

# HTML 5 Forms

# Introduction

- ❖ HTML 5.0 introduces a new set of controls we can use in our forms.
- ❖ The web browser's support for these new controls varies from one browser to another.
- ❖ The support for the HTML old controls still exists.
- ❖ The new HTML 5.0 controls are scriptable similarly to the old ones.

# Introduction

- ❖ The HTML 5.0 specification covers the controls' functional behavior only. It doesn't cover their appearance or display.
- ❖ W3C maintains an up-to-date list of all the HTML 5 form controllers currently supported together with the future ones.

# The `email` Input Type

- ❖ Using the `<INPUT . . . >` controller with `email` as its type we will get a text field that accepts email addresses only.

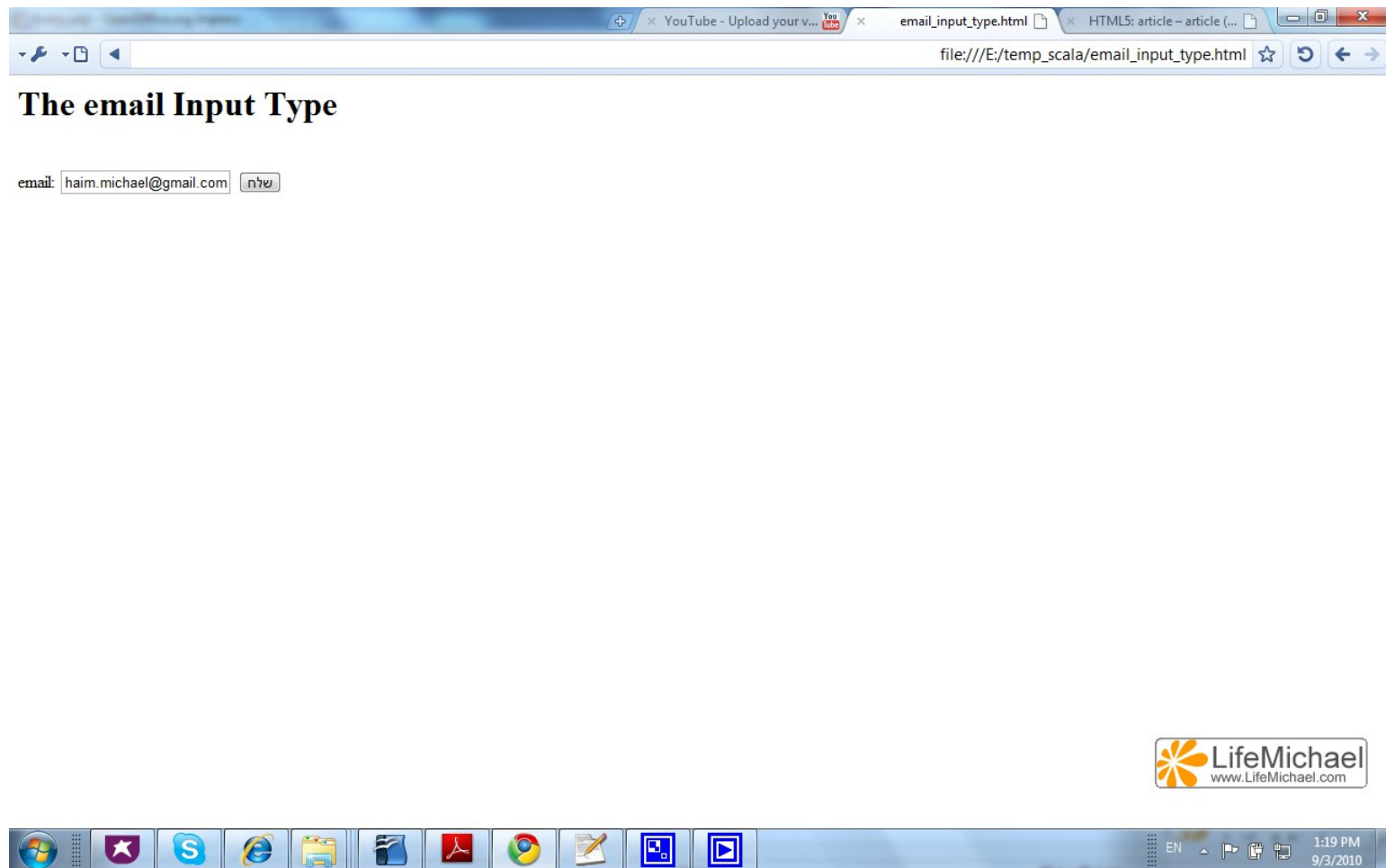
# The email Input Type

```
<h1>The email Input Type</h1>
```

```
<form action="http://www.abelski.com/courses/html5/demo.php" method="get">  
<br>email: <input type="email" name="user_email_address"/>  
<input type="submit"/>  
</form>
```



# The email Input Type



# The `tel` Input Type

- ❖ Using the `<INPUT . . . >` controller with `tel` as its type we will get a text field that accepts telephone numbers only.

# The tel Input Type

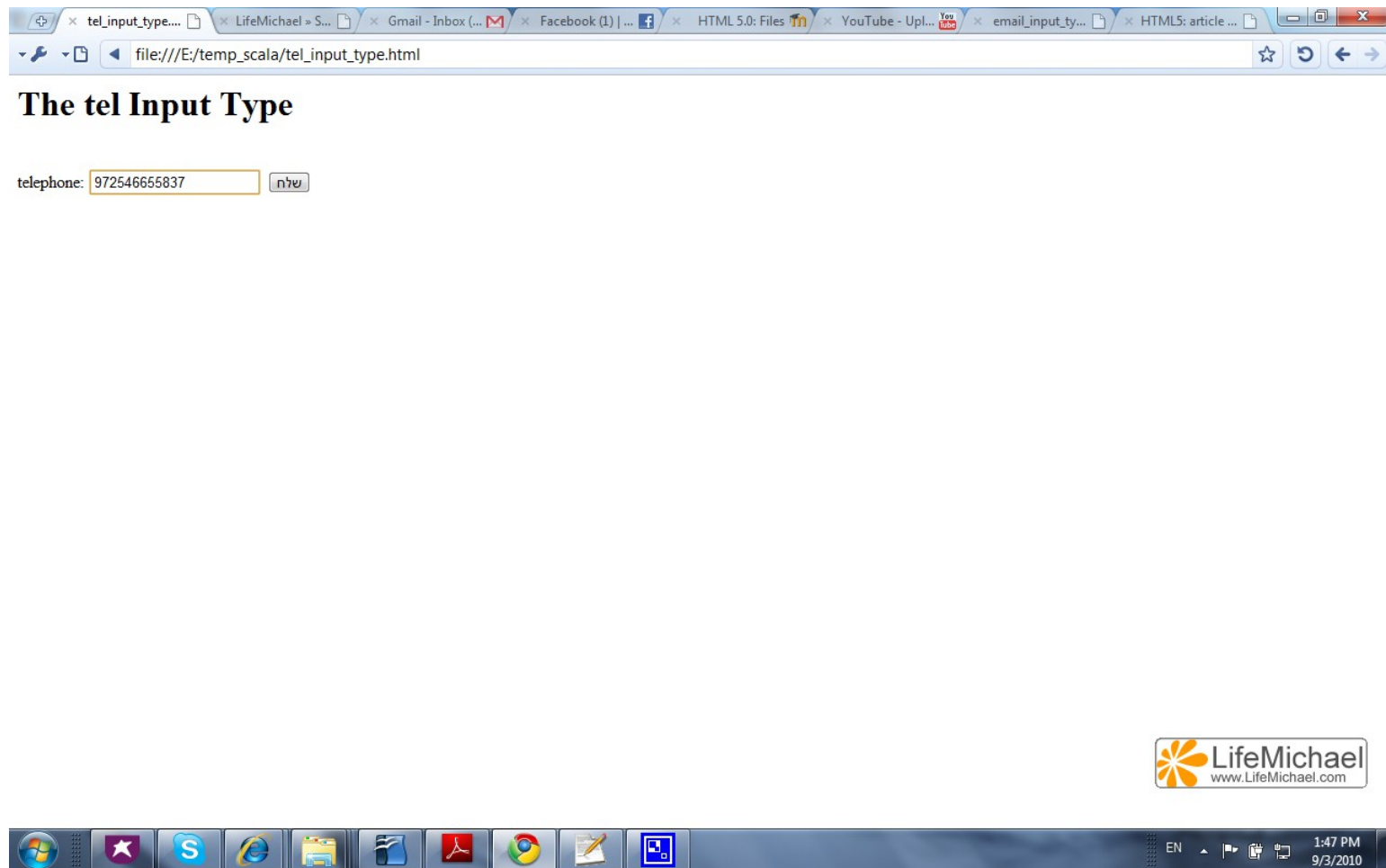
```
<h1>The tel Input Type</h1>
```

```
<form action="http://www.abelski.com/courses/html5/teldemo.php" method="get">  
<br>telephone: <input type="tel" name="user_tel"/>  
<input type="submit"/>  
</form>
```

The support for each one of the special input types varies from one browser to another. Some browsers will provide an hint so the user knows he should enter a telephone number. Others will prevent entering something that canont be a phone number.



# The tel Input Type



# The `range` Input Type

- ❖ Using the `<INPUT . . . >` controller with `range` as its type together with specifying `min` and `max` attributes we will get a controller that accepts values in the specified range only.

# The range Input Type

```
<h1>The range Input Type</h1>
```

```
<form  action="http://www.abelski.com/courses/html5/rangedemo.php"  
      method="get">
```

```
<br>enter your age (in the range 18..90):
```

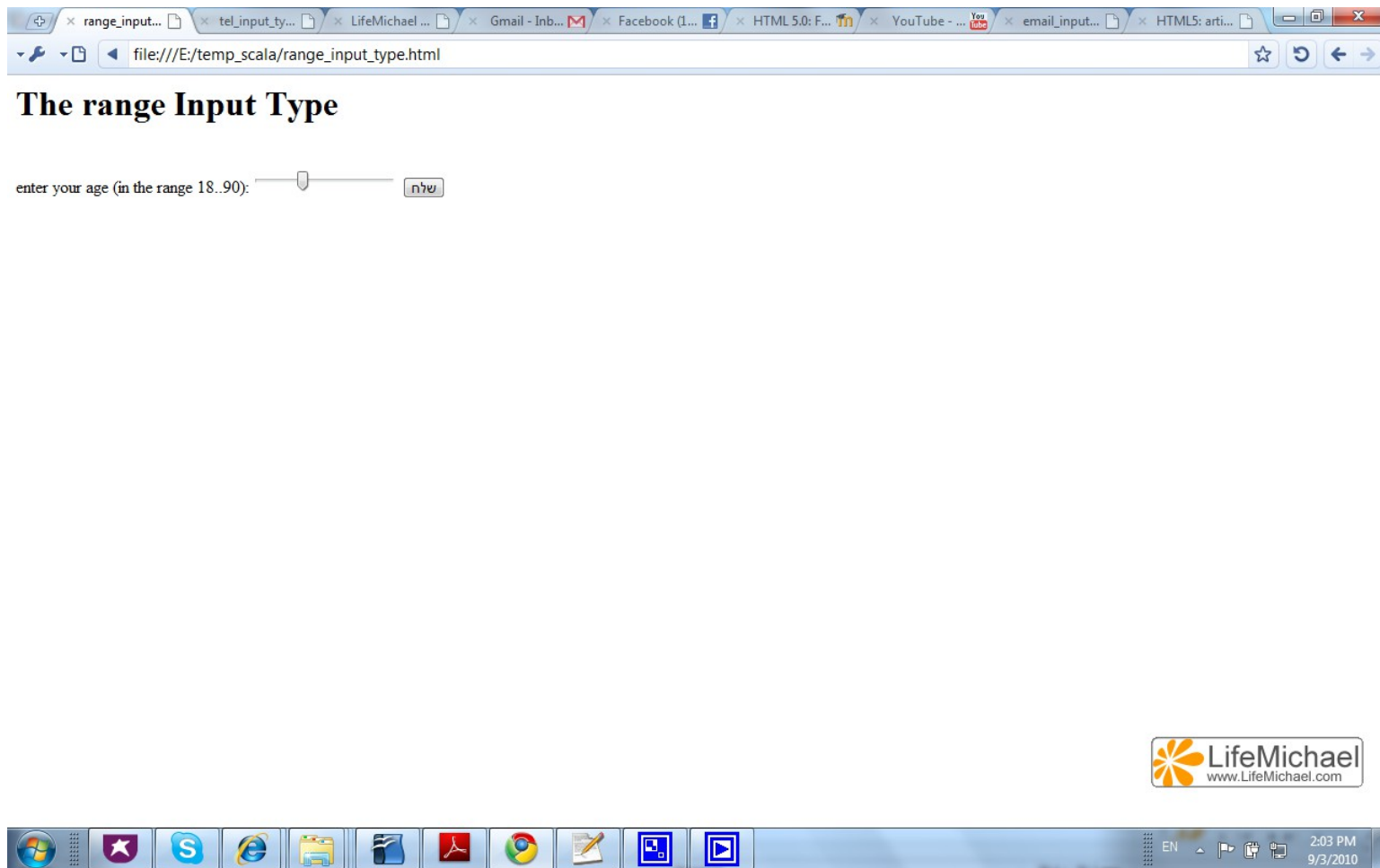
```
<input type="range" name="age" min="18" max="90" />
```

```
<input type="submit"/>
```

```
</form>
```



# The range Input Type



# The range Input Type

- ❖ We can use JavaScript in order to present the selected value.

# The range Input Type

```
<h1>The range Input Type</h1>
```

```
<form    action="http://www.abelski.com/courses/html5/rangedemo.php"  
        method="get">
```

```
<br>enter your age (in the range 18..90):
```

```
<input type="range" name="age" onchange="showTheSelectedValue(this.value)"/>
```

```
<input type="submit"/>
```

```
</form>
```

```
<script type="text/javascript">
```

```
function showTheSelectedValue(number)
```

```
{
```

```
    document.getElementById("selected_age").innerHTML = number;
```

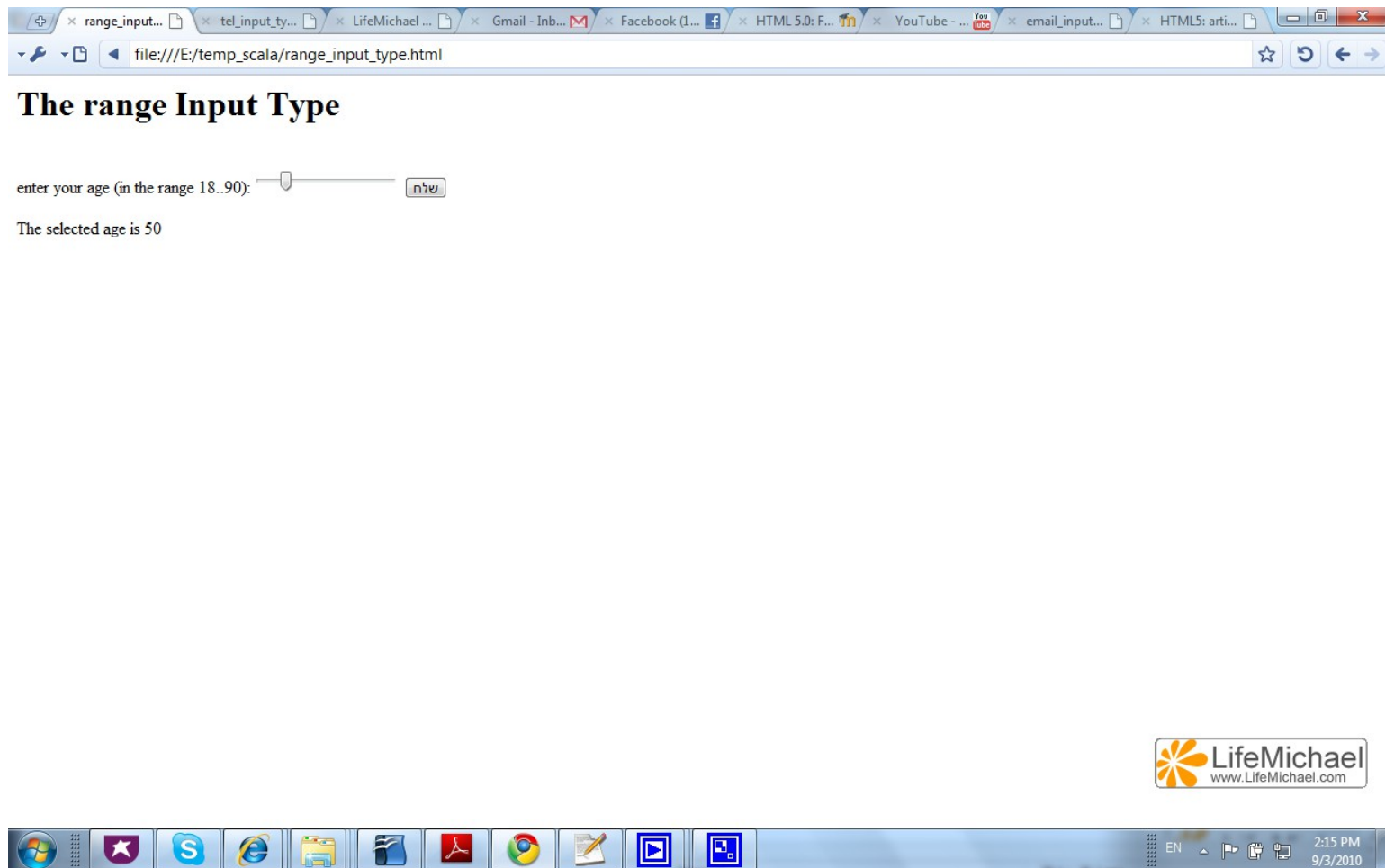
```
}
```

```
</script>
```

```
The selected age is <span id="selected_age">50</span>
```



# The range Input Type



# The `range` Input Type

- ❖ We can use add the `step` attribute in order to specify the size of the steps in which the slider changes its value.



# The range Input Type

```
<h1>The range Input Type</h1>
```

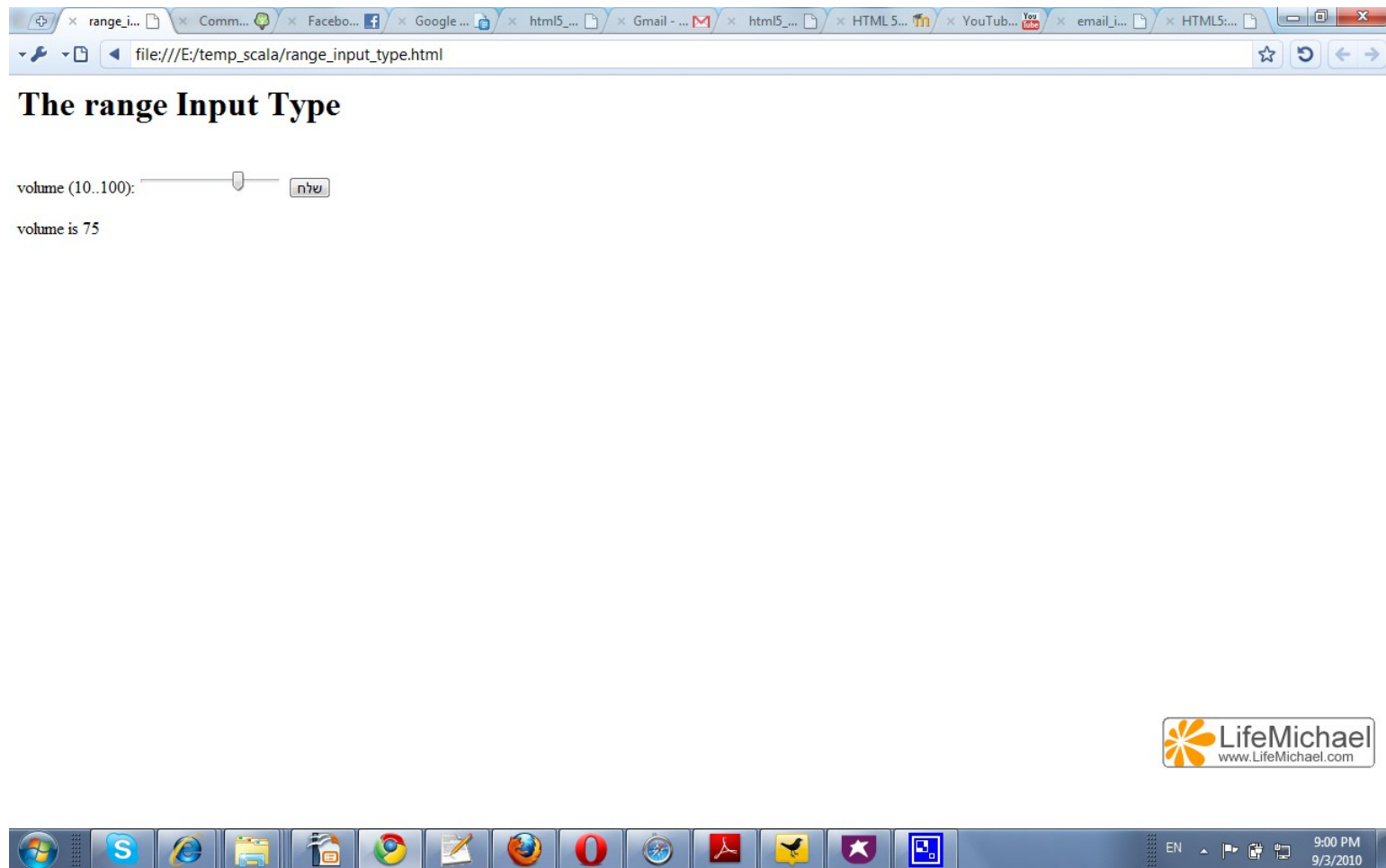
```
<form action="http://www.abelski.com/courses/html5/rangedemo.php" method="get">  
<br>volume (10..100):  
<input type="range" name="age" onchange="showTheSelectedValue(this.value)"  
min="10" max="100" step="5"/>  
<input type="submit"/>  
</form>
```

```
<script type="text/javascript">  
function showTheSelectedValue(number)  
{  
    document.getElementById("selected_age").innerHTML = number;  
}  
</script>
```

```
volume is <span id="selected_age">50</span>
```



# The range Input Type



# The `number` Input Type

- ❖ Specifying `number` as the type of our controller we will get a text field through which the user will be able to enter numeric values only.

The user will be allowed to enter valid numeric values only. Trying to enter non numeric values the user won't be able to send the entered data to the server.

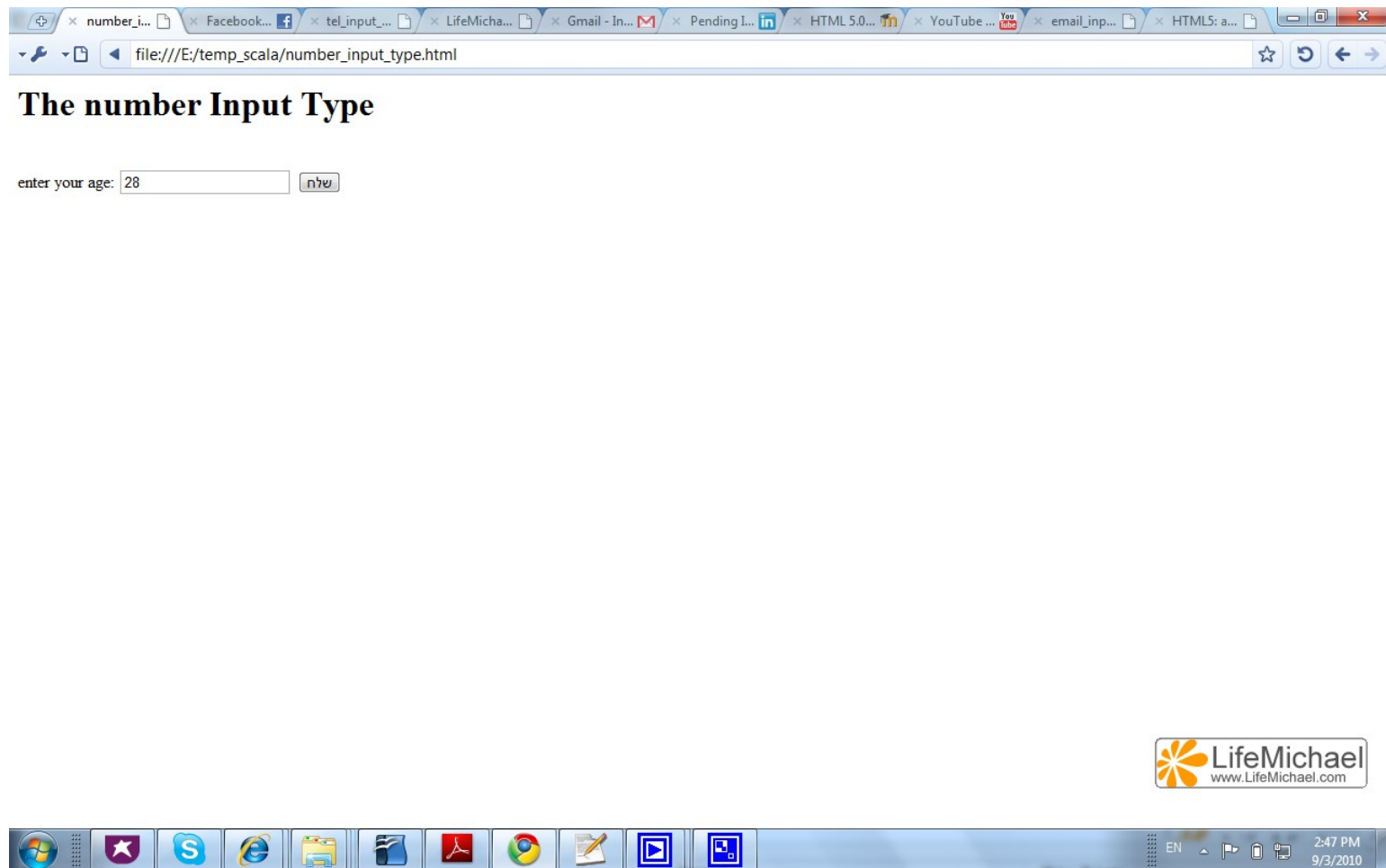
# The number Input Type

```
<h1>The number Input Type</h1>
```

```
<form  action="http://www.abelski.com/courses/html5/rangedemo.php"  
      method="get">  
<br>enter your age: <input type="number" name="age"/>  
<input type="submit"/>  
</form>
```



# The number Input Type



# The `color` Input Type

- ❖ Specifying `color` as the type of our input controller we will get a control that allows us to pick a color only.

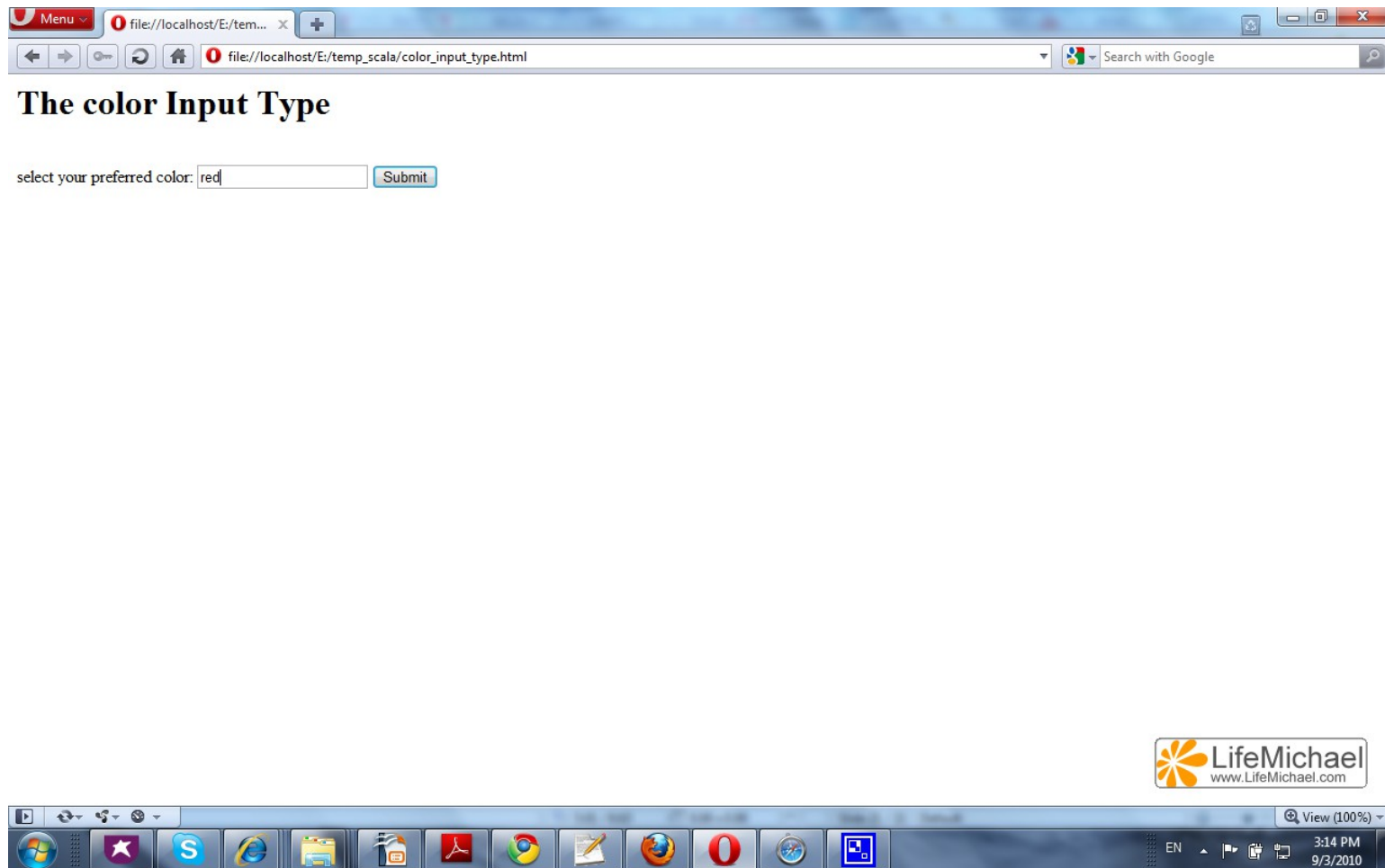
# The color Input Type

```
<h1>The color Input Type</h1>
```

```
<form  action="http://www.abelski.com/courses/html5/colordemo.php"
      method="get">
<br>select your preferred color: <input type="color" name="colorpick"/>
<input type="submit"/>
</form>
```



# The color Input Type





# The `datetime` Input Type

- ❖ Specifying `datetime` as the type of our input control we will get a control that allows us to enter the date, the time and the time zone.

# The datetime Input Type

```
<h1>The datetime Input Type</h1>
```

```
<form  action="http://www.abelski.com/courses/html5/colordemo.php"  
      method="get">
```

```
<br>select your preferred time: <input type="datetime" name="usertime"/>
```

```
<input type="submit"/>
```

```
</form>
```



# The datetime Input Type

Menu file:///localhost/E:/tem... X

file:///localhost/E:/temp\_scala/datetime\_input\_type.html Search with Google

## The datetime Input Type

select your preferred time: 2020-02-02 00:01 UTC Submit

		February 2019						
Week	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
5	28	29	30	31	1	2	3	
6	4	5	6	7	8	9	10	
7	11	12	13	14	15	16	17	
8	18	19	20	21	22	23	24	
9	25	26	27	28	1	2	3	
10	4	5	6	7	8	9	10	

Today None

LifeMichael  
www.LifeMichael.com

View (100%)

EN 3:24 PM 9/3/2010

# The `datetime-local` Input Type

- ❖ Specifying `datetime-local` as the type of our input control we will get a control that allows us to enter the date and the time. The control we get won't allow the user to select the time zone.

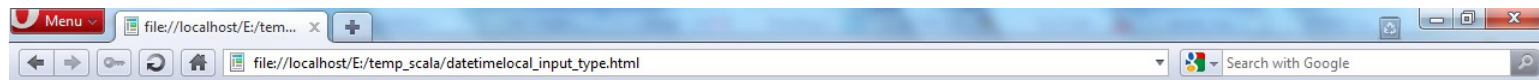
# The datetime-local Input Type

```
<h1>The datetime-local Input Type</h1>
```

```
<form
  action="http://www.abelski.com/courses/html5/datetimedemo.php"
  method="get">
  <br>select your preferred time: <input type="datetime-local"
    name="usertime"/>
  <input type="submit"/>
</form>
```



# The datetime-local Input Type



## The datetime-local Input Type

select your preferred time:  :

September

2010

Week	Mon	Tue	Wed	Thu	Fri	Sat	Sun
35	30	31	1	2	3	4	5
36	6	7	8	9	10	11	12
37	13	14	15	16	17	18	19
38	20	21	22	23	24	25	26
39	27	28	29	30	1	2	3
40	4	5	6	7	8	9	10

Today

None



# The `time` Input Type

- ❖ Specifying `time` as the input type will create a user control for getting the time only.

# The time Input Type

```
<h1>The time Input Type</h1>
```

```
<form  action="http://www.abelski.com/courses/html5/datetimedemo.php"  
      method="get">
```

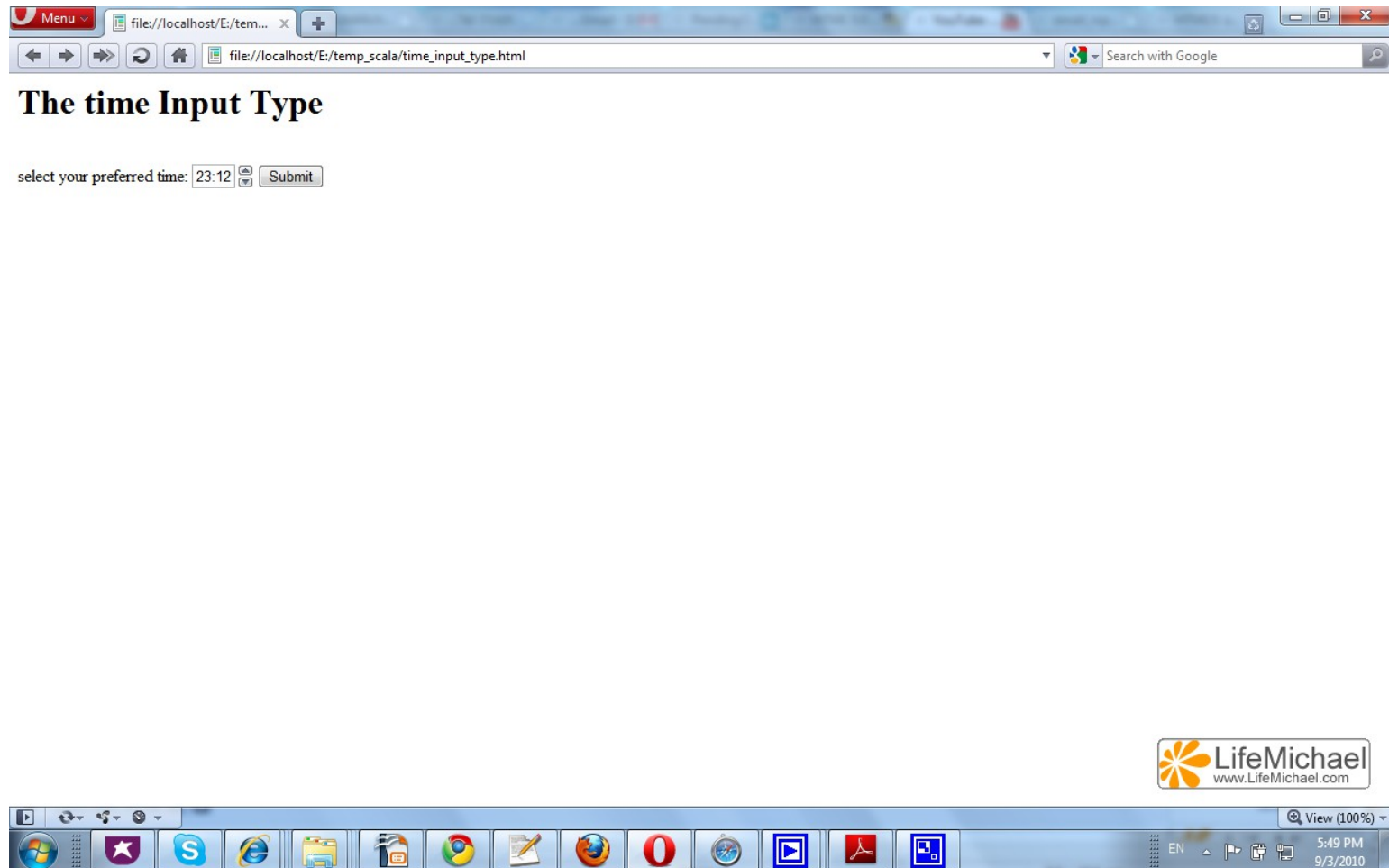
```
<br>select your preferred time: <input type="time" name="usertime"/>  
<input type="submit"/>
```

```
</form>
```





# The time Input Type



# The `date` Input Type

- ❖ Specifying `date` as the input type will create a user control for getting the date only.

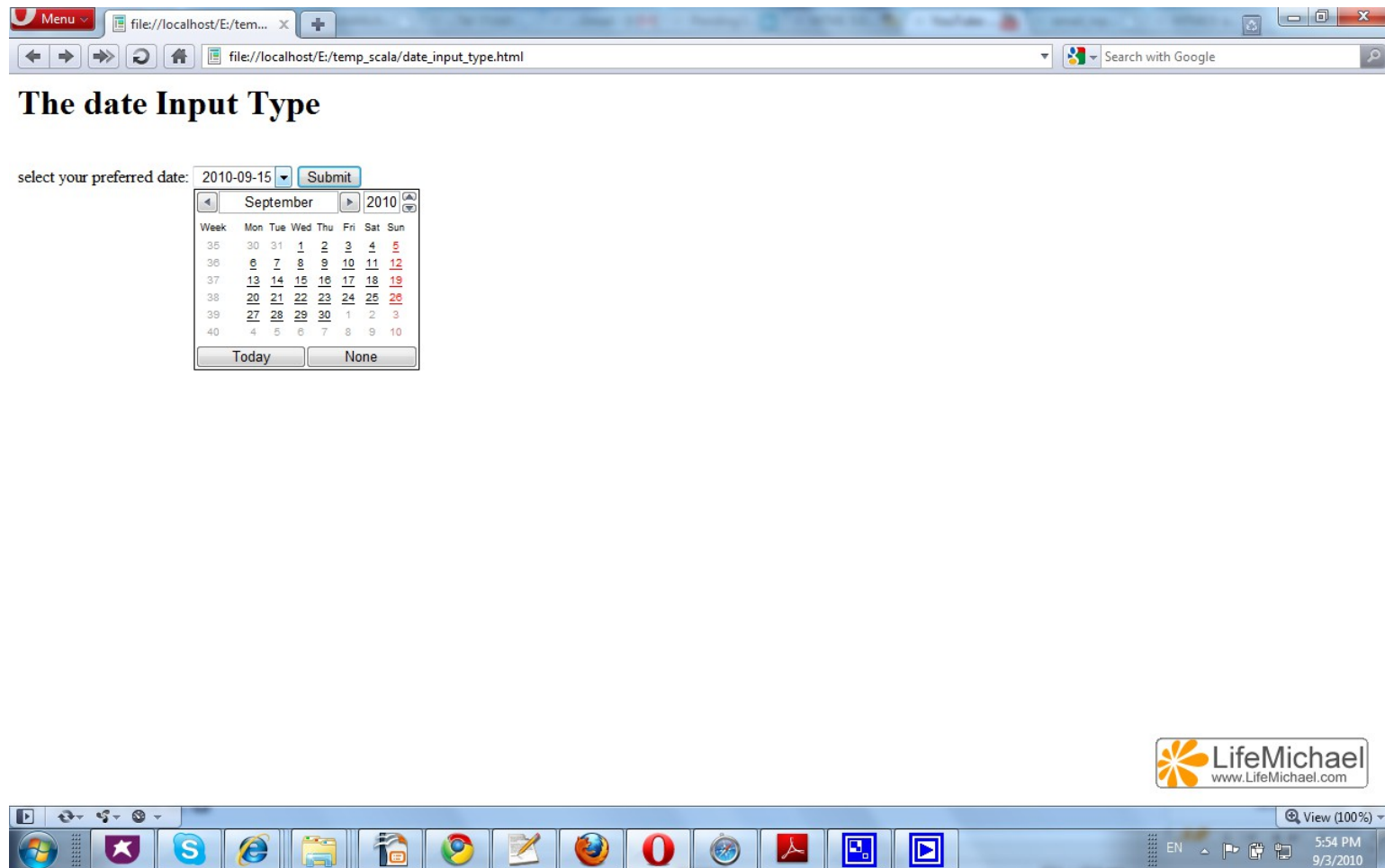
# The date Input Type

```
<h1>The date Input Type</h1>
```

```
<form  action="http://www.abelski.com/courses/html5/datetimedemo.php"  
      method="get">  
<br>select your preferred date: <input type="date" name="usertime"/>  
<input type="submit"/>  
</form>
```



# The date Input Type



# The `week` Input Type

- ❖ Specifying `week` as the input type will create a user control for getting the week only.

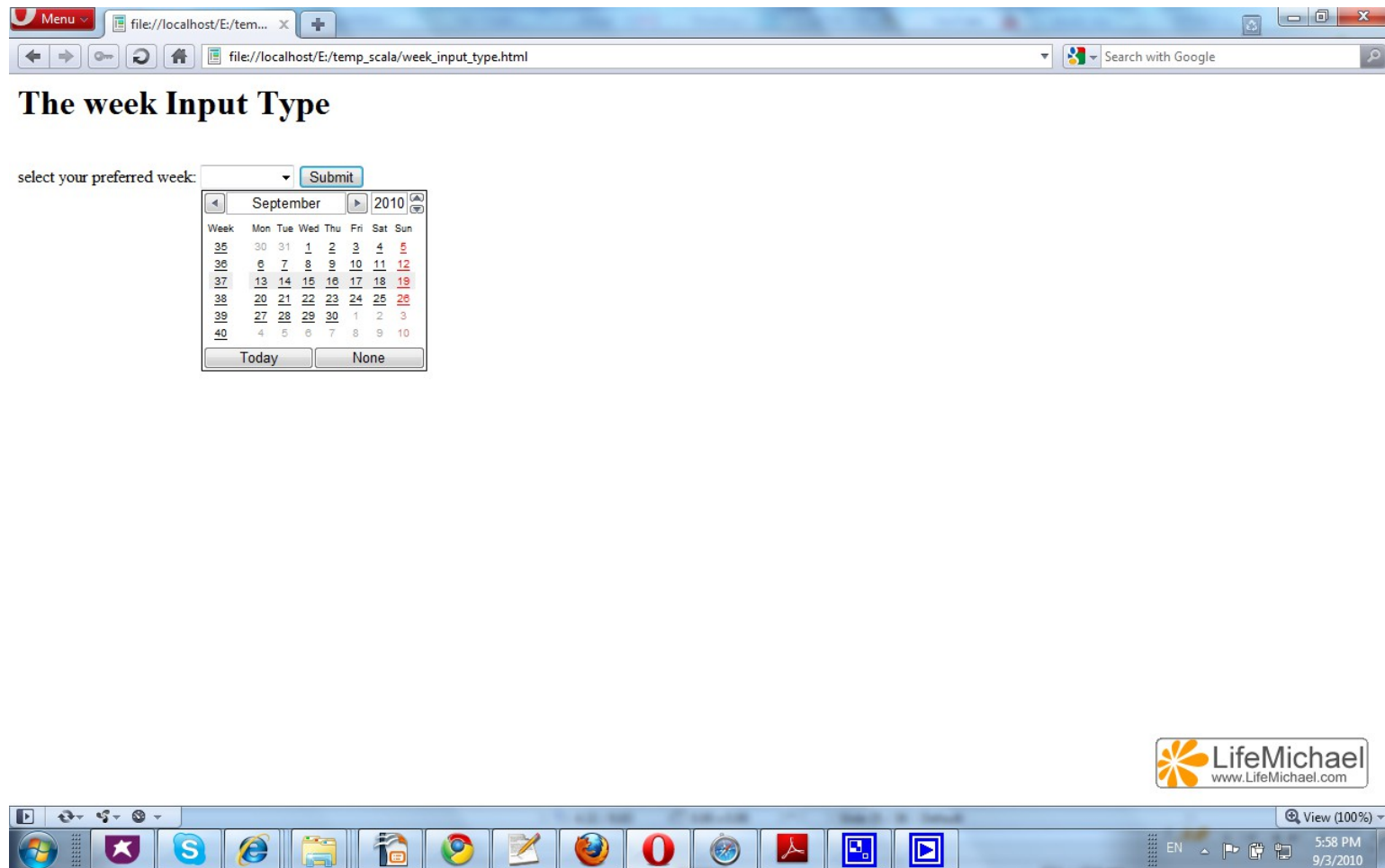
# The week Input Type

```
<h1>The week Input Type</h1>
```

```
<form  action="http://www.abelski.com/courses/html5/datetimedemo.php"
      method="get">
<br>select your preferred week: <input type="week" name="usertime"/>
<input type="submit"/>
</form>
```



# The week Input Type



# The `month` Input Type

- ❖ Specifying `month` as the input type will create a user control for getting the month only.



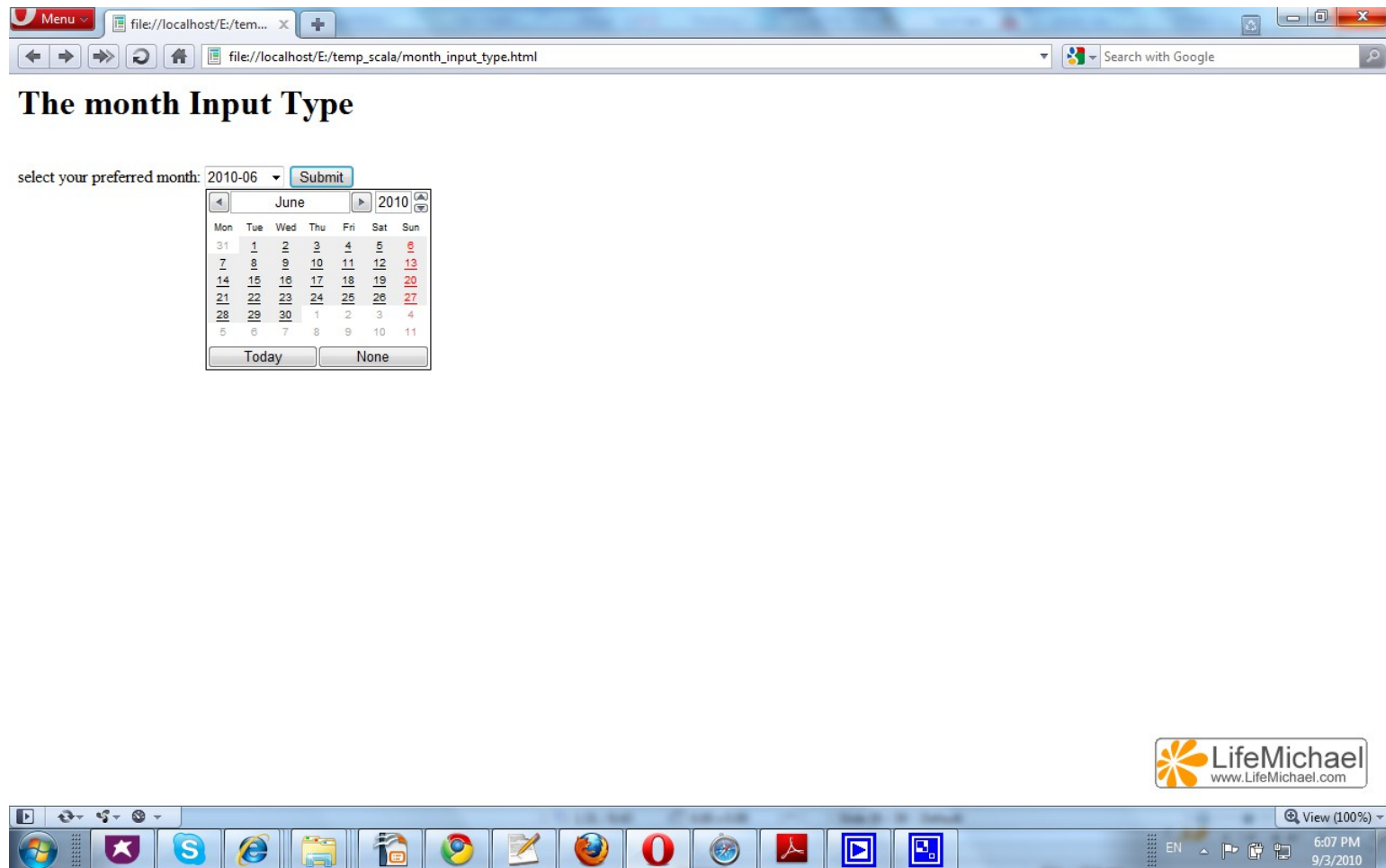
# The month Input Type

```
<h1>The month Input Type</h1>
```

```
<form    action="http://www.abelski.com/courses/html5/datetimedemo.php"  
        method="get">  
<br>select your preferred month: <input type="month" name="usertime"/>  
<input type="submit"/>  
</form>
```



# The month Input Type



# The HTML 5 Forms APIs

- ❖ HTML 5 presents attributes and APIs we can use together with the controls. Both the old ones and the new ones.
- ❖ HTML 5 attributes and APIs aim at reducing the amount of scripting.

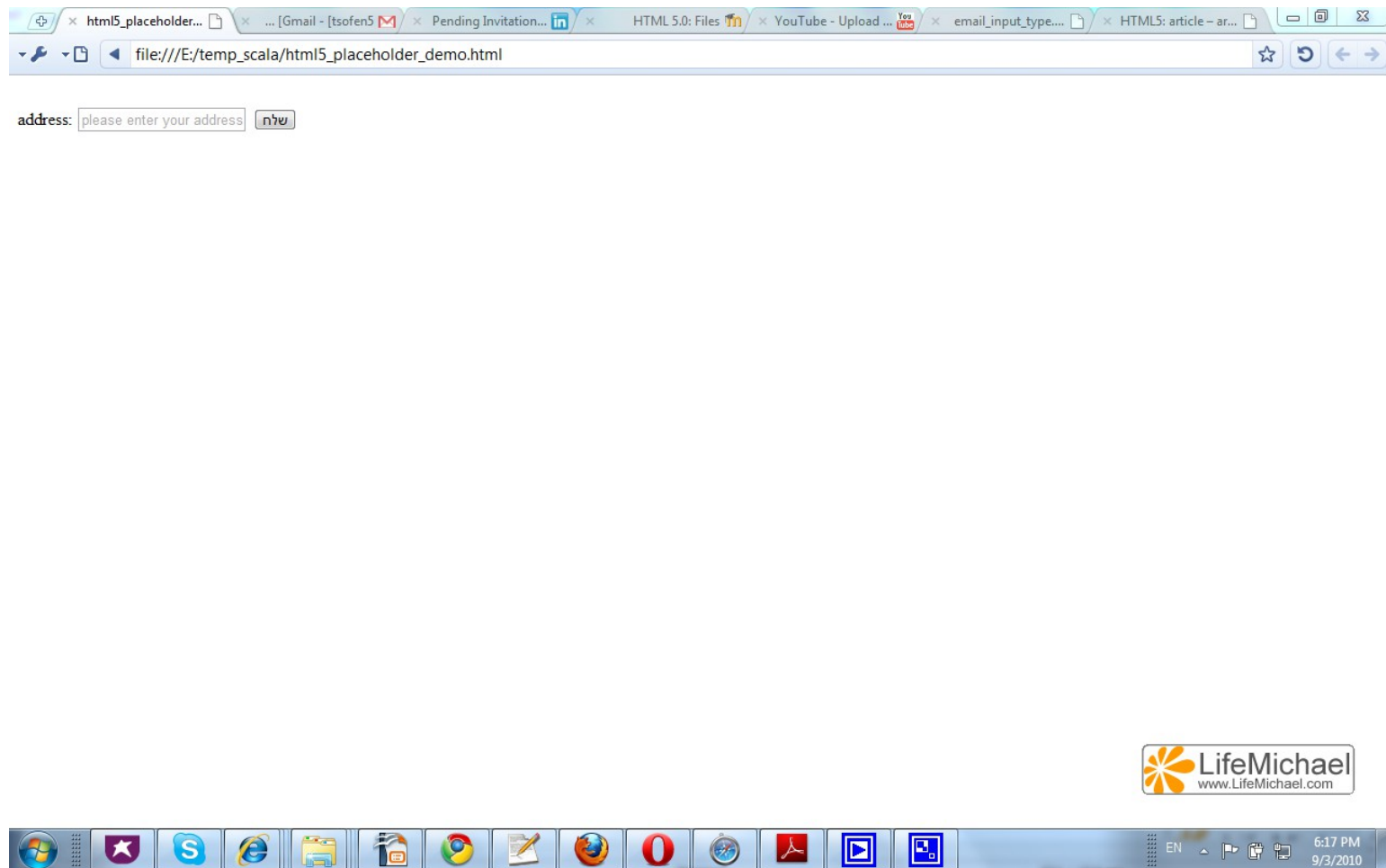
# The placeholder Attribute

- ❖ We can use this attribute for providing hintful descriptive text to the user. This hintful text will be shown as long as the user hasn't entered any text.

```
<form action="http://www.abelski.com/courses/html5/datetimedemo.php"
      method="get">
  <br>address:
  <input type="text"
        name="useraddress"
        placeholder="please enter your address" />
  <input type="submit"/>
</form>
```



# The placeholder Attribute



# The autocomplete Attribute

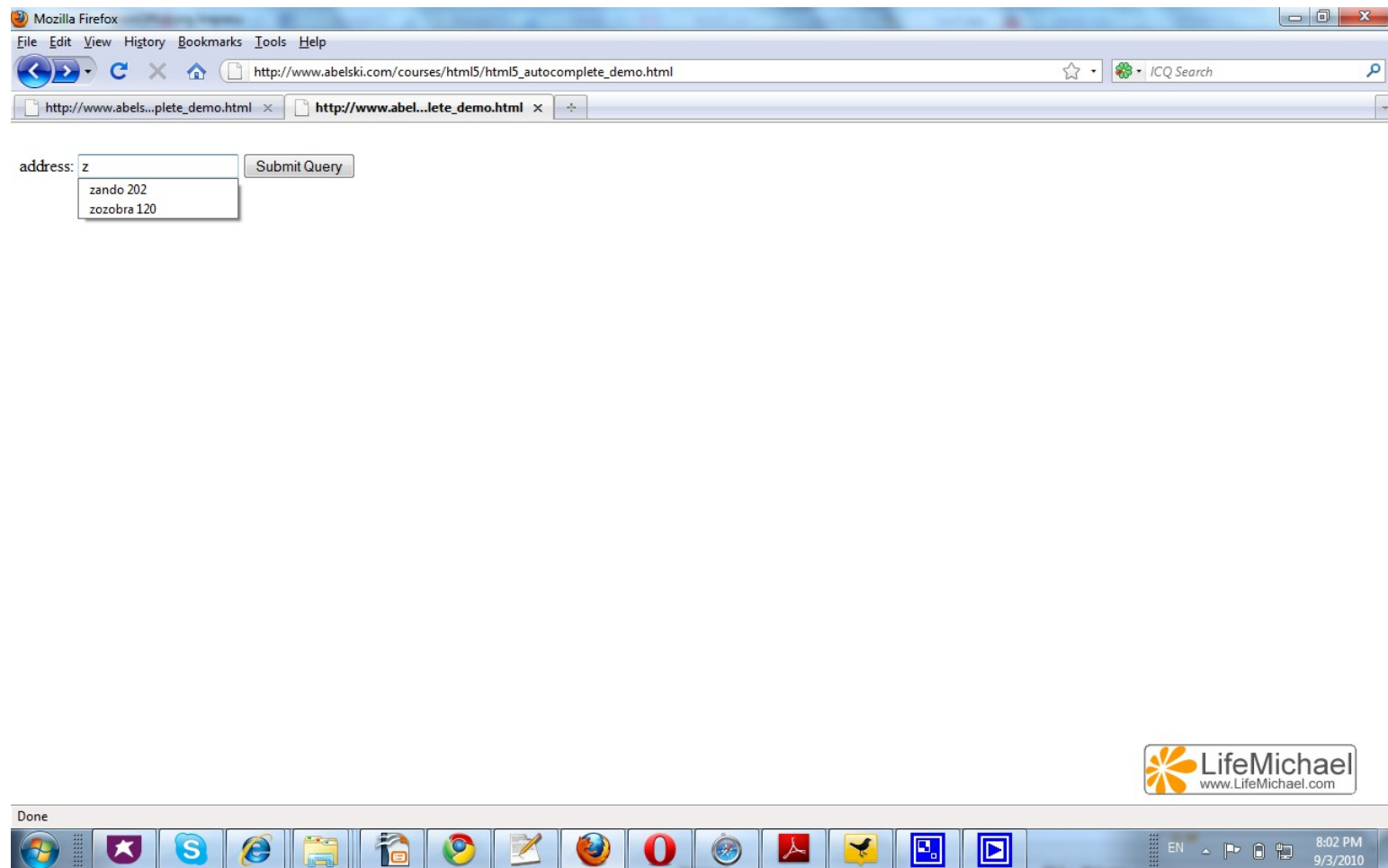
- ❖ This attribute tells the browser whether or not to save the value for future autocomplete. The possible values are `on`, `off` and `unspecified`. The `unspecified` option means applying the setting of the form (if exists). If these setting don't exist then it would be `on`.

# The autocomplete Attribute

```
<form action="http://www.abelski.com/courses/html5/datetimedemo.php"
      method="get">
<br>address:
<input  type="text"
        name="useraddress"
        placeholder="please enter your address"
        autocomplete="on"/>
<input type="submit"/>
</form>
```



# The autocomplete Attribute





# The autofocus Attribute

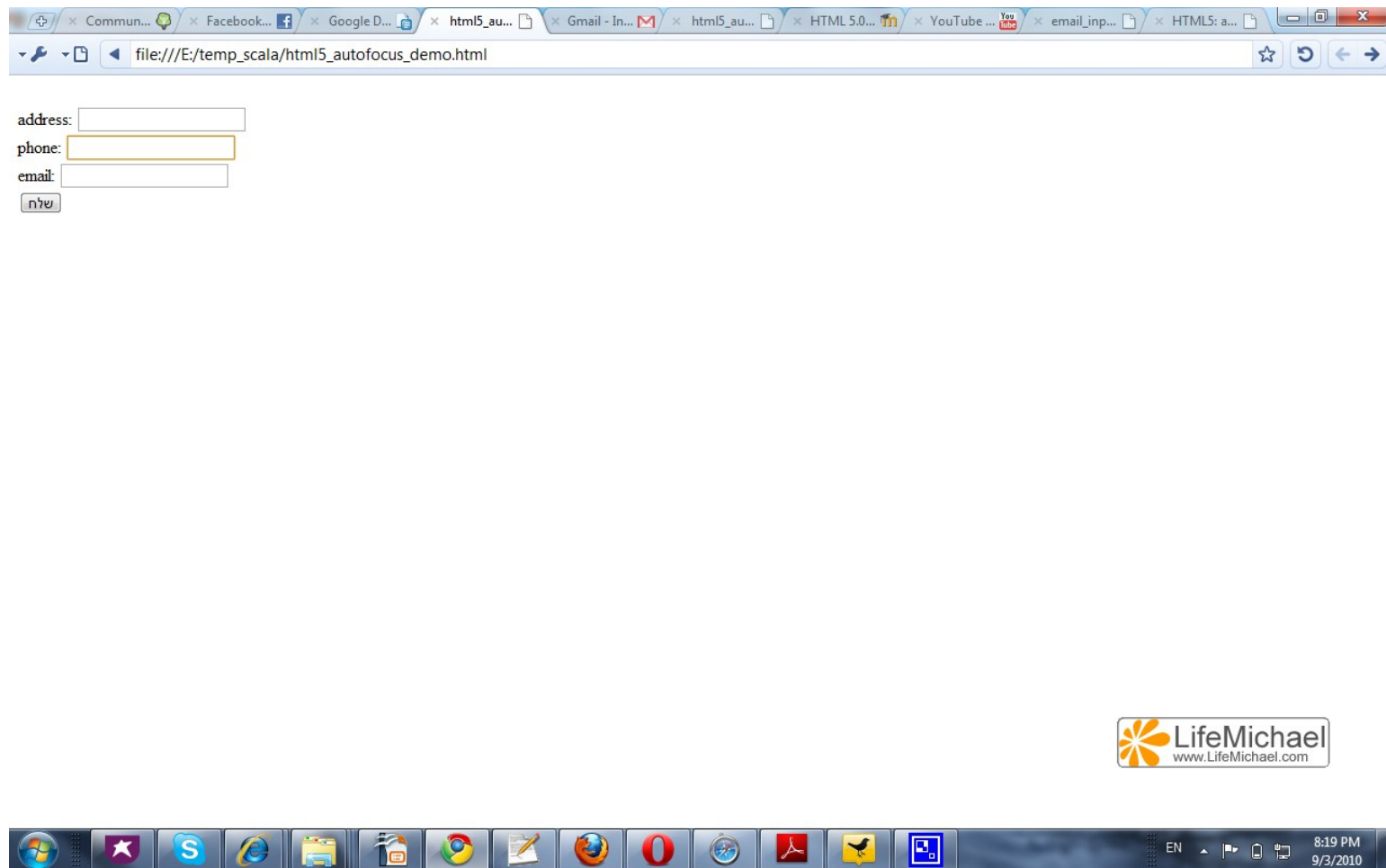
- ❖ We can use this attribute to specify the specific user interface control we want to get the focus when the page completes its loading. We use this attribute without a value.

# The autofocus Attribute

```
<form action="autocomplete.php" method="get">  
<br>address: <input type="text" name="useraddress" />  
<br>phone: <input type="text" name="phone" autofocus />  
<br>email: <input type="text" name="email_address" />  
<br><input type="submit"/>  
</form>
```



# The autofocus Attribute



# The `datalist` Element

- ❖ We can use the `datalist` element for specifying list of possible values we want to display in a specific input control.
- ❖ We use the `list` attribute within the input control in order to connect the input control with the `datalist` element.

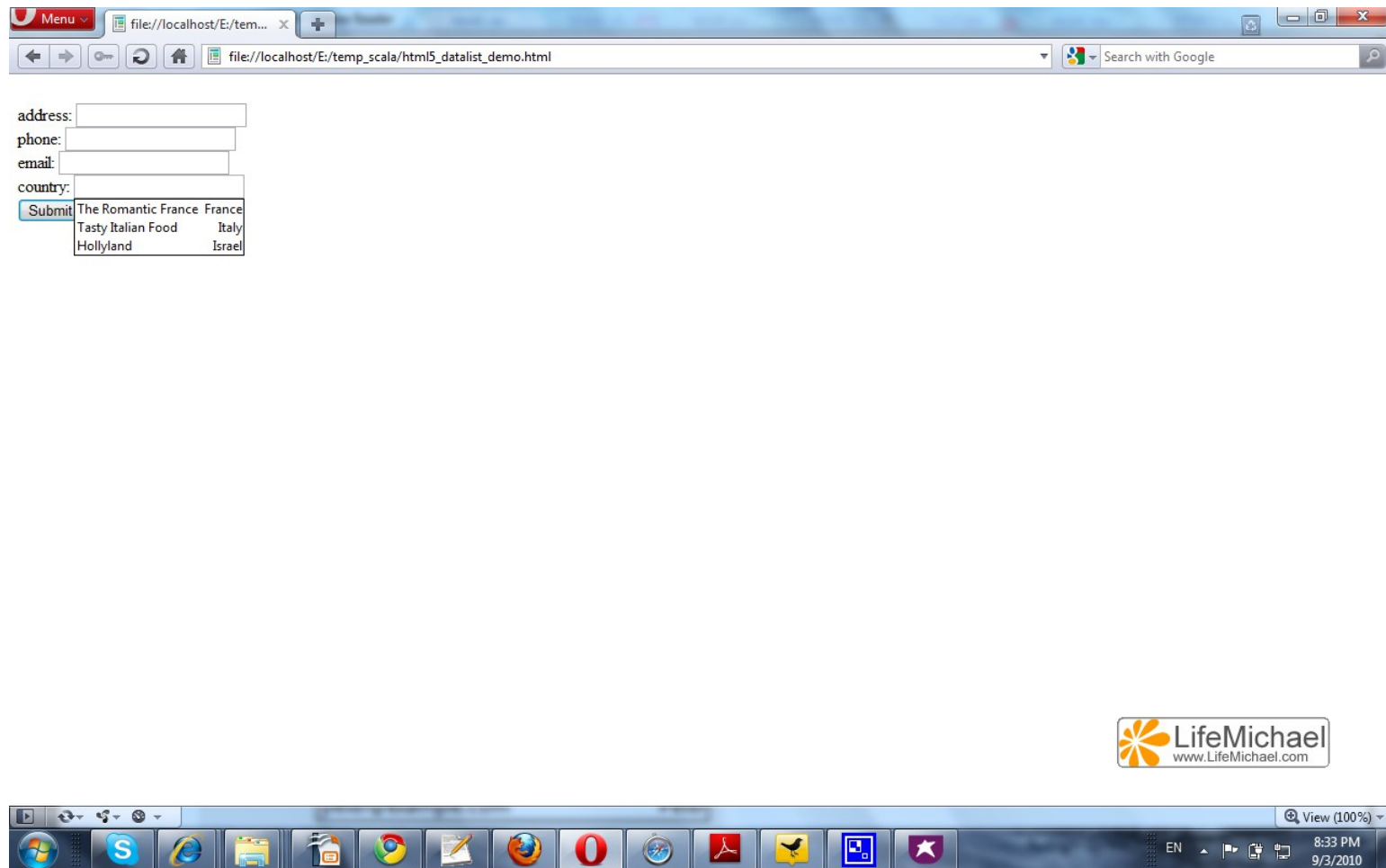
# The datalist Element

```
<datalist id="countries">
  <option value="The Romantic France" label="France" />
  <option value="Tasty Italian Food" label="Italy" />
  <option value="Hollyland" label="Israel" />
</datalist>
```

```
<form action="autocomplete.php" method="get">
<br>address: <input type="text" name="useraddress" />
<br>phone: <input type="text" name="phone" autofocus />
<br>email: <input type="text" name="email_address" />
<br>country: <input type="text" name="country" list="countries" />
<br><input type="submit"/>
</form>
```



# The datalist Element



# The required Attribute

- ❖ We can use this attribute in order to specify specific controls in which the user must enter a value before the form is submitted.

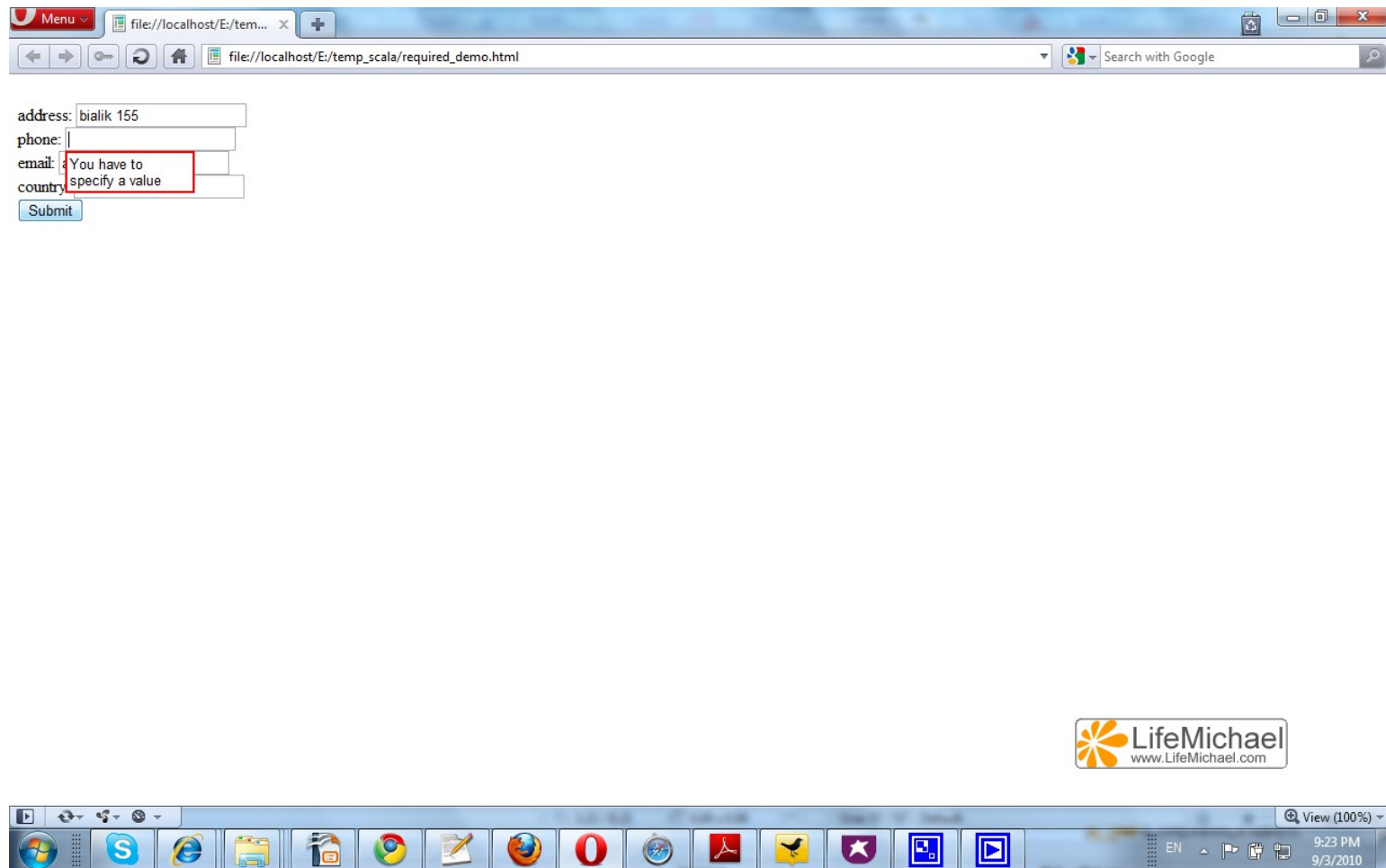
# The required Attribute

```
<form action="date.php" method="get">  
<br>address: <input type="text" name="useraddress" />  
<br>phone: <input type="text" name="phone" required />  
<br>email: <input type="text" name="email_address" />  
<br>country: <input type="text" name="country" />  
<br><input type="submit"/>  
</form>
```





# The required Attribute



# Forms Validation

- ❖ Each user control has a `ValidityState` object that tells whether the specific control holds valid data or not.
- ❖ Referring the `validity` attribute of a user control object we get a reference for its `ValidityState` object.

...

```
val myCheck = document.myForm.firstName.validity;
```

...

# Forms Validation

- ❖ Referring the `valid` attribute each `ValidityState` object has we shall get `true` if all eight validation tests pass successfully.

...

```
val myCheck = document.myForm.firstName.validity;  
if (myCheck.valid)  
{  
    //data is valid  
}
```

...

# Validity Constraints

- ❖ There are eight possible validity constraints on each and every form element. Each one of them can be accessed through the `ValidityState` object. That object has eight fields named after these eight possible validity constraints.
- ❖ We can check separately each and every one of them.

# Validity Constraints

- ❖ The `ValidityState` object is a live object. Once we get a reference for a `ValidityState` object of a specific control we can keep holding it. The validity checks it returns are updated when changes occur.

# The `valueMissing` Constraint

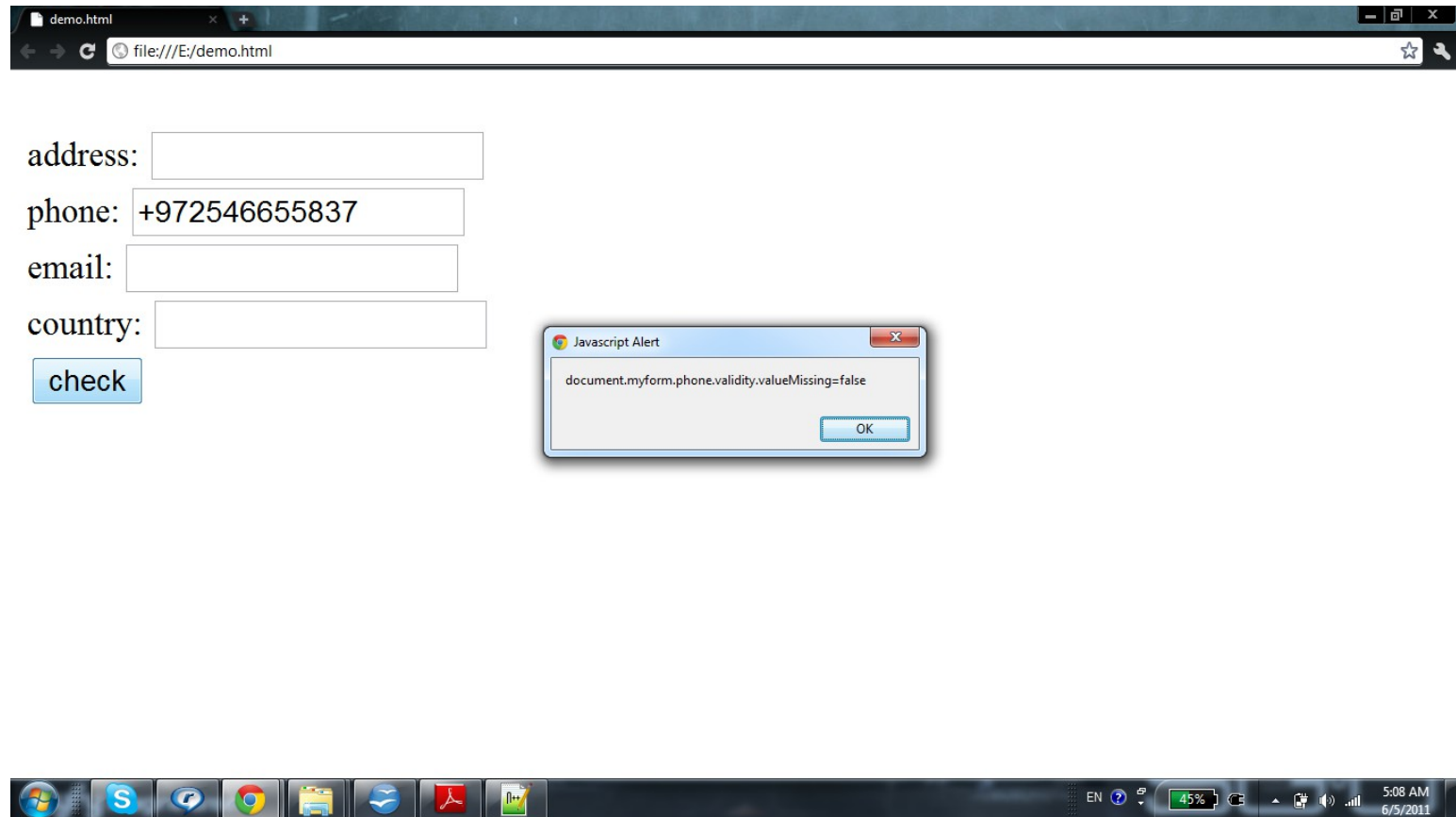
- ❖ This constraint is relevant when using the `required` attribute.
- ❖ Unless a value is entered the value of this constraint is true.

# The valueMissing Constraint

```
<form action="date.php" method="get" name="myform">
<br>address: <input type="text" name="useraddress" />
<br>phone: <input type="text" name="phone" required />
<br>email: <input type="text" name="email_address" />
<br>country: <input type="text" name="country" />
<br><input type="button" value="check" onclick="check()" />
</form>
<script>
function check()
{
    alert("document.myform.phone.validity.valueMissing="+
        document.myform.phone.validity.valueMissing);
}
</script>
```



# The valueMissing Constraint





# The `typeMismatch` Constraint

- ❖ This constraint checks to verify the type of the entered value matches with the expected value (number, tel, email etc.).

# The `patternMismatch` Constraint

- ❖ This constraint checks to verify the value matches with the pattern we set. The pattern is set using the `pattern` attribute.

# The `tooLong` Constraint

- ❖ This constraint checks to ensure the entered value doesn't exceed the maximum length we set.
- ❖ We use the `maxLength` attribute in order to set the maximum allowed length.

# The `rangeOverflow` Constraint

- ❖ This constraint checks to ensure the entered value isn't bigger than the maximum allowed value we set.
- ❖ We use the `max` attribute in order to set the maximum allowed value.

# The `rangeUnderflow` Constraint

- ❖ This constraint checks to ensure the entered value isn't smaller than the minimum value we set.
- ❖ We use the `min` attribute in order to set the minimum allowed value.

# The `stepMismatch` Constraint

- ❖ This constraint checks to ensure the entered value doesn't violate the step we set.
- ❖ We use the `step` attribute in order to set the step value.

# The `customError` Constraint

- ❖ We can write JavaScript code that explicitly sets an error for a specific control. Calling `setCustomValidity(message)` on a specific control we get two outcomes. The control is placed into a an error state and a customized error message is set.
- ❖ When referring `customError` we shall get true if the control we refer was placed into a custom error state.

# The `output` Element

- ❖ The `output` element is been used for presenting the result of some sort of calculation that took place within the scope of a form.
- ❖ We can add the `for` attribute in order to specify (space separated) the ids of those elements that took place in the calculation.



# The output Element

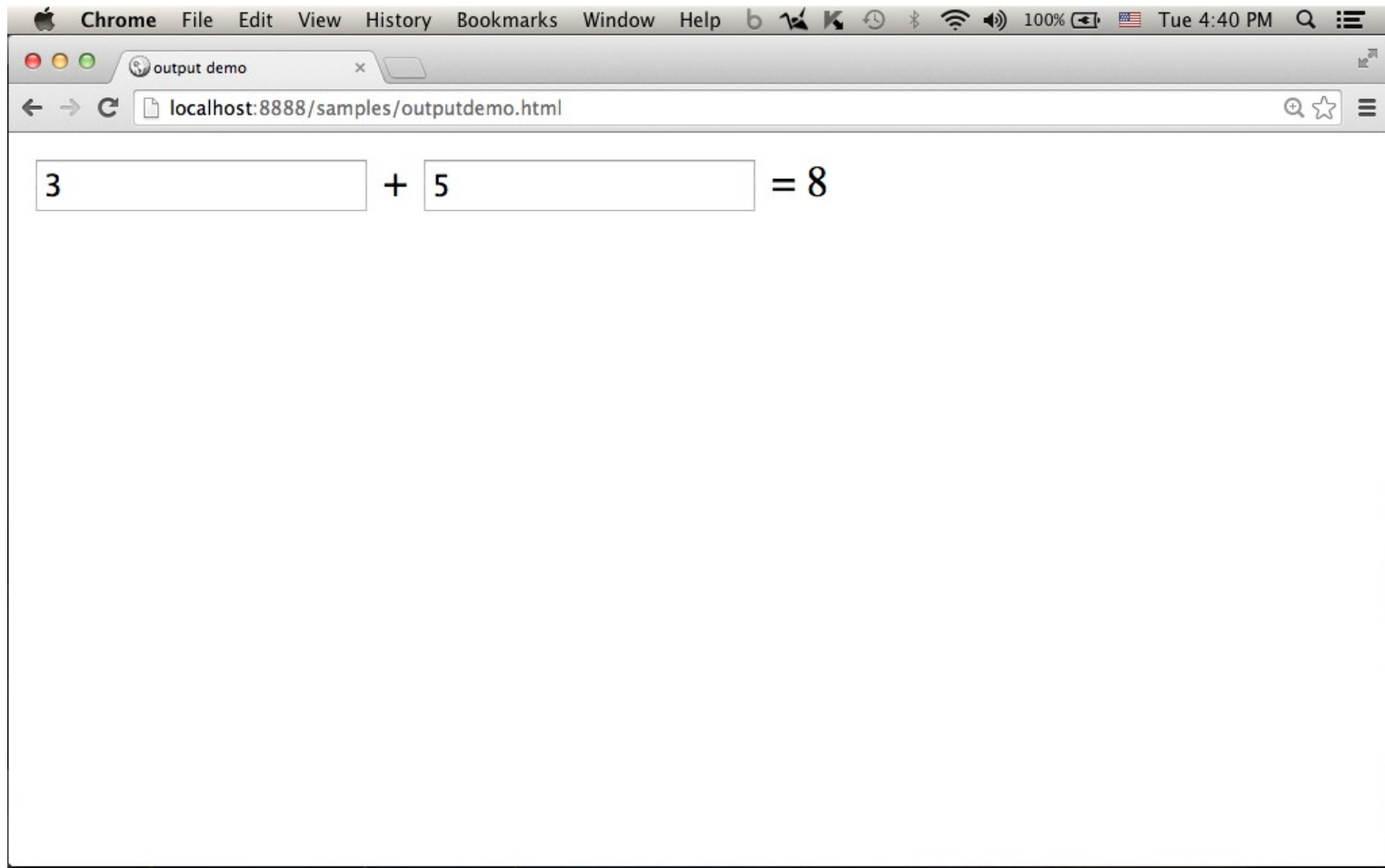
- ❖ We can use the `form` attribute in order to specify the form owner of this output element. The value of this attribute should be the id of a specific form in the very same page.
- ❖ We can use the `name` attribute in order to name the output element we are using.

# The output Element

```
<!DOCTYPE html>
<html>
<head>
  <title>output demo</title>
</head>
<body>
<form oninput="result.value =
  parseInt(numA.value) + parseInt(numB.value)">
  <input name="numA" type="number" > +
  <input name="numB" type="number" > =
  <output name="result"></output>
</form>
</body>
</html>
```



# The output Element



# HTML 5 Forms

## Introduction

- ❖ HTML 5.0 introduces a new set of controls we can use in our forms.
- ❖ The web browser's support for these new controls varies from one browser to another.
- ❖ The support for the HTML old controls still exists.
- ❖ The new HTML 5.0 controls are scriptable similarly to the old ones.

© 2010 Haim Michael. All Rights Reserved.

## Introduction

- ❖ The HTML 5.0 specification covers the controls' functional behavior only. It doesn't cover their appearance or display.
- ❖ W3C maintains an up-to-date list of all the HTML 5 form controllers currently supported together with the future ones.

© 2010 Haim Michael. All Rights Reserved.

## The `email` Input Type

- ❖ Using the `<INPUT . . . >` controller with `email` as its type we will get a text field that accepts email addresses only.

© 2010 Haim Michael. All Rights Reserved.

# The email Input Type

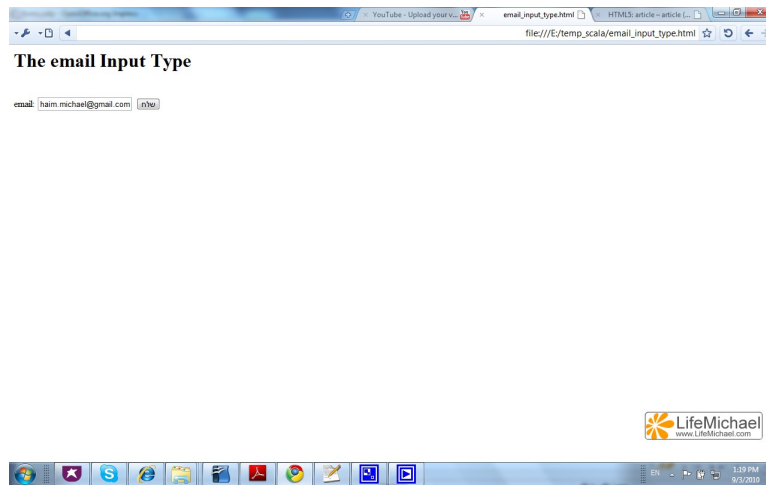
```
<h1>The email Input Type</h1>  
  
<form action="http://www.abelski.com/courses/html5/demo.php" method="get">  
<br>email: <input type="email" name="user_email_address"/>  
<input type="submit"/>  
</form>
```



© 2010 Haim Michael. All Rights Reserved.



# The email Input Type



© 2010 Haim Michael. All Rights Reserved.

## The `tel` Input Type

- ❖ Using the `<INPUT . . .>` controller with `tel` as its type we will get a text field that accepts telephone numbers only.

© 2010 Haim Michael. All Rights Reserved.

## The tel Input Type

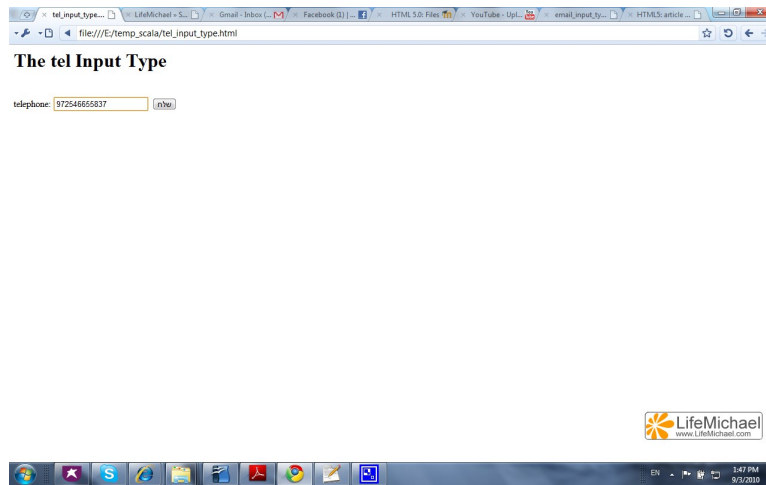
```
<h1>The tel Input Type</h1>
```

```
<form action="http://www.abelski.com/courses/html5/teldemo.php" method="get">  
<br>telephone: <input type="tel" name="user_tel"/>  
<input type="submit"/>  
</form>
```

The support for each one of the special input types varies from one browser to another. Some browsers will provide an hint so the user knows he should enter a telephone number. Others will prevent entering something that cannot be a phone number.

© 2010 Haim Michael. All Rights Reserved.

# The tel Input Type



© 2010 Haim Michael. All Rights Reserved.

## The `range` Input Type

- ❖ Using the `<INPUT . . . >` controller with `range` as its type together with specifying `min` and `max` attributes we will get a controller that accepts values in the specified range only.

© 2010 Haim Michael. All Rights Reserved.

# The range Input Type

```
<h1>The range Input Type</h1>

<form  action="http://www.abelski.com/courses/html5/rangedemo.php"
      method="get">

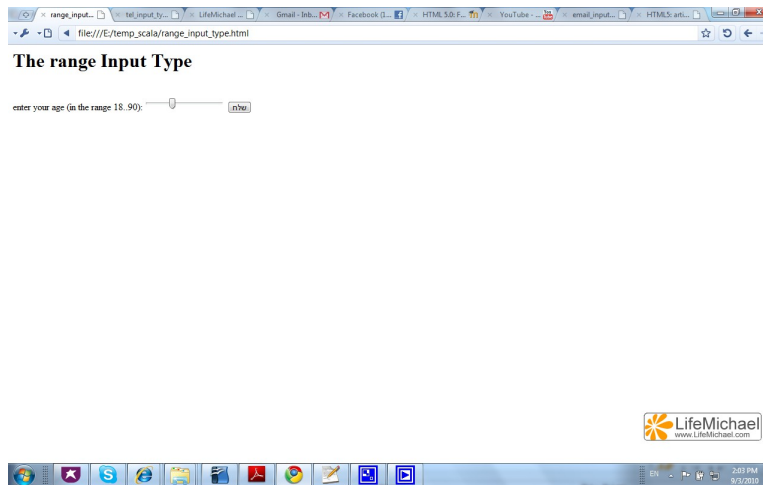
  <br>enter your age (in the range 18..90):
  <input type="range" name="age" min="18" max="90" />
  <input type="submit"/>

</form>
```



© 2010 Haim Michael. All Rights Reserved.

# The range Input Type



© 2010 Haim Michael. All Rights Reserved.

## The `range` Input Type

- ❖ We can use JavaScript in order to present the selected value.

© 2010 Haim Michael. All Rights Reserved.



## The range Input Type

```
<h1>The range Input Type</h1>

<form action="http://www.abelski.com/courses/html5/rangedemo.php"
      method="get">

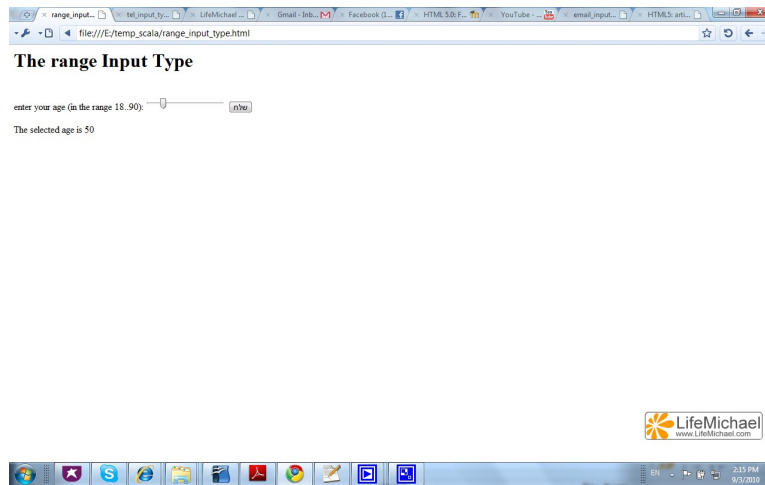
  <br>enter your age (in the range 18..90):
  <input type="range" name="age" onchange="showTheSelectedValue(this.value)"/>
  <input type="submit"/>
</form>

<script type="text/javascript">
function showTheSelectedValue(number)
{
    document.getElementById("selected_age").innerHTML = number;
}
</script>
The selected age is <span id="selected_age">50</span>
```



© 2010 Haim Michael. All Rights Reserved.

# The range Input Type



© 2010 Haim Michael. All Rights Reserved.

## The `range` Input Type

- ❖ We can use add the `step` attribute in order to specify the size of the steps in which the slider changes its value.

© 2010 Haim Michael. All Rights Reserved.

## The range Input Type

```
<h1>The range Input Type</h1>

<form action="http://www.abelski.com/courses/html5/rangedemo.php" method="get">
<br>volume (10..100):
<input type="range" name="age" onchange="showTheSelectedValue(this.value)"
min="10" max="100" step="5"/>
<input type="submit"/>
</form>

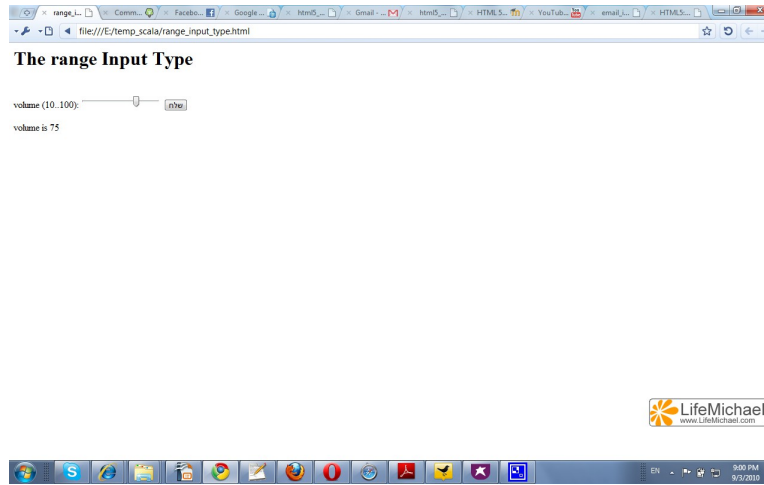
<script type="text/javascript">
function showTheSelectedValue(number)
{
    document.getElementById("selected_age").innerHTML = number;
}
</script>

volume is <span id="selected_age">50</span>
```



© 2010 Haim Michael. All Rights Reserved.

# The range Input Type



© 2010 Haim Michael. All Rights Reserved.

## The `number` Input Type

- ❖ Specifying `number` as the type of our controller we will get a text field through which the user will be able to enter numeric values only.

The user will be allowed to enter valid numeric values only. Trying to enter non numeric values the user won't be able to send the entered data to the server.

© 2010 Haim Michael. All Rights Reserved.

# The number Input Type

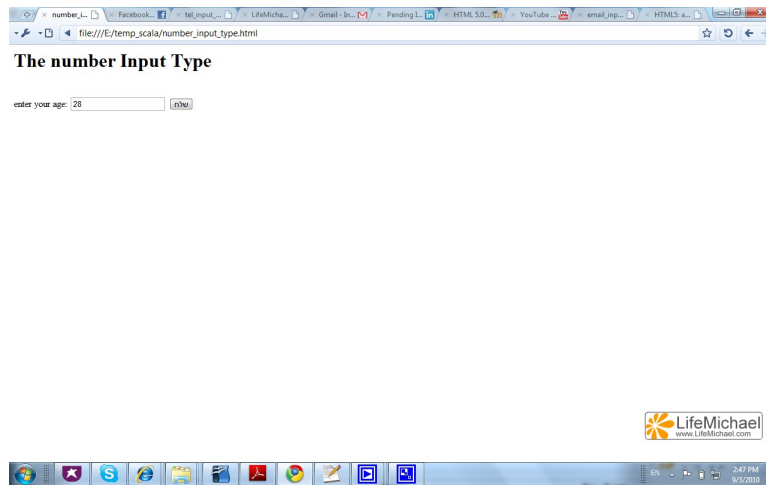
```
<h1>The number Input Type</h1>

<form  action="http://www.abelski.com/courses/html5/rangedemo.php"
      method="get">
<br>enter your age: <input type="number" name="age"/>
<input type="submit"/>
</form>
```



© 2010 Haim Michael. All Rights Reserved.

# The number Input Type



© 2010 Haim Michael. All Rights Reserved.



## The `color` Input Type

- ❖ Specifying `color` as the type of our input controller we will get a control that allows us to pick a color only.

© 2010 Haim Michael. All Rights Reserved.

## The color Input Type

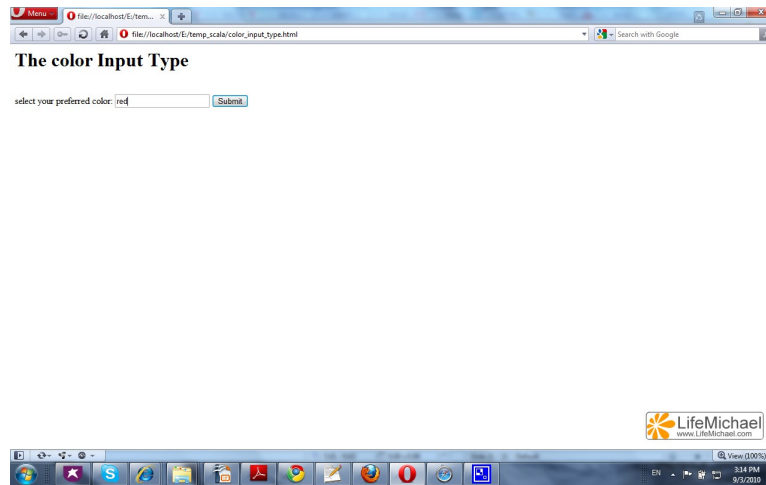
```
<h1>The color Input Type</h1>

<form  action="http://www.abelski.com/courses/html5/colordemo.php"
      method="get">
  <br>select your preferred color: <input type="color" name="colorpick"/>
  <input type="submit"/>
</form>
```



© 2010 Haim Michael. All Rights Reserved.

# The color Input Type



© 2010 Haim Michael. All Rights Reserved.

## The `datetime` Input Type

- ❖ Specifying `datetime` as the type of our input control we will get a control that allows us to enter the date, the time and the time zone.

© 2010 Haim Michael. All Rights Reserved.

# The datetime Input Type

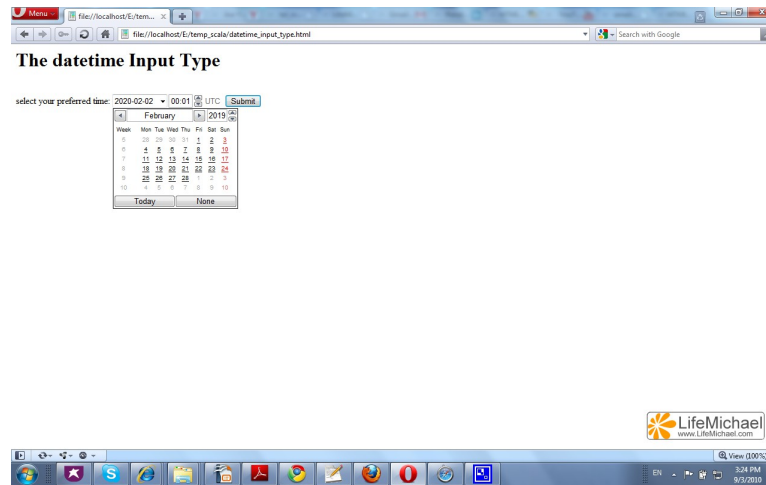
```
<h1>The datetime Input Type</h1>

<form  action="http://www.abelski.com/courses/html5/colordemo.php"
        method="get">
<br>select your preferred time: <input type="datetime" name="usertime"/>
<input type="submit"/>
</form>
```



© 2010 Haim Michael. All Rights Reserved.

# The datetime Input Type



© 2010 Haim Michael. All Rights Reserved.

## The `datetime-local` Input Type

- ❖ Specifying `datetime-local` as the type of our input control we will get a control that allows us to enter the date and the time. The control we get won't allow the user to select the time zone.

© 2010 Haim Michael. All Rights Reserved.

## The datetime-local Input Type

```
<h1>The datetime-local Input Type</h1>

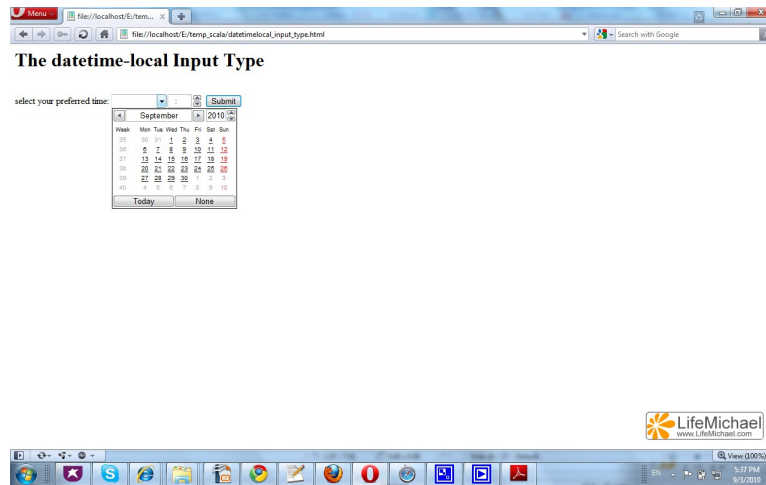
<form
  action="http://www.abelski.com/courses/html5/datetimedemo.php"
  method="get">
  <br>select your preferred time: <input type="datetime-local"
    name="usertime"/>
  <input type="submit"/>
</form>
```



© 2010 Haim Michael. All Rights Reserved.



# The datetime-local Input Type



© 2010 Haim Michael. All Rights Reserved.

## The `time` Input Type

- ❖ Specifying `time` as the input type will create a user control for getting the time only.

© 2010 Haim Michael. All Rights Reserved.

# The time Input Type

```
<h1>The time Input Type</h1>

<form  action="http://www.abelski.com/courses/html5/datetimedemo.php"
        method="get">

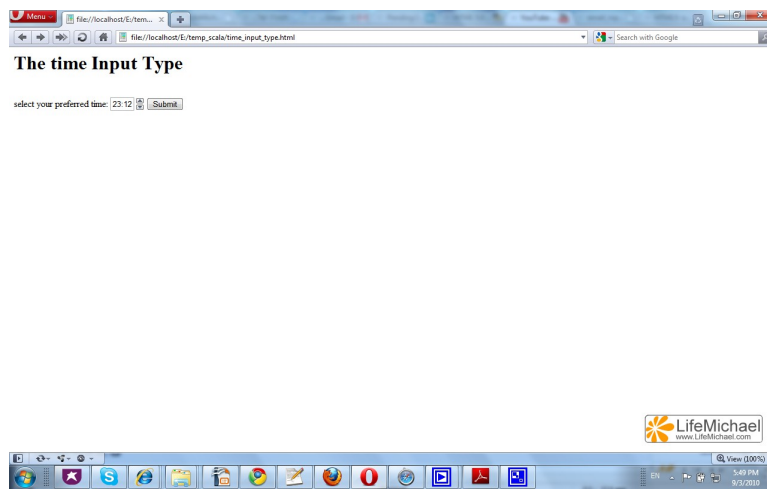
<br>select your preferred time: <input type="time" name="usertime"/>
<input type="submit"/>

</form>
```



© 2010 Haim Michael. All Rights Reserved.

# The `time` Input Type



© 2010 Haim Michael. All Rights Reserved.

## The `date` Input Type

- ❖ Specifying `date` as the input type will create a user control for getting the date only.

© 2010 Haim Michael. All Rights Reserved.

# The date Input Type

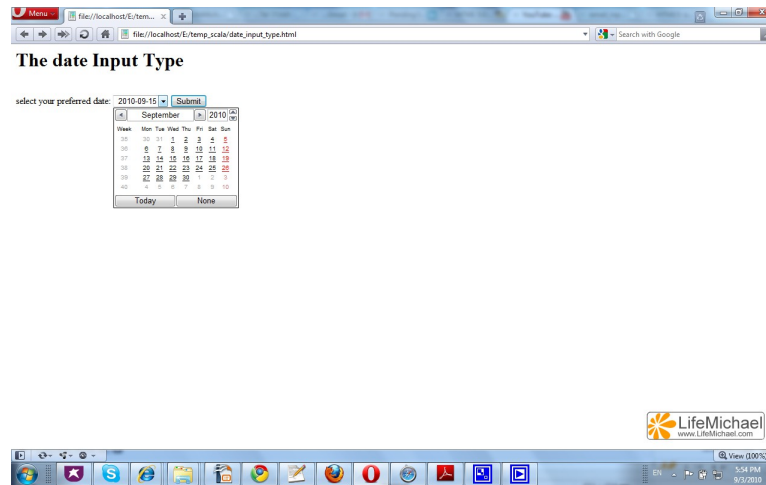
```
<h1>The date Input Type</h1>

<form  action="http://www.abelski.com/courses/html5/datetimedemo.php"
      method="get">
<br>select your preferred date: <input type="date" name="usertime"/>
<input type="submit"/>
</form>
```



© 2010 Haim Michael. All Rights Reserved.

# The date Input Type



© 2010 Haim Michael. All Rights Reserved.

## The `week` Input Type

- ❖ Specifying `week` as the input type will create a user control for getting the week only.

© 2010 Haim Michael. All Rights Reserved.



## The week Input Type

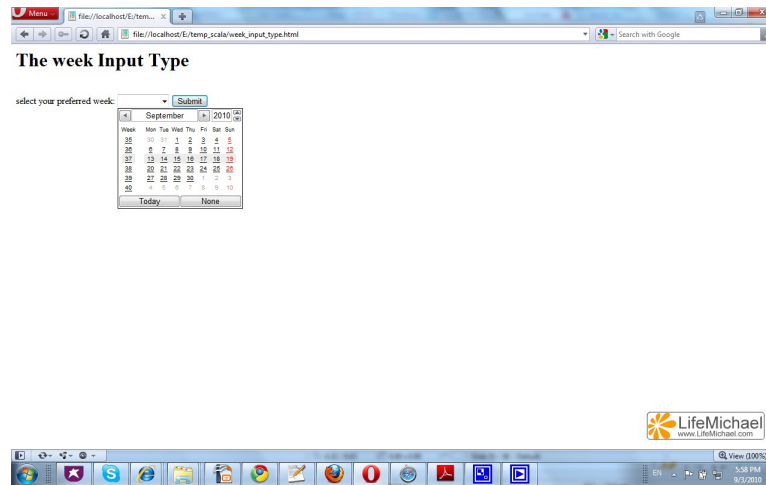
```
<h1>The week Input Type</h1>

<form  action="http://www.abelski.com/courses/html5/datetimedemo.php"
      method="get">
<br>select your preferred week: <input type="week" name="usertime"/>
<input type="submit"/>
</form>
```



© 2010 Haim Michael. All Rights Reserved.

# The week Input Type



© 2010 Haim Michael. All Rights Reserved.

## The `month` Input Type

- ❖ Specifying `month` as the input type will create a user control for getting the month only.

© 2010 Haim Michael. All Rights Reserved.

## The month Input Type

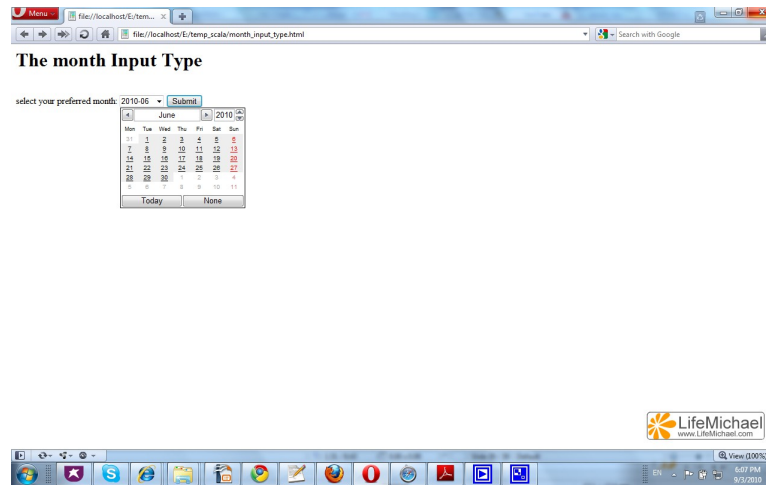
```
<h1>The month Input Type</h1>

<form  action="http://www.abelski.com/courses/html5/datetimedemo.php"
      method="get">
<br>select your preferred month: <input type="month" name="usertime"/>
<input type="submit"/>
</form>
```



© 2010 Haim Michael. All Rights Reserved.

# The month Input Type



© 2010 Haim Michael. All Rights Reserved.

## The HTML 5 Forms APIs

- ❖ HTML 5 presents attributes and APIs we can use together with the controls. Both the old ones and the new ones.
- ❖ HTML 5 attributes and APIs aim at reducing the amount of scripting.

© 2010 Haim Michael. All Rights Reserved.

## The placeholder Attribute

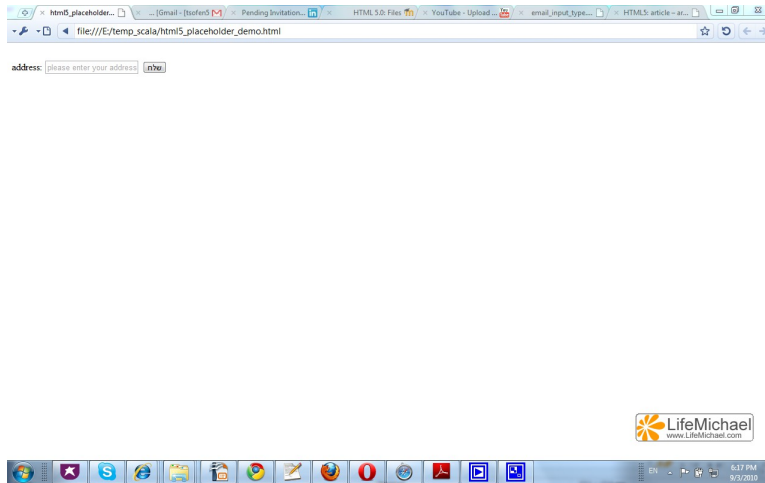
- ❖ We can use this attribute for providing hintful descriptive text to the user. This hintful text will be shown as long as the user hasn't entered any text.

```
<form action="http://www.abelski.com/courses/html5/datetimedemo.php"
      method="get">
  <br>address:
  <input type="text"
        name="useraddress"
        placeholder="please enter your address" />
  <input type="submit"/>
</form>
```



© 2010 Haim Michael. All Rights Reserved.

# The placeholder Attribute



© 2010 Haim Michael. All Rights Reserved.



## The autocomplete Attribute

- ❖ This attribute tells the browser whether or not to save the value for future autocomplete. The possible values are `on`, `off` and `unspecified`. The `unspecified` option means applying the setting of the form (if exists). If these setting don't exist then it would be `on`.

© 2010 Haim Michael. All Rights Reserved.

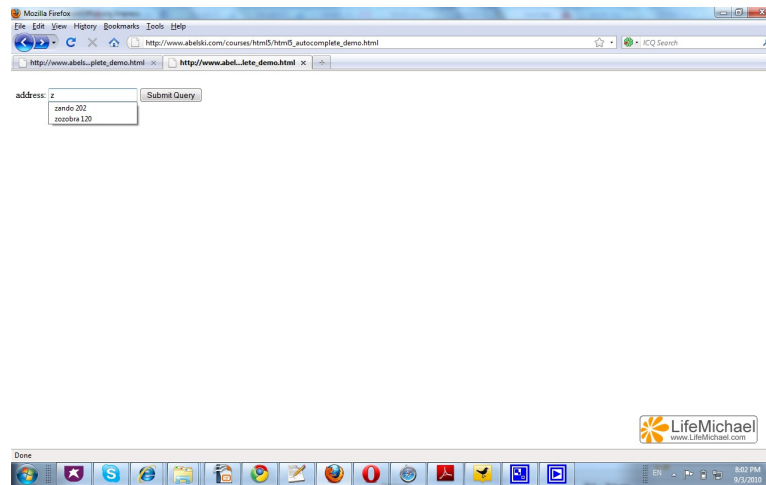
## The autocomplete Attribute

```
<form action="http://www.abelski.com/courses/html5/datetimedemo.php"
      method="get">
  <br>address:
  <input type="text"
        name="useraddress"
        placeholder="please enter your address"
        autocomplete="on"/>
  <input type="submit"/>
</form>
```



© 2010 Haim Michael. All Rights Reserved.

# The autocomplete Attribute



© 2010 Haim Michael. All Rights Reserved.

## The autofocus Attribute

- ❖ We can use this attribute to specify the specific user interface control we want to get the focus when the page completes its loading. We use this attribute without a value.

© 2010 Haim Michael. All Rights Reserved.

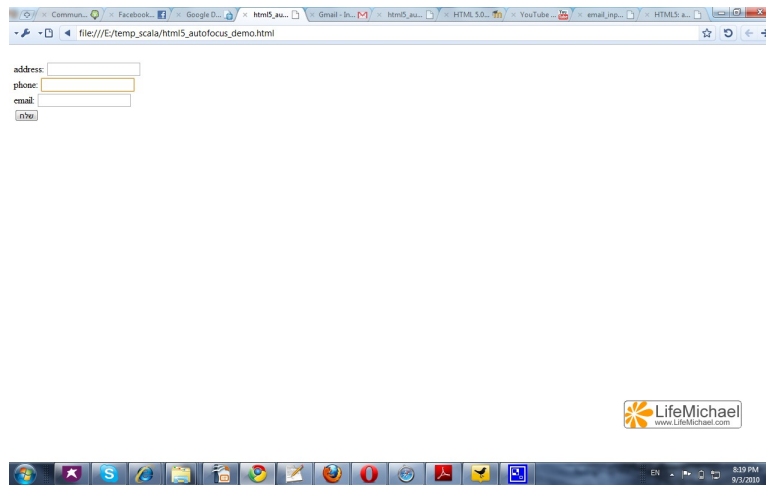
## The autofocus Attribute

```
<form action="autocomplete.php" method="get">  
<br>address: <input type="text" name="useraddress" />  
<br>phone: <input type="text" name="phone" autofocus />  
<br>email: <input type="text" name="email_address" />  
<br><input type="submit"/>  
</form>
```



© 2010 Haim Michael. All Rights Reserved.

# The autofocus Attribute



© 2010 Haim Michael. All Rights Reserved.

## The `datalist` Element

- ❖ We can use the `datalist` element for specifying list of possible values we want to display in a specific input control.
- ❖ We use the `list` attribute within the input control in order to connect the input control with the `datalist` element.

© 2010 Haim Michael. All Rights Reserved.

## The datalist Element

```
<datalist id="countries">
  <option value="The Romantic France" label="France" />
  <option value="Tasty Italian Food" label="Italy" />
  <option value="Hollyland" label="Israel" />
</datalist>

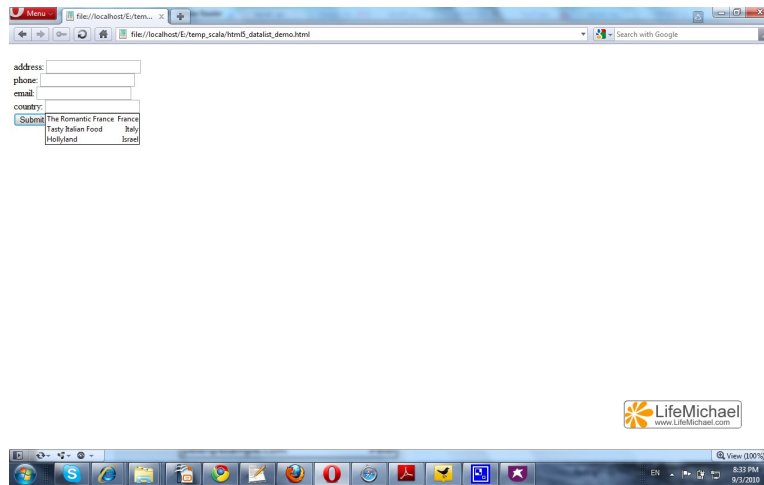
<form action="autocomplete.php" method="get">
<br>address: <input type="text" name="useraddress" />
<br>phone: <input type="text" name="phone" autofocus />
<br>email: <input type="text" name="email_address" />
<br>country: <input type="text" name="country" list="countries" />
<br><input type="submit"/>
</form>
```



© 2010 Haim Michael. All Rights Reserved.



# The datalist Element



© 2010 Haim Michael. All Rights Reserved.

## The required Attribute

- ❖ We can use this attribute in order to specify specific controls in which the user must enter a value before the form is submitted.

© 2010 Haim Michael. All Rights Reserved.

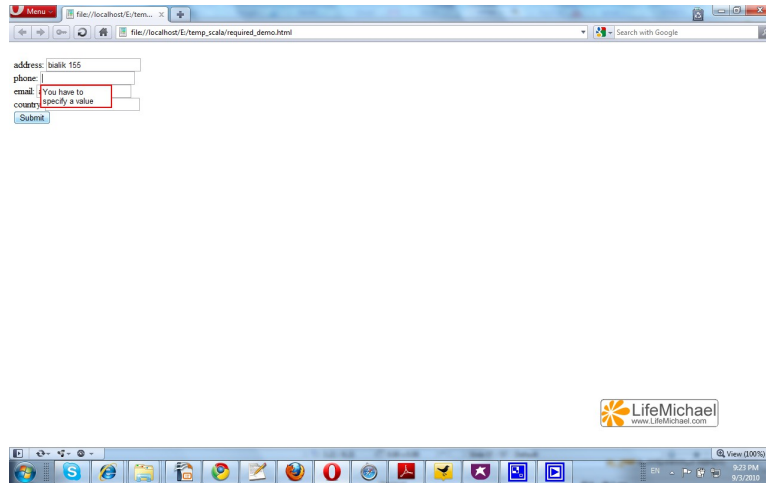
## The required Attribute

```
<form action="date.php" method="get">  
<br>address: <input type="text" name="useraddress" />  
<br>phone: <input type="text" name="phone" required />  
<br>email: <input type="text" name="email_address" />  
<br>country: <input type="text" name="country" />  
<br><input type="submit"/>  
</form>
```



© 2010 Haim Michael. All Rights Reserved.

## The required Attribute



The screenshot shows a web browser window with the address bar displaying "file://localhost/E:/temp/required\_demo.html". The page contains a form with the following fields:

- address: baikk 155
- phone:
- email: (You have to specify a value)
- country:

A red box highlights the email field, indicating it is required. Below the fields is a "Submit" button. The browser's taskbar at the bottom shows various application icons and the system clock displaying 9:21 PM on 9/3/2010. A "LifeMichael" logo is visible in the bottom right corner of the browser window.

© 2010 Haim Michael. All Rights Reserved.

## Forms Validation

- ❖ Each user control has a `ValidityState` object that tells whether the specific control holds valid data or not.
- ❖ Referring the `validity` attribute of a user control object we get a reference for its `ValidityState` object.

```
...  
val myCheck = document.myForm.firstName.validity;  
...
```

© 2010 Haim Michael. All Rights Reserved.

## Forms Validation

- ❖ Referring the `valid` attribute each `ValidityState` object has we shall get `true` if all eight validation tests pass successfully.

```
...  
val myCheck = document.myForm.firstName.validity;  
if (myCheck.valid)  
{  
    //data is valid  
}  
...
```

© 2010 Haim Michael. All Rights Reserved.

## Validity Constraints

- ❖ There are eight possible validity constraints on each and every form element. Each one of them can be accessed through the `ValidityState` object. That object has eight fields named after these eight possible validity constraints.
- ❖ We can check separately each and every one of them.

© 2010 Haim Michael. All Rights Reserved.

## Validity Constraints

- ❖ The `ValidityState` object is a live object. Once we get a reference for a `ValidityState` object of a specific control we can keep holding it. The validity checks it returns are updated when changes occur.

© 2010 Haim Michael. All Rights Reserved.



## The `valueMissing` Constraint

- ❖ This constraint is relevant when using the `required` attribute.
- ❖ Unless a value is entered the value of this constraint is true.

© 2010 Haim Michael. All Rights Reserved.

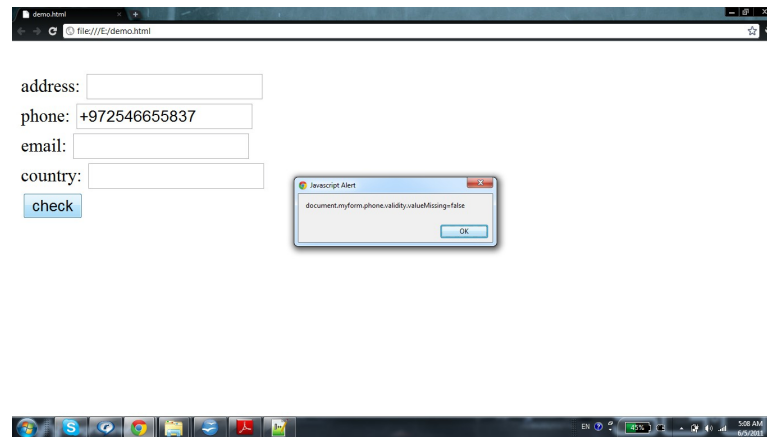
## The valueMissing Constraint

```
<form action="date.php" method="get" name="myform">
<br>address: <input type="text" name="useraddress" />
<br>phone: <input type="text" name="phone" required />
<br>email: <input type="text" name="email_address" />
<br>country: <input type="text" name="country" />
<br><input type="button" value="check" onclick="check()" />
</form>
<script>
function check()
{
    alert("document.myform.phone.validity.valueMissing="+
        document.myform.phone.validity.valueMissing);
}
</script>
```



© 2010 Haim Michael. All Rights Reserved.

## The valueMissing Constraint



© 2010 Haim Michael. All Rights Reserved.

## The `typeMismatch` Constraint

- ❖ This constraint checks to verify the type of the entered value matches with the expected value (number, tel, email etc.).

© 2010 Haim Michael. All Rights Reserved.

## The `patternMismatch` Constraint

- ❖ This constraint checks to verify the value matches with the pattern we set. The pattern is set using the `pattern` attribute.

© 2010 Haim Michael. All Rights Reserved.

## The `tooLong` Constraint

- ❖ This constraint checks to ensure the entered value doesn't exceed the maximum length we set.
- ❖ We use the `maxLength` attribute in order to set the maximum allowed length.

© 2010 Haim Michael. All Rights Reserved.

## The `rangeOverflow` Constraint

- ❖ This constraint checks to ensure the entered value isn't bigger than the maximum allowed value we set.
- ❖ We use the `max` attribute in order to set the maximum allowed value.

© 2010 Haim Michael. All Rights Reserved.

## The `rangeUnderflow` Constraint

- ❖ This constraint checks to ensure the entered value isn't smaller than the minimum value we set.
- ❖ We use the `min` attribute in order to set the minimum allowed value.

© 2010 Haim Michael. All Rights Reserved.



## The `stepMismatch` Constraint

- ❖ This constraint checks to ensure the entered value doesn't violate the step we set.
- ❖ We use the `step` attribute in order to set the step value.

© 2010 Haim Michael. All Rights Reserved.

## The `customError` Constraint

- ❖ We can write JavaScript code that explicitly sets an error for a specific control. Calling `setCustomValidity(message)` on a specific control we get two outcomes. The control is placed into a an error state and a customized error message is set.
- ❖ When referring `customError` we shall get true if the control we refer was placed into a custom error state.

© 2010 Haim Michael. All Rights Reserved.

## The `output` Element

- ❖ The `output` element is been used for presenting the result of some sort of calculation that took place within the scope of a form.
- ❖ We can add the `for` attribute in order to specify (space separated) the ids of those elements that took place in the calculation.

© 2010 Haim Michael. All Rights Reserved.

## The output Element

- ❖ We can use the `form` attribute in order to specify the form owner of this output element. The value of this attribute should be the id of a specific form in the very same page.
- ❖ We can use the `name` attribute in order to name the output element we are using.

© 2010 Haim Michael. All Rights Reserved.

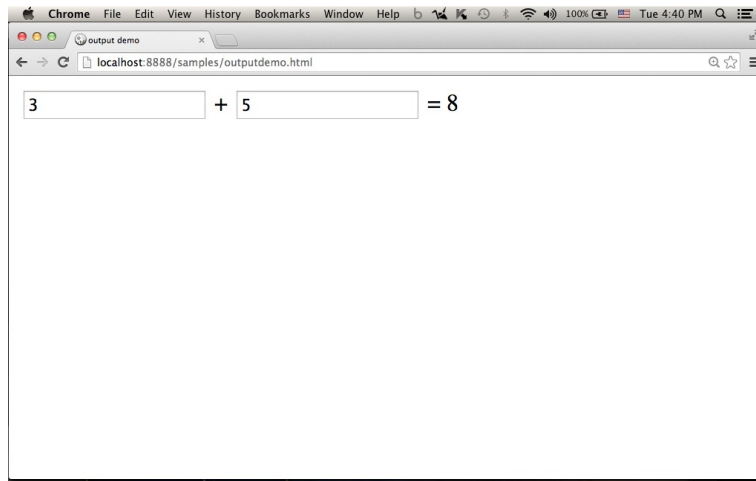
## The output Element

```
<!DOCTYPE html>
<html>
<head>
  <title>output demo</title>
</head>
<body>
<form oninput="result.value =
  parseInt(numA.value) + parseInt(numB.value)">
  <input name="numA" type="number" > +
  <input name="numB" type="number" > =
  <output name="result"></output>
</form>
</body>
</html>
```



© 2010 Haim Michael. All Rights Reserved.

## The output Element



© 2010 Haim Michael. All Rights Reserved.