The University Of Sheffield.

COM1008: Web and Internet Technology

Lecture 12: Forms and events

Dr. Steve Maddock
s.maddock@sheffield.ac.uk

1. Introduction

- The Web browser supports the following advanced features:
  - A programming environment to create and delete and manipulate elements of the Web page (Tuesday's lecture)
  - An event-driven programming model to respond to user input
- Today we'll focus on two things:
- HTML Forms
  - Forms make it possible for Web sites to collect information from their visitors, e.g. order form on Amazon
- Events
  - Handle user interaction with elements of the Web page
  - *Today*: focus on forms
  - *Next week*: Graphics and events on the canvas

2. Forms

- A form is composed of a number of labelled fields and buttons

Name:

Sex: Male Female

green

Likes: Football Rugby Union Golf Tiddlywinks

Add your comments here...

Enter my information

Product Care Programme

If you have a problem with any Desperate Software product, please fill in the form below. Fields marked with a * are required.

Your name: * First Last

Your email address: *

Country of residence: UK

Telephone number:

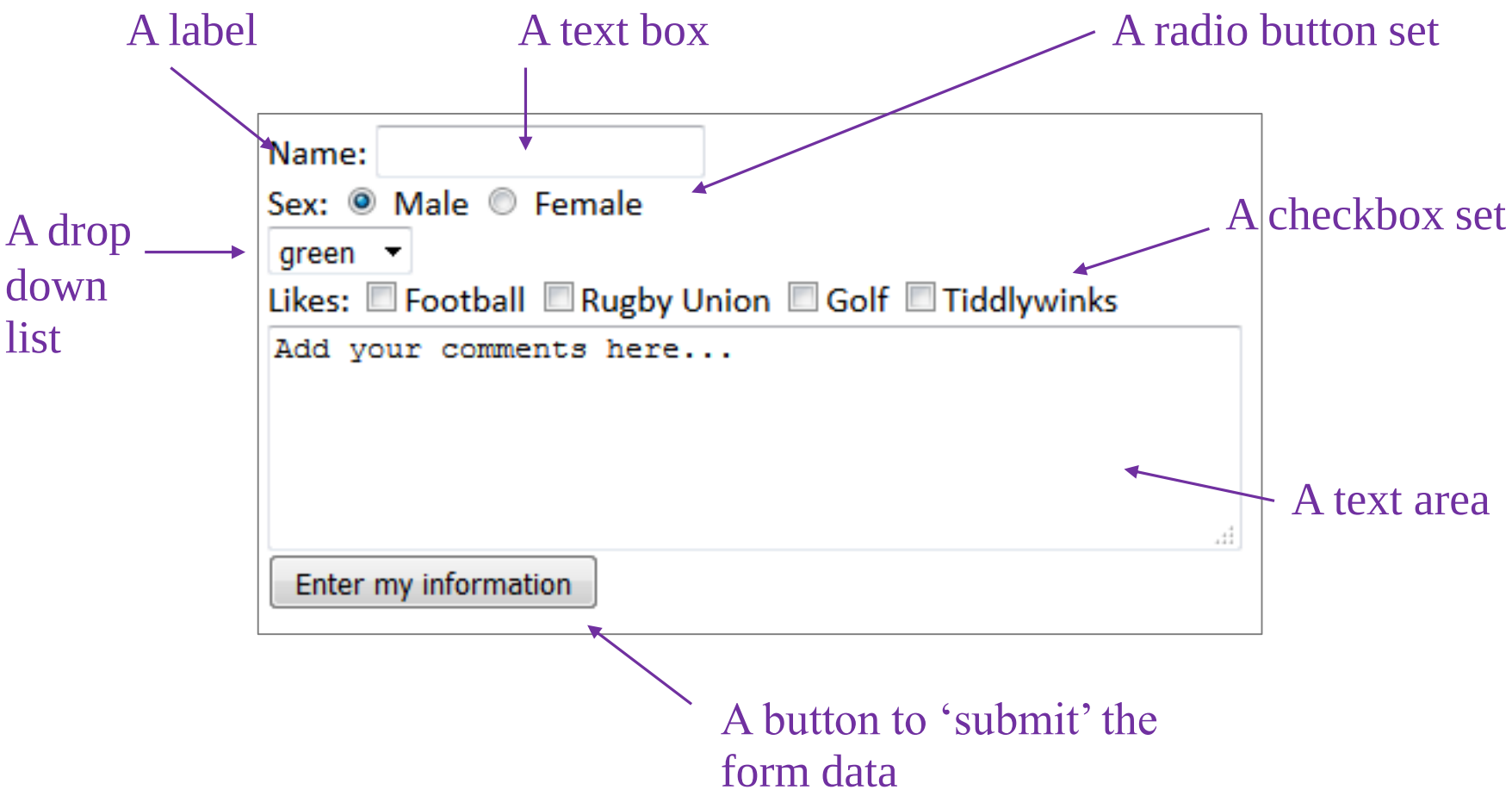
Customer ID number: * XX-nnnn

Please describe your problem as clearly as possible. *

Submit form

Chapman and Chapman, 06

3. Creating a HTML form

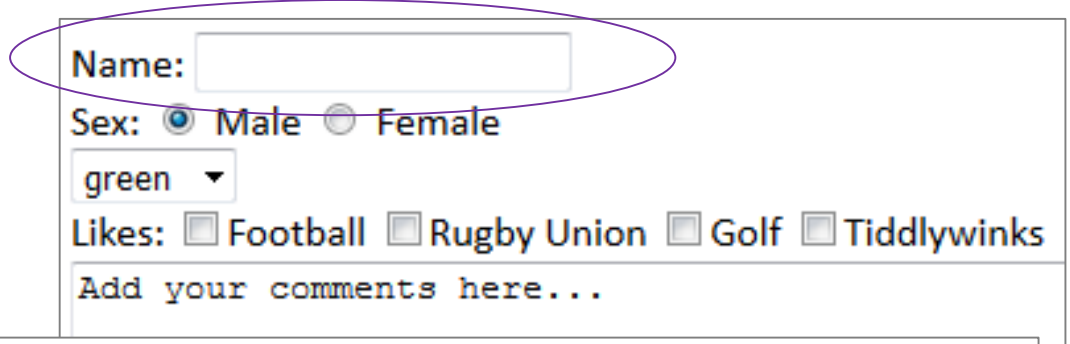


3. Creating a HTML form

- The HTML file uses the ‘form’ element

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8" />
  <title>Form examples</title>
</head>

<body>
  <form name="myform" action="">
     <!-- contents of the form go here -->
  </form>
</body>
</html>
```

3. Creating a HTML form

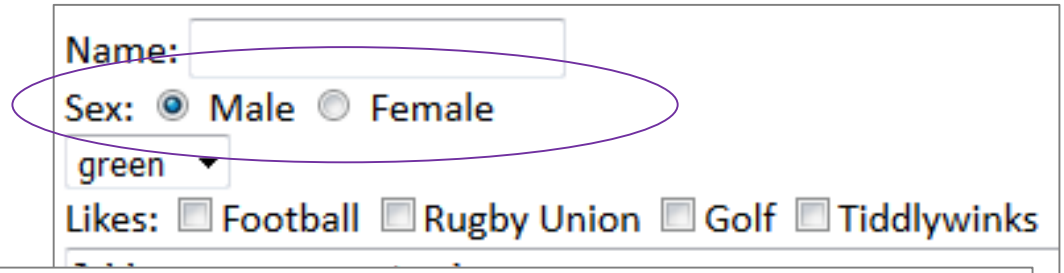



```
<form name="myform" action="">

    <label for="GET-name">Name:</label>
    <input id="GET-name" type="text" name="name"
           maxlength="40" size="20" />
    <br>

    <!-- rest of the form goes here -->
</form>
```

- A label is used for the name of the field
- The for attribute matches the id attribute of the matching input text field

- Each input field has a number of attributes associated with it
- Both **name** and **id** attributes should (typically) be set to the same value:
  - name is used in the submission of the form
  - id can be used to find the specific field in the form, e.g. for CSS styling
- The **value** attribute, if supplied, defines the default value for the input text field

3. Creating a HTML form




```
<form name="myform" action="">
    <!-- rest of the form goes here -->

    <label>Sex:</label>
    <input id="GET-male" type="radio" name="sex"
           value="male" checked>
    <label for="GET-male">Male</label>
    <input id="GET-female" type="radio" name="sex"
           value="female">
    <label for="GET-female">Female</label>

    <!-- rest of the form goes here -->
</form>
```

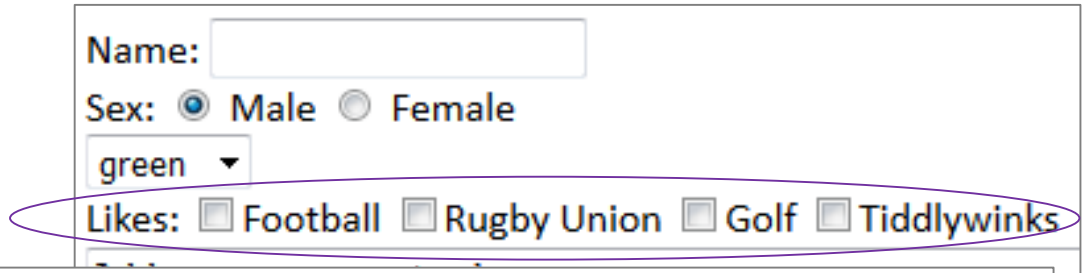
- A ‘radio button set’ shares the same name so that only one of them can be ‘on’
- The initial one that is ‘on’ is indicated by the attribute ‘checked’
  - `checked="checked"` or just `checked`

3. Creating a HTML form


```
<form name="myform" action="">
    <!-- rest of the form goes here -->

    <select name="eye colour">
      <option value="blue">blue</option>
      <option value="brown">brown</option>
      <option value="green" selected>green</option>
      <option value="other">other</option>
    </select>
    <br>

    <!-- rest of the form goes here -->
</form>
```

- The select element is used to create a drop-down list
- The option element is used to define each option
- The initial one that is ‘on’ is indicated by the attribute selected

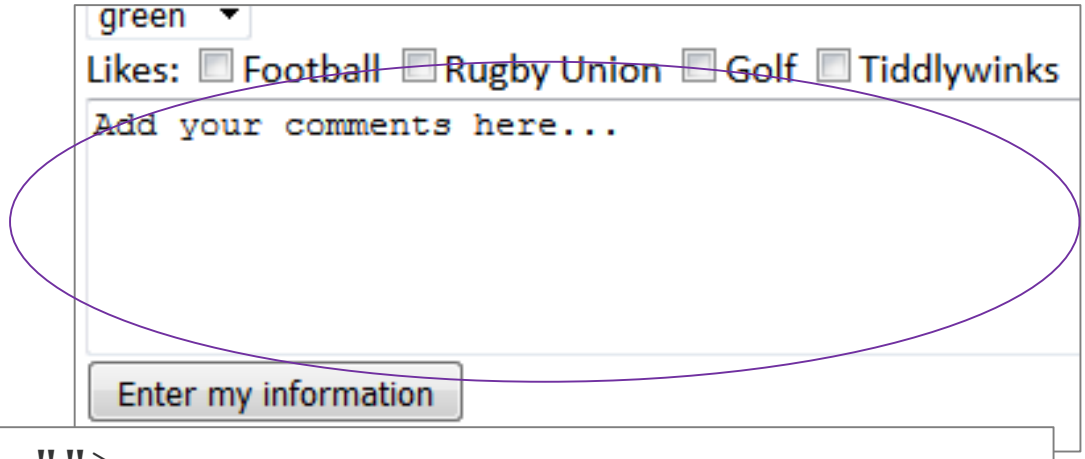
3. Creating a HTML form




```
<form name="myform" action="">
    <!-- rest of the form goes here -->
    <label>Likes:</label>
    <input type="checkbox" name="likes"
           value="football">Football</input>
    <input type="checkbox" name="likes"
           value="rugby union">Rugby Union</input>
    <input type="checkbox" name="likes"
           value="golf">Golf</input>
    <input type="checkbox" name="likes"
           value="tiddlywinks">Tiddlywinks</input>
    <br>
    <!-- rest of the form goes here -->
</form>
```

- A checkbox set shares the same name
- The attribute checked can be used to initially set whether or not the specific checkbox is selected

3. Creating a HTML form




```
<form name="myform" action="">
    <!-- rest of the form goes here -->

     <textarea name="comments" rows="5" cols="50" >
       Add your comments here...
     </textarea>
    <br>

    <input type="submit" value="Enter my information">

</form>
```

- The size of the text area is set using the rows and cols attributes
- We'll revisit the <input type="submit"… button later.

4. Form elements in HTML5

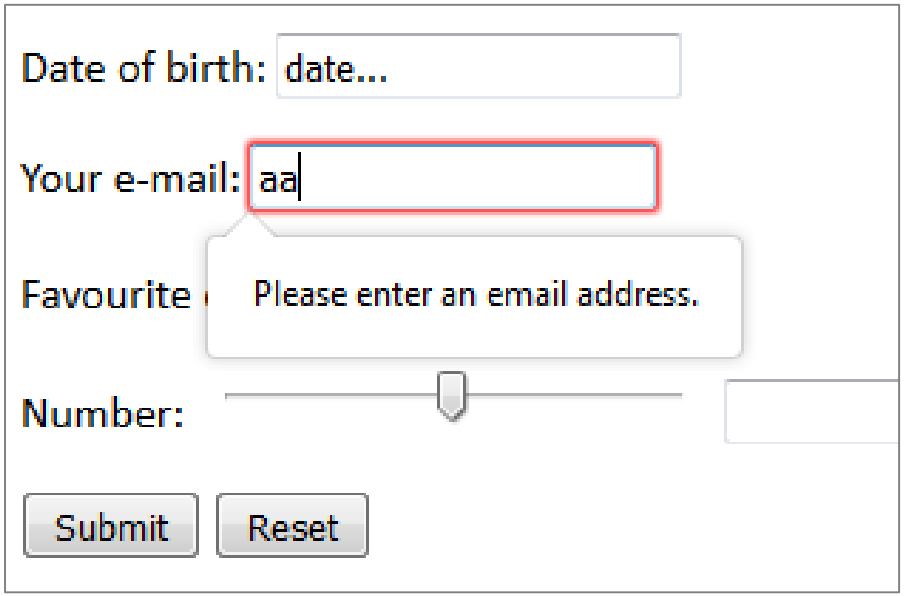
- There are other form elements we haven't met in this example:
  - <button>, <fieldset>, <file>, <object>, <keygen>, <meter>, <output>, <progress>, …

- In addition, in HTML5, form input elements can exist outside the main form
  - Use the 'form' attribute to relate them

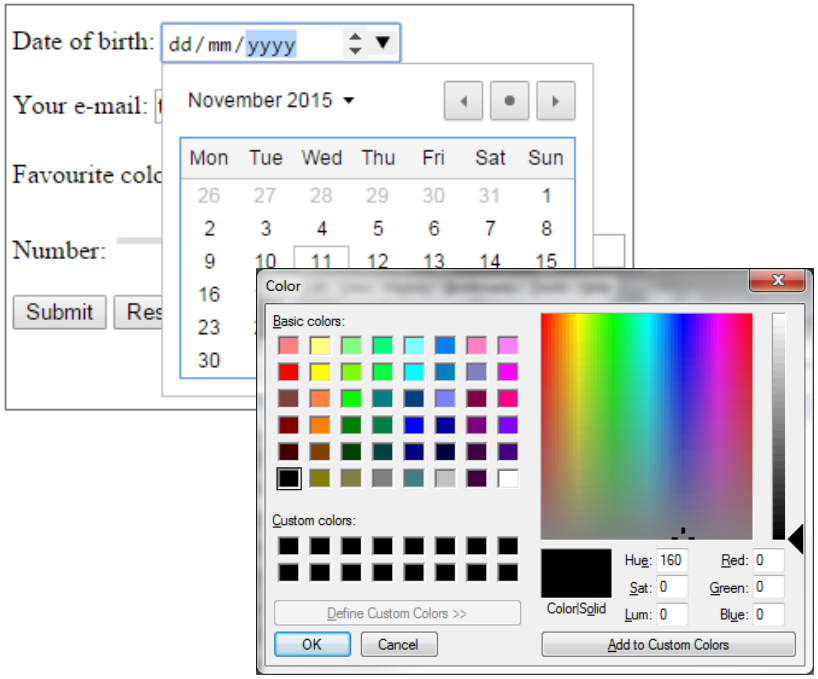
```
<form id="foo">
  <input type="text">...
</form>

<textarea form="foo"></textarea>
```

4.1 HTML5: New input types

- Automatic validation for these:
  - <input type="email">
  - <input type="url">



- Other input types
  - date, time, datetime, month, week, number, range, search, tel, color



- Some browsers supply widgets for some input types
  - Chrome calendar widget
  - Firefox colour widget

4.2 New form attributes in HTML5

- We saw the use of checked and selected in previous examples
- New attributes include: autofocus, placeholder, required, multiple, pattern, autocomplete, min, max, step

```
<p>
<label for="myemail">Your e-mail:</label>
<input type="email" name="myemail" id="myemail"
       value="type your email..." maxlength="40" size="20"
       required="required" />
</p>
```

```
<p>
<label for="myemail">Your e-mail:</label>
<input type="email" name="myemail" id="myemail"
       value="type your email..." maxlength="40" size="20"
       required />
</p>
```

4.2 New form attributes in HTML5

Your name: type your name...

Submit Reset

```
<p>
  <label for="from">Your name:</label>
  <input type="text" name="from" id="from"
         value="type your name..." maxlength="40" size="20" />
</p>
```

- The **placeholder** attribute can be used to define the default, initial value for the field

Your name: type your name...

Submit Reset

```
<p>
<label for="from">Your name: <input type="text" name="from"
id="from" placeholder="type your name..." maxlength="40"
size="20" />
</p>
```

5. Dealing with forms

Database

Web server
(form processing script)

Client
(validation or other actions)

Form submission data

Response page

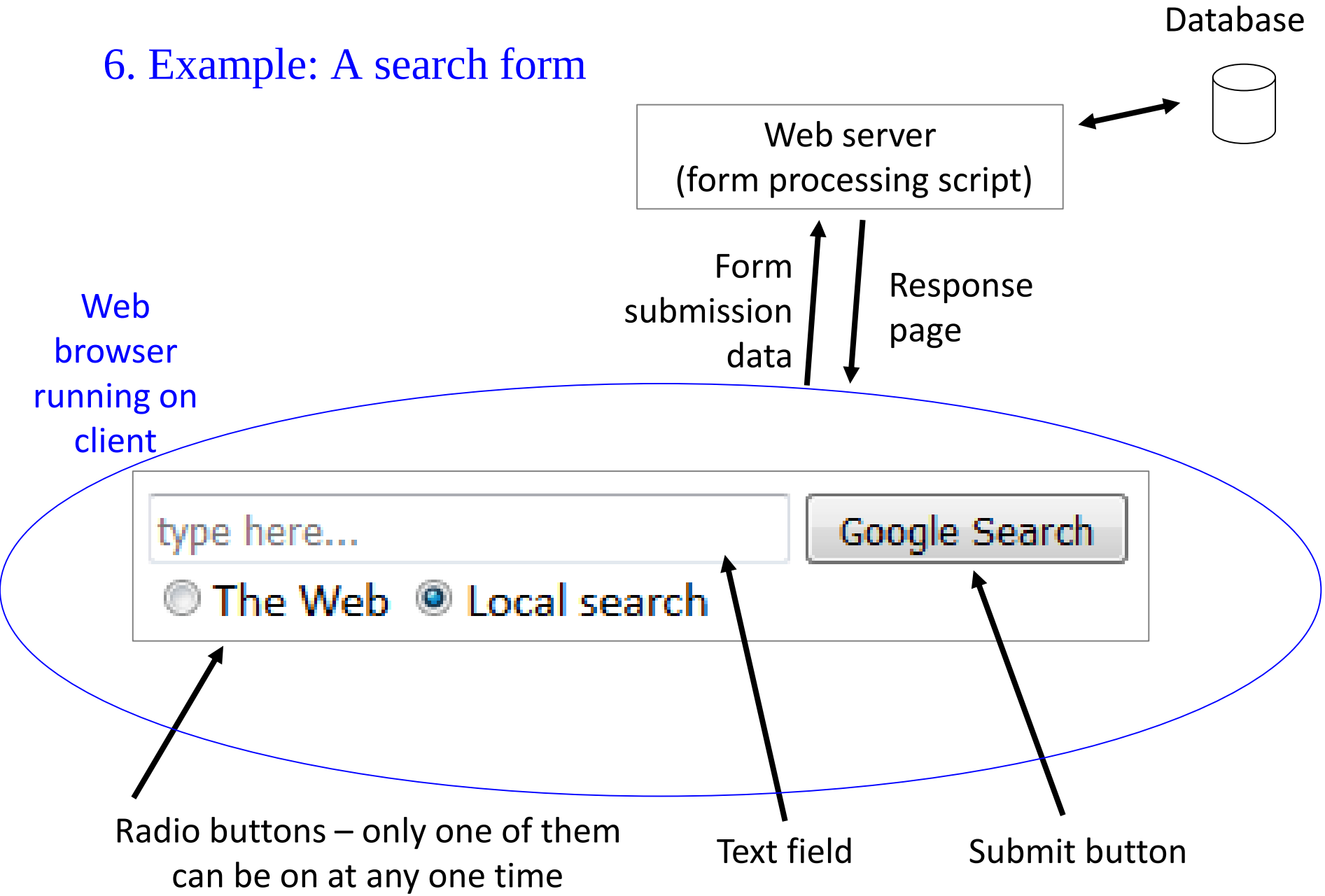
E-mail

Name:
Sex: Male Female
green
Likes: Football Rugby Union Golf Tiddlywinks
Add your comments here...
Enter my information

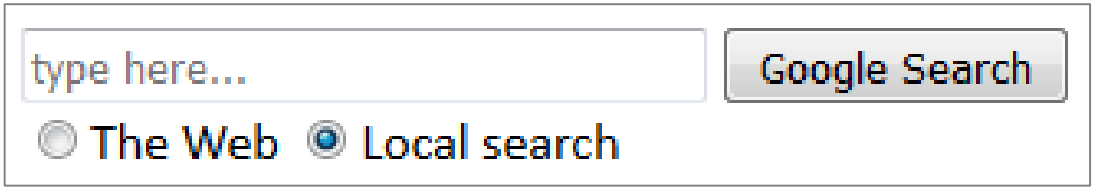
Web browser running on client

HTML5 - validation is inbuilt for common data types

6. Example: A search form

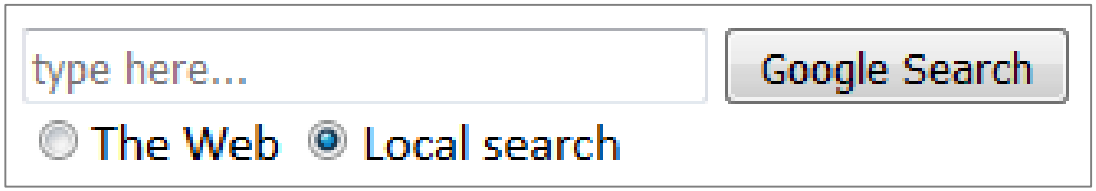


6.1 The form




```
<form action="http://www.google.com/search" method="get">

  <input type="text" name="q" size="31" maxlength="255"
         placeholder="type here..." />

  <input type="submit" value="Google Search" /><br />

  <input type="radio" name="sitesearch" value="" />The Web
  <input type="radio" name="sitesearch"
         value="http://staffwww.dcs.shef.ac.uk/people/S.Maddock"
         checked />Local search<br />
</form>
```

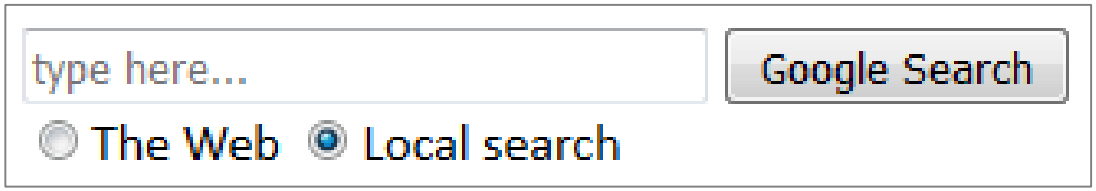
6.2 What to do with the form data




```
<form action="http://www.google.com/search" method="get">
  <!-- rest of form here -->
  <input type="submit" value="Google Search" /><br />
  <!-- rest of form here -->
</form>
```

- A form typically has a submit button
- This is created using the input type submit
- When it is clicked the data in the form is transferred to the program specified in the action attribute of the form

6.2 What to do with the form data

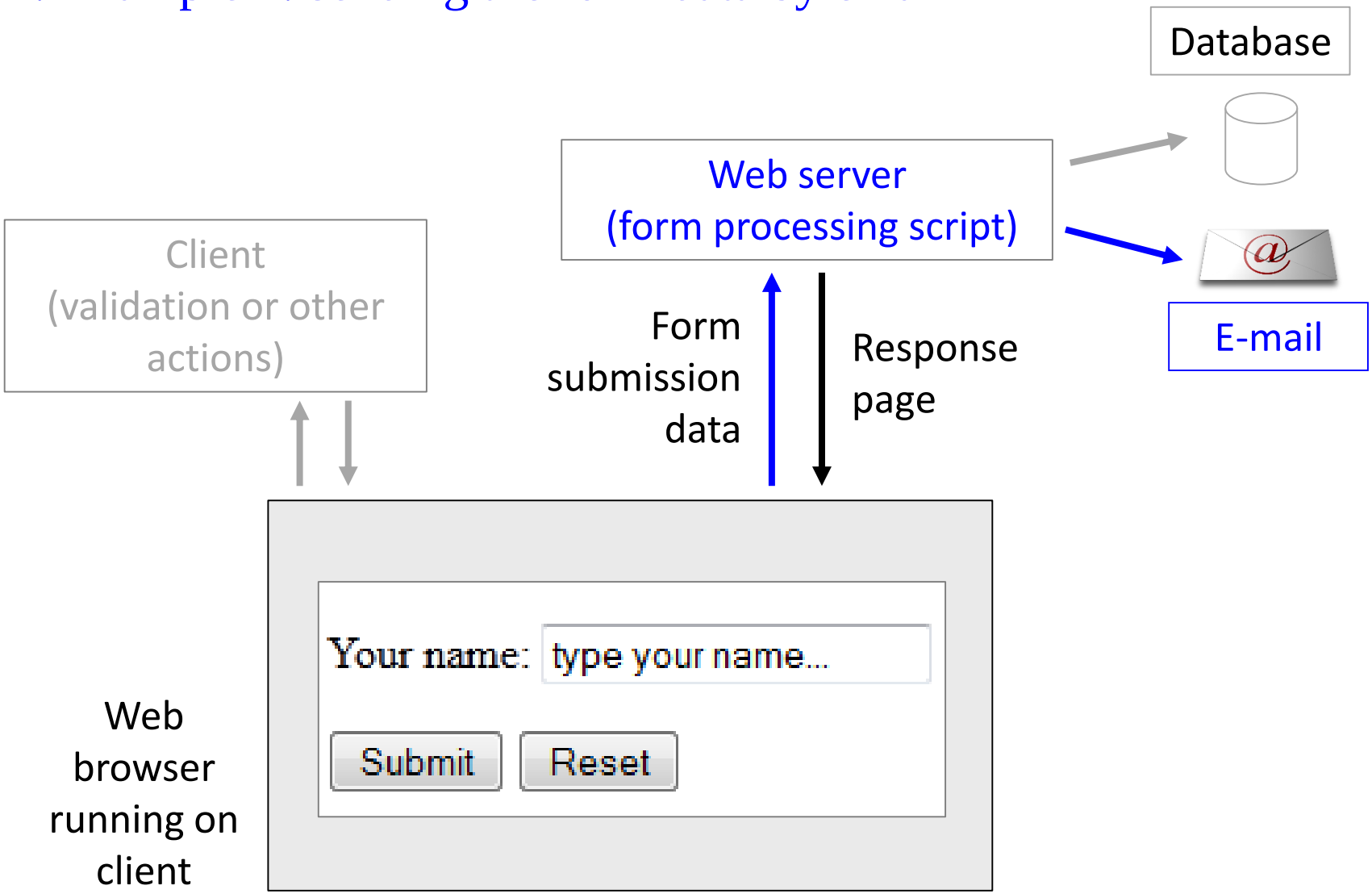



```
<form action="http://www.google.com/search" method="get">
  <!-- rest of form here -->
</form>
```

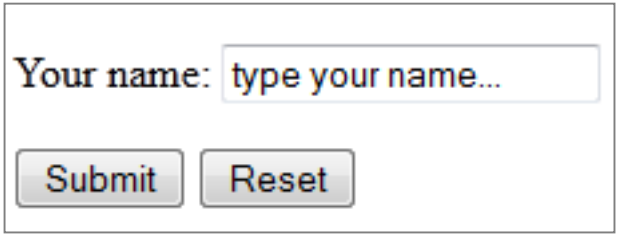
```
<form name="myName" action="howToHandleTheData"
      method="howToSendTheData">
  <!-- form contents go here -->
</form>
```

- **action** – URL of the program that will process the data
- **method** – determines how data is sent to the server:
  - get – use when processing of data has no side-effects
  - post – used if there are side-effects, e.g. update a database

7. Example 2: sending the form data by email



7. Example 2




```
<form name="myform"
      action="http://www.dcs.shef.ac.uk/cgi-bin/FormMail.pl"
      method="get">

  <input type="hidden" name="recipient"
         value="s.maddock@sheffield.ac.uk" />

  <p>
    <label for="from">Your name:</label>
    <input type="text" name="from" id="from"
           placeholder="type your name..."
           maxlength="40" size="20" />
  </p>

  <p>
    <input type="submit" name="submit" id="submit"
           value="Submit" />
    <input type="reset" />
  </p>

</form>
```

7.1 Button types

Your name: type your name...

Submit Reset

```
<form name="myform"
      action="http://www.dcs.shef.ac.uk/cgi-bin/FormMail.pl"
      method="get">

  <input type="hidden" name="recipient"
         value="s.maddock@sheffield.ac.uk" />

  <p>
    <l
    <i

  </p>

  <p>
    <input type="submit" name="submit" id="submit"
           value="Submit" />
    <input type="reset" />
  </p>

</form>
```

- Three button types can be created for a form:
  - submit button – submits the form
  - reset button – resets all controls to their initial values
  - Anonymous button – attach a client-side script to these (see later)

7.2 action

Your name: type your name...

Submit Reset

```
<form name="myform"
      action="http://www.dcs.shef.ac.uk/cgi-bin/FormMail.pl"
      method="get">

  <input type="hidden" name="recipient"
         value="s.maddock@sheffield.ac.uk" />
```

- The Department's Web manager has made available a cgi script for use with forms
- It e-mails the 'recipient' the data in the form as a set of pairs: (name, value) for each field in the form
- *More info*: http://www.scriptarchive.com/readme/formmail.html

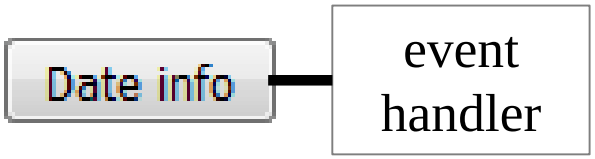
- The cgi script requires one specific input command to be present in the form
- The type is set to 'hidden' so that it does not appear on the Web page
- name="recipient" must be present
- The **value** is the e-mail address to send the contents of the form to

```
</form>
```

8. Event handling

- Dealing with user interaction on a web page
- Some example events:
  - Loading a document, Clicking a button , Validating user entry in a form, Browser screen changing size
- Attach an event handler to an element
  - A JavaScript function
  - Called when the user interacts with the element, e.g. the click of a button
- There are lots of events for different HTML elements
  - https://developer.mozilla.org/en-US/docs/Web/Events

JavaScript program *attaches* an event handler (a function) to the button to catch a specific event



User clicks the button with the mouse and the relevant event handler for that event is automatically called by the JavaScript system

Event Name	Meaning
load	All the content of a page has bee
unload	The document has been removec window
click	The mouse was clicked with the c the element.
dblclick	The mouse was double-clicked wi sor over the element.
mousedown	The mouse button was pressed wi sor over the element.
mouseup	The mouse button was released cursor over the element.
mouseover	The cursor was moved onto the e
mousemove	The cursor was moved while it wa element.
mouseout	The cursor was moved away fror ment.
focus	The element has received the fo will accept input.
blur	The element has lost the focus.
keypress	A key was pressed and released

Chapman and Chapman, 06

9. Example: button and click event

Date info

```
<body>
  <p>
    <button name="dateinfo" id="dateinfo">Date info</button>
  </p>
  <script src="./js/event1.js"></script>
</body>
```

- The button element represents a clickable button
- name field only required is part of a form
- id is more useful for locating the button on the page

```
function printDateInf
  var now = new Date();
  alert(now);
}

// main program
var elementB = document.getElementById('dateinfo');
console.log(elementB);
elementB.addEventListener('click', printDateInfo, false);
```

9. Example: button and click event

Date info

```
<body>
  <p>
    <button name="dateinfo" id="dateinfo">Date info</button>
  </p>
  <script src="./js/event1.js"></script>
</body>
```

- The button element is found using its id
- The event listener is added using **addEventListener**

```
function printDateInfo() {
  var now = new Date();
  alert(now);
}

// main program
var elementB = document.getElementById('dateinfo');
console.log(elementB);
elementB.addEventListener('click', printDateInfo, false);
```

9. Example: button and click event

Date info

```
<body>
  <p>
    <button name="dateinfo" id="dateinfo">Date info</button>
  </p>
  <script src="./js/event1.js"></script>
</body>
```

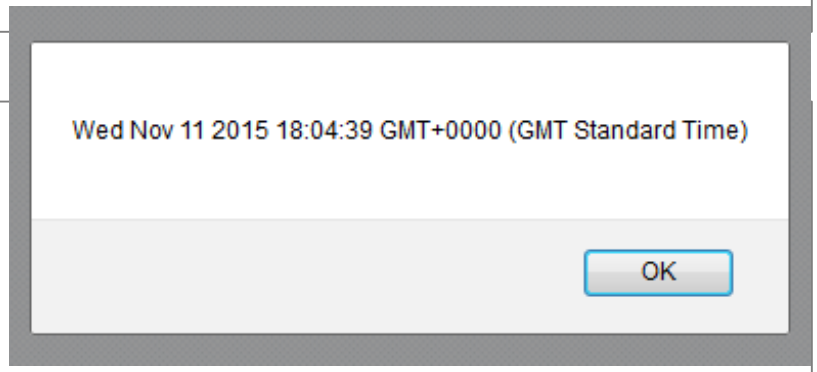
```
function printDateInfo() {
  var now = new Date();
  alert(now);
}

// main program
var elementB = document.getElementById('dateinfo');
console.log(elementB);
elementB.addEventListener('click', printDateInfo, false);
```

Details

- Every element has a set of properties, e.g. onclick, to handle specific events
- Thus the function is stored in this property (as an object)

9. Example: button and click event

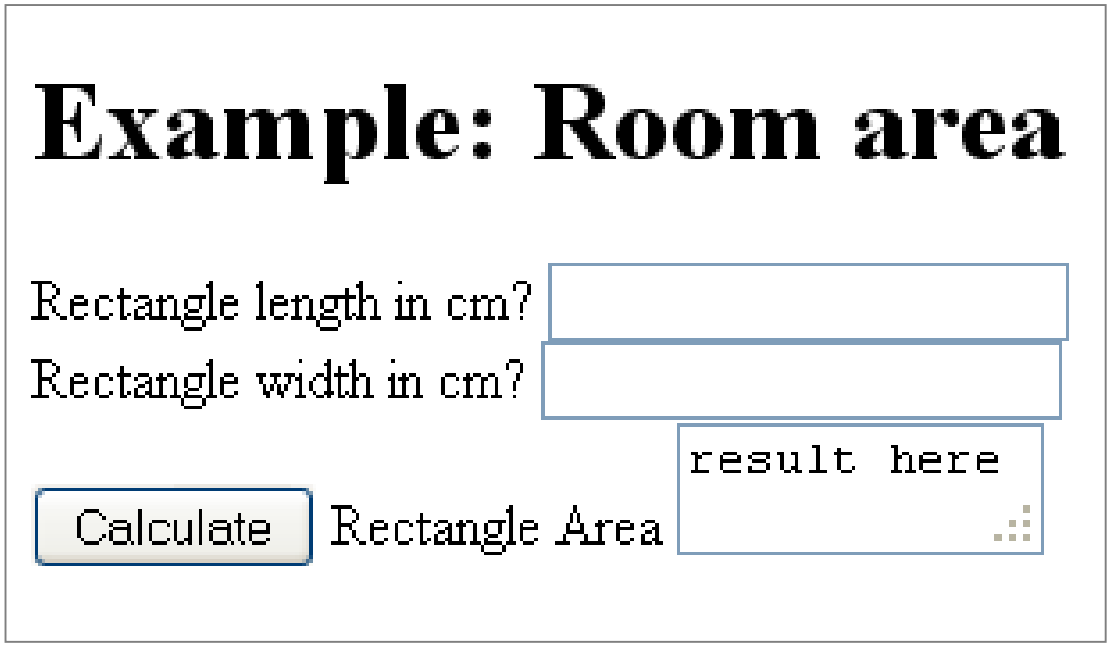

```
<body>
  <p>
    <button name="dateinfo" id="dateinfo">Date info</button>
  </p>
  <script src="./js/event1.js"></script>
</body>
```



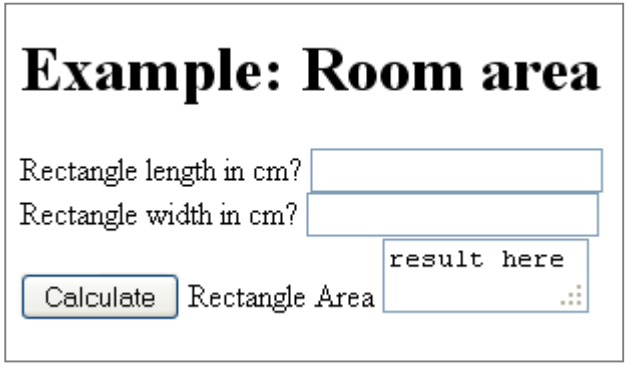

```
function printDateInfo() {
  var now = new Date();
  alert(now);
}

// main program
var elementB = document.getElementById('dateinfo');
console.log(elementB);
elementB.addEventListener('click', printDateInfo, false);
```

10. Example: input and output using a form

- Use the onclick event handler to make use of the data input in a form to calculate the data to insert in another part of the form
- Use an id for each part of the form so it is easy to access



10.1 Create the form



- First, create the form

```
<form id="calcarea" name="calcarea" >
  <label for="mylength">Rectangle length in cm?</label>
  <input type="text" id="mylength" name="mylength" required  />
  <br />
  <label for="mywidth">Rectangle width in cm?</label>
  <input type="text" id="mywidth" name="mywidth"
         required="required"  />
  <br />
  <input type="button" name="calculate" id="calculate"
         value="Calculate" />
  <label for="output">Rectangle Area</label>
  <textarea cols="10" rows="1" id="output" name="output" >
    result here
  </textarea>
</form>
```

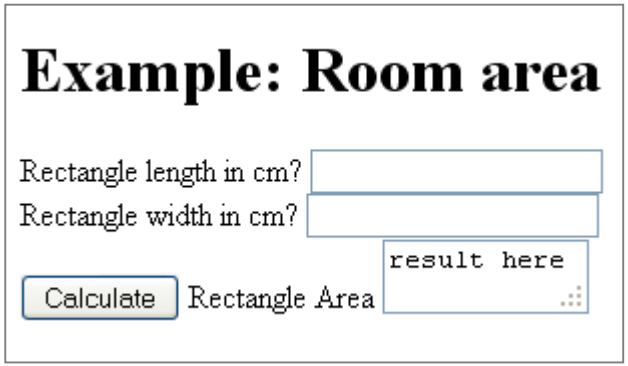
Text field

Text field

Button

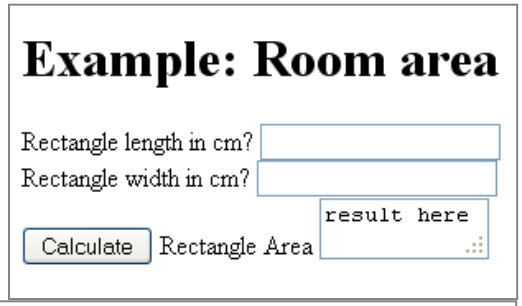
Text area

10.2 Attach the event handler




```
function area() {
  var length = document.getElementById('mylength').value;
  var width = document.getElementById('mywidth').value;
  document.getElementById('output').value = length*width;
}

function clearOutput() {
  document.getElementById('output').value = "";
}

var myButton = document.getElementById('calculate');
myButton.addEventListener('click', area, false);
```

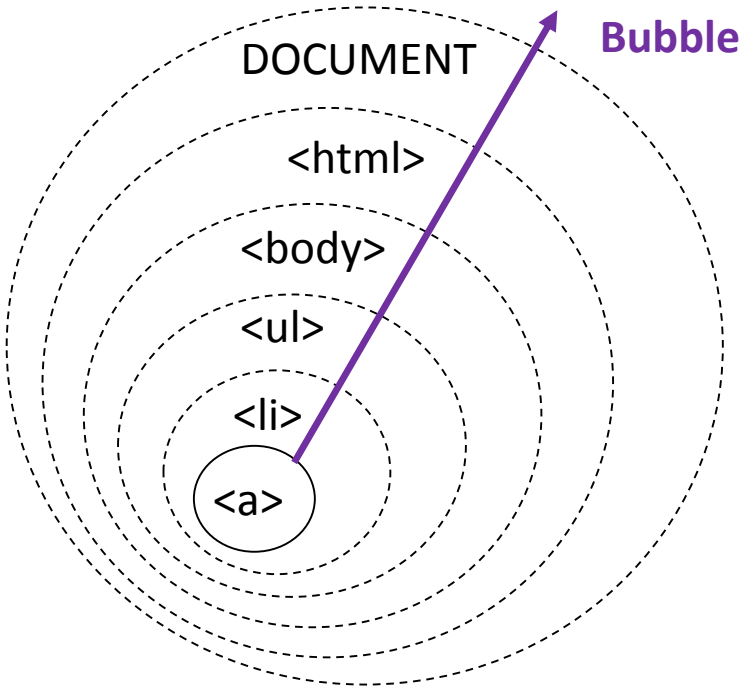
10.3 Further event handlers

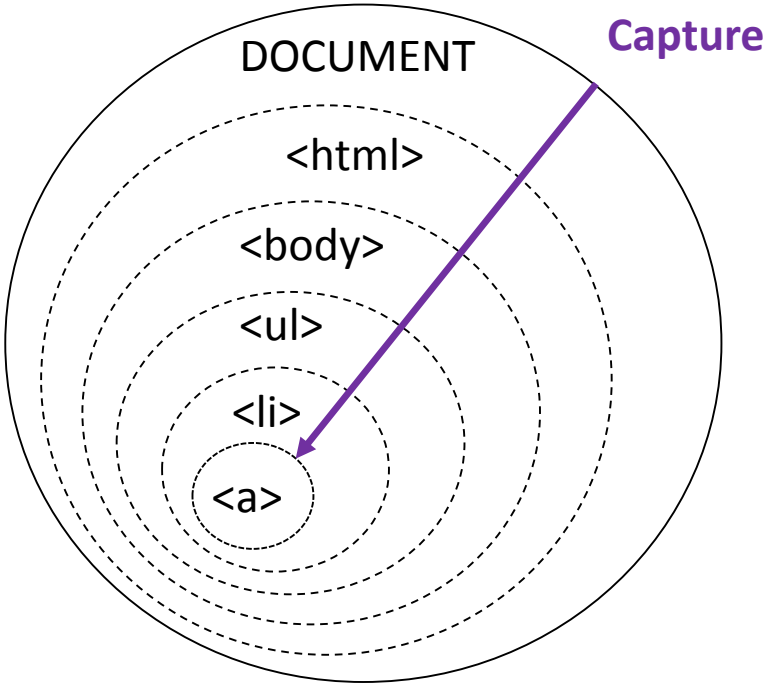



```
function area() {
  var length = document.getElementById('mylength').value;
  var width = document.getElementById('mywidth').value;
  document.getElementById('output').value = length*width;
}

function clearOutput() {
  document.getElementById('output').value = "";
}

var myButton = document.getElementById('calculate');
myButton.addEventListener('click', area, false);

var lengthInputElement = document.getElementById('mylength');
lengthInputElement.addEventListener('blur', clearOutput, false);

var widthInputElement = document.getElementById('mywidth');
widthInputElement.addEventListener('blur', clearOutput, false);
```

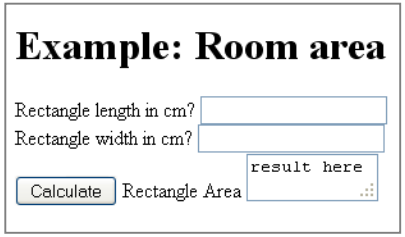
11. Bubbling and capturing

- If an event listeners are attached at multiple levels in a nested hierarchy of elements, then the default is that the event bubbles outwards and is handled by each level in turn
  - `myButton.addEventListener('click', area, false);`
- The alternative is event capturing, where true is set for the event listener





12. Form or GUI?



- *Previous example:* the input elements were wrapped in the form element

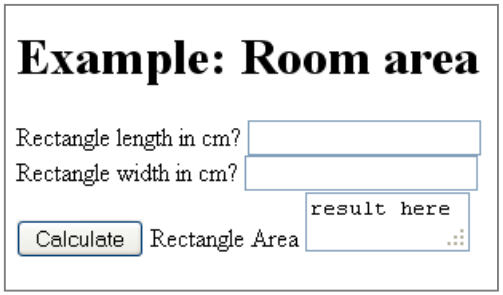
```
<form id="calcarea" name="calcarea" >
  <label for="mylength">Rectangle length in cm?</label>
  <input type="text" id="mylength" name="mylength" required  />
  <!-- rest of form here -->
</form>
```

- Since there is no intention of submitting this data to a Web server in this example, we do not need to use the form element
  - *Mozilla*: "The HTML <form> element represents a document section that contains interactive controls to submit information to a web server." [https://developer.mozilla.org/en/docs/Web/HTML/Element/form]
- Instead we can wrap the interface elements in something else, e.g. a div element, or distribute the input elements anywhere on a Web page in order to create a graphical user interface
  - We'll see more of this when we look at Graphics and the Canvas and events.

13. Summary

- Forms make it possible for Web sites to collect information from their visitors
  - More info: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Forms/
- Events are the result of user interaction
  - E.g. Loading a document, Clicking a button , Validating user entry in a form, Browser screen changing size
- We use an event handler to deal with the events
  - E.g. clicking a button causes a specific JavaScript function to be called
- We'll look at more events when we deal with graphics and the canvas
- *Next lecture*: objects

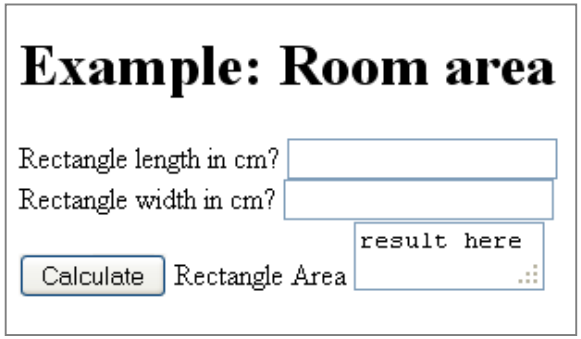
Appendix A. Some more examples



- There are alternative ways to access form fields
- The DOM creates a set of arrays that represent the forms and the elements in them. The name of the field of the form can be used to index these arrays or to access the field directly, as shown below.
- However, the use of DOM methods is preferred.

```
function area() {
  var length =
    document.forms["calcarea"].elements["mylength"].value;
  var width = document.calcarea.mywidth.value;
  document.forms["calcarea"]["output"].value=length*width;
}

// main program
var myButton = document.getElementById('calculate');
myButton.addEventListener('click', area, false);
```

Appendix A. Some more examples



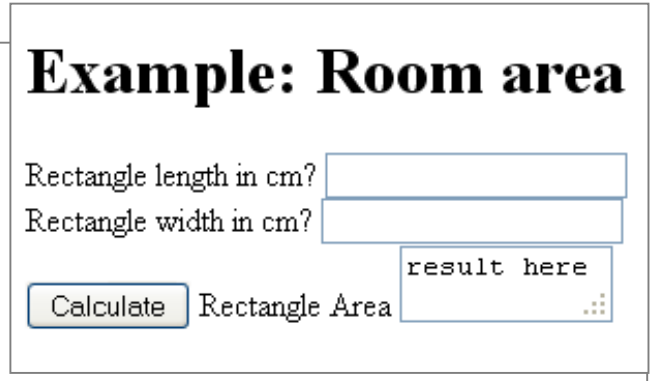
- BEWARE: DON'T DO THIS
- Since the script is run after the page has loaded, document.write() will rewrite the entire page, so all that is seen is the result of the calculation!!

```
function area() {
  var length =
    document.forms["calcarea"].elements["mylength"].value;
  var width = document.calcarea.mywidth.value;
  document.write("Area = " + length*width);
}

// main program
var myButton = document.getElementById('calculate');
myButton.addEventListener('click', area, false);
```

Area = 12

Appendix A. Some more examples

```
function area() {
  console.log("area");
  var length = $("#mylength").val();
  var width = $("#mywidth").val();
  console.log(length+", "+width);
  $("#output").val(length*width);
}

function clearOutput() {
  console.log(clearOutput);
  $("#output").val("");
}

// main program
console.log("running...");

//$("#calculate").on("click", area);   // alternative jQuery approach
$("#calculate").click(area);
```



- Can also use a JavaScript library such as jQuery: http://jquery.com/

Appendix A. Some more examples

- There is also an **output** element. The following example uses the form element to handle the oninput event when the user changes any of the nested items in the form (the event bubbles out to the form level)
- oninput would be discouraged now since it means JavaScript appears in the HTML file; addEventListener would be preferred

```
<body>
  <h1>Example: calculation using the output element</h1>
  <!-- https://developer.mozilla.org/en-
US/docs/Web/HTML/Element/output -->

  <form oninput
          ="result.value=parseInt(a.value)+parseInt(b.value)">
    <input type="range" name="b" value="50" /> +
    <input type="number" name="a" value="10" /> =
    <output name="result"></output>
  </form>
 </body>
```

Appendix A. Some more examples

- Extending the example to demonstrate bubbling of events
- Add event listener to both the div and the slider
- Event fires on the slider first and then the enclosing div

```
<div id="calculation">
  <input type="range" id="b" min="0" max="100" step="1" /> +
  <input type="number" id="a" value="10" /> =
  <output id="result"></output>
</div>

<div>
  <label for="slideroutput">Slider value: </label>
  <output id="slideroutput"></output>
</div>
```

```
<script>
  function changeResult() {
    var a = document.getElementById("a");
    var b = document.getElementById("b");
    var result = document.getElementById("result");
    result.value = parseInt(a.value) + parseInt(b.value);
  }

  function sliderOutput() {
    var b = document.getElementById("b");
    var s = document.getElementById("slideroutput");
    s.value =  parseInt(b.value);
  }

  // main program body
  // example of event bubbling
  var calc = document.getElementById("calculation");
  calc.addEventListener('input', changeResult, false);
  var myinput = document.getElementById("b");
  b.addEventListener('input', sliderOutput, false);
</script>
```

Appendix A. Some more examples

- In HTML5, the content of any element on a page can be made editable by setting the attribute contentEditable to true.
- Using some event handlers, a text editor could be created.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>JavaScript examples</title>
</head>
<body>
  <div contentEditable="true">
     This text can be edited by the user.
  </div>
</body>
</html>
```