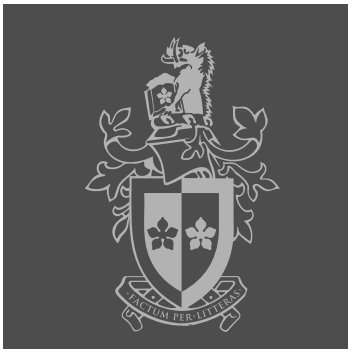
SWINBURNE UNIVERSITY OF TECHNOLOGY

COS10026
Computing Technology Inquiry Project

Lecture 3
HTML 3 – Forms and Structural Elements

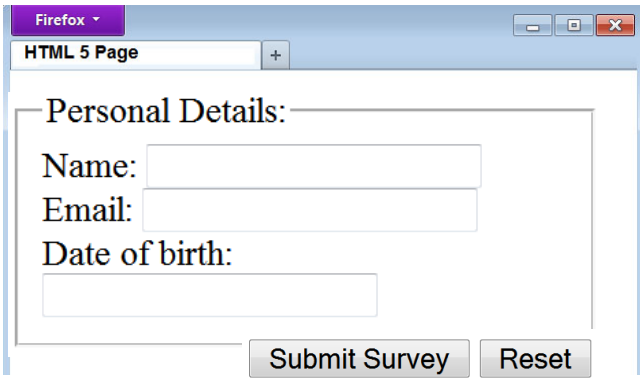


Topics

- Form, Form Attributes and Form Elements
- Form Validation with Regular Expressions
- HTML Structural Elements

Form element

- **<form> … </form>** provides a mechanism to allow a user to enter information into a web page.
- Entered information can be submitted to a server, which in turn can receive the data, process the data and generate a response.
- Possible responses may include:
  - ☐ display information on a web page;
  - ☐ adding data to a database; or
  - ☐ sending an email message.



Form (continued)

1. Form filled

Client requests a web page containing a form by entering a URL on the web browser

Server responds by sending the webpage of the form as HTML

Client

Server

2. Form result

Client clicks the **submit** button on the form which sends the **form data** to the form **action** URL for processing on the server

Server responds by processing the data received then sends the resulting HTML webpage

Form Data

- Form data are submitted in the form of parameter name-value pairs

  parameterName = parameterValue

  - E.g., username = "s123456"

    password = "abcdef"

    gender = "female"

- Multiple such pairs can be sent in one submission to the server

Form attributes (continued)

```
<form id="survey" method="post" action="process.php">
    <!-- Form elements here -->
</form>
```

- **id** - unique identifier of the form
- **method** -HTTP method used to submit the form – **get** or **post**
  - **get** is often used to submit data to obtain something
    e.g. search, or see a product (URL is visible in the browser)
  - **post** is often used to submit data for storage
    e.g. registration (URL is not visible in the browser)
- **action** - URL referring to where the data is to be submitted for processing
  - Absolute path is used if processing is from a different site
- ***Usually*** the <form> element contains all ***form control elements*** and all other form structuring elements.
- Nothing will be displayed or actioned, unless there are ***form control elements***.

HTML Forms

Form control elements: *Note: input is an empty or void element*

- `<input ../>` defines a ***form control*** for the user to enter ***data***.
  - It can have the following attributes:
    `type, name, value, id`
  - The ***type*** attribute specifies the type of the input element, including:

    `text, checkbox, radio, password, submit, reset, hidden, file, image, button`

- `<select>` defines a ***form control*** for the ***selection of options*** from a ***selection list*** and can have the following *attributes:*

  `size, multiple, tabindex, disabled`

- `<textarea>` defines a ***form control*** for the user to enter ***multi-line text input*** and can have the following *attributes:*

  `rows, cols, readonly, tabindex, accesskey, disabled`

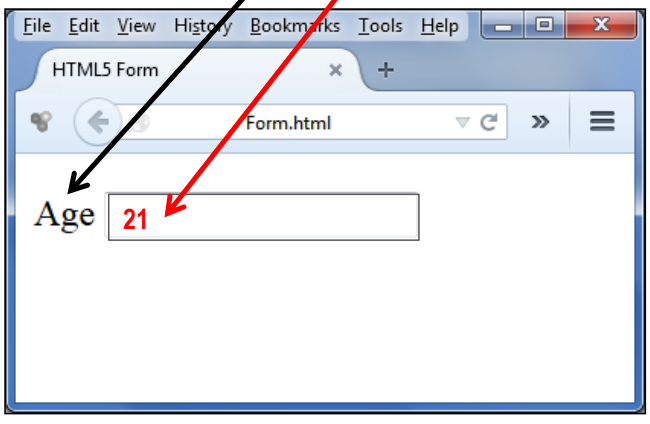
HTML Content

Form Elements: Label

- **<label>…</label>** element associates a ***label*** with a ***form control.***

```
<form id="survey" method="post" action="process.php">
  <p><label for="age">Age</label>

  <input type="text" name="years" id="age"/></p>
 <!--  other form controls -->
</form>
```



Note: Each ***form control*** element has a **name** attribute. Data is passed in **name = value** pairs e.g. **years**=21

When label text is 'clicked' or 'touched' the form control is focused. This provides a better / larger target on touch screens.

Form Elements: Label

Technique #1

Preferred Technique! `for` is explicitly connected to `id`

```
<form id="survey" method="post" action="process.php">
  <p><label for="age">Age</label>
  <input type="text" name="age" id="age" /></p>
</form>
```

Technique #2

```
<form id="survey" method="post" action="process.php">
  <p><label>Age
      <input type="text" name="age" />
  </label></p>
</form>
```

This technique is very common but the **label** is ***not explicitly*** connected to an **id**entifier

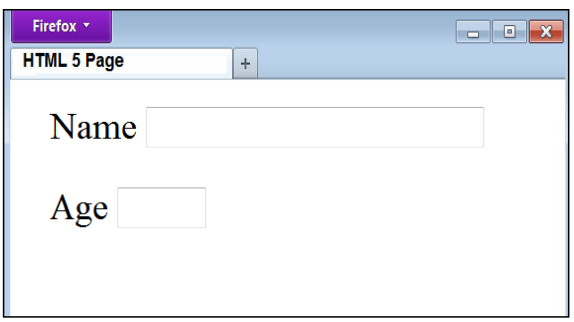
Form Elements: Input Text

```
<p><label>Name
 <input type="text" id= "tbName" name="name" maxlength="40"
                                size="20"/>
</label></p>
<p><label>Age
 <input type="text" name="age" maxlength="2"
            size="2" />
</label></p>
```

If **type** is not included, or is unidentified, type="text" is assumed.

type="text" is used for both text and numbers
name attribute is used to pass data for form processing
maxlength specify the maximum number of characters allowed
size sets the visible width of the text box

name=Qiang
age=30



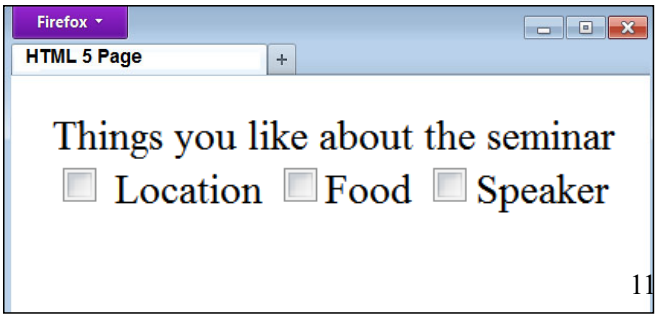
Form Elements: Input Checkbox

```
<p>Things you like about the seminar
  <br />
  <label><input type="checkbox" name="things[]"
       value="loc" />Location</label>
  <label><input type="checkbox" name="things[]"
       value="fud" />Food</label>
  <label><input type="checkbox" name="things[]"
       value="spk" />Speaker</label>
</p>
```

An array called "things"

As multiple checks are allowed, the **name** (sent to the server) must either

- Be different
- or the same but terminate with [ ]

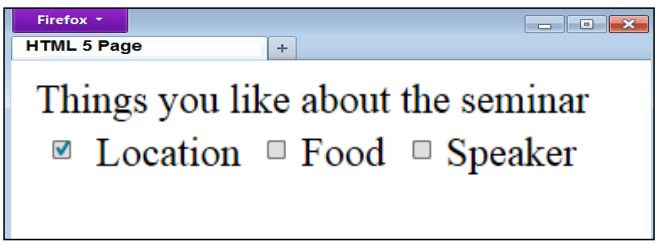


Form Elements: Input Checkbox

```
<p>Things you like about the seminar
  <br />
  <label><input type="checkbox" name="things[]"
      value="loc" checked="checked" />
      Location</label>
  <label><input type="checkbox" name="things[]"
       value="fud /">Food</label>
  <label><input type="checkbox" name="things[]"
       value="spk" />Speaker</label>
</p>
```

checked is used to initialised checkbox with a default check

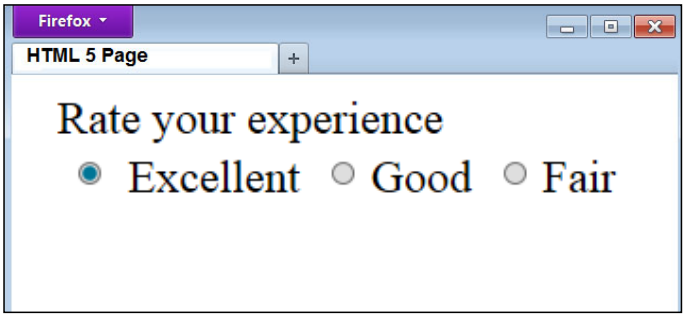
checked="checked" is used for XHTML compliant code



Form Elements: Input Radio Button

Best to group radio inputs with a fieldset and legend - see later.

```
<p>Rate your experience<br />
   <label><input type="radio" name="rating"
       value="Excel" checked="checked" />
      Excellent</label>
   <label><input type="radio" name="rating"
       value="Good" />Good</label>
   <label><input type="radio" name="rating"
       value="Fair" />Fair</label>
</p>
```



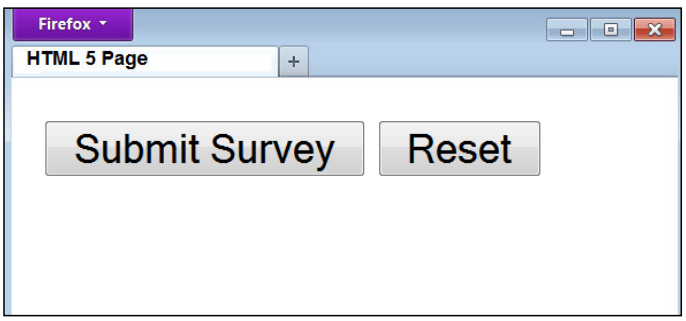
Note: Only ***one*** choice is allowed, *the **name** must be the same rating=Excel*

Form Elements: Input Submit and Reset

```
<p>
 <input type="submit" value="Submit Survey"/>
 <input type="reset" value="Reset" />
</p>
```

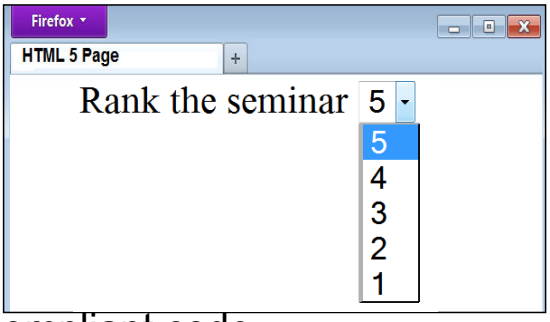
Make sure that the form has an input of type submit.

Note: Reset means set all input form fields to its initial value, and not clear its value.
It only has the effect of clearing, if the initial values are blank or empty



Form Elements: Select & Option

```
<p><label for="rank">Rank the seminar</label>
  <select name="rank" id="rank">
    <option value="5" selected="selected">5</option>
    <option value="4">4</option>
    <option value="3">3</option>
    <option value="2">2</option>
    <option value="1">1</option>
  </select>
</p>
```



- **selected="selected"** is used for XHTML compliant code

Form Elements: Text Area

- **<textarea>…</textarea>** defines a form control for the user to enter multi-line text input
- Text Areas can have the following attributes: rows, cols, readonly, tabindex, accesskey, disabled

```
<p><label>Comments<br />
  <textarea name="comments"
      rows="4" cols="20">
      Enter comments here.
  </textarea>
</label></p>
```

Note: all characters in the text area's element content are displayed verbose. So do not add blank spaces

Forms - Common errors – Watch out!

- Errors in *Form Control elements* may lead to *data errors*

input type = "radio"

one, from a limited number of choices

mutually exclusive – one checked, all others unchecked

same 'name'

***different 'value'** for elements in the group*

input type = "checkbox"

one or more, from a limited number of choices

*usually have **different 'name'** or use an array[]*

***select** and **option** (dropdown box)*

one (or more) options, from a limited number of choices

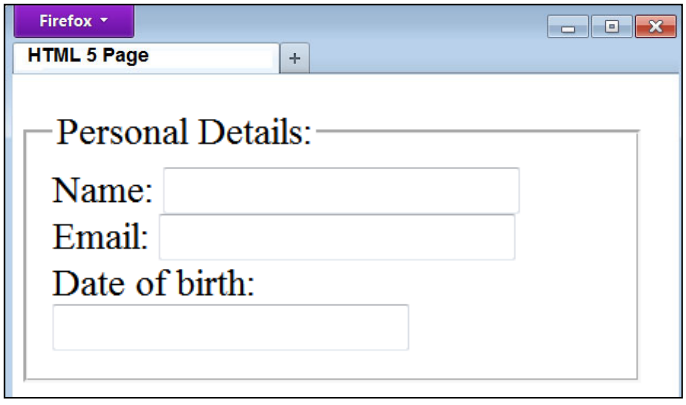
***'name'** only for **select** (can have a 'multiple' attribute)*

***different 'value'** for '**option**' elements (not 'name')*

HTML Forms: Fieldset & Legend

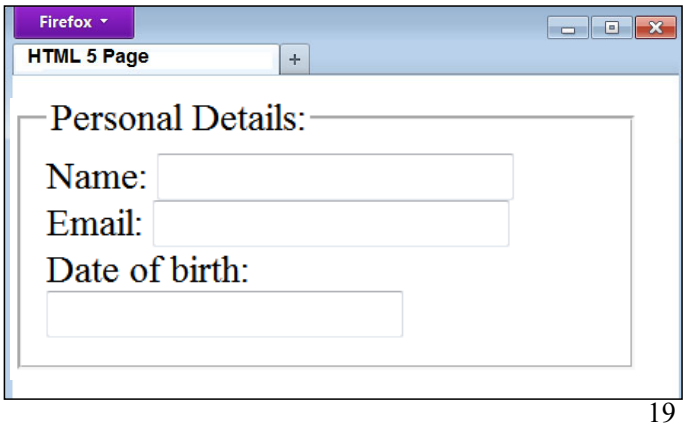
Other form elements:

- `<fieldset>` element is used for ***grouping*** related form controls, so authors can divide a form into smaller, more manageable parts, improving the usability of the form.
- `<legend>` element defines a ***caption for a fieldset*** and must be at the start of a fieldset, before any other elements. A legend can include an `accesskey` attribute.



Form Elements: Fieldset & Legend

```
<fieldset>
  <legend>Personal Details:</legend>
  <label>Name:
  <input type="text" name="pname" /></label>
  <label>Email:
  <input type="text" name="pemail" /></label>
  <label>Date of birth:
  <input type="text" name="pdob" /></label>
</fieldset>
```



HTML5 Form Elements

- HTML5 has new form input **types**
  Note that these are not yet universally supported by all browsers

http://html5test.com/

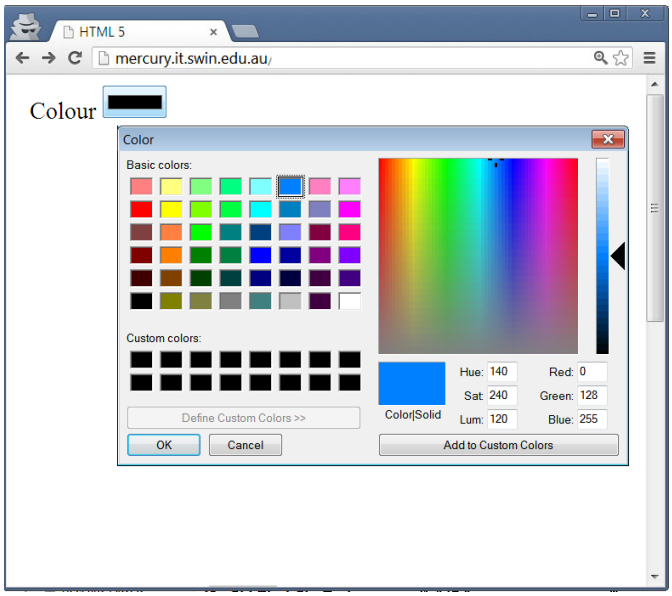
- Examples generated using Chrome.
  - ☐ color
  - ☐ date
  - ☐ datetime
  - ☐ email
  - ☐ month
  - ☐ number
  - ☐ range
  - ☐ search
  - ☐ tel
  - ☐ time
  - ☐ url
  - ☐ week

Note: If the browser does not understand the type, it will default to type="text"

- Other **new attributes** include:
  autofocus, placeholder, pattern, required

HTML5 Form Elements: Colour

```
<p><label>Colour
    <input type="color" name="favcolor"
        autofocus />
</label></p>
```



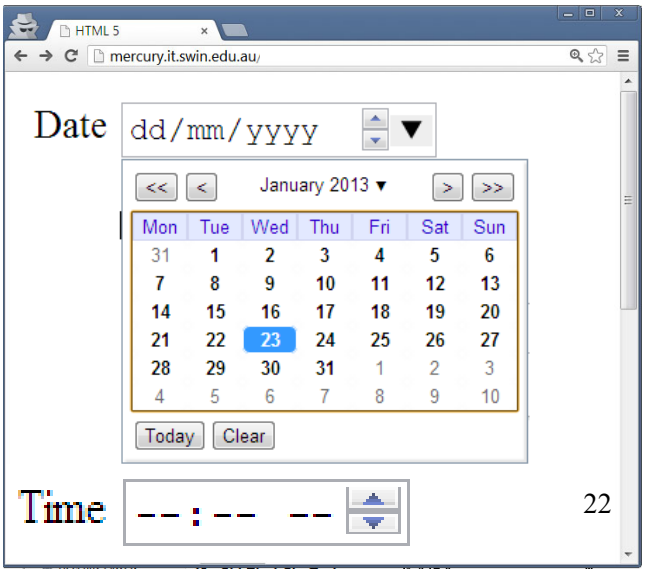
The ***autofocus*** attribute defines which text input should have the default cursor position.

There can only be one field with autofocus. If there is more than 1 the first instance gets the focus.

HTML5 Form Elements: Date

```
<p><label>Date
    <input type="date" name="bday" />
</label></p>

<p><label>Time</label>
    <input type="time"
      name="starttime" />
</label></p>
```

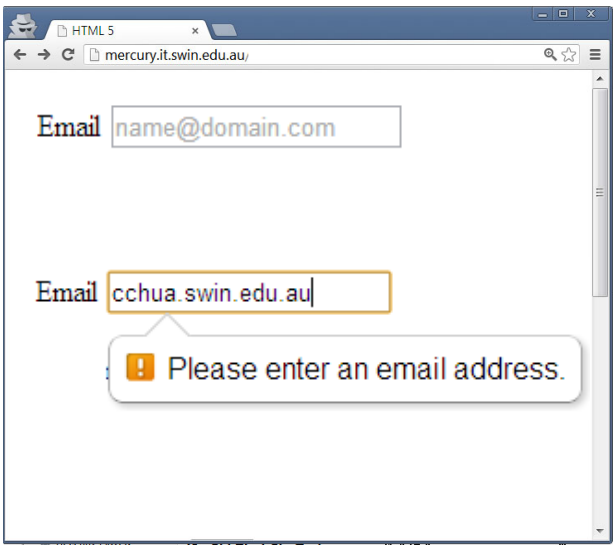


HTML5 Form Elements: Email

```
<p><label>Email
    <input type="email" name="contactemail"
     placeholder="name@domain.com"
     required="required" />
</label></p>
```

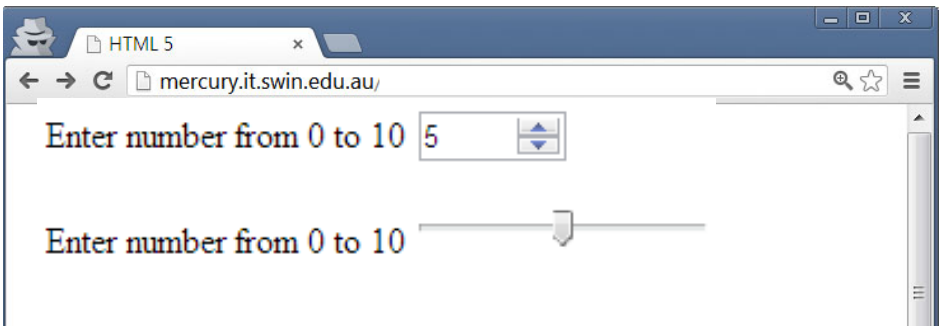
The **placeholder** attribute specifies a short hint that describes the expected value of an input field

The **required** attribute indicates that email field must be filled prior to submission



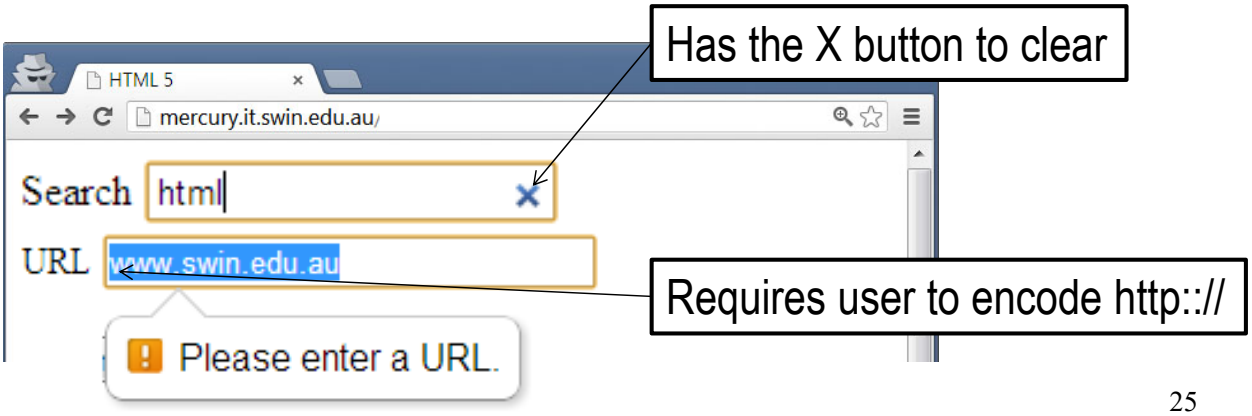
HTML5 Form Elements: Number

```
<p><label>Enter number from 0 to 10
    <input type="number" name="score" min="0"
           max="10" step="1" value="5" />
</label></p>
<p><label>Enter number from 0 to 10
    <input type="range" name="rating" min="0"
           max="10" value="5" />
</label></p>
```



HTML5 Form Elements: Search

```
<p><label>Search
   <input type="search" name="searchquery"
    placeholder="search query" />
</label></p>
<p><label>URL
   <input type="url" name="website"
    placeholder="http://www.domainname.au" />
</label></p>
```

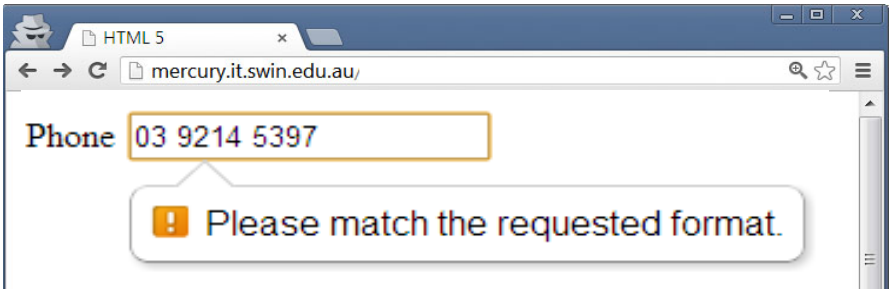


HTML5 Form Elements: Phone

```
<p><label>Phone
   <input type="tel" name="phone"
     placeholder="(##) ####-####"
     pattern="\(\d{2}\) +\d{4}-\d{4}" />
</label></p>
```

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

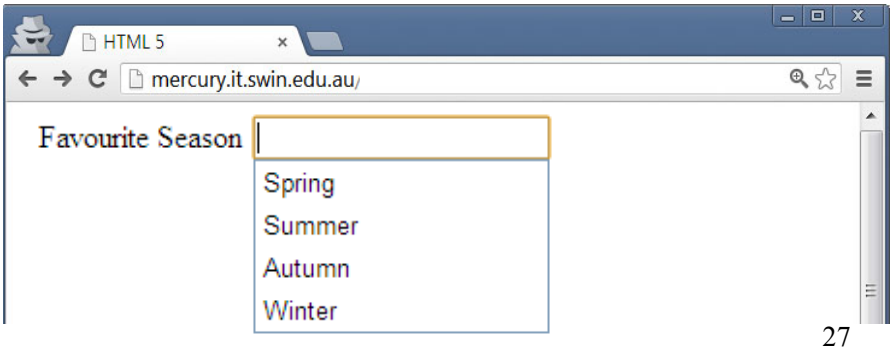
Works with the following input types: text, search, url, tel, email, and password.



HTML5 Form Elements: Data List

```
<p><label>Favourite Season
   <input type="text" name="favseason"
      list="seasons">
   <datalist id="seasons">
     <option value = "Spring">
     <option value = "Summer">
     <option value = "Autumn">
     <option value = "Winter">
   </datalist>
</label></p>
```

Make sure the id matches the list



Topics

- Form, Form Attributes and Form Elements
- Form Validation with Regular Expressions
- HTML Structural Elements

HTML Forms Input Data Control and Checking

- Examples

```
<form id="regForm" method="post“ action="….php">
    …
    <p><label for="catname">Cat&amp;apos;s Name</label>
    <input type="text" name= "catname" id="catname"
        maxlength="20"
        size="10"
        required="required"/>
    </p>
    …
    <input type="email" name= "email" id="email" required="required"/>
</form>
```

Restricts the # characters

Size of text box

A value must be entered

HTML5 input control

Using patterns in HTML

- The pattern attribute uses a 'regular expression' to define the characters that can be entered into a field

```
<input type="text" name= "catname" id="catname"        maxlength="20"
        pattern="^[a-zA-Z ]+$"
        required="required"/>
```

Alpha characters or space only

```
<input type="text" name= "dob" id="dob"  maxlength="10" size="10“
        placeholder="dd/mm/yyyy"
        pattern="\d{1,2}\/\d{1,2}\/\d{4}"
        required="required"/>
```

Placeholders provide prompt to the user

dd/mm/yyyy
???? no range

```
/(?=d)(?:(?:31(?!.(?:0?[2469]|11))|(?: 30|29)(?!.0?2)|29(?=.0?2.(?:(?:(?:1[6--9]| [2--9]d)?(?:0[48]|[2468][048]|[13579]
[26])|(?:(?:16|[2468][048]|[3579] [26])00)))(?:x20|$))|(?:2[0--8]|1d|0? [1--9]))([--/])(?:1[012]|0?[1--9])1(?: 1[6--9]|[2--
-9]d)?dd(?:(?=x20d)x20| $))?/
```

Regular expressions not necessarily the best solution to every check!!

http://html5pattern.com/

What are Regular Expressions?

- ***Regular Expressions:***
  - are strings that describe the 'pattern' or 'rules' for strings

    e.g. **^[a-zA-Z]+$**
  - are strings that follow a set of syntax rules
  - can be used as a concise and consistent way to 'test' for matching patterns.
- ***Regular expressions can be great for checking form values!***
  - A simple regular expression can be the equivalent of many lines of code.

Where are they Used?

EVERYWHERE!

- **Regular Expressions** have a history in unix environments, became popular in many text editors, and were supported in text processing (programming) languages like Perl and Tcl.

  … but the catch was there were some variations in syntax ☹ mostly now the same basic syntax ☺
- **Regular Expressions** are used in many text search routines
- **Regular Expressions** are *very easy* to use ☺ .... but *not so easy* to define the 'pattern' ☹

```
/(?=d)(?:(?:31(?!.(?:0?[2469]|11))|(?: 30|29)(?!.0?2)|29(?=.0?2.(?:(?:(?:1[6--9]| [2--9]d)?(?:0[48]|[2468][048]|[13579] [26])|(?:(?:16|[2468][048]|[3579] [26])00)))(?:x20|$))|(?:2[0--8]|1d|0? [1--9]))([--/])(?:1[012]|0?[1--9])1(?: 1[6--9]|[2--
-9]d)?dd(?:(?=x20d)x20| $))?/
```

RegExp - Basic Syntax

`/pattern/modifiers`

Pattern Basics

^	Start of string
$	End of string
.	Match any single character
(a|b)	a or b
(…)	Group section
[abc]	match any character in the set
[^abc]	not match in the set
[a-z]	match the range
\d	match a single digit from 0 to 9

Pattern Quantifiers

a?	0 or 1 of a
a*	0 or more of a
a+	one or more instance of a
a{3}	exactly 3 a's = aaa
a{3,}	3 or more a's
a{3,6}	between 3 to 6 a's
!(pattern)	"not" pattern

[\ ^ $. | ? * + ( ) are the 11 meta-characters, or special characters, used in the syntax.
If you want to include these in a RegExp literal you need to escape them with \ eg. **\(**

RegExp - Basic Examples

JavaScript	matches "Isn't JavaScript great?"
^JavaScript	matches "JavaScript rules!", not "What is JavaScript?"
JavaScript$	matches "I love JavaScript", not "JavaScript is great!"
^JavaScript$	matches "JavaScript", and nothing else

bana?na	matches "banana" and "banna", but not "banaana".
bana+na	matches "banana" and "banaana", but not "banna".
bana*na	matches "banna", "banana", and "banaaana", but not "bnana"
^[a-zA-z]+$	matches any string of one or more letters and nothing else.

http://www.sitepoint.com/article/expressions-javascript

x?	0 or 1 of x
x*	0 or more of x
x+	one or more instance of x

RegExp - Basic Syntax

Groups & Ranges

- . Any character (except \n)
- (a|b) a or b
- (…) group
- (?:…} passive group
- [abc] set (“range”) a, b or c
- [^abc] not a, b or c
- [a-g] set range a to g
- [3-6] set range of digits 3,4,5 and 6
- \n “nth” group or subpattern

Pattern Modifiers

/i case insensitive

/x allow comments and white space in pattern

…

RegExp - Some Sample Patterns

[A-Za-z0-9-]+ = Letters, numbers and hyphens

`\d{1,2}\/\d{1,2}\/\d{4}` = date as 19/9/2006 \ = escape character

[^\s]+(?=\.(jpg|gif|png))\.\2 = jpg, gif or png filename

^[1-9]{1}$|^[1-2]{1}[0-9]{1}$|^30$ = any number from 1 to 30

#?([A-Fa-f0-9]){3}(([A-Fa-f0-9]){3})? = valid hex colour code

.{6,} = password six or more characters

(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,15} = password string
(at least 1 uppercase letter, 1 lowercase letter and 1 digit)

^.+@.+\..{2,3}$ = email address

\<(/?[^\>]+)\>/ = HTML tags

Remember: Same pattern may be used for checking form values, ***both*** on client-side and server-side.

Topics

- Form, Form Attributes and Form Elements
- Form Validation with Regular Expressions
- HTML Structural Elements (Container Elements)

HTML Structure

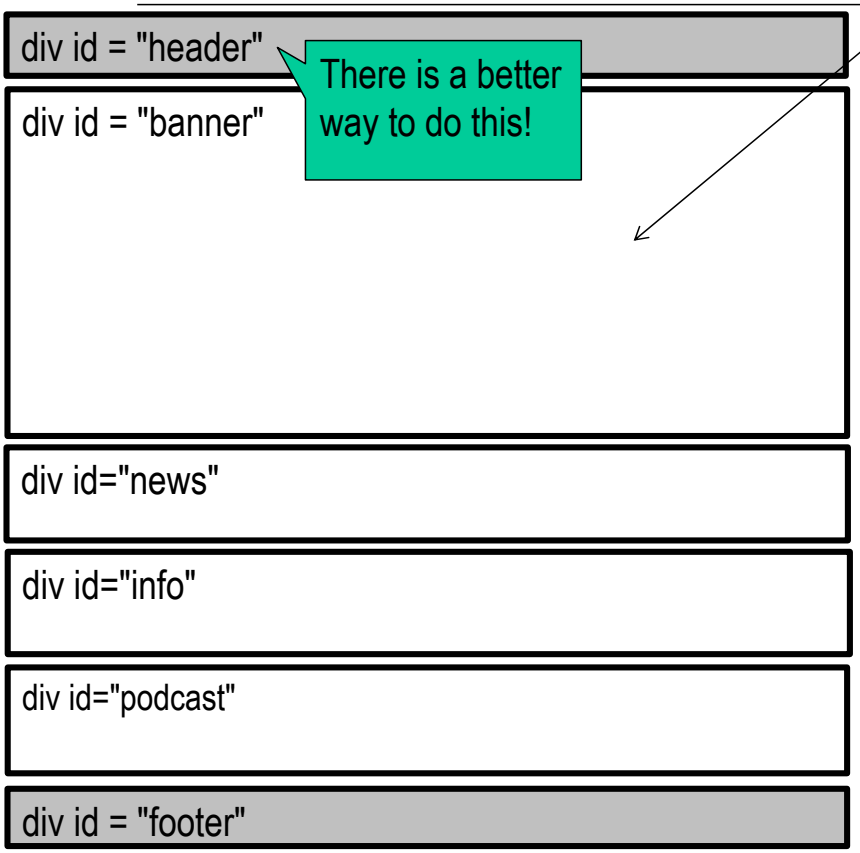
Division

- ** ... ** is a generic logical block level container used to divide content, e.g. section
- It has **no default** meaning or rendering behaviour, as it is a logical “division”.
- It plays a role in providing an arbitrary block container where a style can be applied to use CSS.
- **Do not** use a when you should be using a logical element like <p> or <nav>.

Span

- **…** is a generic inline level container used to group other inline elements, such as text.
- It has **no default** meaning or presentation qualities.
- Similar to the element, the tag has a role in providing an arbitrary container but for inline elements.
- **Do not** use a when you should be using a logical element like or .

Division (continued)



Presented without CSS

- Always remember that **HTML** is only about **content** and **structure**, where these will appear on the screen will be specified through **CSS**

HTML5 Semantic Elements

HTML5 added some new semantic elements such as

<nav> <article> <section> <header> <footer>…

Non-semantic elements such as and - Tell nothing about its content. The defined structure is semantically *neutral* in terms of meaning to the browser.

Semantic elements describe their meaning to both the browser and the developer. For example,

<nav> defines a set of navigation links.

<section> defines a section in a document.

<footer> defines a footer for a document or section.

….

Navigation

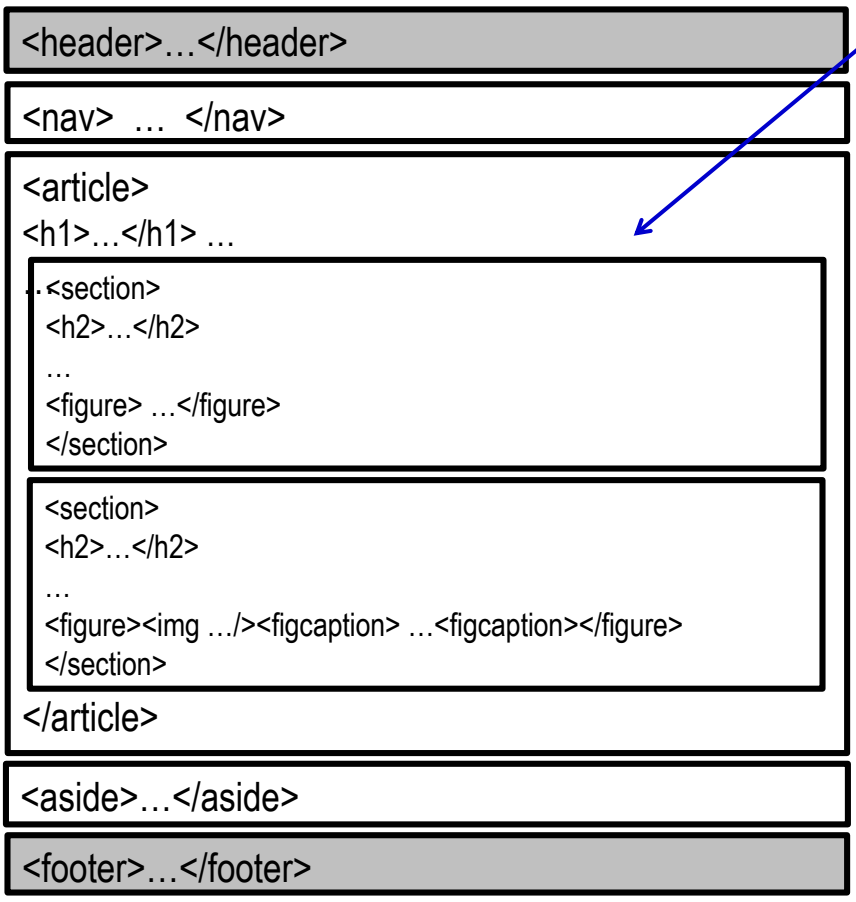
- **<nav>…</nav>** specifies a section of navigation links.
- It is intended only for major block of navigation links.
  - Not all links of a document must be in a <nav> element
- Browsers, such as screen readers for disabled users, can use this element to determine whether to omit the initial rendering of this content.

Navigation (continued)

```
<nav>
<a href="index.php">Home</a> |
<a href="product.php">Products</a> |
<a href="download.php">Download</a> |
<a href="contact.php">Contact Us</a>
</nav>
```

An unordered list <ul> is often used inside <nav>

Structure: Putting it all together



Presented without CSS

- Always remember that **HTML** is only about **content** and **structure**, where these will appear on the screen will be specified through **CSS**

Structure: Putting it all together