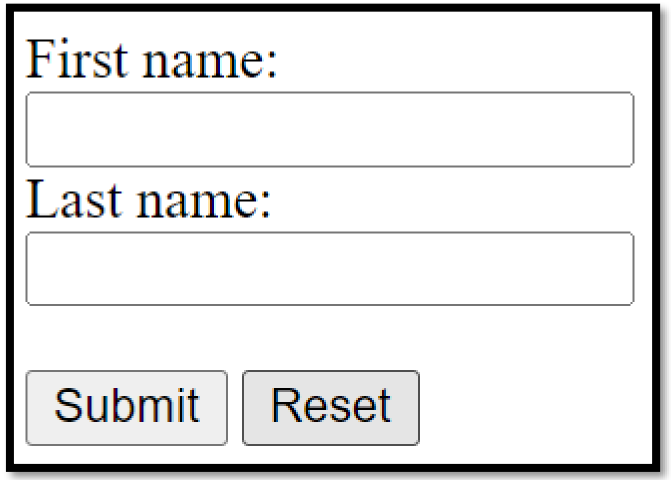
- The **for** attribute of the **<label>** tag **should be equal** to the **id attribute** of the **<input> element** to **bind them together**.

Reset button

- The **<input type="reset">** defines a **reset button** that will **reset** all form values to their **default values.** In this case pressing the Reset button will **clear** the text from First name and Last name fields.

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML Forms</title>
</head>
<body>
<form>
    <label for="fname">First name:</label><br>
    <input type="text" id="fname" name="fname" value=""><br>
    <label for="lname">Last name:</label><br>
    <input type="text" id="lname" name="lname" value=""><br><br>
    <input type="submit" value="Submit">
    <input type="reset">
</form>
</body>

</html>
```

This is how the HTML code will be displayed in a browser:

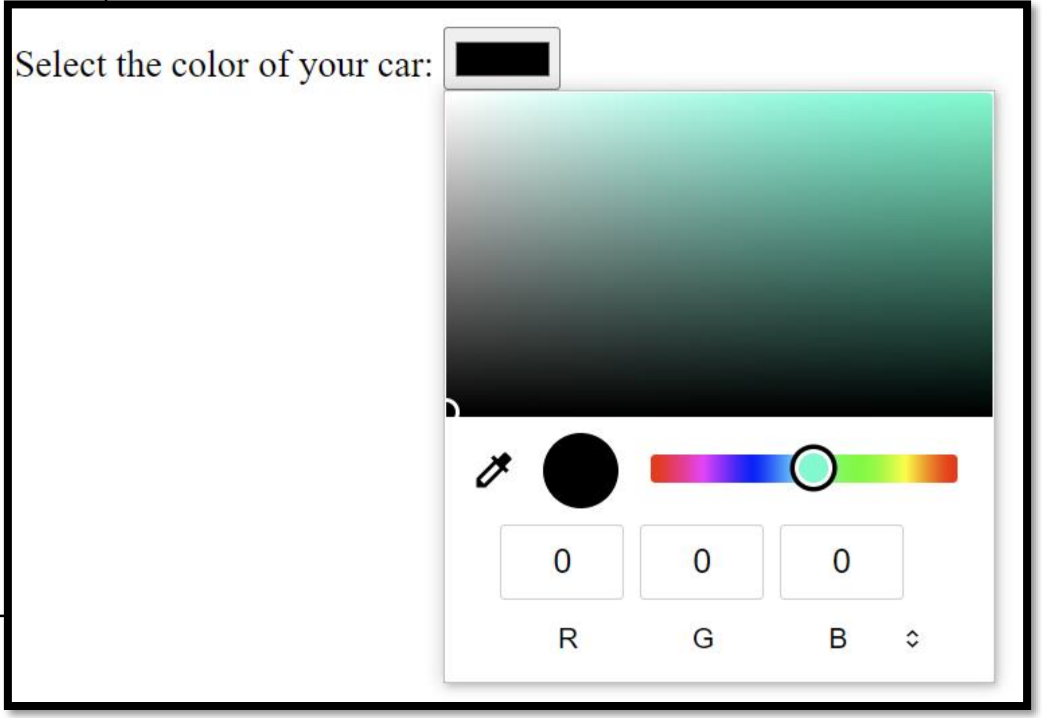


Color

- The **<input type="color">** is used for **input fields** that should **contain a color**. Depending on browser support, a **color picker** can appear in the input field.

```
<!DOCTYPE html>
<html>

<head>
    <title>HTML Forms</title>
</head>

<body>
<form>
    <label for="carcolor">Select the color of your car:</label>
    <input type="color" id="carcolor" name="carcolor">
</form>
</body>

</html>
```

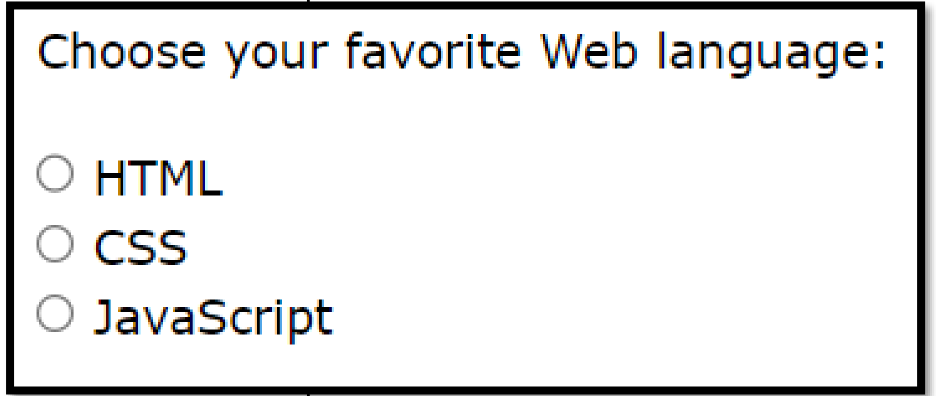
This is how the HTML code will be displayed in a browser:


Select the color of your car:

Radio Buttons

- The **<input type="radio">** defines a **radio button**. Radio buttons let a user **select ONE** of a **limited number** of choices with **the same name value**.

```
<!DOCTYPE html>
<html>

<head>
    <title>HTML Forms</title>
</head>

<body>
    <p>Choose your favorite Web language:</p>
    <form>
        <input type="radio" id="html" name="fav_language" value="HTML">
        <label for="html">HTML</label><br>
        <input type="radio" id="css" name="fav_language" value="CSS">
        <label for="css">CSS</label><br>
        <input type="radio" id="javascript" name="fav_language" value="JavaScript">
        <label for="javascript">JavaScript</label>
    </form>
</body>

</html>
```


Choose your favorite Web language:

○ HTML
○ CSS
○ JavaScript

This is how the HTML code will be displayed in a browser.

Radio Buttons

- In this example, we have a **group of radio buttons** with the **name attribute** set to "**fav_language**".

- To **get the value** of the **selected radio button** using JavaScript, you can use the following code:

This CSS selector targets an **<input>** element that has a **name attribute** with a **value** of "fav_language" and is **currently checked** (i.e., selected by the user).

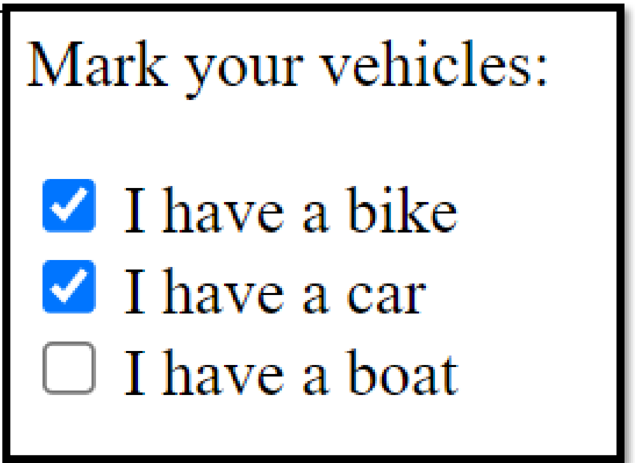
```
const selectedLang = document.querySelector('input[name="fav_language"]:checked').value;
console.log(selectedLang);
```

- In this code, we use the **querySelector** method to find the **checked** radio button by its **name attribute**. We use the **:checked pseudo-class** to **select only** the **checked radio button**. Then, we **retrieve** the **value attribute** of the selected radio button and log it to the console.

Checkboxes

- The **<input type="checkbox">** defines a **check box**. Checkboxes let a user **select ZERO or MORE options** of a **limited number** of choices.

```
<!DOCTYPE html>
<html>

<head>
    <title>HTML Forms</title>
</head>

<body>
    <p>Mark your vehicles:</p>
    <form>
        <input type="checkbox" id="vehicle1" class="veh" name="vehicle1" value="Bike">
        <label for="vehicle1"> I have a bike</label><br>
        <input type="checkbox" id="vehicle2" class="veh" name="vehicle2" value="Car">
        <label for="vehicle2"> I have a car</label><br>
        <input type="checkbox" id="vehicle3" class="veh" name="vehicle3" value="Boat">
        <label for="vehicle3"> I have a boat</label>
    </form>
</body>

</html>
```


Mark your vehicles:

✓ I have a bike
✓ I have a car
☐ I have a boat

This is how the HTML code will be displayed in a browser.

Checkboxes

- In this example, **get the value** of the **selected check boxes** using JavaScript, you can use the following code:

This CSS selector targets all **<input>** elements that has a **type attribute** with a **value** of "checkbox" and with **class attribute equal to** "veh" and are **currently checked** (i.e., selected by the user).

```
const checkboxes = document.querySelectorAll("input[type='checkbox'].veh:checked");

for (let i = 0; i < checkboxes.length; i++){
    console.log(checkboxes[i].id + ':' + checkboxes[i].value);
}
```

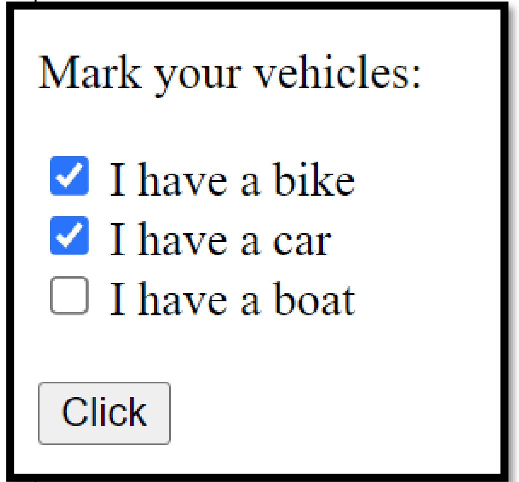
In this code, we use the **querySelectorAll** method to find the all **checked** checkboxes by their **type attribute and class name**. We use the **:checked pseudo-class** to **select only** the **checked checkboxes button**. Then, we **retrieve** the **id** and **value attributes** of the selected checkboxes and log these to the console.

Example

```
function getValues() {
    const checkboxes = document.querySelectorAll("input[type='checkbox’].veh:checked");
    for (let i = 0; i < checkboxes.length; i++) {
        console.log(checkboxes[i].id + ':' + checkboxes[i].value);
    }
}
```

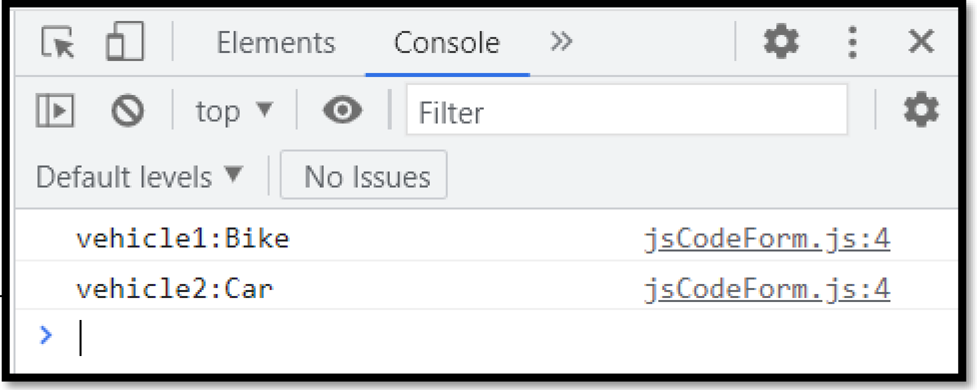
JS/jsCodeForm.js

```
<!DOCTYPE html>
<html>

<head>
    <script src="JS/jsCodeForm.js" defer></script>
    <title>HTML Forms</title>
</head>

<body>
    <p>Mark your vehicles:</p>
    <form>
        <input type="checkbox" id="vehicle1" name="vehicle1" class="veh" value="Bike">
        <label for="vehicle1"> I have a bike</label><br>
        <input type="checkbox" id="vehicle2" name="vehicle2" class="veh" value="Car">
        <label for="vehicle2"> I have a car</label><br>
        <input type="checkbox" id="vehicle3" name="vehicle3" class="veh" value="Boat">
        <label for="vehicle3"> I have a boat</label>
        <br><br>
        <button type="button" onclick="getValues()">Click</button>
    </form>
</body>

</html>
```



Mark your vehicles:

- [x] I have a bike
- [x] I have a car
- [ ] I have a boat

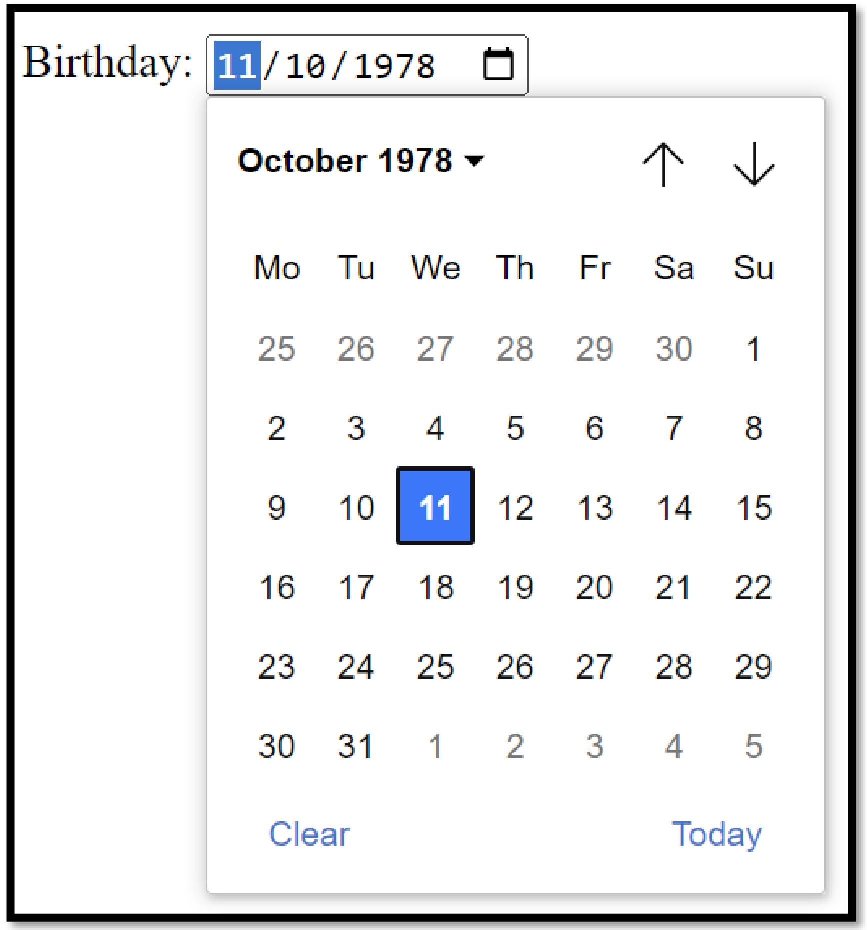


```
vehicle1:Bike                 jsCodeForm.js:4
vehicle2:Car                  jsCodeForm.js:4
```

Date

- The **<input type="date">** is used for **input fields** that should contain a **date** (for example a Birthday field). Depending on browser support, a **date picker** can show up in the input field.

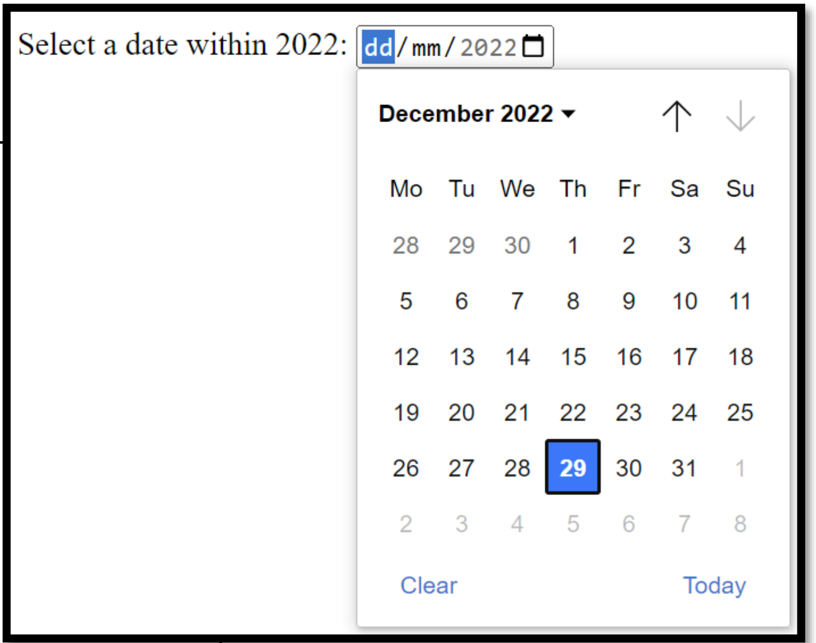
```
<!DOCTYPE html>
<html>

<head>
    <title>HTML Forms</title>
</head>

<body>
<form>
    <label for="birthday">Birthday:</label>
    <input type="date" id="birthday" name="birthday">
</form>
</body>

</html>
```



Date

- You can also use the **min** and **max** **attributes** to add **restrictions to the dates allowed to be selected**. For example, if you want to allow the user to select a date that falls within 2022.

```
<!DOCTYPE html>
<html>

<head>
    <title>HTML Forms</title>
</head>

<body>
<form>
    <label for="datemax">Select a date within 2022:</label>
    <input type="date" id="date" name="date" min="2022-01-01" max="2022-12-31"><br><br>
</form>
</body>

</html>
```

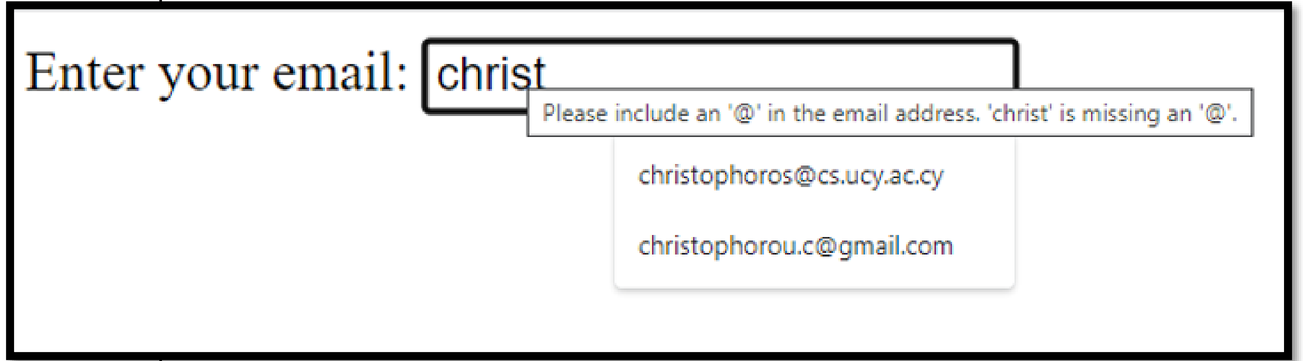


Select a date within 2022: dd/mm/2022

Email

- The **<input type="email">** is used for **input fields** that should contain **an e-mail address**. **Depending** on **browser support**, the e-mail address can be **automatically validated** when **submitted**. Some smartphones recognize the email type, and add ".com" to the keyboard to match email input.

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML Forms</title>
</head>
<body>
<form>
    <label for="email">Enter your email:</label>
    <input type="email" id="email" name="email">
</form>
</body>
</html>
```


Enter your email: christ

Please include an '@' in the email address. 'christ' is missing an '@'.

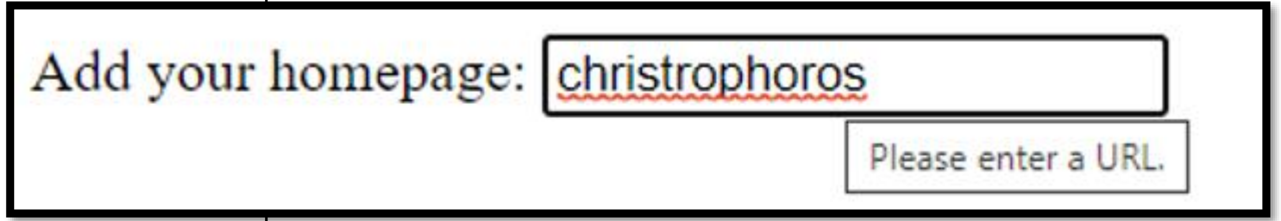
christophoros@cs.ucy.ac.cy

christophorou.c@gmail.com

Url

- The <input **type**="**url**"> is used for input fields that should contain a **URL address**. Depending on browser support, the url field can be **automatically validated** when submitted. Some smartphones recognize the url type, and adds ".com" to the keyboard to match url input.

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML Forms</title>
</head>
<body>
<form>
    <label for="homepage">Add your homepage:</label>
    <input type="url" id="homepage" name="homepage">
</form>
</body>
</html>
```

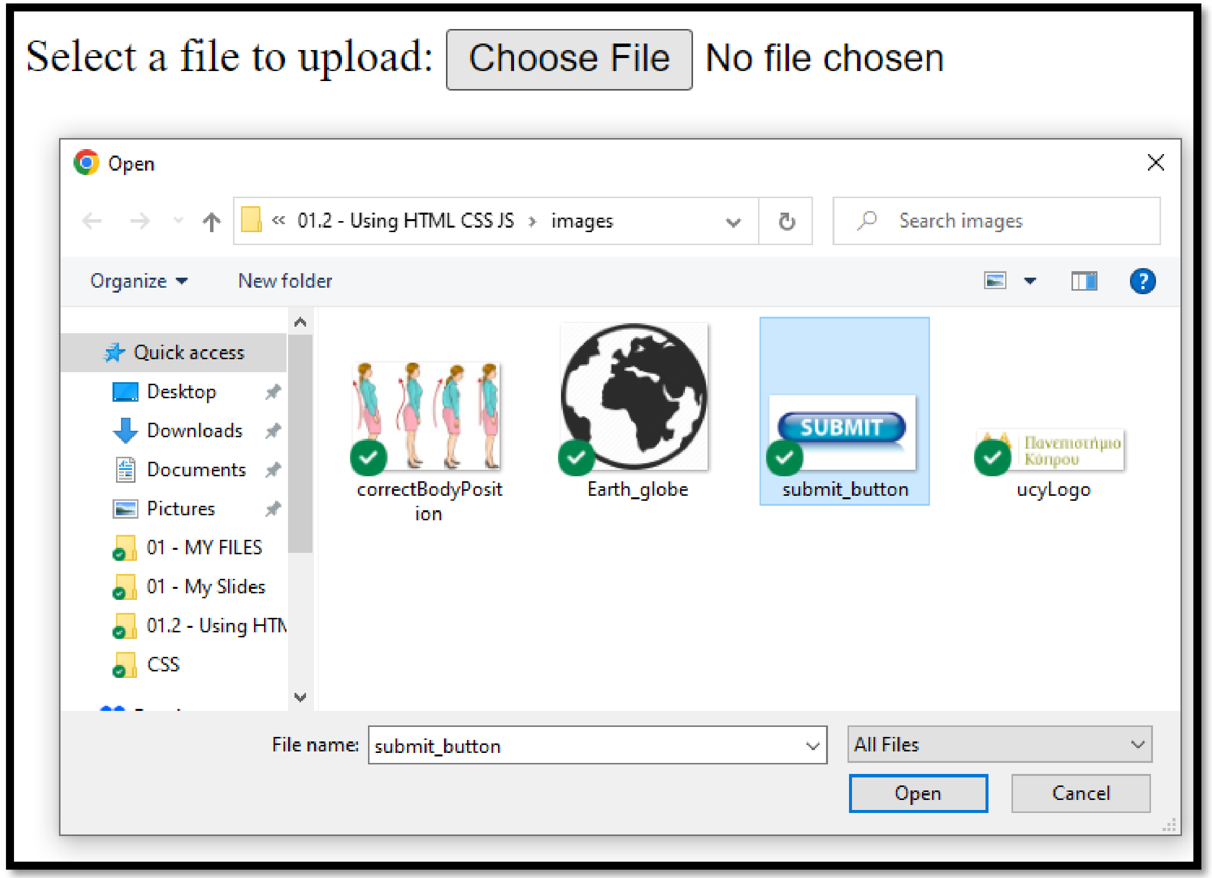

Add your homepage: christrophoros

Please enter a URL.

File

- The **<input type="file">** defines a **file-select field** and a **"Choose File"** button for file uploads.

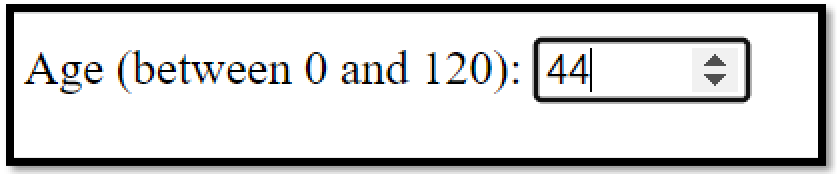
```
<!DOCTYPE html>
<html>

<head>
    <title>HTML Forms</title>
</head>

<body>
<form>
    <label for="myfile">Select a file to upload:</label>
    <input type="file" id="myfile" name="myfile">
</form>
</body>

</html>
```


Select a file to upload: Choose File No file chosen

Number

- The **<input type="number">** **defines** a **numeric input field**. You can also **set restrictions** on **what numbers are accepted**. The following example displays a numeric input field, where you can enter a value only from 0 to 120:

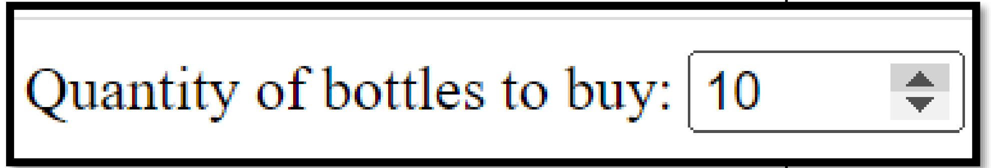
```
<!DOCTYPE html>
<html>
<head>
    <title>HTML Forms</title>
</head>
<body>
<form>
    <label for="age">Age (between 0 and 120):</label>
    <input type="number" id="age" name="age" min="0" max="120">
</form>
</body>
</html>
```


Age (between 0 and 120): 44

Number

- The following example displays a numeric input field, where you can enter a value from 0 to 100 in **steps** of 10 and with **default value** 10:

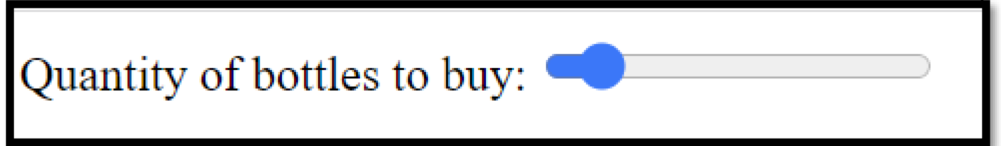
```
<!DOCTYPE html>
<html>

<head>
    <title>HTML Forms</title>
</head>

<body>
<form>
    <label for="quantity">Quantity of bottles to buy:</label>
    <input type="number" id="quantity" name="quantity" min="0" max="100" step="10" value="10">
</form>
</body>

</html>
```


Quantity of bottles to buy: 10

Range

- The **<input type="range">** defines a **control** for entering a number whose exact value is not important (like a slider control). **Default range** is **0** to **100**. However, you can **set restrictions** on what **numbers** are accepted with the **min** and **max** and **step** attributes.

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML Forms</title>
</head>
<body>
<form>
    <label for="quantity">Quantity of bottles to buy:</label>
    <input type="range" id="quantity" name="quantity" min="0" max="100" step="10" value="10">
</form>
</body>
</html>
```


Quantity of bottles to buy:

Note here that NO value IS DISPLAYED on the slider control….

Range – How to Display the Values on the slider control

- Here we **want to show** the **min** and **max** **values** of the slider → We can do this by **including these values as text before** and **after** the **slider tag**.

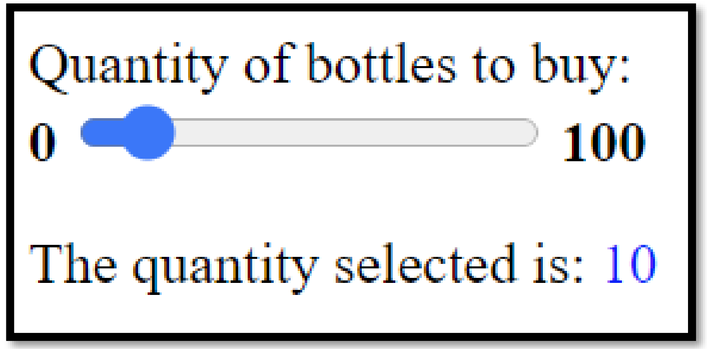
- Also we want **to display the current value** of the slider → We can do this by adding a **<span> element** with **blue text** to **hold the value of the slider** and **add onchange="show_value(this.value)";** in the attributes of <input **type**="**range**">

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML Forms</title>
    <script src="JS/jsCodeForm.js"></script>
</head>

<form>
    <label for="quantity">Quantity of bottles to buy:</label><br>
    <b>0</b>
    <input type="range" id="quantity" name="quantity" min="0" max="100" step="10" value="10" onchange="show_value(this.value);">
    <b>100</b>
    <br>
    <p>The quantity selected is: <span id="slider_value" style="color:blue;"></span> </p>
</form>
</body>
</html>
```

Range – How to Display the Values on the slider control

- The following JavaScript function is used to **get as input the value** of the **slider** when it **changes** and **display it** in the **<span> block**.

```
function show_value(value) {                       JS/jsCodeForm.js
    document.getElementById("slider_value").innerHTML = value;
}
```


Quantity of bottles to buy:
0 100
The quantity selected is: 10

…And this is how the HTML code will be displayed in a browser.

Tel

- The <input **type**="**tel**"> is used for **input fields** that should contain a **telephone number**. Unlike the email and url input types, the **tel input** type value is **not automatically validated** to a particular format before the form can be submitted, **because formats** for **telephone numbers vary so much around the world**. A **pattern** attribute **can be used**.

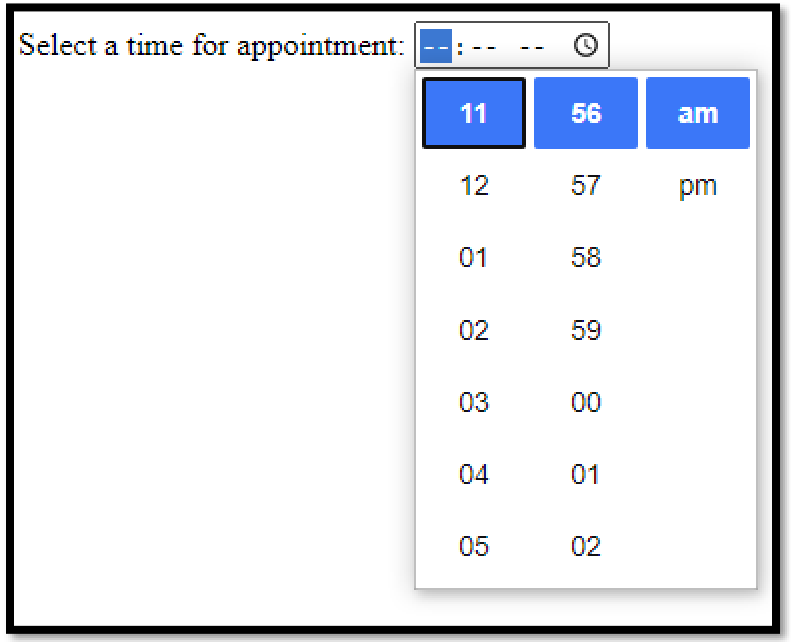
```
<!DOCTYPE html>
<html>
<head>
    <title>HTML Forms</title>
</head>
<body>
<form>
    <label for="phone">Enter your phone number:</label> <br>
    <input type="tel" id="phone" name="phone" pattern="[0-9]{3}-[0-9]{3}-[0-9]{4}" required> <br>
    <small>Format: 123-456-7890</small>
</form>
</body>
</html>
```

The **pattern** attribute **specifies** a **regular expression** that the <input> element's value is **checked against** on **form submission.**

Time

- The <input type="time"> allows the user to select a time (no time zone). Depending on browser support, a time picker can show up in the input field.

```
<!DOCTYPE html>
<html>

<head>
    <title>HTML Forms</title>
</head>

<body>
<form>
    <label for="appt">Select a time for appointment: </label>
    <input type="time" id="appt" name="appt">
</form>
</body>

</html>
```


Select a time for appointment: --:-- --

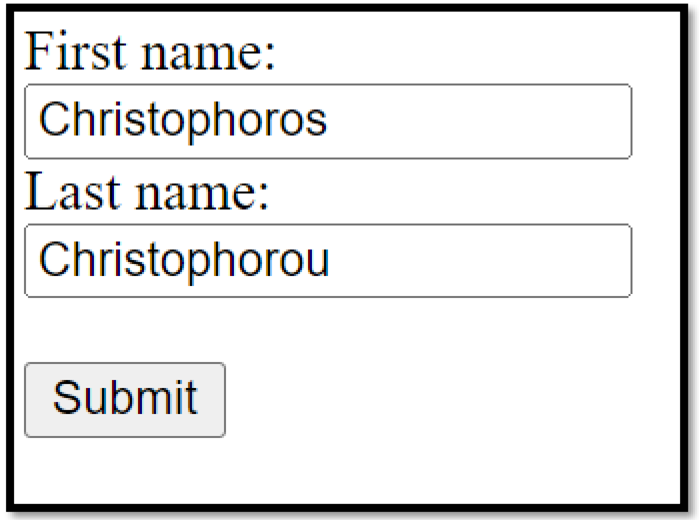
The Submit Button

- The <input **type="submit"**> defines a button **for submitting** the form data to a **form-handler**. The form-handler is typically a file on the server with a script for processing input data (i.e., php) and is specified in the Form's **action** attribute.

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML Forms</title>
</head>
<body>
    <p>Choose your favorite Web language:</p>
    <form action="PHP/action_page.php" method="POST">
        <label for="fname">First name:</label><br>
        <input type="text" id="fname" name="fname" value="Christophoros"><br>
        <label for="lname">Last name:</label><br>
        <input type="text" id="lname" name="lname" value="Christophorou"><br><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

Note: When the **Submit button** is **clicked** a "**submit**" **event is fired** in the <form> element!

We will study PHP in a later lecture, however some simple examples are provided next in this lecture

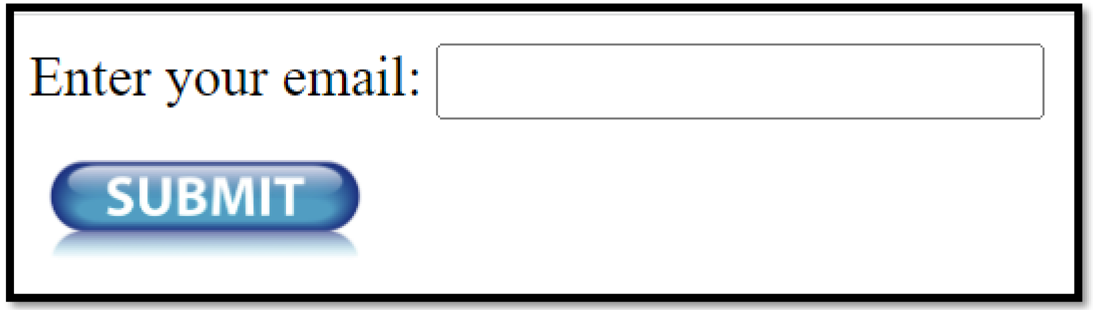

First name:
Christophoros
Last name:
Christophorou

This is how the HTML code will be displayed in a browser.

Image

- The <input **type**="**image**"> is used for defining an image as a **submit button**. You should also specify the **width** and **height** of the image.

```
<!DOCTYPE html>
<html>

<head>
    <title>HTML Forms</title>
</head>

<body>
form>
    <label for="email">Enter your email:</label>
    <input type="email" id="email" name="email"><br>
    <input type="image" src="images/submit_button.jpg" alt="Submit" width="100px" height="auto">
</form>
</body>

</html>
```



Input Restrictions

- Below is a list of some common **input restrictions** that can be set with the input types discuss so far.

Attribute	Description
checked	Specifies that an input field should be pre-selected when the page loads (for type="checkbox" or type="radio")
disabled	Specifies that an input field should be disabled
max	Specifies the maximum value for an input field
maxlength	Specifies the maximum number of character for an input field
min	Specifies the minimum value for an input field
pattern	Specifies a regular expression to check the input value against. For this JavaScript can be used. The pattern attribute works with the following input types: text, date, search, url, tel, email, and password.

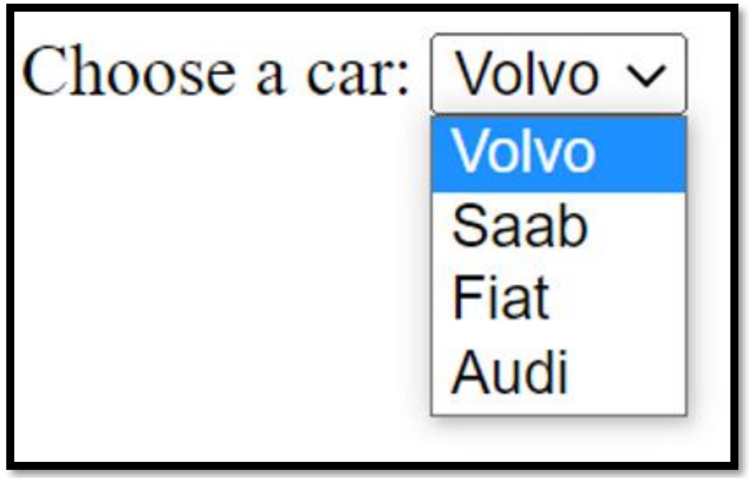
Input Restrictions

Attribute	Description
readonly	Specifies that an input field is read only (cannot be changed)
required	Specifies that an input field is required (must be filled out)
size	Specifies the width (in characters) of an input field
step	Specifies the legal number intervals for an input field
value	Specifies the default value for an input field

The <select> Element

- The **<select>** HTML element is used to **define** a **drop-down list**. The **<option>** element defines an **option that can be selected**. **By default**, the **first item** in the drop-down list **is selected**.

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML Forms</title>
</head>
<body>
    <form>
    <label for="cars">Choose a car:</label>
    <select id="cars" name="cars">
        <option value="volvo">Volvo</option>
        <option value="saab">Saab</option>
        <option value="fiat">Fiat</option>
        <option value="audi">Audi</option>
    </select>
    </form>
</body>
</html>
```

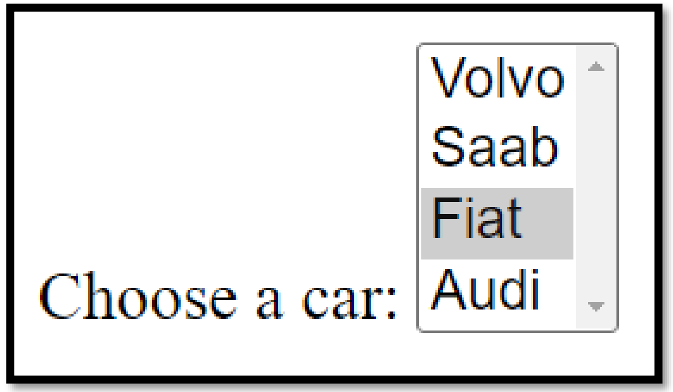


This is how the HTML code will be displayed in a browser.

The <select> Element

- To define a **pre-selected option**, add the **selected attribute** to the option. Also, you can use the **size attribute** to **specify** the **number of visible values!**

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML Forms</title>
</head>
<body>
    <form>
    <label for="cars">Choose a car:</label>
    <select id="cars" name="cars" size="4">
        <option value="volvo">Volvo</option>
        <option value="saab">Saab</option>
        <option value="fiat" selected>Fiat</option>
        <option value="audi">Audi</option>
    </select>
    </form>
</body>
</html>
```

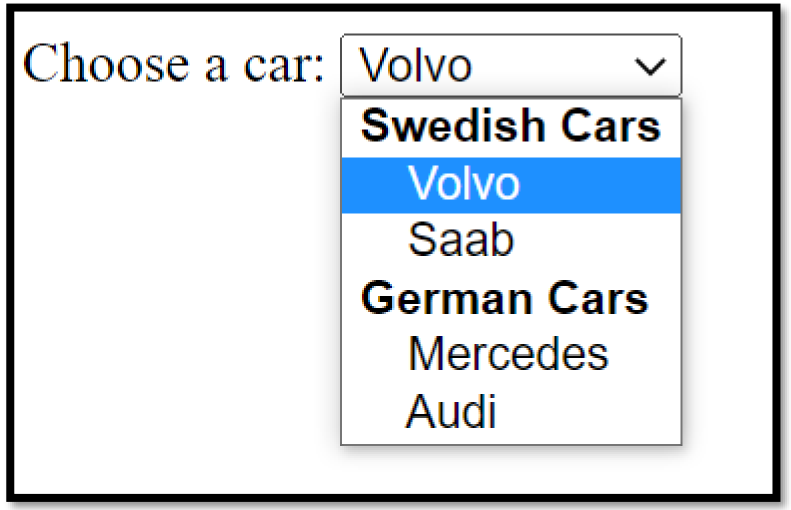


This is how the HTML code will be displayed in a browser.

The <optgroup> Element

- The **<optgroup>** HTML element is used to **group related options** in **<select>** element (for example **group the cars based** on their **country of origin**). If you have a long list of options, groups of related options are easier to handle for a user.

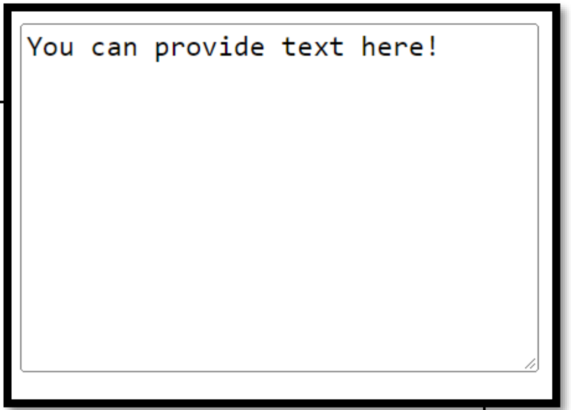
```
<!DOCTYPE html>
<html>
<head>
    <title>HTML Forms</title>
</head>
<body>
<form>
    <label for="cars">Choose a car:</label>
    <select name="cars" id="cars">
        <optgroup label="Swedish Cars">
            <option value="volvo">Volvo</option>
            <option value="saab">Saab</option>
        </optgroup>
        <optgroup label="German Cars">
            <option value="mercedes">Mercedes</option>
            <option value="audi">Audi</option>
        </optgroup>
    </select>
</form>
</body>
</html>
```



This is how the HTML code will be displayed in a browser.

The <textarea> Element

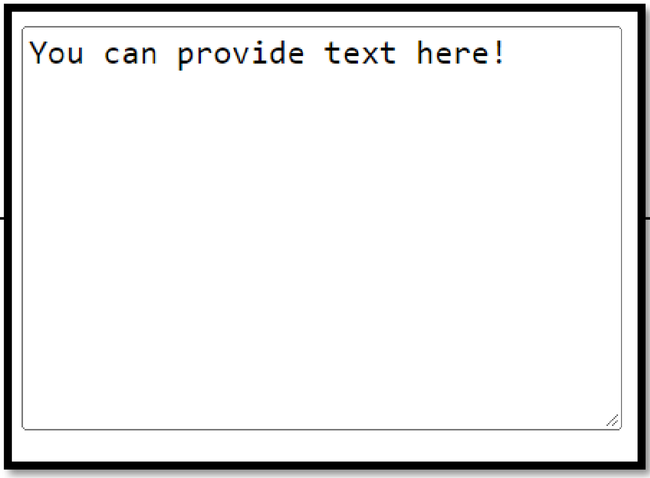
- The **<textarea>** HTML element defines a **multi-line input field** (a text area). It is useful when you want to **allow users** to **enter a sizeable amount of free-form text**, for example a **comment** on a **review** or **feedback form**.
- The **rows** and **cols** **attributes** defines the **visible number** of **lines** and **width** of the text area!


You can provide text here!

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML Forms</title>
</head>
<body>
    <form>
        <textarea name="message" id="textarea1" rows="10" cols="30">You can provide text here!</textarea>
    </form>
</body>
</html>
```

The <textarea> Element

- You can also define the size of the text area (width and heigh) by using CSS


You can provide text here!

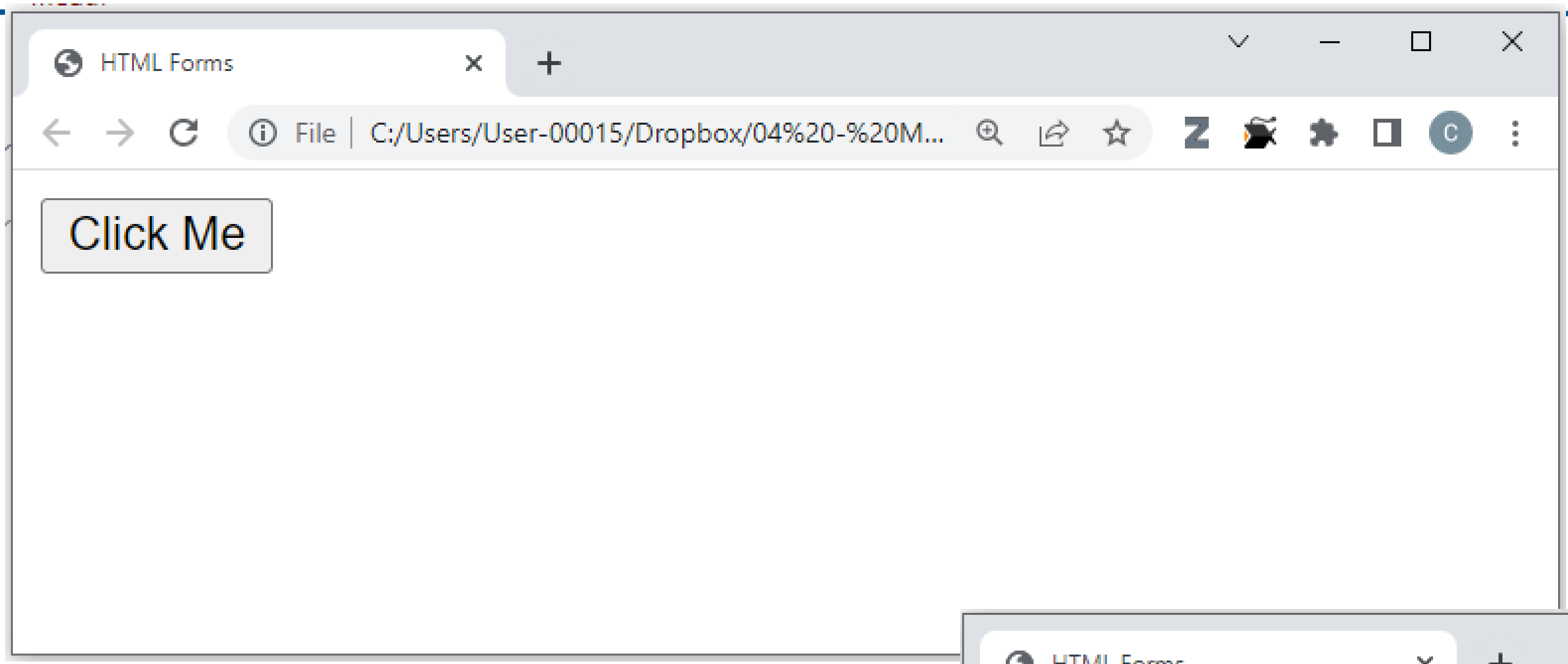
```
<!DOCTYPE html>
<html>
<head>
    <title>HTML Forms</title>
</head>
<body>
    <form>
        <textarea name="message" id="textarea1" style="width:200px; height:600px;">You can provide text here!</textarea>
    </form>
</body>
</html>
```

The <button> Element

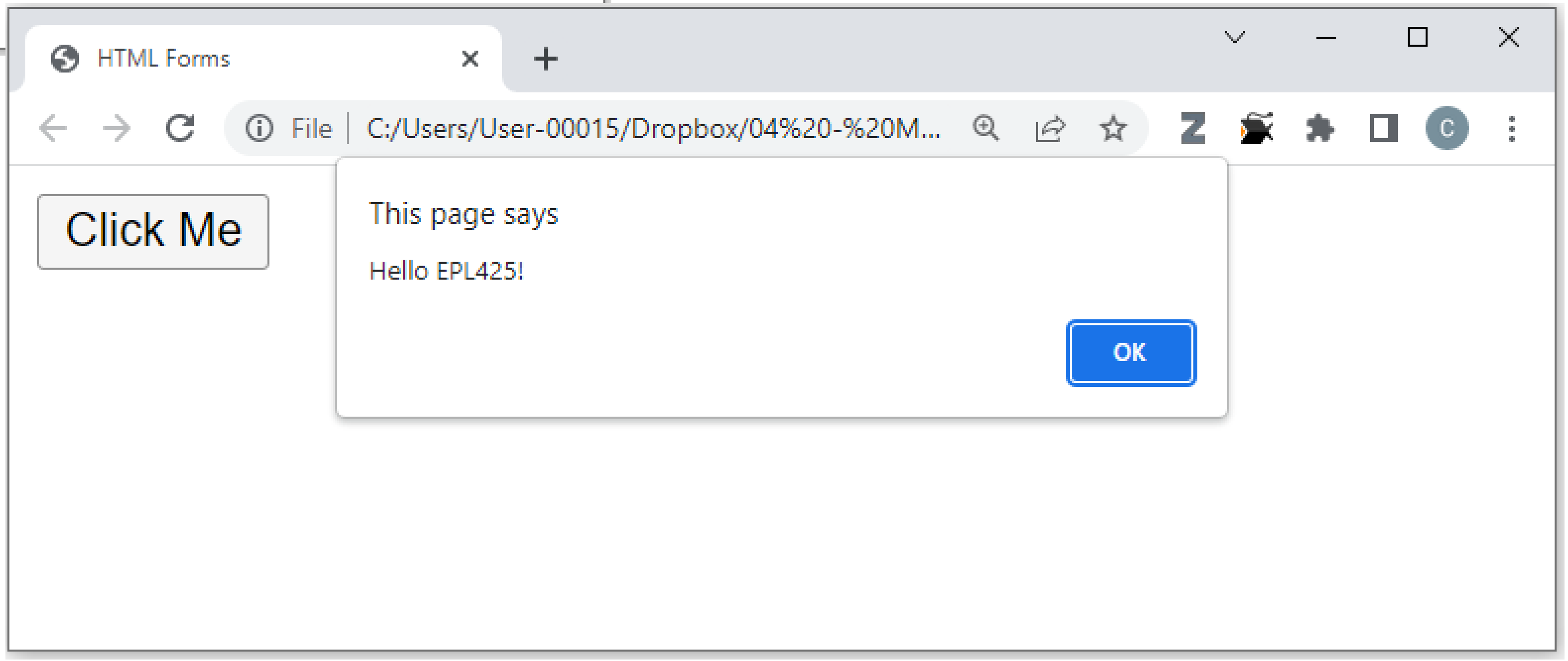
- The **<button>** HTML element defines a **clickable element with no default behavior** that **take actions** **(e.g., we can invoke a JavaScript function to submit the form).**
- Always **specify** the **type attribute** for the **<button> element**. Different browsers may use different default types for the button element.

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML Forms</title>
</head>
<body>
    <form>
        <button type="button" id="btn1" onclick="alert('Hello EPL425!')">Click Me</button>
    </form>
</body>
</html>
```

The <button> Element



This is how the HTML code will be displayed in a browser.



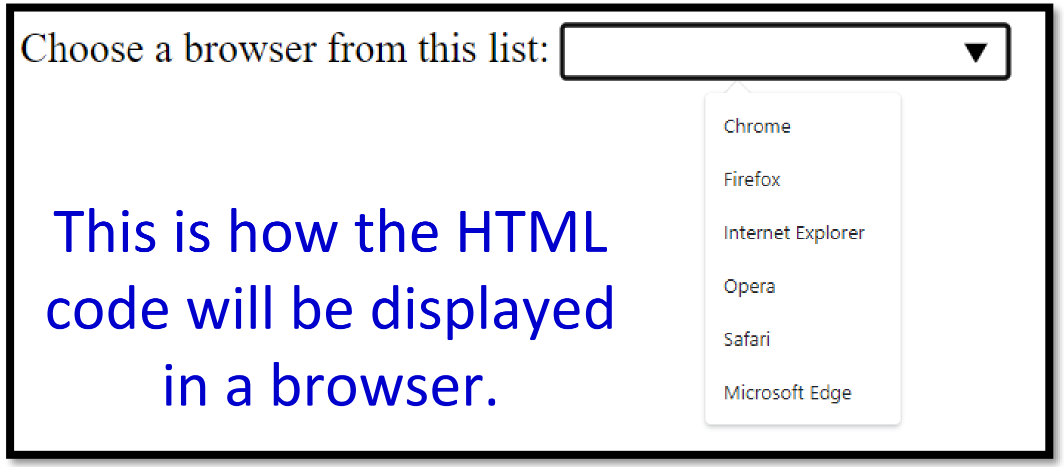
This page says

Hello EPL425!

The <datalist> Element

- The **<datalist>** HTML element **specifies** a **list of pre-defined options** for an **<input>** element. Users will see a **drop-down list** of the **pre-defined options** as they **input data**. The **list attribute** of the **<input>** element, **must refer** to the **id attribute** of the **<datalist>** element.

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML Forms</title>
</head>
<body>
<form>
    <label for="myBrowser">Choose a browser from this list:</label>
    <input list="browsers" id="myBrowser" name="myBrowser" />
    <datalist id="browsers">
        <option value="Chrome"></option>
        <option value="Firefox"></option>
        <option value="Internet Explorer"></option>
        <option value="Opera"></option>
        <option value="Safari"></option>
        <option value="Microsoft Edge"></option>
    </datalist>
</form>
</body>
</html>
```


Choose a browser from this list:

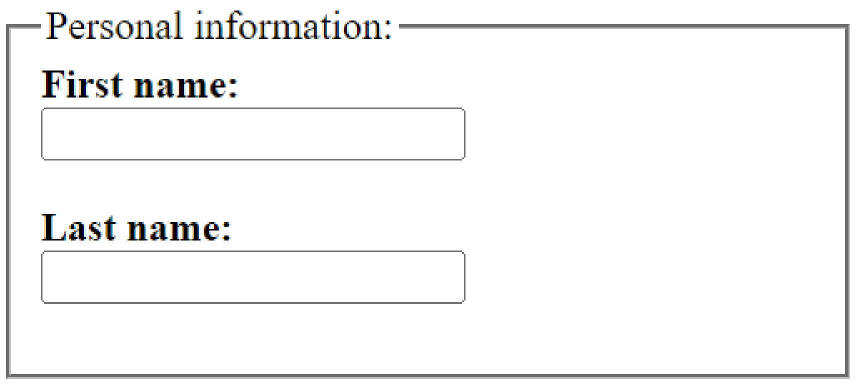
This is how the HTML code will be displayed in a browser.

*The main difference between <select> and <datalist> elements is that **with the <datalist>**, the user **can enter its own input** and **add that as an option**, whereas the <select> tag doesn't provide this feature.*

Grouping Form Data with <fieldset> and <legend> Elements

- The **<fieldset>** element **groups** **related data** in a form while the **<legend>** element defines a **caption** for the **<fieldset>** element.

```
<!DOCTYPE html>
<html>
<title>The first Input Form</title>
<body>
    <form action="action_page.php">
        <fieldset>
            <legend>Personal information:</legend>
            <b>First name: </b> <br>
            <input type="text" name="firstname"> <br><br>
            <b>Last name: </b> <br>
            <input type="text" name="lastname"> <br><br>
        </fieldset>
    </form>
</body>
</html>
```

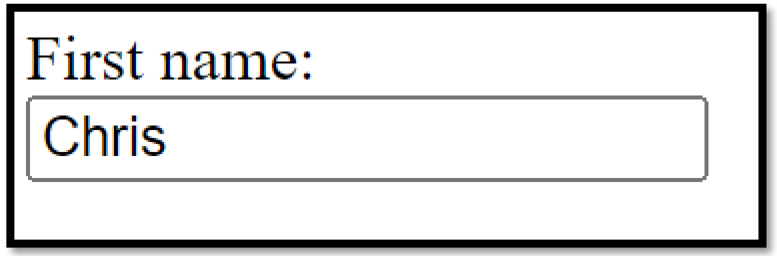


In a browser it will look like this!

HTML Input Attributes

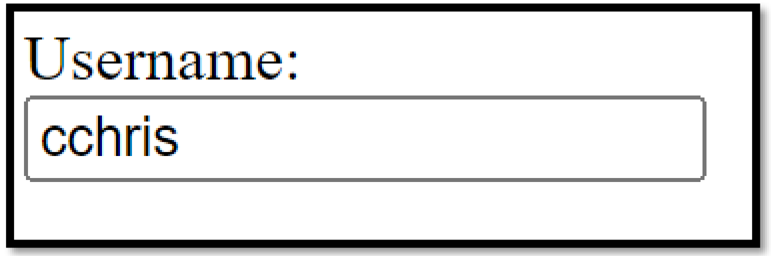
- In the following slides a **list of attributes** that can be **applied** to the HTML <input> elements included in a <form> are provided.

Attribute	Description	Example
value	Specifies an **initial/default value** for an input field	<label for="fname">First name:</label> <input type="text" id="fname" name="fname" **value="Chris"**>


First name:
Chris

HTML Input Attributes

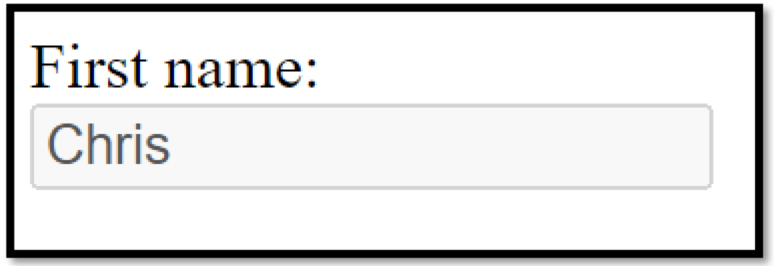
Attribute	Description	Example
readonly	Specifies that an input field is read-only. A read-only input field **cannot be modified** (however, a user can tab to it, highlight it, and copy the text from it). The value of a read-only input field **will be sent** when **submitting the form**!	`<label for="uname">Username:</label><br>` `<input type="text" id="uname" name="uname" value="cchris" **readonly**><br>`


Username:

cchris

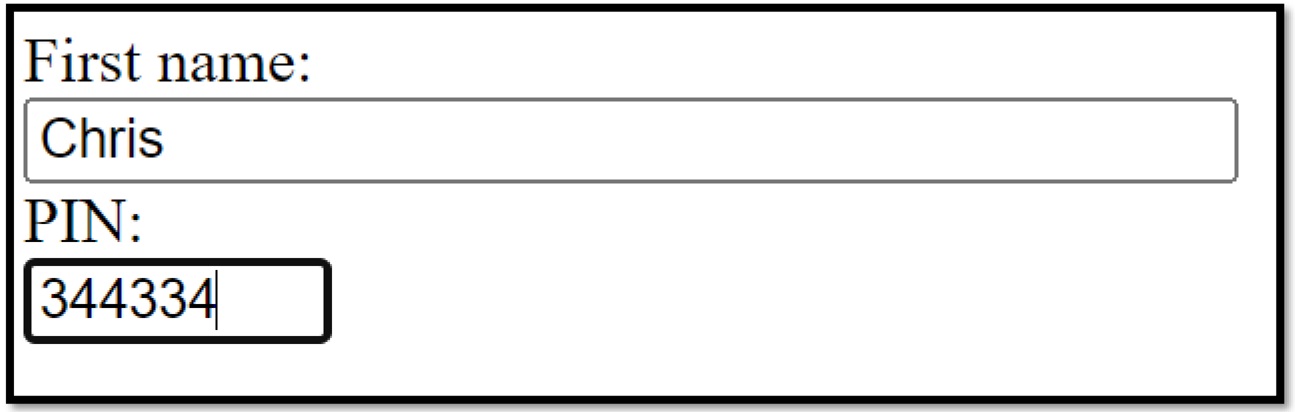
HTML Input Attributes

Attribute	Description	Example
disabled	Specifies that an input field **should be disabled**. A disabled input field is **unusable** and **un-clickable**. The value of a disabled input field **will not be sent** when submitting the form!	`<label for="fname">First name:</label><br>` `<input type="text" id="fname" name="fname" value="Chris" disabled>`


First name:

Chris

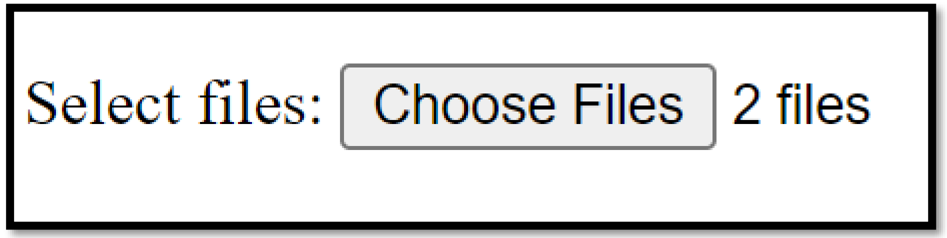
HTML Input Attributes

Attribute	Description	Example
size & **maxlength**	**size specifies** the **visible width, in characters**, of an input field. The default value for size is 20. Works with the following input types: text, search, tel, url, email, and password. **maxlength** specifies the **maximum number of characters allowed** in an **input field**. When a maxlength is set, the input field will **not accept more than the specified number of characters**. However, this attribute does not provide any feedback. So, **if you want to alert the user**, you must write **JavaScript** code.	`<label for="fname">First name:</label><br>`<br>`<input type="text" id="fname" name="fname" value="Chris" size="40"><br>`<br>`<label for="pin">PIN:</label><br>`<br>`<input type="text" id="pin" name="pin" size="6" maxlength="6">`  First name: Chris PIN: 344334

HTML Input Attributes

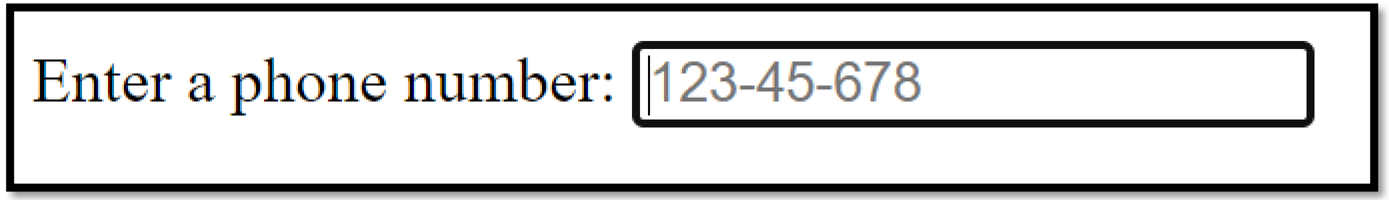
Attribute	Description	Example

multiple	Specifies that the user is **allowed to enter more than one value** in an input field. The **multiple** attribute works with the following input types: **email**, and **file**.	`<label for="files">Select files:</label>`
		`<input type="file" id="files" name="files" multiple>`



HTML Input Attributes

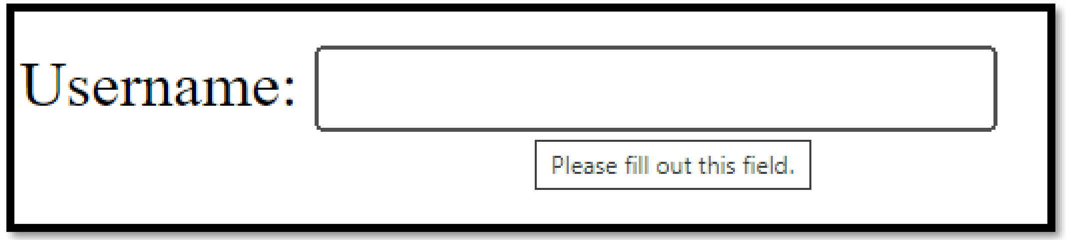
Attribute	Description	Example

placeholder	**Specifies** a **short hint** that **describes** the **expected value of an input field** (a sample value or a short description of the expected format).
	
	The short hint is **displayed** in the input field **before the user enters a value.**
	
	The placeholder attribute works with the following input types: text, search, url, tel, email, and password.
		`<label for="phone">Enter a phone number:</label>`
		`<input type="tel" id="phone" name="phone" placeholder="123-45-678">`
		
		Enter a phone number: 123-45-678

HTML Input Attributes

Attribute	Description	Example

required	Specifies that an input field **must be filled** out before submitting the form.
	
	The required attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.
		`<label for="username">Username:</label>`
		`<input type="text" id="username" name="username" required>`

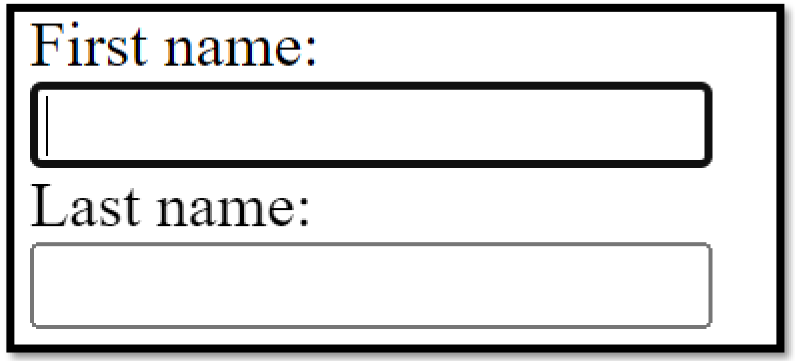


Username:

Please fill out this field.

HTML Input Attributes

Attribute	Description	Example
autofocus	Specifies that an input field should **automatically get focus** when the **page loads**.	`<label for="fname">First name:</label><br>`<br>`<input type="text" id="fname" name="fname" autofocus><br>`



Submitting the form

- **Data transfer** from browser to server can be performed via:
  - HTML form submission (the **default way;** e.g., an example is shown in the next slide)
  - AJAX (Asynchronous JavaScript and XML)
- **Both methods** **send** the **form data** to a **PHP file** on the server side **using** **HTTP messages** but **in a different way**.

The HTTP Basics

HTTP provides **4 basic methods** for CRUD (**C**reate, **R**ead, **U**pdate, **D**elete) operations for resources:

- **GET** - **Request/Retrieve** a resource from the web server (**downloads data**)
- **POST** - **Send/submit/create** a new resource on the web server. Usually used for **uploading data.** This request is usually sent by the browser whenever you submit a form
- **PUT** - **Update/modify** existing resource (or create a new resource)
- **DELETE** - **Delete** an existing resource

The HTTP Basics

Another 2 less commonly used methods:

- **HEAD** - Fetch meta-data of representation only (i.e., a metadata representation)
- **OPTIONS** - Check which HTTP methods a particular resource supports.

HTTP GET Request Message Example – No Query parameters

www.chris.com/EPL425/getStudents.php

Resource URI

Protocol version

Request line
(GET, POST commands)

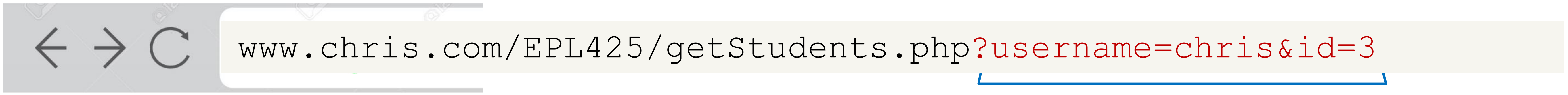
Header lines

Carriage return, line feed
indicates end of headers

Entity-Body:

```
GET /EPL425/getStudents.php HTTP/1.1
Host: www.chris.com
User-agent: Mozilla/4.0
Connection: close
Accept-language: *

```

HTTP GET Request Message Example


www.chris.com/EPL425/getStudents.php?username=chris&id=3

Query Parameters passing for data filtering

Resource URI

Request line (GET, POST commands)

```
GET /EPL425/getStudents.php?username=chris&id=3 HTTP/1.1
Host: www.chris.com
User-agent: Mozilla/4.0
Connection: close
Accept-language: *
```

Header lines

Entity-Body
Empty when GET method is used

When using HTTP GET, the data is sent as a series of key=value pairs, appended to the URL after ?

HTTP POST Request Message Example

www.chris.com/EPL425/getStudents.php

Resource URI

Request line
(GET, POST commands)

```
POST /EPL425/getStudents.php HTTP/1.1
Host: www.chris.com
User-agent: Mozilla/4.0
Connection: close
Accept-language: *

username=chris&id=3
```

Header lines

Entity-Body
With POST method data are included here

When using HTTP POST, the data is sent as a series of key=value pairs, similar to HTTP GET. However, instead of appending the parameters to the URL, they are included in the message body.

Install PHP and start PHP Server

- Since PHP is running on the server, before continuing you have to **make sure** that **PHP** is **installed** on your PC and the **System Variables Path** is **set correctly**!

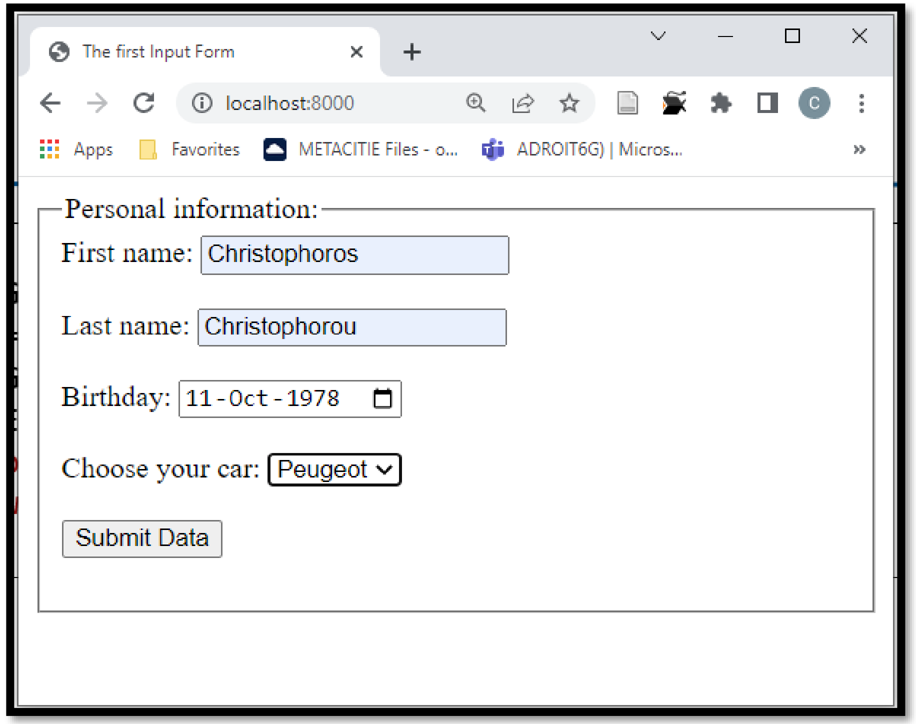
- For this, follow these steps to download and configure PHP on you PC: https://www.geeksforgeeks.org/how-to-install-php-in-windows-10/

- Then, go to the folder of your web site and **start the PHP server** using the following command.

```
PHP –S  localhost:8000
```

```html
<!DOCTYPE html>
<html>

<head>
    <title>The first Input Form</title>
</head>

<body>
    <form id="form1" action="PHP/submit_data.php" method="GET">
        <fieldset>
            <legend>Personal information:</legend>
            <label>First name: </label>
            <input type="text" name="firstname" id="firstname" /> <br><br>
            <label>Last name: </label>
            <input type="text" name="lastname" id="lastname" /> <br><br>
            <label for="birthday">Birthday:</label>
            <input type="date" id="birthday" name="birthday" /> <br><br>
            <label for="cars">Choose your car:</label>
            <select id="cars" name="cars">
                <option value="volvo">Volvo</option>
                <option value="peugeot">Peugeot</option>
                <option value="BMW">BMW</option>
                <option value="audi">Audi</option>
            </select> <br><br>
            <input type="submit" value="Submit Data" /> <br><br>
        </fieldset>
    </form>
</body>

</html>
```

HTML form submission – The default way using "GET"



Personal information:

First name: Christophoros

Last name: Christophorou

Birthday: 11-Oct-1978

Choose your car: Peugeot

Submit Data

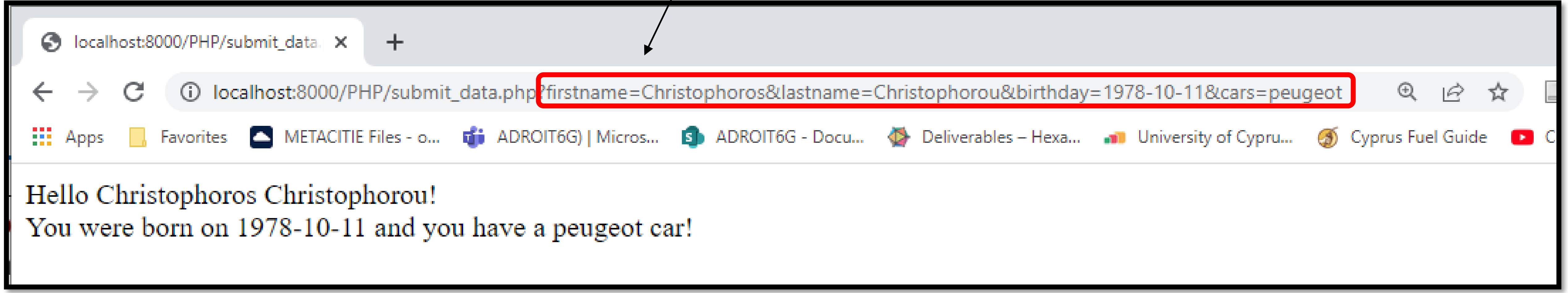
HTML form submission – The default way using "GET"

In PHP, **$_GET** is actually a **superglobal associative array** that contains the **values of variables** passed to the current script via the **URL parameters (query string).** To access these you can use the syntax **$_GET['variable_name']**, where "variable_name" is the **name** of the variable you want to access.

PHP/submit_data.php

```php
<?php
$name = $_GET["firstname"];
$epitheto = $_GET["lastname"];
$date = $_GET["birthday"];
$car = $_GET["cars"];
echo "Hello $name $epitheto! <br>";
echo "You were born on $date and you have a $car car!"
?>
```

After the Submit Data button is clicked



localhost:8000/PHP/submit_data.php?firstname=Christophoros&lastname=Christophorou&birthday=1978-10-11&cars=peugeot

```
Hello Christophoros Christophorou!
You were born on 1978-10-11 and you have a peugeot car!
```

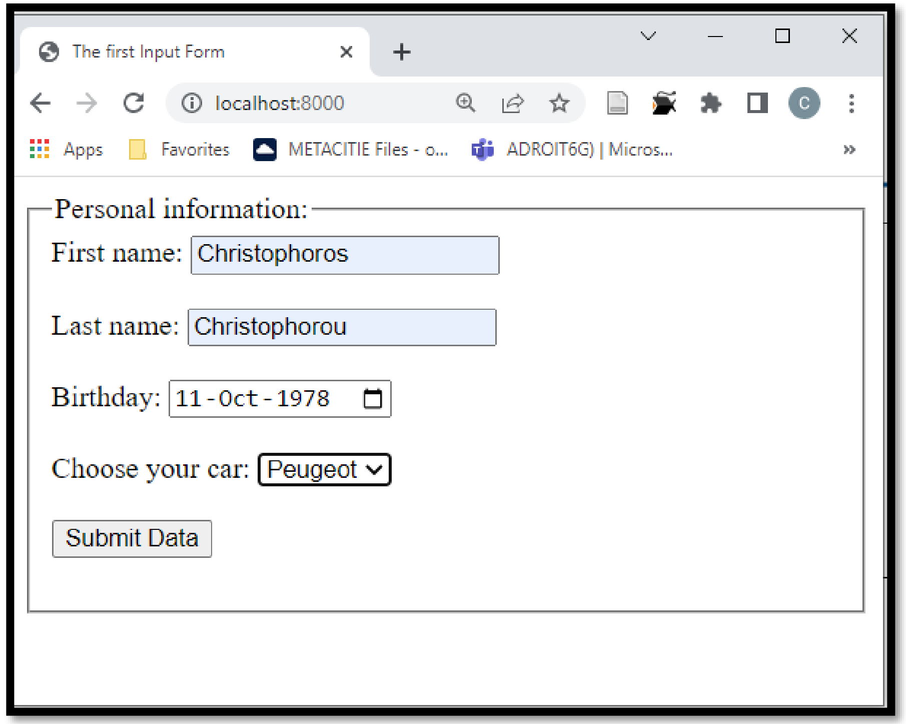
Parenthesis …… Arrays in PHP

- **Associative arrays** in PHP are arrays where **each element** is **identified** by a **key** instead of an index number. The key can be a string or a number. Here's an example:

```php
<?php
$person = array("name" => "Christophoros", "age" => 44, "height" => 1.75);

echo $person["name"]; // Outputs "Christophoros"
echo $person["age"]; // Outputs 44
echo $person["height"]; // Outputs 1.75
?>
```

```html
<!DOCTYPE html>
<html>

<head>
    <title>The first Input Form</title>
</head>

<body>
    <form id="form1" action="PHP/submit_data.php" method="post">
        <fieldset>
            <legend>Personal information:</legend>
            <label>First name: </label>
            <input type="text" name="firstname" id="firstname" /> <br><br>
            <label>Last name: </label>
            <input type="text" name="lastname" id="lastname" /> <br><br>
            <label for="birthday">Birthday:</label>
            <input type="date" id="birthday" name="birthday" /> <br><br>
            <label for="cars">Choose your car:</label>
            <select id="cars" name="cars">
                <option value="volvo">Volvo</option>
                <option value="peugeot">Peugeot</option>
                <option value="BMW">BMW</option>
                <option value="audi">Audi</option>
            </select> <br><br>
            <input type="submit" value="Submit Data" /> <br><br>
        </fieldset>
    </form>
</body>

</html>
```

HTML form submission – The default way using "POST"



Personal information:

First name: Christophoros

Last name: Christophorou

Birthday: 11-Oct-1978

Choose your car: Peugeot

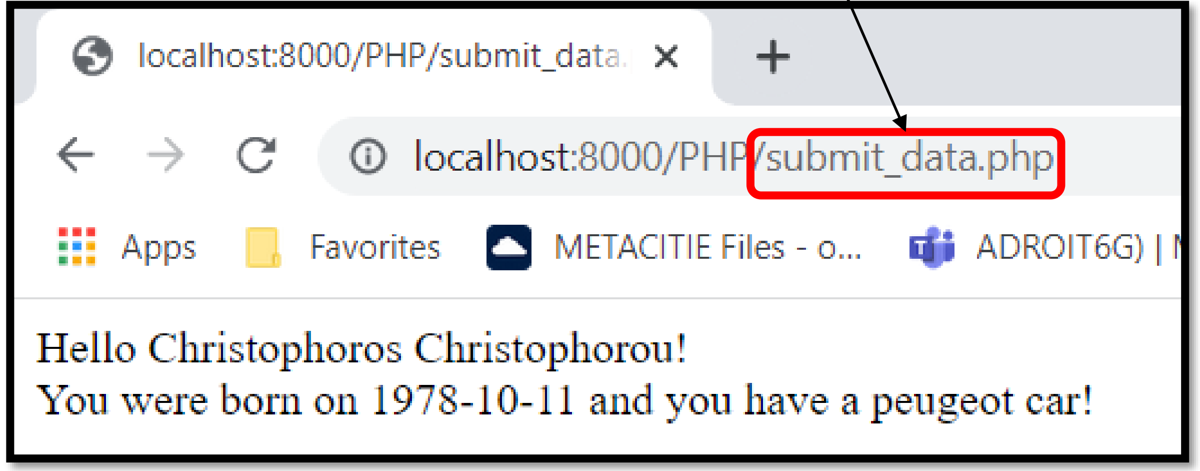
Submit Data

HTML form submission – The default way using "POST"

The key difference in this example is that the **method attribute** of the form element is set to "**POST**" instead of "GET". This **causes the form data** to be **submitted** using the HTTP POST method, which **sends the data in the body** of the **HTTP request** instead of as URL parameters.
Also we access the form data using the **$_POST** superglobal **associative array** instead of the **$_GET array**, since the data is being sent in the **entity body** instead of as URL parameters.

PHP/submit_data.php

```
<?php
$name = $_POST["firstname"];
$epitheto = $_POST["lastname"];
$date = $_POST["birthday"];
$car = $_POST["cars"];
echo "Hello $name $epitheto! <br>";
echo "You were born on $date and you have a $car car!"
?>
```


localhost:8000/PHP/submit_data
localhost:8000/PHP/submit_data.php
Apps
Favorites
METACITIE Files - o...
ADROIT6G) |

Hello Christophoros Christophorou!
You were born on 1978-10-11 and you have a peugeot car!

After the Submit Data button is clicked

```
POST /PHP/submit_data.php  HTTP/1.1
Host: localhost:8000
User-agent: Mozilla/4.0
Connection: close
Accept-language: *

firstname=Christophoros&lastname=Ch
ristophorou&birthday=1978-10-
11&cars=pegeout
```

Entity-Body
With POST method data are included here

Ερωτήσεις?