

Client Side JavaScript Week 3 Lecture

**COMMONWEALTH OF
Copyright Regulations 1969
WARNING**

This material has been reproduced and communicated to you by or on behalf of the University of Sydney pursuant to Part VB of the Copyright Act 1968 (**the Act**).

The material in this communication may be subject to copyright under the Act. Any further reproduction or communication of this material by you may be the subject of copyright protection under the Act.

Do not remove this notice.

Most materials and diagrams in this slides are from chapter 4 and 6 of Fundamentals of Web Development

Admin

- We have two more labs on Wednesday 5-6pm
 - Lab 115
 - Lab 130A
- Most evening labs are overcrowded
 - If Wednesday 5-6pm suits your timetable, please allocate yourself, or let the course TA (XIANG DAI <dai.xiang.au@gmail.com>) know
- Assignment Demo
 - Usually based on submitted version
- Group assignment
 - Ok to form group across labs
 - Coordinator may decide which lab you should go for the demo
- Quiz arrangement
 - All students will sit the quiz at the same (or at least similar) time
 - Arrangement to be released later

Outline

- **More HTML**
 - **Table**
 - **Elements**
 - **Styling**
 - **Form**
 - **Controls**
- **JavaScript**
 - **Location and Basic Syntax**
 - **Variables, Control Structure, Function, Object, Array**
 - **Windows and DOM object**
 - **Event model**




HTML Table example

Chapter 4

	Free	Basic	Premium
Upload Space	50MB	200MB	Unlimited
Daily Uploads	1	10	Unlimited
Total Uploads	20	100	Unlimited
Social Sharing		✓	✓
Analytics			✓
Price per year	Free	\$ 9.99	\$ 19.99

Chapter 4

Artist Inventory






Artist	Work Details			
	Title	Year	Home	
 Jacques-Louis David		The Death of Marat	1793	Royal Museums of Fine Arts of Belgium
		The Intervention of the Sabine Women	1793	Royal Museums of Fine Arts of Belgium

Chapter 4

October 2014

S	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	
« Sep						Nov »

Paintings

Title	Artist	Year	Genre
 Death of Marat	David, Jacques-Louis	1793	Romanticism
 Lictors Bearing to Brutus the Bodies of his Sons	David, Jacques-Louis	1789	Romanticism
 Liberty Leading the People	Delacroix, Eugene	1830	Romanticism
 Arrangement in Grey and Black	Whistler, James Abbott	1871	Realism
 Mademoiselle Caroline Riviere	Ingres, Jean-Auguste	1806	Neo-Classicism

HTML Table basic mark ups

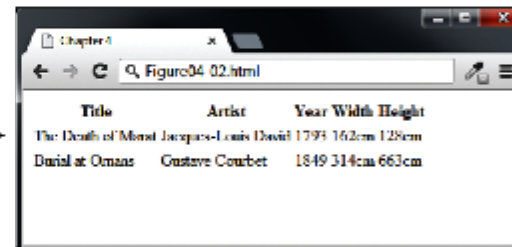
- Tables can be used to display
 - Many types of content
 - Calendars, financial data, etc
 - Any type of data
 - Images, text, links etc
- A **table** in HTML is created using the **<table>** element
 - A basic table contains rows **<tr>** and cells **<td>**
 - Many table contains headings which is a special row to indicate what each cell is about: **<th>**

Basic able example

<table>					
<tr>	Title	Artist	Year	Width	Height
<th>	<th>	<th>	<th>	<th>	<th>
<tr>	The Death of Marat	Jacques-Louis David	1793	162cm	128cm
<td>	<td>	<td>	<td>	<td>	<td>
<tr>	Burial at Ornans	Gustave Courbet	1849	314cm	663cm
<td>	<td>	<td>	<td>	<td>	<td>

th

```
<table>
  <tr>
    <th>Title</th>
    <th>Artist</th>
    <th>Year</th>
    <th>Width</th>
    <th>Height</th>
  </tr>
  <tr>
    <td>The Death of Marat</td>
    <td>Jacques-Louis David</td>
    <td>1793</td>
    <td>162cm</td>
    <td>128cm</td>
  </tr>
  <tr>
    <td>Burial at Ornans</td>
    <td>Gustave Courbet</td>
    <td>1849</td>
    <td>314cm</td>
    <td>663cm</td>
  </tr>
</table>
```



Title	Artist	Year	Width	Height
The Death of Marat	Jacques-Louis David	1793	162cm	128cm
Burial at Ornans	Gustave Courbet	1849	314cm	663cm

Spanning Rows and Columns

- Simplest table is of a grid structure, with each row having the same number of cells
- It is possible to merge cells horizontally or vertically, e.g. having some cells covering a few rows or columns

<table>					
<tr>	Title	Artist	Year	Size (width x height)	
	<th>	<th>	<th>	<th colspan=2>	
<tr>	The Death of Marat	Jacques-Louis David	1793	162cm	128cm
	<td>	<td>	<td>	<td>	<td>
<tr>	Burial at Ornans	Gustave Courbet	1849	314cm	663cm
	<td>	<td>	<td>	<td>	<td>

use the
colspan or
rowspan
attributes

Notice that this row
now only has four
cell elements.

```
<table>
<tr>
  <th>Title</th>
  <th>Artist</th>
  <th>Year</th>
  <th colspan="2">Size (width x height)</th>
</tr>
<tr>
  <td>The Death of Marat</td>
  <td>Jacques-Louis David</td>
  <td>1793</td>
  <td>162cm</td>
  <td>128cm</td>
</tr>
...
</table>
```

Row Spanning Example

Artist	Title	Year
Jacques-Louis David	The Death of Marat	1793
	The Intervention of the Sabine Women	1799
	Napoleon Crossing the Alps	1800

Notice that these two rows now only have two cell elements.

```

<table>
<tr>
  <th>Artist</th>
  <th>Title</th>
  <th>Year</th>
</tr>
<tr>
  <td rowspan="3">Jacques-Louis David</td>
  <td>The Death of Marat</td>
  <td>1793</td>
</tr>
<tr>
  <td>The Intervention of the Sabine Women</td>
  <td>1799</td>
</tr>
<tr>
  <td>Napoleon Crossing the Alps</td>
  <td>1800</td>
</tr>
...
</table>

```


Additional table tags

- `<caption>`

A title for the table is good for accessibility.

```
<table>
```

```
<caption>19th Century French Paintings</caption>
```

- `<col>`, `<colgroup>`

These describe our columns, and can be used to aid in styling.

```
<col class="artistName" />
```

```
<colgroup id="paintingColumns">
```

```
<col />
```

```
<col />
```

```
</colgroup>
```

- `<thead>`

Table header could potentially also include other `<tr>` elements.

```
<thead>
```

```
<tr>
```

```
<th>Title</th>
```

```
<th>Artist</th>
```

```
<th>Year</th>
```

```
</tr>
```

```
</thead>
```

- `<tfoot>`

Yes, the table footer comes *before* the body.

```
<tfoot>
```

```
<tr>
```

```
<td colspan="2">Total Number of Paintings</td>
```

```
<td>2</td>
```

```
</tr>
```

```
</tfoot>
```

- `<tbody>`

Potentially, with styling the browser can scroll this information, while keeping the header and footer fixed in place.

```
<tbody>
```

```
<tr>
```

```
<td>The Death of Marat</td>
```

```
<td>Jacques-Louis David</td>
```

```
<td>1793</td>
```

```
</tr>
```

```
<tr>
```

```
<td>Burial at Ornans</td>
```

```
<td>Gustave Courbet</td>
```

```
<td>1849</td>
```

```
</tr>
```

```
</tbody>
```

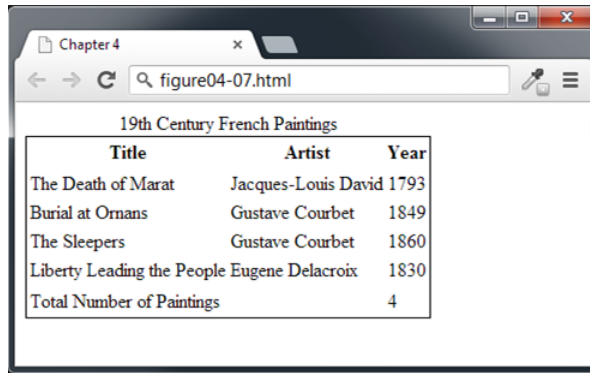
```
</table>
```



Title	Artist	Year
The Death of Marat	Jacques-Louis David	1793
Burial at Ornans	Gustave Courbet	1849
Total Number of Paintings		2

Styling tables

- Most box model styling can be applied to `<table>`, `<tr>`, `<td>` and other tags

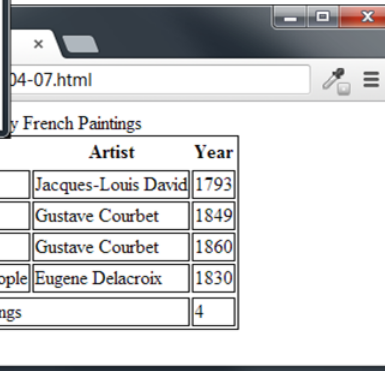


Chapter 4 x
figure04-07.html

19th Century French Paintings

Title	Artist	Year
The Death of Marat	Jacques-Louis David	1793
Burial at Ornans	Gustave Courbet	1849
The Sleepers	Gustave Courbet	1860
Liberty Leading the People	Eugene Delacroix	1830
Total Number of Paintings		4

```
table {  
    border: solid 1pt black;  
}
```

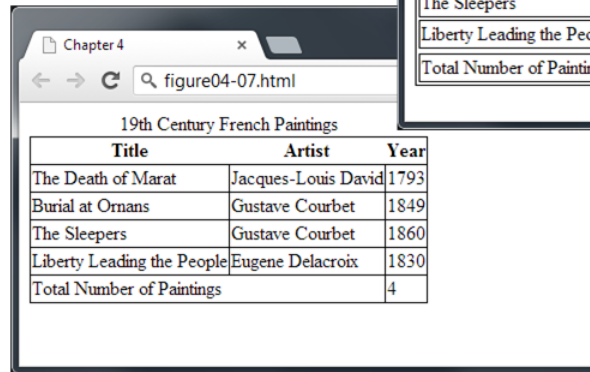


Chapter 4 x
figure04-07.html

19th Century French Paintings

Title	Artist	Year
The Death of Marat	Jacques-Louis David	1793
Burial at Ornans	Gustave Courbet	1849
The Sleepers	Gustave Courbet	1860
Liberty Leading the People	Eugene Delacroix	1830
Total Number of Paintings		4

```
table {  
    border: solid 1pt black;  
}  
td {  
    border: solid 1pt black;  
}
```



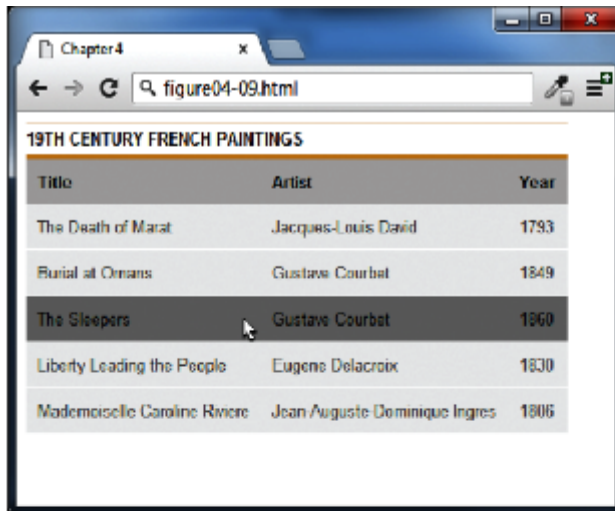
Chapter 4 x
figure04-07.html

19th Century French Paintings

Title	Artist	Year
The Death of Marat	Jacques-Louis David	1793
Burial at Ornans	Gustave Courbet	1849
The Sleepers	Gustave Courbet	1860
Liberty Leading the People	Eugene Delacroix	1830
Total Number of Paintings		4

```
table {  
    border: solid 1pt black;  
    border-collapse: collapse;  
}  
td {  
    border: solid 1pt black;  
}
```

Nifty Table styling tricks: hover effect and zebra-stripes

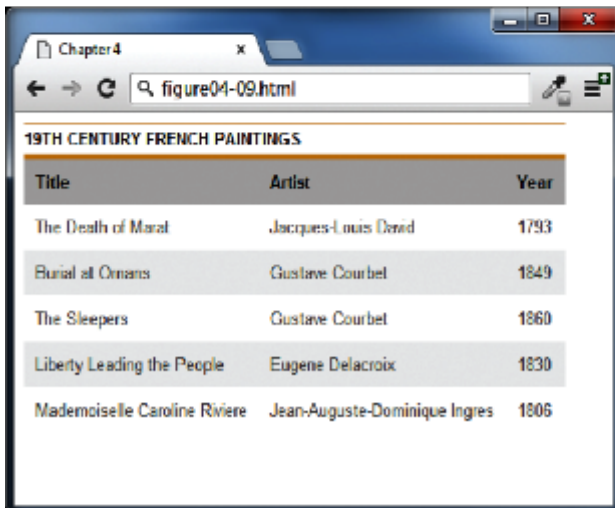


A screenshot of a web browser window titled 'Chapter4' showing a table of 19th-century French paintings. The table has three columns: Title, Artist, and Year. The row for 'The Sleepers' by Gustave Courbet is highlighted with a dark gray background, indicating a hover effect. The other rows have a light gray background.

Title	Artist	Year
The Death of Marat	Jacques-Louis David	1793
Burial at Omsans	Gustave Courbet	1849
The Sleepers	Gustave Courbet	1860
Liberty Leading the People	Eugene Delacroix	1830
Mademoiselle Caroline Riviere	Jean-Auguste-Dominique Ingres	1806

Pseudo class

```
tbody tr:hover {  
    background-color: #9e9e9e;  
    color: black;  
}
```



A screenshot of a web browser window titled 'Chapter4' showing the same table of 19th-century French paintings. The table has three columns: Title, Artist, and Year. The rows alternate between light gray and white backgrounds, creating a zebra-stripe effect. The row for 'The Sleepers' is highlighted with a dark gray background.

Title	Artist	Year
The Death of Marat	Jacques-Louis David	1793
Burial at Omsans	Gustave Courbet	1849
The Sleepers	Gustave Courbet	1860
Liberty Leading the People	Eugene Delacroix	1830
Mademoiselle Caroline Riviere	Jean-Auguste-Dominique Ingres	1806

```
tbody tr:nth-child(odd) {  
    background-color: white;  
}
```

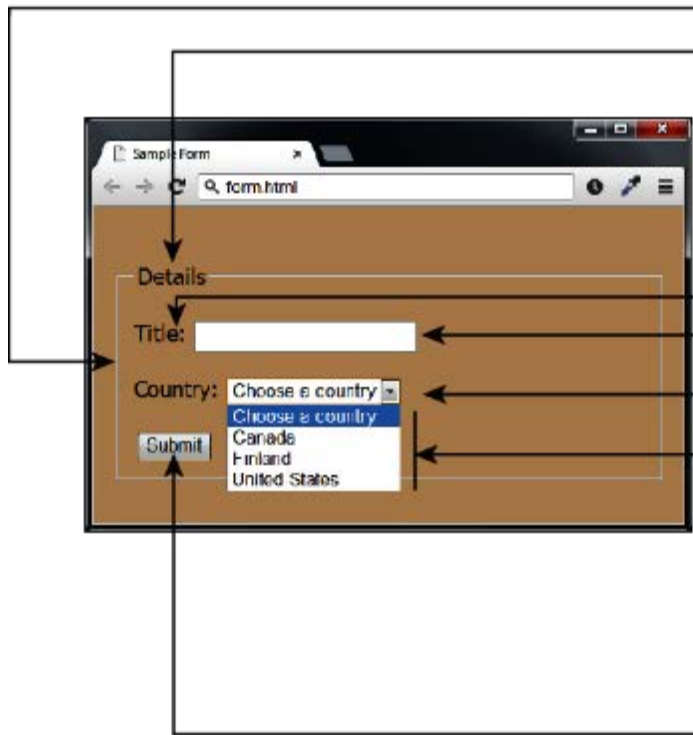
Outline

- **More HTML**
 - Table
 - Elements
 - Styling
 - **Form**
 - **Controls**
- **JavaScript**
 - Location and Basic Syntax
 - Variables, Control Structure, Function, Object, Array
 - Windows and DOM object
 - Event model

HTML Forms

- Forms provide a way for users to interact with a web server
- Forms contain elements similar to desktop GUI
 - Plain text or password input
 - Selection
 - Radio and check boxes
 - Buttons

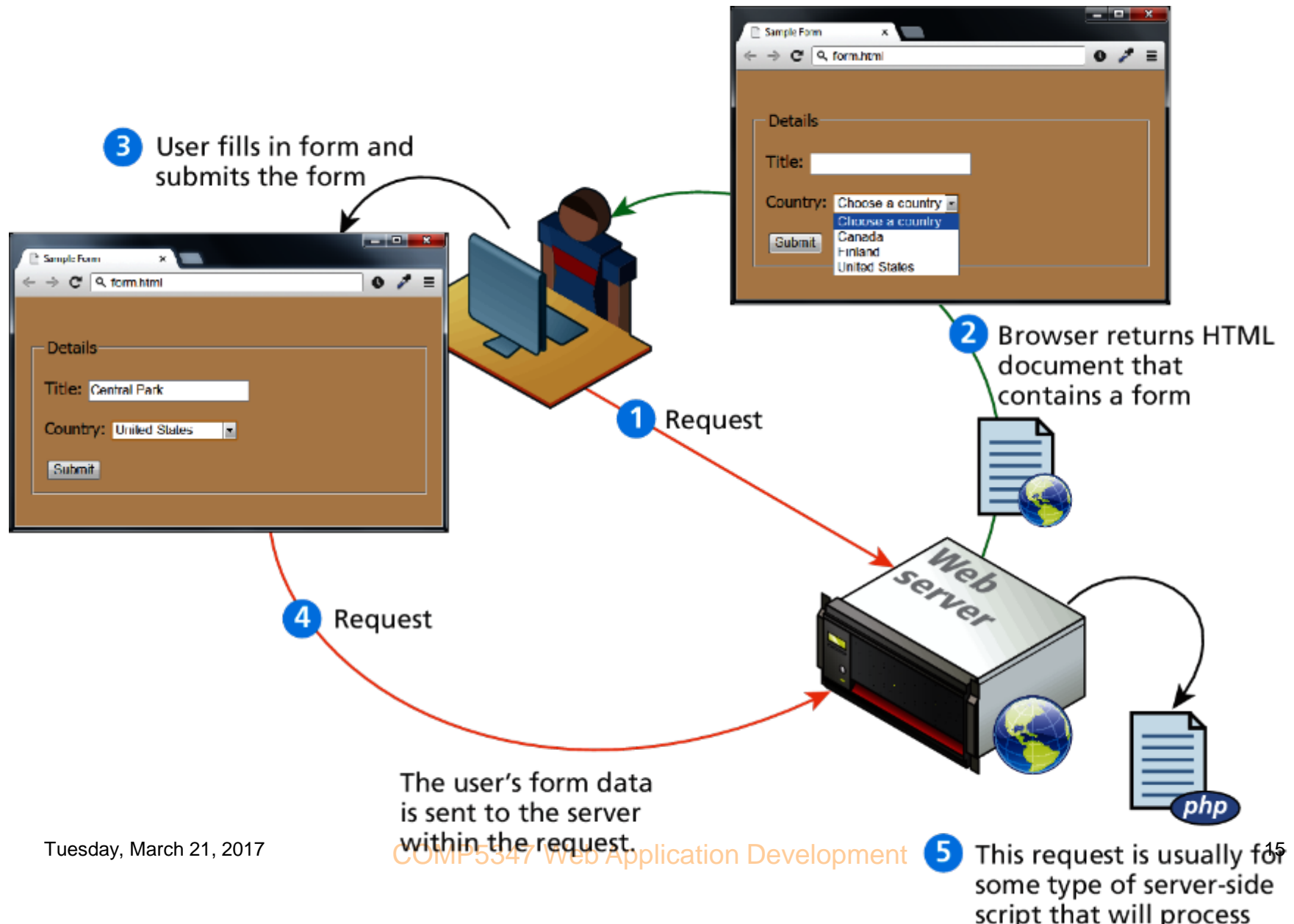
Form structures



```
<form method="get" action="process.php">  
  <fieldset>  
    <legend>Details</legend>  
    <p>  
      <label>Title: </label>  
      <input type="text" name="title" />  
    </p>  
    <p>  
      <label>Country: </label>  
      <select name="where">  
        <option>Choose a country</option>  
        <option>Canada</option>  
        <option>Finland</option>  
        <option>United States</option>  
      </select>  
    </p>  
    <input type="submit" />  
  </fieldset>  
</form>
```

Form is main element to allow users enter information and get passed to the server application

How forms interact with servers



Form-Related HTML Elements

Type	Description
<button>	Defines a clickable button.
<datalist>	An HTML5 element form defines lists to be used with other form elements.
<fieldset>	Groups related elements in a form together.
<form>	Defines the form container.
<input>	Defines an input field. HTML5 defines over 20 different types of input.
<label>	Defines a label for a form input element.
<legend>	Defines the label for a fieldset group.
<option>	Defines an option in a multi-item list.
<optgroup>	Defines a group of related options in a multi-item list.
<select>	Defines a multi-item list.
<textarea>	Defines a multiline text entry box.

Text Input Controls

Type	Description
text	Creates a single line text entry box. <code><input type="text" name="title" /></code>
textarea	Creates a multiline text entry box. <code><textarea rows="3" ... /></code>
password	Creates a single line text entry box for a password <code><input type="password" ... /></code>
search	Creates a single-line text entry box suitable for a search string. This is an HTML5 element. <code><input type="search" ... /></code>
email	Creates a single-line text entry box suitable for entering an email address. This is an HTML5 element. <code><input type="email" ... /></code>
tel	Creates a single-line text entry box suitable for entering a telephone. This is an HTML5 element. <code><input type="tel" ... /></code>
url	Creates a single-line text entry box suitable for entering a URL. This is an HTML5 element. <code><input type="url" ... /></code>

Example text input controls

```
<input type="search" placeholder="enter search text" ... />
```

Search: Search:

```
<input type="email" ... />
```

Email: *In Opera*

Please enter a valid email address

Email: *In Chrome*

! Please enter an email address.

```
<input type="url" ... />
```

url:

! Please enter a URL.

```
<input type="tel" ... />
```

Tel:

Key motivations of new form controls in HTML5

- Usability
- Styling
- Client side validation

Select Lists

Search City:

- Paris
- Prague

```
<input type="text" name="city" list="cities" />
```

```
<datalist id="cities">
  <option>Calcutta</option>
  <option>Calgary</option>
  <option>London</option>
  <option>Los Angeles</option>
  <option>Paris</option>
  <option>Prague</option>
</datalist>
```

datalist

Select:

```
<select name="choices">
  <option>First</option>
  <option selected>Second</option>
  <option>Third</option>
</select>
```

Select:

- First
- Second
- Third

```
<select ... >
  <optgroup label="North America">
    <option>Calgary</option>
    <option>Los Angeles</option>
  </optgroup>
  <optgroup label="Europe">
    <option>London</option>
    <option>Paris</option>
    <option>Prague</option>
  </optgroup>
</select>
```

Cities:

- North America
- Calgary
- Los Angeles
- Europe
- London
- Paris
- Prague

Which Value to send

Select:

Second	▼
First	
Second	
Third	

```
<select name="choices">  
  <option>First</option>  
  <option>Second</option>  
  <option>Third</option>  
</select>
```

```
<select name="choices">  
  <option value="1">First</option>  
  <option value="2">Second</option>  
  <option value="3">Third</option>  
</select>
```

?choices=Second

?choices=2

Radio buttons and checkboxes

Continent:

- ☐ North America
- ☒ South America
- ☐ Asia

```
<input type="radio" name="where" value="1">North America<br/>  
<input type="radio" name="where" value="2" checked="">South America<br/>  
<input type="radio" name="where" value="3">Asia
```

I accept the software license ☒

```
<label>I accept the software license</label>  
<input type="checkbox" name="accept" >
```

Where would you like to visit?

- ☒ Canada
- ☐ France
- ☒ Germany

```
<label>Where would you like to visit? </label><br/>  
<input type="checkbox" name="visit" value="canada">Canada<br/>  
<input type="checkbox" name="visit" value="france">France<br/>  
<input type="checkbox" name="visit" value="germany">Germany
```

?accept=on&visit=canada&visit=germany

Button Controls

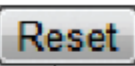
Type	Description
<code><input type="submit"></code>	Creates a button that submits the form data to the server.
<code><input type="reset"></code>	Creates a button that clears any of the user's already entered form data.
<code><input type="button"></code>	Creates a custom button. This button may require Javascript for it to actually perform any action.
<code><input type="image"></code>	Creates a custom submit button that uses an image for its display.
<code><button></code>	<p>Creates a custom button. The <code><button></code> element differs from <code><input type="button"></code> in that you can completely customize what appears in the button; using it, you can, for instance, include both images and text, or skip server-side processing entirely by using hyperlinks.</p> <p>You can turn the button into a submit button by using the <code>type="submit"</code> attribute.</p>

Example Button Controls

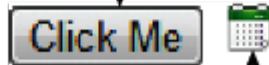
```
<input type="submit" />
```



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



```
<input type="button" value="Click Me" />
```



```
<input type="image" src="appointment.png" />
```



```
<button>  
  <a href="email.html">  
      
    Email  
  </a>  
</button>
```

```
<button type="submit" >  
    
  Edit  
</button>
```

Number and Ranges

Rate this photo:

2

Grumpy ————— Ecstatic

```
<label>Rate this photo: <br/>
```

```
<input type="number" min="1" max="5" name="rate" />
```

Grumpy

```
<input type="range" min="0" max="10" step="1" name="happiness" />
```

Ecstatic

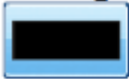
Rate this photo:

Grumpy ————— Ecstatic

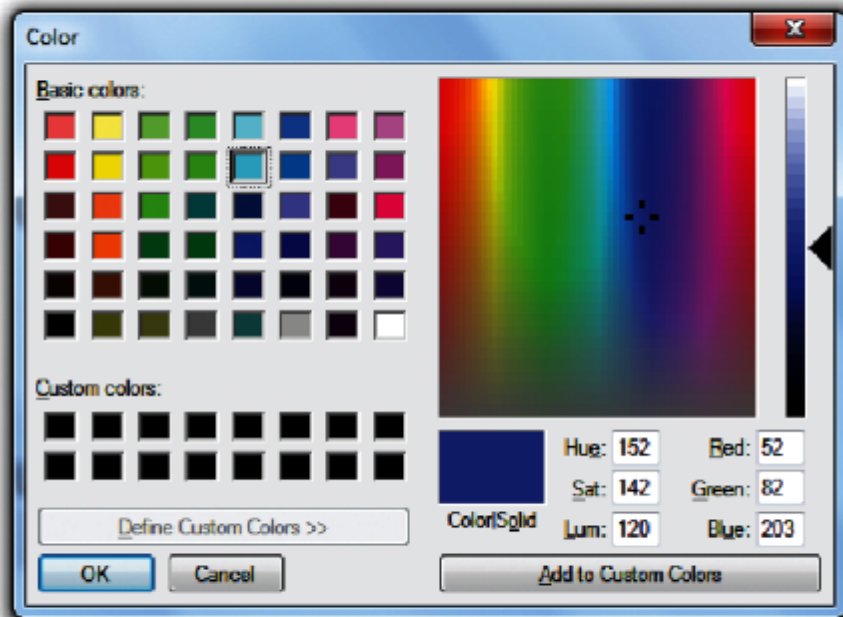
Controls as they appear in browser
that doesn't support these input types

Color

Background Color:



```
<label>Background Color: <br/>
<input type="color" name="back" />
```



Background Color:

Control as it appears in browser that
doesn't support this input type

Date and Time

Date:

March 2013

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

Today

```
<label>Date: <br/>
<input type="date" ... />
```

Time:

02:02 AM

```
<input type="time" ... />
```

DateTime:

2013-03-08 05:46 UTC

```
<input type="datetime" ... />
```

DateTime Local:

2013-03-13 12:02

```
<input type="datetime-local" ... />
```

Outline

- **More HTML**
 - Table
 - Elements
 - Styling
 - Form
 - Controls
- **JavaScript**
 - **Location and Basic Syntax**
 - Variables, Control Structure, Function, Object, Array
 - Windows and DOM object
 - Event model

JavaScript

- JavaScript is an object based, dynamically typed scripting language
 - Has been the primary client-side scripting language for HTML and CSS
 - It has also a server side implementation
 - W3C school simply defines it as “the most popular programming language in the world”
- As a client-side scripting language
 - It runs inside the browser
 - Able to interact with many browser managed resources: DOM, Browser’s object (BOM) such as windows, screen, history, cookies and more
 - It can be written as inline (discouraged!), embedded or as external file

Brief history

- Was created in 10 days in May 1995 by Brendan Erich, then working at Netscape and now of Mozilla
 - Considered as not properly defined and targeting amateurs
- JavaScript became a much more important part of web development in the mid 2000s with **AJAX**.
 - Initial effort from Microsoft in late 1999, get adopted by other browsers
 - Made very popular by Gmail and Google maps
 - Receives a lot more professional programming attention
- A lot of JavaScript frameworks have since created: jQuery, Prototype, AngularJS, etc.
- Server side JavaScript also gaining popularity

Location of JavaScript code

```
<a href="JavaScript:OpenWindow();"more info</a>  
<input type="button" onclick="alert('Are you sure?');" />
```

LISTING 6.1 Inline JavaScript example

```
<script type="text/javascript">  
/* A JavaScript Comment */  
alert ("Hello World!");  
</script>
```

LISTING 6.2 Embedded JavaScript example

```
<head>  
  <script type="text/JavaScript" src="greeting.js">  
  </script>  
</head>
```

LISTING 6.3 External JavaScript example

JavaScript Variables

- Declaring a variable
 - `var name;`
- JavaScript does not require variables to have a type before they can be used in a program
- A variable in JavaScript can contain a value of any data type, and in many situations, JavaScript automatically converts between values of different types for you
- Variable has various scopes

`var x;`  a variable `x` is defined

`var y = 0;`  `y` is defined and initialized to 0

`y = 4;`  `y` is assigned the value of 4

Conditional Control

```
var hourOfDay;    // var to hold hour of day, set it later...
var greeting;     // var to hold the greeting message.
if (hourOfDay > 4 && hourOfDay < 12){
    // if statement with condition
    greeting = "Good Morning";
}
else if (hourOfDay >= 12 && hourOfDay < 20){
    // optional else if
    greeting = "Good Afternoon";
}
else{ // optional else branch
    greeting = "Good Evening";
}
```

LISTING 6.4 Conditional statement setting a variable based on the hour of the day

Loops

```
var i=0; // initialise the Loop Control Variable
while(i < 10){ //test the loop control variable
    i++; //increment the loop control variable
}
```

```
for (var i = 0; i < 10; i++){
    //do something with i
}
```

Functions

- **Functions** are the building block for modular code in JavaScript
 - They are defined by using the reserved word **function** and then the function name and (optional) parameters.

Therefore a function to raise x to the yth power might be defined as:

```
function power(x,y){  
    var pow=1;  
    for (var i=0;i<y;i++){  
        pow = pow*x;  
    }  
    return pow;  
}
```

And called as

```
power(2,10);
```

Object

- JavaScript is different to classic OOP, which is class-based.
 - It has a clear concept of **Object** which is similar to object in other OOP languages
 - The concept of **Class** is the source of confusion
- We usually start by introducing *Object*
 - An object is a collection of related data and/or functionality
 - The “data” part is referred to as “property”
 - The “functionality” part is referred to as “method”
 - In JavaScript, almost “everything” is an object
 - Most data types
 - Functions
 - The easiest way of creating an object is to use Object Literal

Creating Object using literal

```
var person = {  
    firstName:"John",  
    lastName:"Doe",  
    age:50,  
    eyeColor:"blue"  
};
```

```
var person = {  
    firstName:"John",  
    lastName:"Doe",  
    age:50,  
    eyeColor:"blue",  
    fullName : function() {  
        return this.firstName + " " + this.lastName;  
    }  
};
```

Variable types

- Primitive types
 - Boolean, string and number
 - Null, undefined
- Complex types
 - Object
 - Array
 - Function

Arrays

- Arrays are used to store multiple values in a single variable.
 - `var greetings = ["Good Morning", "Good Afternoon"];`
 - Array element is accessible with index, starting from 0
 - E.g. `greetings[0] = "Good Morning"`

Variable Scope

- Each variable in a program has a scope
- The scope of a variable is the portion of the program in which the variable can be used
- JavaScript has function scope
 - The scope changes inside functions
- A variable declared outside a function has global scope
 - In the HTML context, all scripts and functions on a web page can access it.
- Variables declared inside a function has local scope
 - They can only be accessed within in the function

Outline

- **More HTML**
 - Table
 - Elements
 - Styling
 - Form
 - Controls
- **JavaScript**
 - Location and Basic Syntax
 - Variables, Control Structure, Function, Object, Array
 - **Windows and DOM object**
 - Event model

JavaScript Objects

- JavaScript contains some build-in objects for common processing
 - String, Date, Math and so on
- Client side JavaScript is able to access browser object
 - window, history, location, etc
- Client side JavaScript is able to access HTML elements as a set of objects (DOM)
 - document, various element and other objects

<https://www.w3schools.com/jsref/default.asp>

DOM standards

- Most commonly implemented specification: DOM level 2
- Several sub category
 - Core
 - Interface for manipulating hierarchically organized node sets
 - HTML
 - Support for specific HTML elements
 - Style
 - Dealing with element style and style sheets
 - Events
 - Dealing with how event handlers are attached or removed from DOM nodes

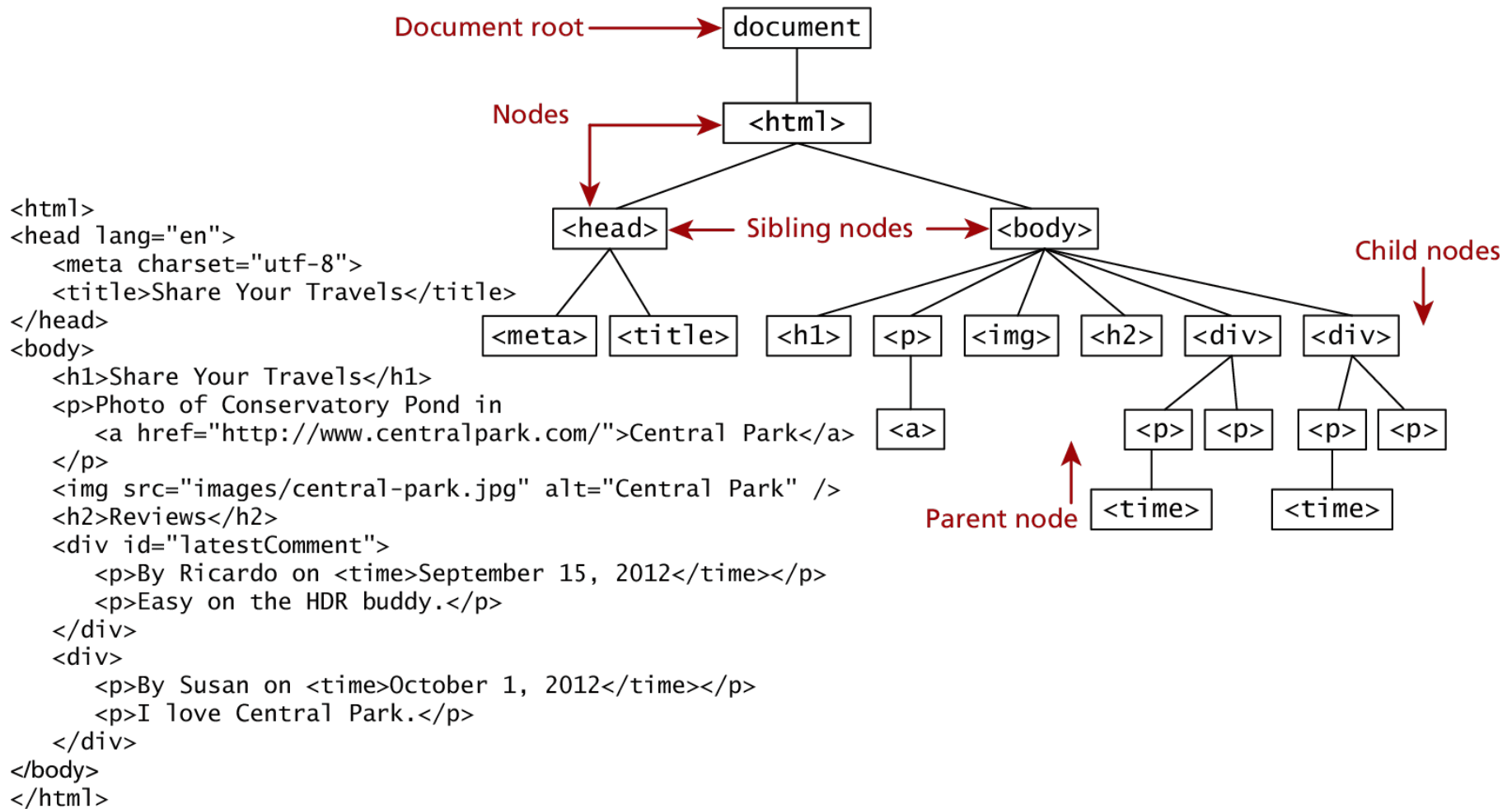
DOM Basics

- The DOM presents documents as a hierarchy of Node objects
 - Node is the most abstract concept
 - There are different types of node
 - Document: represent the root of the tree
 - Element: an HTML or XML element
 - Attr: an attribute of an element (not considered as part of a DOM tree)
 - Comment: an HTML comment
 - Text: the textual content of an Element or Attr
 - ...
 - Some node may have child node
 - Element may have other element or text as child node
- The Document Object Model gives programmers access to all the elements on a web page. Using JavaScript, programmers can create, modify and remove elements in the page dynamically.

DOM nodes and Trees

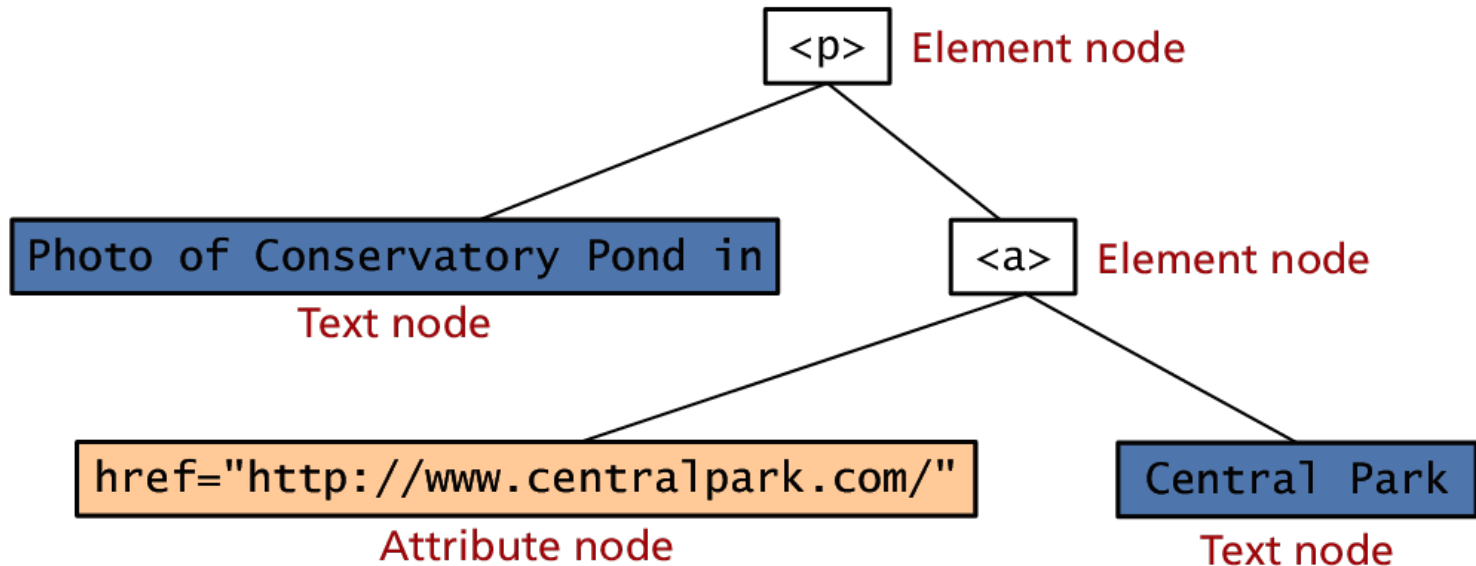
- The nodes in a document make up the page's DOM tree, which describes the relationships among elements
- Nodes are related to each other through child-parent relationships
- A node may have multiple children, but only one parent
- Nodes with the same parent node are referred to as **siblings**
- The document node in a DOM tree is called the root node, because it has no parent

The DOM



DOM Nodes

```
<p>Photo of Conservatory Pond in  
  <a href="http://www.centralpark.com/">Central Park</a>  
</p>
```



Essential Node Properties

Property	Description
attributes	Collection of node attributes
childNodes	A NodeList of child nodes for this node
firstChild	First child node of this node.
lastChild	Last child of this node.
nextSibling	Next sibling node for this node.
nodeName	Name of the node
nodeType	Type of the node
nodeValue	Value of the node
parentNode	Parent node for this node.
previousSibling	Previous sibling node for this node.

Document Object

Method	Description
<code>createAttribute()</code>	Creates an attribute node
<code>createElement()</code>	Creates an element node
<code>createTextNode()</code>	Create a text node
<code>getElementById(id)</code>	Returns the element node whose id attribute matches the passed id parameter.
<code>getElementsByTagName(name)</code>	Returns a nodeList of elements whose tag name matches the passed name parameter.

Accessing nodes

```
var abc = document.getElementById("latestComment");
```

```
<body>
  <h1>Reviews</h1>
  <div id="latestComment">
    <p>By Ricardo on <time>September 15, 2012</time></p>
    <p>Easy on the HDR buddy.</p>
  </div>
  <hr/>
  <div>
    <p>By Susan on <time>October 1, 2012</time></p>
    <p>I love Central Park.</p>
  </div>
  <hr/>
</body>
```

```
var list = document.getElementsByTagName("div");
```

Modifying a DOM element

```
var latest = document.getElementById("latestComment");
var oldMessage = latest.innerHTML;
var newMessage = oldMessage + "<p>Updated this div with JS</p>";
latest.removeChild(latest.firstChild);
latest.appendChild(document.createTextNode(newMessage));
```

LISTING 6.9 Changing the HTML using `createTextNode()` and `appendChild()`

```
var commentTag = document.getElementById("specificTag");
commentTag.style.backgroundColor = "#FFFF00";
commentTag.style.borderWidth="3px";
```

```
var commentTag = document.getElementById("specificTag");
commentTag.className = "someClassName";
```

Outline

- **More HTML**
 - Table
 - Elements
 - Styling
 - Form
 - Controls
- **JavaScript**
 - Location and Basic Syntax
 - Variables, Control Structure, Function, Object, Array
 - Windows and DOM object
 - **Event model**

Events

- HTML events are “things” that happen to HTML elements
- When JavaScript is used in HTML pages, JavaScript can “react” on these events
- An HTML event can be something the browser does, or something a user does
- Examples:
 - An HTML web page has finished loading
 - An HTML input field was changed
 - An HTML button was clicked
- Event handler
 - A function describes what we want to do when an event happens

Registering Event Handler

```
function displayTheDate() {  
    var d = new Date();  
    alert ("You clicked this on " + d.toString());  
}  
var element = document.getElementById('example1');  
element.onclick = displayTheDate;  
  
// or using the other approach  
element.addEventListener('click',displayTheDate);
```

LISTING 6.12 Listening to an event with a function

```
var element = document.getElementById('example1');  
element.onclick = function() {  
    var d = new Date();  
    alert ("You clicked this on " + d.toString());  
};
```

LISTING 6.13 Listening to an event with an anonymous function

Common HTML Events

- Mouse Events
 - onclick, onmousedown, onmouseenter,...
- Keyboard Events
 - onkeydown, onkeyup, ...
- Form events
 - onfocus, onblur, onsubmit, ...
- Frame/Object events
 - onload, onscroll, ...
- Not all browsers implements all events

The onload event

- Both frame and object can fire onload event
 - **Frame** refers to the browser frame that contains the current web page.
 - Onload event fires when “something” is loaded
 - A whole page or a single element

```
window.onload= function(){  
    //all JavaScript initialization here.  
}
```

The event Object and this

- Event object stores contextual information about the event
 - This can be passed to the event handler
 - The object has a number of properties and methods
- In an event-handling function, `this` refers to the target DOM node on which the event occurred

```
document.getElementById("loginForm").onsubmit = function(e){
    var fieldValue=document.getElementById("username").value;
    if(fieldValue==null || fieldValue== ""){
        // the field was empty. Stop form submission
        e.preventDefault();
        // Now tell the user something went wrong
        alert("you must enter a username");
    }
}
```

LISTING 6.18 A simple validation script to check for empty fields

References

- W3C schools [<http://www.w3schools.com/>]
 - HTML tutorial
 - JavaScript Tutorial
- Fundamentals of Web Developments
 - Chapter 4,6