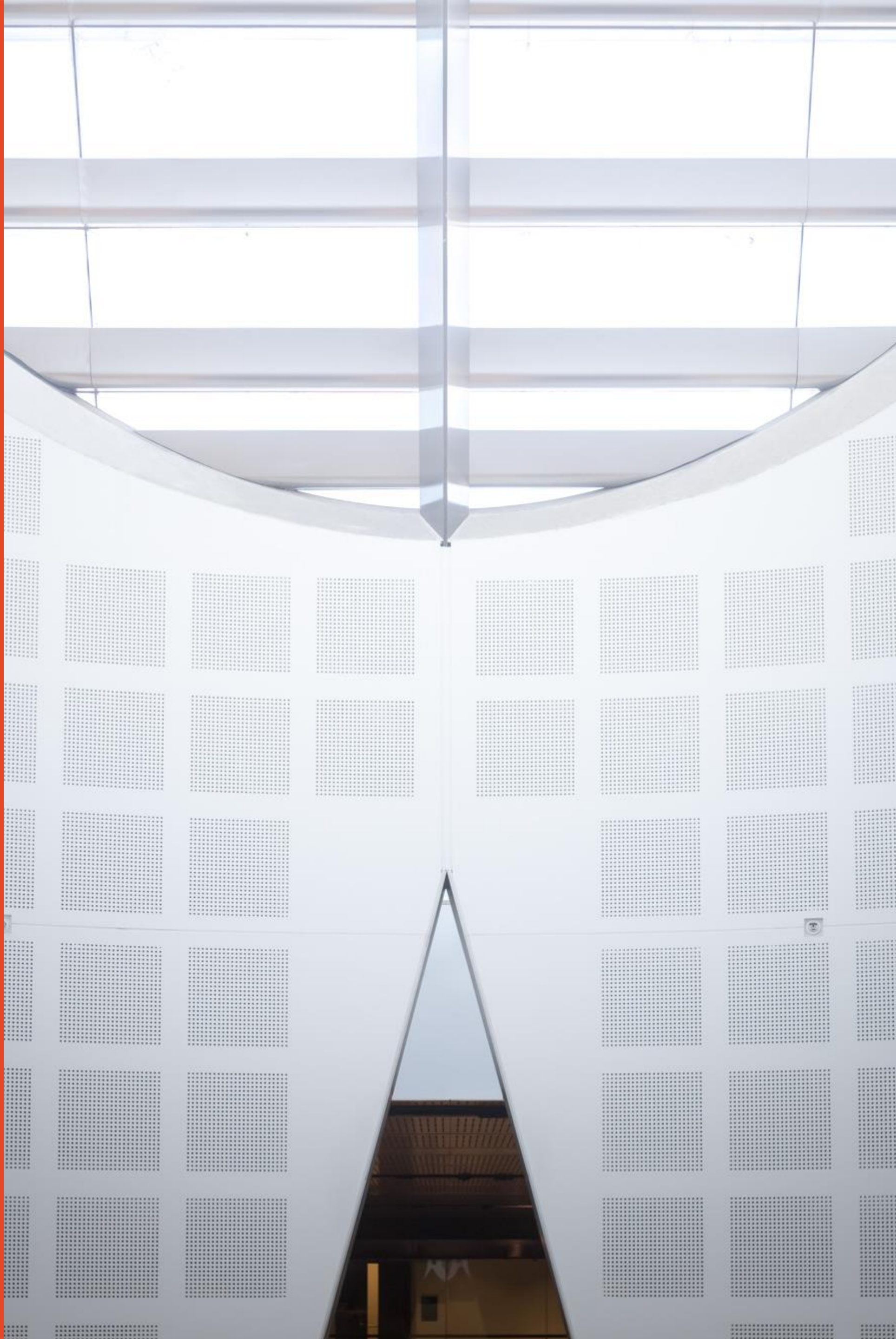
COMP5347: Web Application Development

HTML and Client-Side JavaScript

Dr. Basem Suleiman

School of Computer Science



Outline

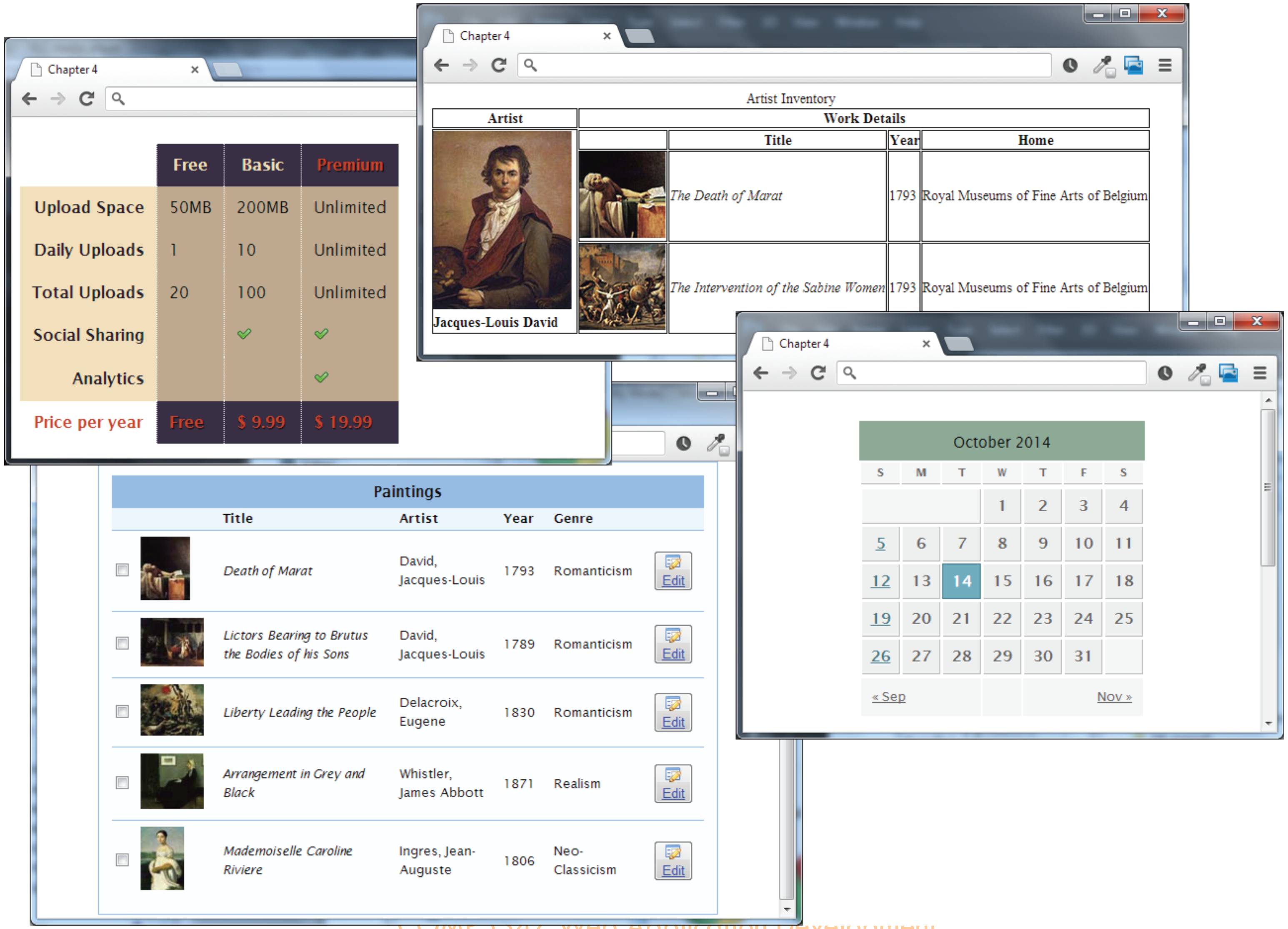
- More HTML
  - Table
    - Elements
    - Styling
  - Form
    - Controls
- JavaScript
  - Location and Basic Syntax
    - Variables, Control Structure, Function, Object, Array
    - More about functions, objects, variable scopes, passing function as parameter
  - Windows and DOM object
  - Event model

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>**

HTML Table Examples



	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

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

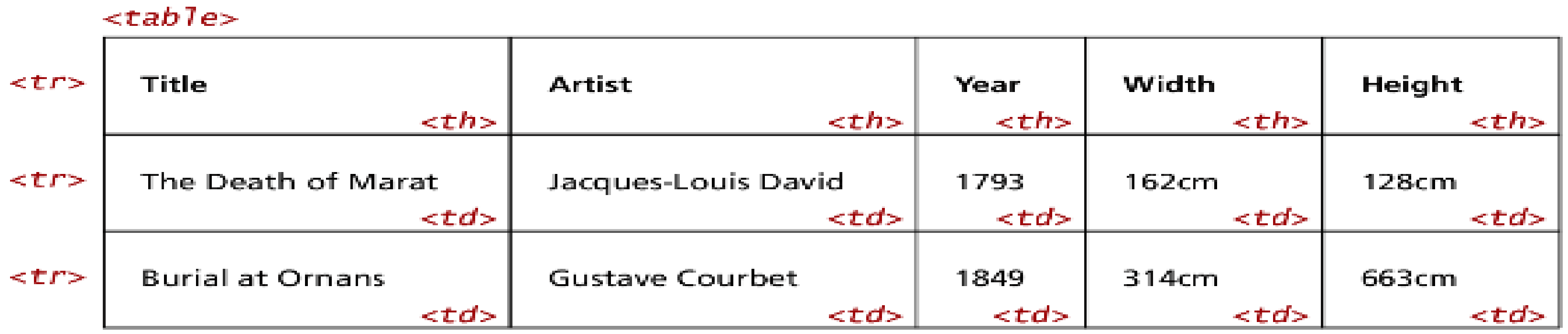
Paintings

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

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 »	

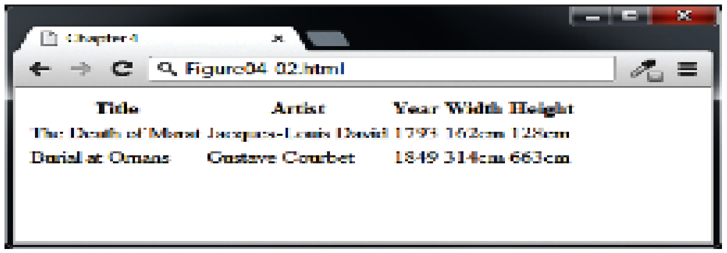
Basic Table Example



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

th

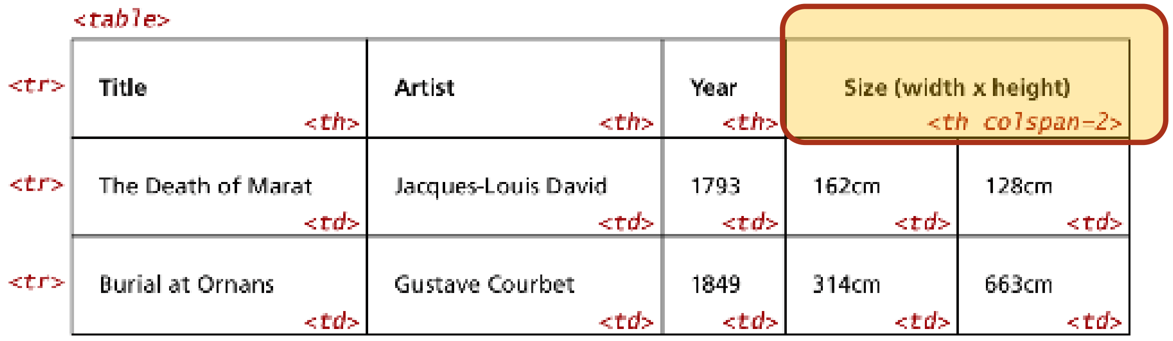
```
<table>
  <tr>
     <th>Title</th>
     <th>Artist</th>
     <th>Year</th>
     <th>Width</th>
     <th>Height</th>
  </tr>
  <tr>
     <td>The Death of Marat</td>
     <td>Jacques-Louis David</td>
     <td>1793</td>
     <td>162cm</td>
     <td>128cm</td>
  </tr>
  <tr>
     <td>Burial at Ornans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



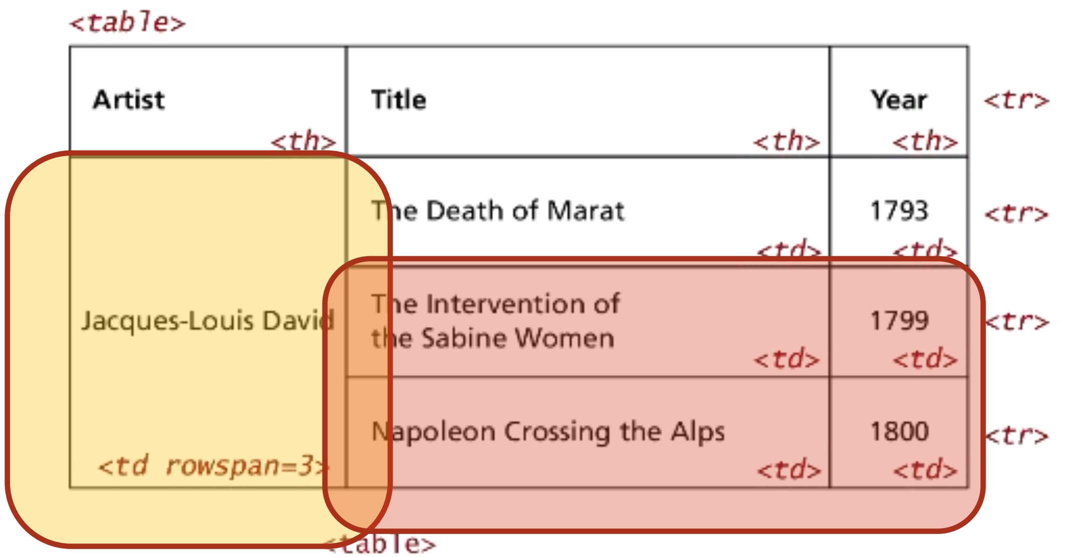
Title	Artist	Year	Size (width x height)	
The Death of Marat	Jacques-Louis David	1793	162cm	128cm
Burial at Ornans	Gustave Courbet	1849	314cm	663cm

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>
```

use the **colspan or rowspan attributes**

Row Spaning Example



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

```
<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>
```

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

Additional Table Elements

<caption>

<col>

<colgroup>

<thead>

<tfoot>

<tbody>

A title for the table is good for accessibility.

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

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

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

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

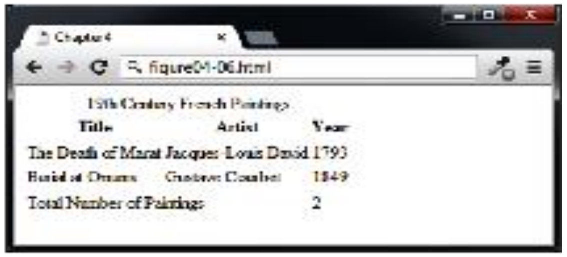
```
<table>
   <caption>19th Century French Paintings</caption>
   <col class="artistName" />
   <colgroup id="paintingColumns">
       <col />
       <col />
   </colgroup>

   <thead>
      <tr>
         <th>Title</th>
         <th>Artist</th>
         <th>Year</th>
      </tr>
   </thead>

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

   <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>
```



19th Century French Paintings

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

Tables – Layout

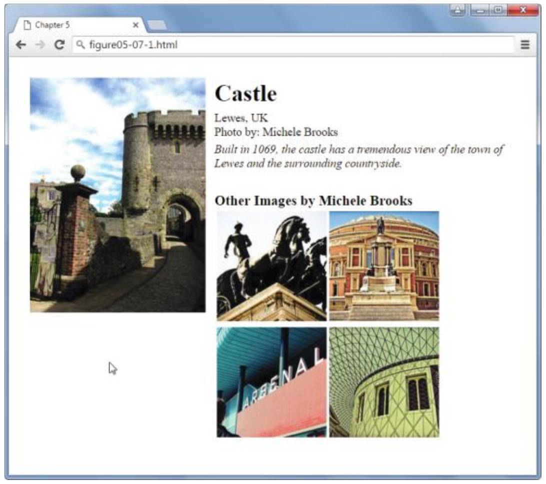

Chapter 5

figure05-07-1.html

Castle

Lewes, UK

Photo by: Michele Brooks

Built in 1069, the castle has a tremendous view of the town of Lewes and the surrounding countryside.

Other Images by Michele Brooks

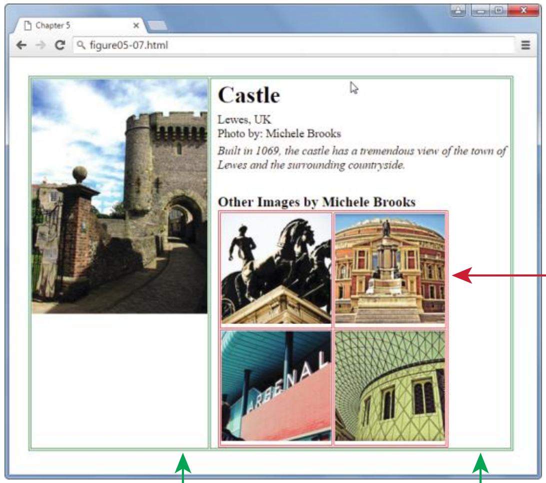

Chapter 5

figure05-07.html

Castle

Lewes, UK

Photo by: Michele Brooks

Built in 1069, the castle has a tremendous view of the town of Lewes and the surrounding countryside.

Other Images by Michele Brooks

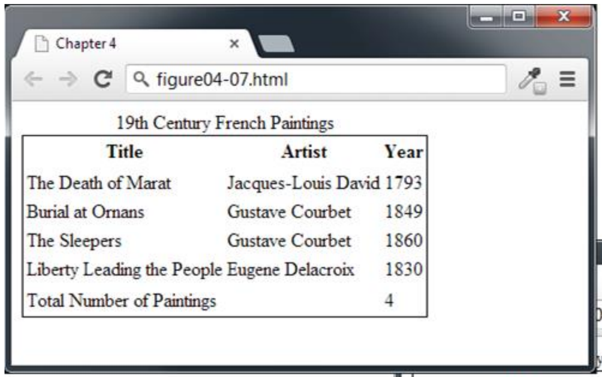
```
<table>
  <tr>
    <td>
      <img src="images/959.jpg" alt="Castle"/>
    </td>
    <td>
      <h2>Castle</h2>
      <p>Lewes, UK</p>
      <p>Photo by: Michele Brooks</p>
      <p>Built in 1069, the castle has a tremendous
         view of the town of Lewes and the
         surrounding countryside.
      </p>

      <h3>Other Images by Michele Brooks</h3>

      <table>
        <tr>
           <td><img src="images/464.jpg" /></td>
           <td><img src="images/537.jpg" /></td>
        </tr>
        <tr>
           <td><img src="images/700.jpg" /></td>
           <td><img src="images/828.jpg" /></td>
        </tr>
      </table>
    </td>
  </tr>
</table>
```

Styling Tables

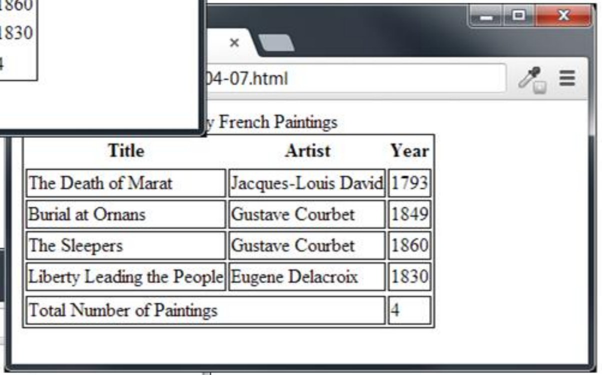
– Most box model styling can be applied to **<table>**, **<tr>**, **<td>** and other tags



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

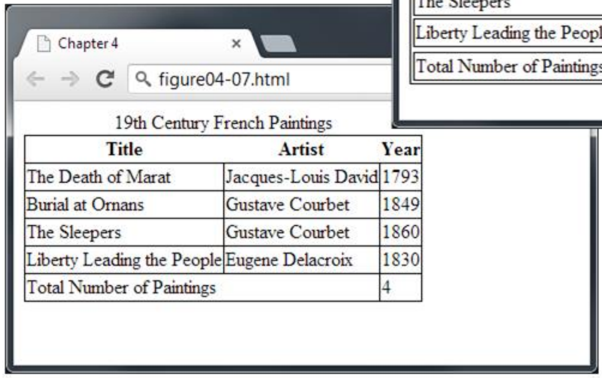
```
table {
    border: solid 1pt black;
}
```



y 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;
}
```

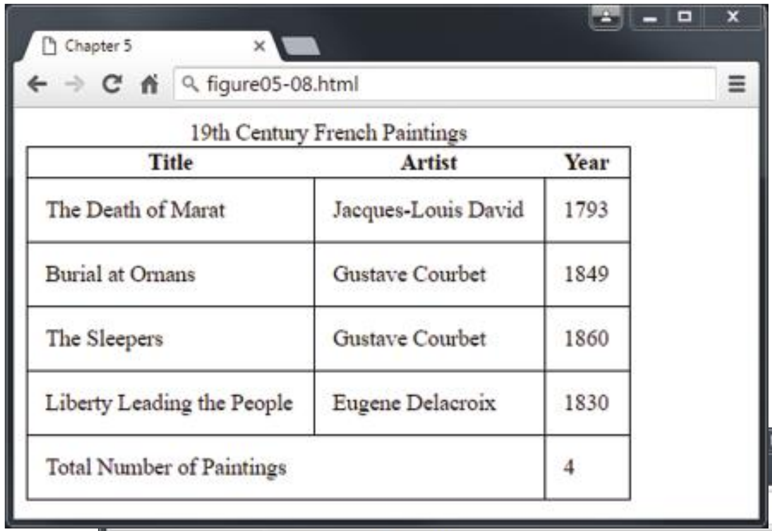


19th Century French Paintings

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;
}
```

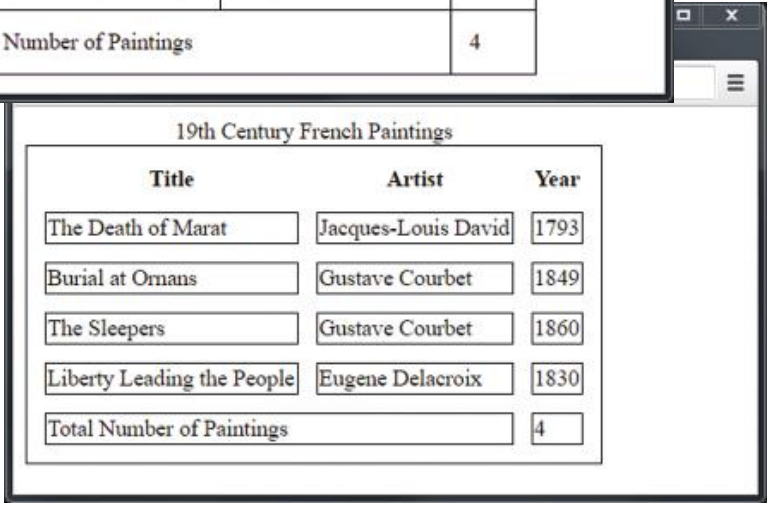
Styling Tables


Chapter 5
figure05-08.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

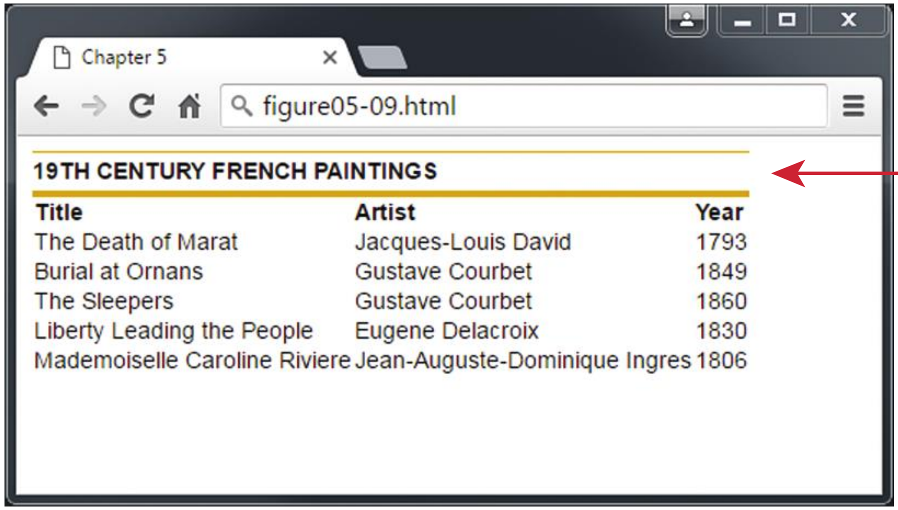
```
table {
   border: solid 1pt black;
   border-collapse: collapse;
}
td {
   border: solid 1pt black;
   padding: 10pt;
}
```


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-spacing: 10pt;
}
td {
   border: solid 1pt black;
}
```

Styling Tables

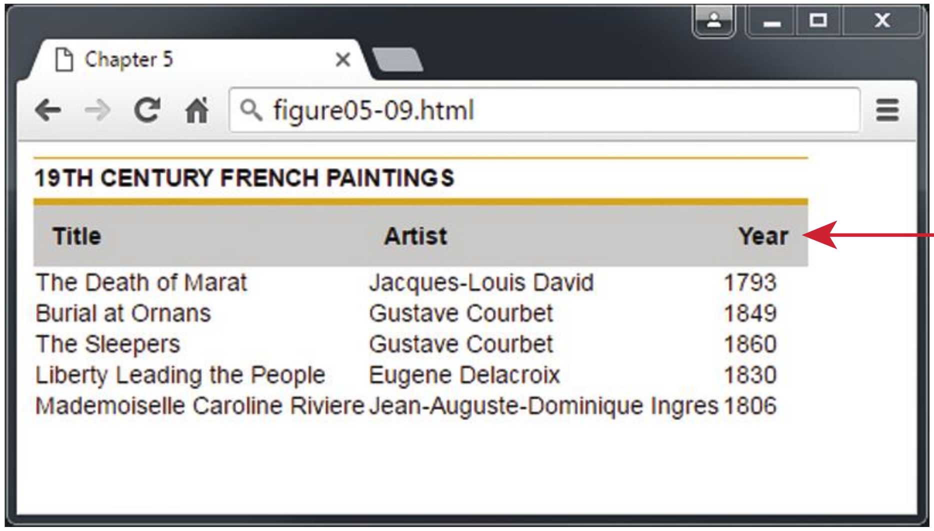


19TH CENTURY FRENCH PAINTINGS

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
Mademoiselle Caroline Riviere	Jean-Auguste-Dominique Ingres	1806

```
caption {
   font-weight: bold;
   padding: 0.25em 0 0.25em 0;
   text-align: left;
   text-transform: uppercase;
   border-top: 1px solid #DCA806;
}
table {
   font-size: 0.8em;
   font-family: Arial, sans-serif;
   border-collapse: collapse;
   border-top: 4px solid #DCA806;
   border-bottom: 1px solid white;
   text-align: left;
}
```

Styling Tables

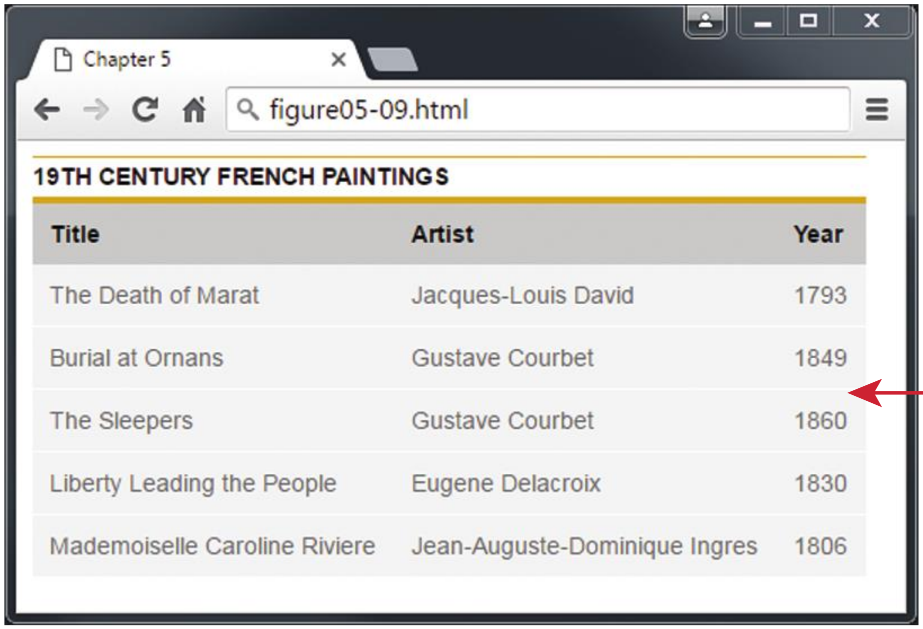


19TH CENTURY FRENCH PAINTINGS

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
Mademoiselle Caroline Riviere	Jean-Auguste-Dominique Ingres	1806

```
thead tr {
   background-color: #CACACA;
}
th {
   padding: 0.75em;
}
```

Styling Tables

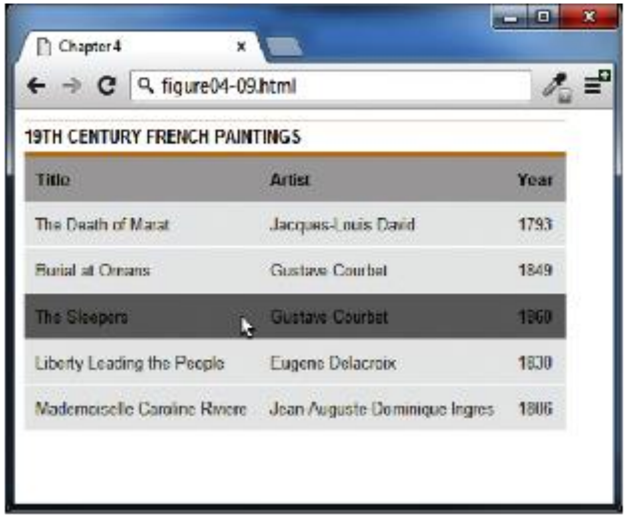


19TH CENTURY FRENCH PAINTINGS

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
Mademoiselle Caroline Riviere	Jean-Auguste-Dominique Ingres	1806

```
tbody tr {
   background-color: #F1F1F1;
   border-bottom: 1px solid white;
   color: #6E6E6E;
}
tbody td {
   padding: 0.75em;
}
```

Nifty Table Styling Tricks: hover effect and zebra-stripes

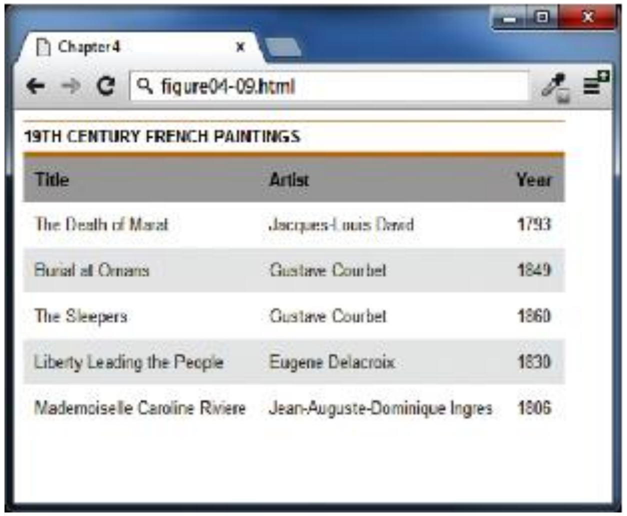


19TH CENTURY FRENCH PAINTINGS

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
Mademoiselle Caroline Riviere	Jean Auguste Dominique Ingres	1806

Pseudo class

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



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
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

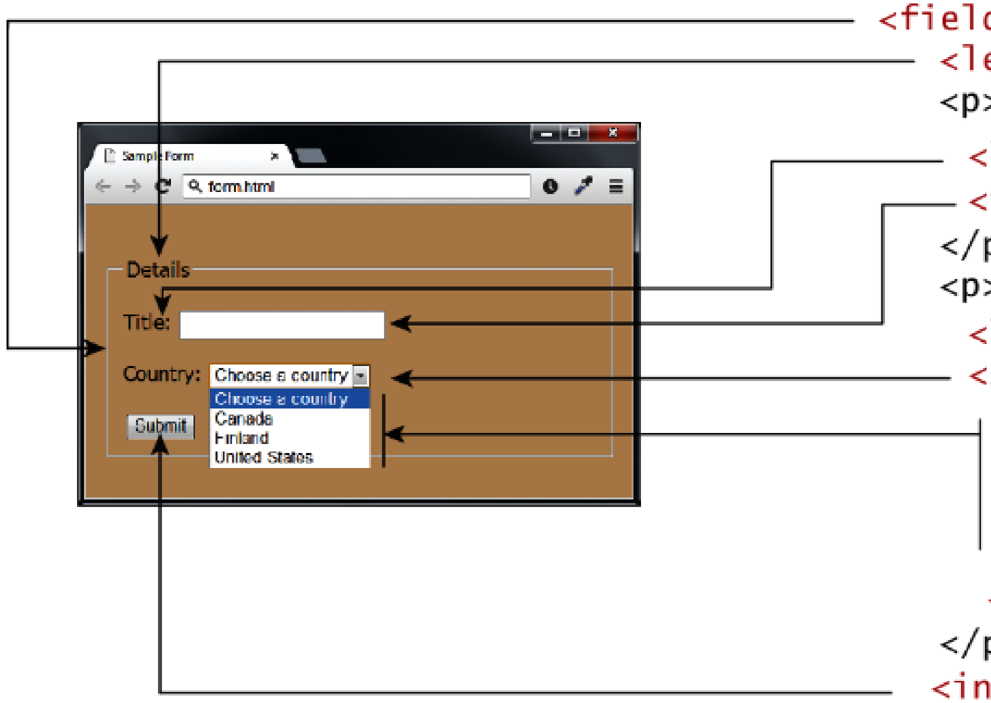
HTML Forms

- 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

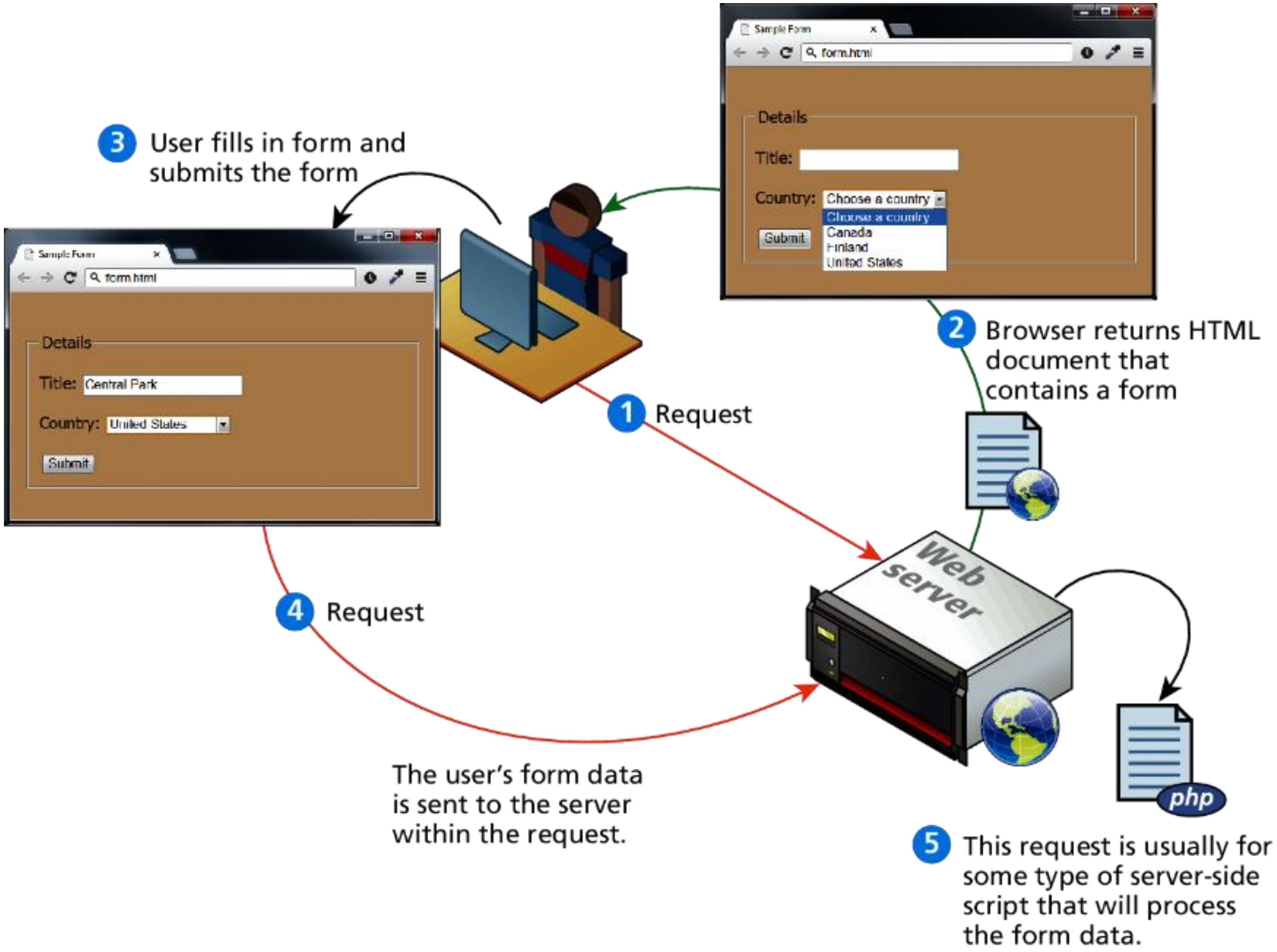
Form Structures

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



```
<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>
```

How Forms Work?



Sample Form
form.html
Details
Title:
Country:
Choose a country
Canada
Finland
United States
Submit
2 Browser returns HTML document that contains a form
3 User fills in form and submits the form
Sample Form
form.html
Details
Title: Central Park
Country: United States
Submit
1 Request
4 Request
Web server
php
The user's form data is sent to the server within the request.
5 This request is usually for some type of server-side script that will process the form data.

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

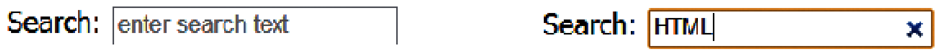
Text Input Controls

Key motivations of new form controls in HTML5

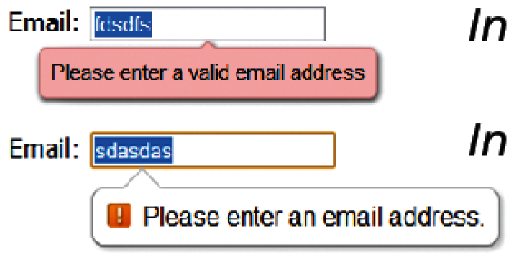
- Usability
- Styling
- Client-side validation

Text Input Controls – Examples

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



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



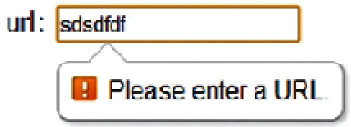
Please enter a valid email address

In Opera

Please enter an email address.

In Chrome

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

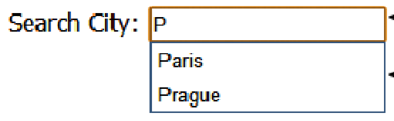


Please enter a URL.

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

Select Lists

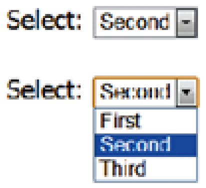
Datalist element



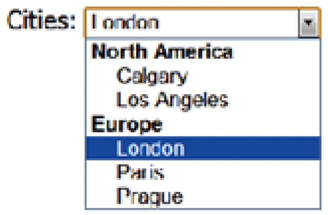
```
<input type="text" name="city" list="cities"  />

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

Select Lists



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



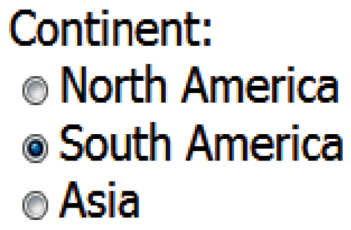
```
<select … >
 <optgroup label="North America">
   <option>Calgary</option>
   <option>Los Angeles</option>
 </optgroup>
 <optgroup label="Europe">
   <option>London</option>
   <option>Paris</option>
   <option>Prague</option>
 </optgroup>
</select>
```

HTML Forms – Query Strings

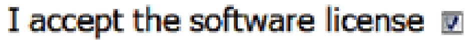
How the browser sends the data to the server

- Through HTTP requests
- The browser packages user's data into a query string
- Query string: a series of name=value pairs separated by &
    - HTML form element's name attribute
    - User input data

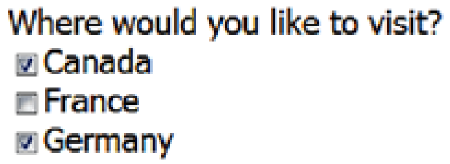
Radio Buttons and Checkboxes


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" >
```

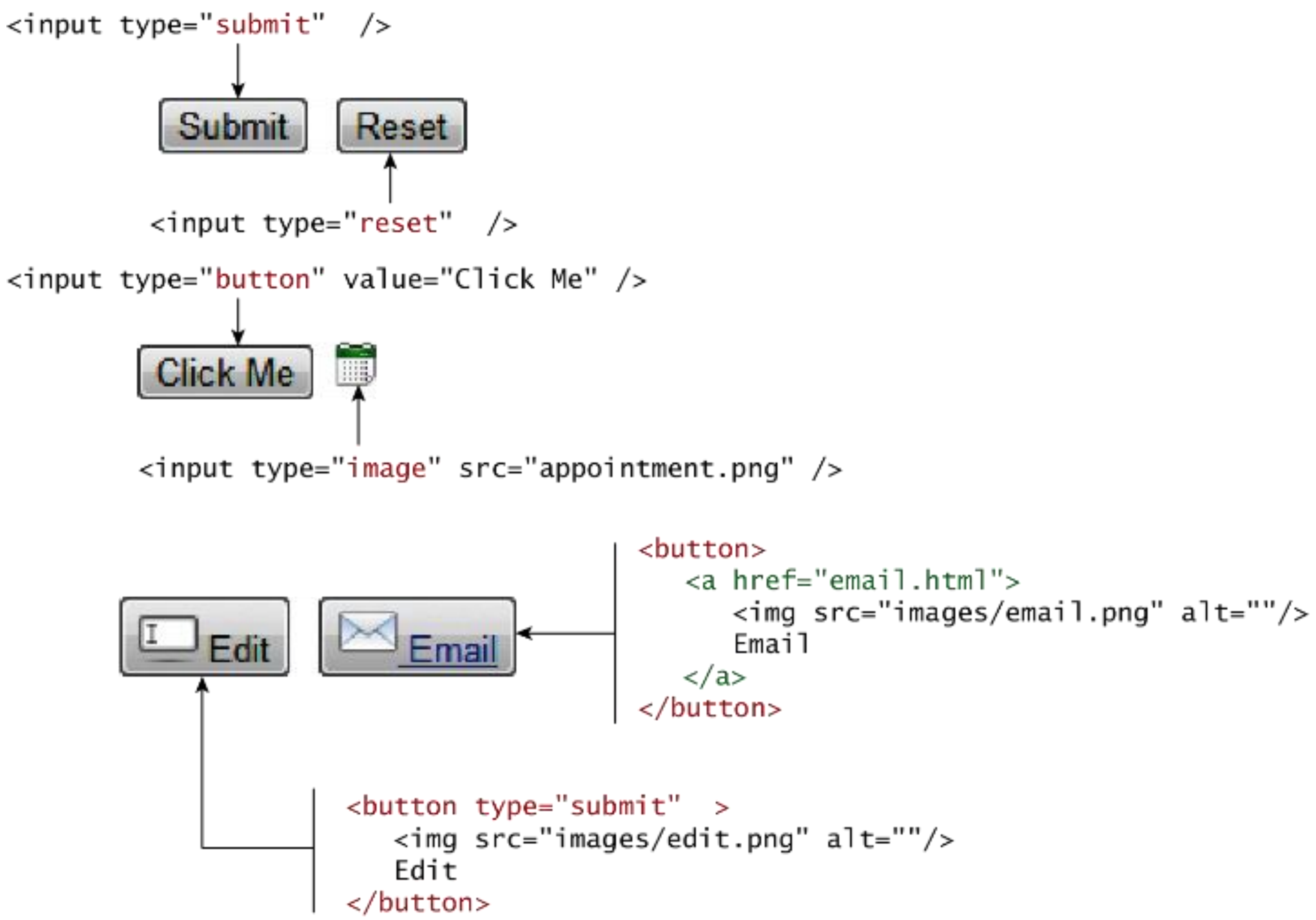

Where would you like to visit?
☑ Canada
☐ France
☑ Germany

```
<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
```

Button Controls

Type	Description
<input type="submit">	Creates a button that submits the form data to the server.
<input type="reset">	Creates a button that clears any of the user's already entered form data.
<input type="button">	Creates a custom button. This button may require Javascript for it to actually perform any action.
<input type="image">	Creates a custom submit button that uses an image for its display.
<button>	Creates a custom button. The <button> element differs from <input type="button"> 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. You can turn the button into a submit button by using the type="submit" attribute.

Button Controls – Example



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

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

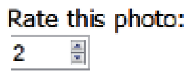
```
<input type="button" value="Click Me" />
```

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

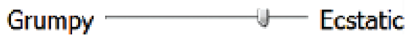
```
<button>
   <a href="email.html">
      <img src="images/email.png" alt=""/>
      Email
   </a>
</button>
```

```
<button type="submit"  >
   <img src="images/edit.png" alt=""/>
   Edit
</button>
```

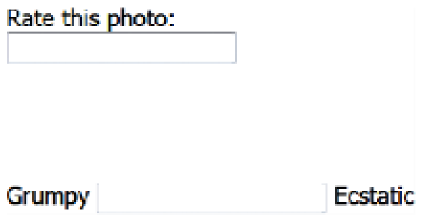
Form Control Elements – Number and Ranges



```
<label>Rate this photo: <br/>
<input type="number" min="1" max="5" name="rate" />
```

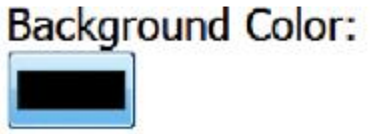


```
Grumpy
<input type="range" min="0" max="10" step="1" name="happiness" />
Ecstatic
```

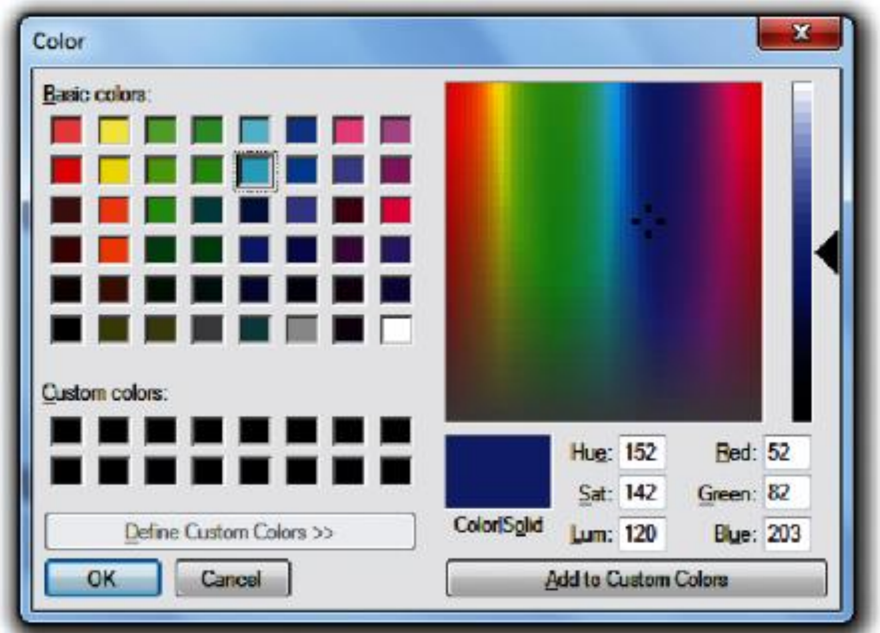


Controls as they appear in browser that doesn't support these input types

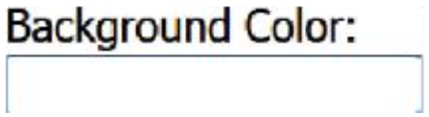
Form Control Elements – Color



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

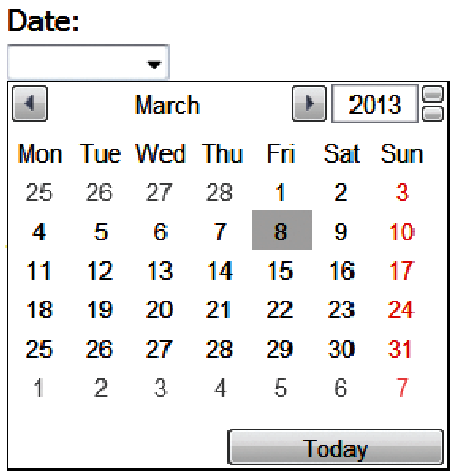


Hue: 152
Sat: 142
Lum: 120
Red: 52
Green: 82
Blue: 203



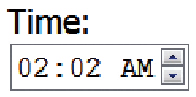
Control as it appears in browser that doesn't support this input type

Form Control Elements – Date and Time

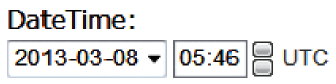


Mon	Tue	Wed	Thu	Fri	Sat	Sun
25	26	27	28	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

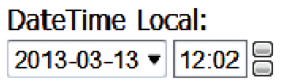
```
<label>Date: <br/>
<input type="date" ... />
```



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

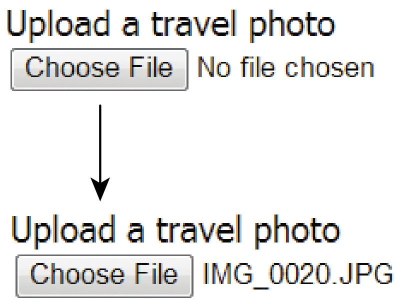


```
<input type="datetime" ... />
```



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

Form Control Elements – File Upload



```
<form method="post" enctype="multipart/form-data" ... >
   ...
   <label>Upload a travel photo</label>
   <input type="file" name="photo" />
   ...
</form>
```

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
  - client-side scripting language for HTML and CSS
  - Also a server-side implementation
  - "*the most popular programming language in the world*" - W3C school

- 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

- Created in 10 days in May 1995 by Brendan Erich, then working at Netscape and now of Mozilla

- Became a much more important part of web development in the mid 2000s with **AJAX**
  - Microsoft 1999, get adopted by other browsers
  - Made very popular by Google
  - Received a lot more professional programming attention

- JavaScript frameworks: jQuery, Prototype, AngularJS, etc.

- Server-side JavaScript also gaining popularity

JavaScript Code – Location

Inline

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

Embedded

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

External

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

JavaScript Variables

- Declaring a variable
  - *`var name;`*
- Does not require specifying data types
- Can contain a value of any data type
- JavaScript automatically converts between values of different types (in many cases)
- 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
```

Conditionals

```
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";
}
```

```
/* x conditional assignment */
x = (y==4) ? "y is 4" : "y is not 4";
```

Condition   Value if true   Value if false

```
/* equivalent to */
if (y==4) {
   x = "y is 4";
}
else {
   x = "y is not 4";
}
```

Conditionals

```
switch (artType) {
    case "PT":
        output = "Painting";
        break;
    case "SC":
        output = "Sculpture";
        break;
    default:
    output = "Other";
}
```

Loops

```
          initialization    condition   post-loop operation
for (var i = 0; i < 10; i++) {
  // do something with i
  // ...
}
```

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

Variable types

- Primitive types: represent simple forms of data
  - Boolean, string and number
  - Null, undefined

- Complex types
  - Object (reference types)
  - Array
  - Function

Primitive Types vs. Reference Types

What is the difference between primitive types and reference types? Use the following examples to expain it to your classmate.

```
var abc = 27;

var def = ”hello”;

var foo = [45, 35, 25]

var xyz = def;

var bar = foo;

bar[0] = 200;
```

Primitive Types vs. Reference Types

```
var abc = 27;
var def = "hello";
```

variables with primitive types

```
var foo = [45, 35, 25];
```

variable with reference type (i.e., array object)

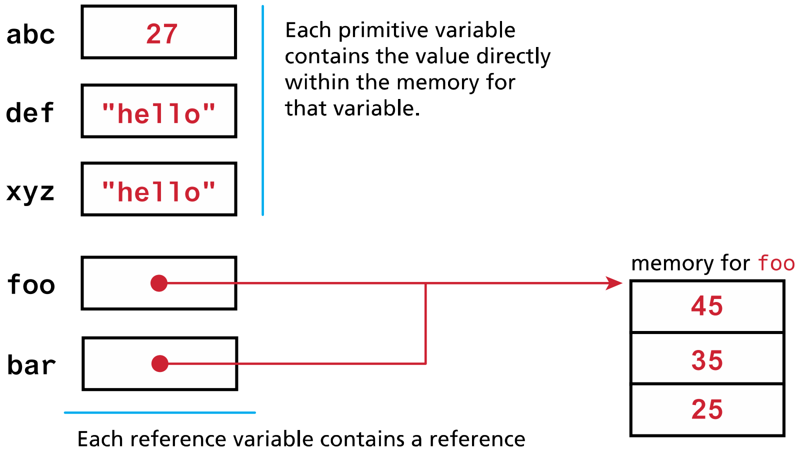
```
var xyz = def;
var bar = foo;
```

these new variables differ in important ways (see below)

```
bar[0] = 200;
```

changes value of the first element of array

Memory representation



Each primitive variable contains the value directly within the memory for that variable.

This element will get changed to the value of 200. Thus both `foo[0]` and `bar[0]` will have the same value (200).

Each reference variable contains a reference (or pointer) to the memory that contains the contents of that object.

JavaScript – Objects

- JavaScript is different to classic OOP, which is class-based
  - It has a clear concept of ***Object*** similar to object in other OOP
  - 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

Object Creation using Literal

```
var objName = {
      name1: value1,
      name2: value2,
      // …
      nameN: valueN
};
```

– Access using either of:
- objName.name1
- objName["name1"]

Object Creation 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;
            }
      };
```

JavaScripts – Arrays

- Arrays are used to store multiple values in a single variable
- Object literal notation
  - var greetings = ["Good Morning", "Good Afternoon"];
- Array() constructor
  - Var greetings = new Array("Good Morning", "Good Afternoon");

- Array element is accessible with index, starting from 0, greetings[0] = “Good Morning”

- Useful methods length(), push(), reverse(), sort(),

JavaScript Functions

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

Example:

```
function subtotal(price,quantity) {
    return price * quantity;
}
```

- Call/invoke function:

```
var result = subtotal(10,2);
```

Functions – Function Expression

A function can be defined using an ***anonymous*** *function expression*

```
var calculateSubtotal = function (price,quantity) {
    return price * quantity;
};
// invokes the function
var result = calculateSubtotal (10,2);
```

JavaScript – Nested Functions

```
function calculateTotal (price,quantity) {
    var subtotal = price * quantity;
    return subtotal + calculateTax(subtotal);
    // this function is nested
    function calculateTax(subtotal) {
        var taxRate = 0.05;
        var tax = subtotal * taxRate;
        return tax;
    }
}
```

Functions – Hoisting

Listing 1

```
function calculateTotal(price,quantity) {
    var subtotal = price * quantity;
    return subtotal + calculateTax(subtotal);

    function calculateTax(subtotal) {
        var taxRate = 0.05;
        var tax = subtotal * taxRate;
        return tax;
    }
}
```

Listing 2

```
function calculateTotal(price,quantity) {
    var subtotal = price * quantity;
    return subtotal + calculateTax(subtotal);

    var calculateTax = function (subtotal) {
        var taxRate = 0.05;
        var tax = subtotal * taxRate;
        return tax;
    };
}
```

Functions – Hoisting

Function declaration is **hoisted** to the beginning of its scope

```
function calculateTotal(price,quantity) {
    var subtotal = price * quantity;
    return subtotal + calculateTax(subtotal);

    function calculateTax(subtotal) {
        var taxRate = 0.05;
        var tax = subtotal * taxRate;
        return tax;
    }
}
```

Variable declaration is hoisted to the beginning of its scope

BUT

Variable assignment is **not** hoisted

```
function calculateTotal(price,quantity) {
    var subtotal = price * quantity;
    return subtotal + calculateTax(subtotal);

    var calculateTax = function (subtotal) {
        var taxRate = 0.05;
        var tax = subtotal * taxRate;
        return tax;
    };
}
```

THUS

The value of the `calculateTax` variable here is undefined

JavaScript – Callback Functions

```
var calculateTotal = function (price, quantity, tax) {
    var subtotal = price * quantity;
    return subtotal + tax(subtotal);
};

var calcTax = function (subtotal) {
    var taxRate = 0.05;
    var tax = subtotal * taxRate;
    return tax;
};

var temp = calculateTotal(50,2,calcTax);
```

2 The local parameter variable tax is a reference to the calcTax() function

1 Passing the calcTax() function object as a parameter

We can say that calcTax variable here is a callback function

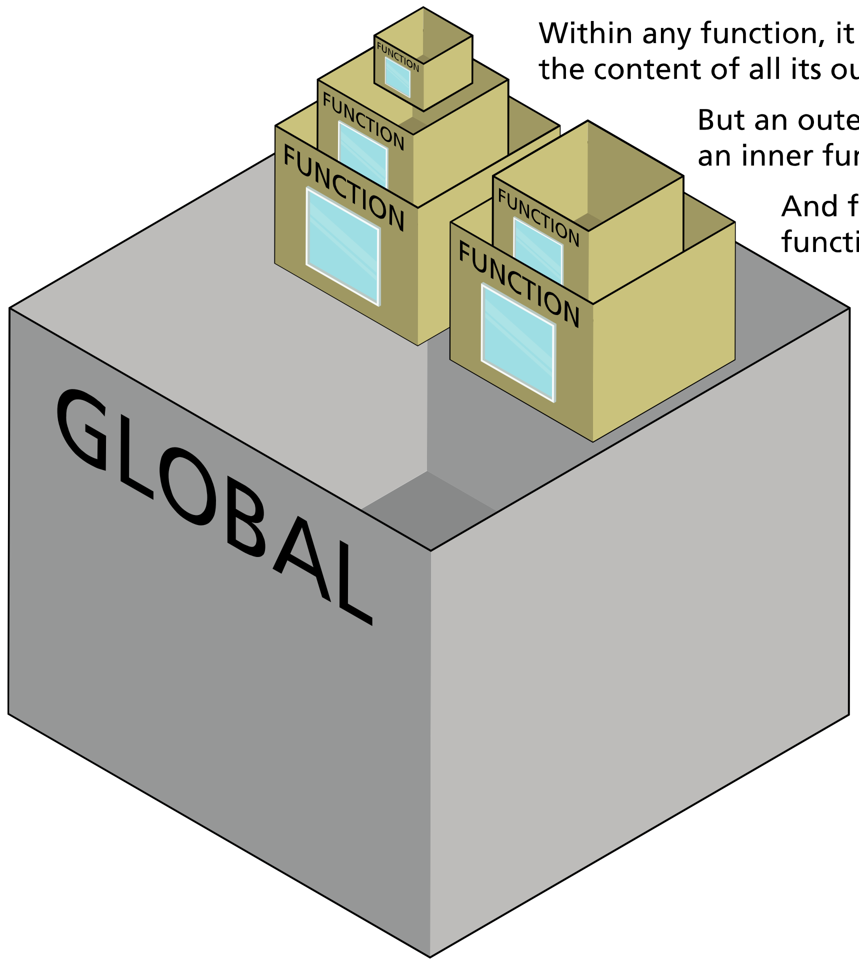
JavaScript – Callback Anonymous Function

Passing an anonymous function definition as a callback function parameter

```
var temp = calculateTotal( 50, 2,
                        function (subtotal) {
                           var taxRate = 0.05;
                           var tax = subtotal * taxRate;
                           return tax;
                        }
                       );
```

JavaScript – Variable Scope

Each function is like a box
with a one-way window



Within any function, it can see out at
the content of all its outer boxes

But an outer function can't look into
an inner function

And functions can't see into other
functions at the same level

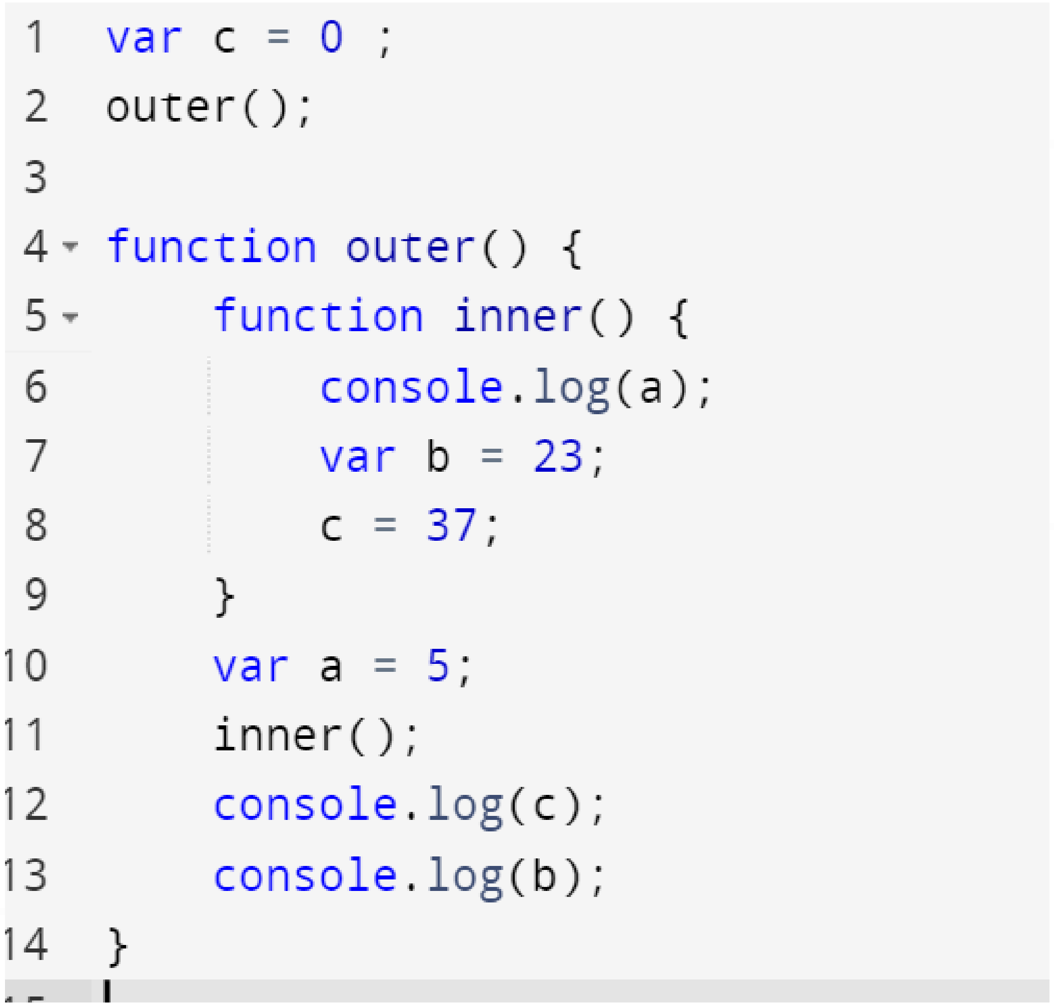
All functions can see anything
within global scope

Scope ends at global ...
functions can't see outside
of the global box

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 <u>outside</u> a function has *<u>globa</u>*l scope
  - In the HTML context, all scripts and functions on a webpage can access it.
- Variables declared <u>inside</u> a function has *<u>local</u>* scope
  - They can only be accessed within in the function

Functions and Variable Scope – Exercise

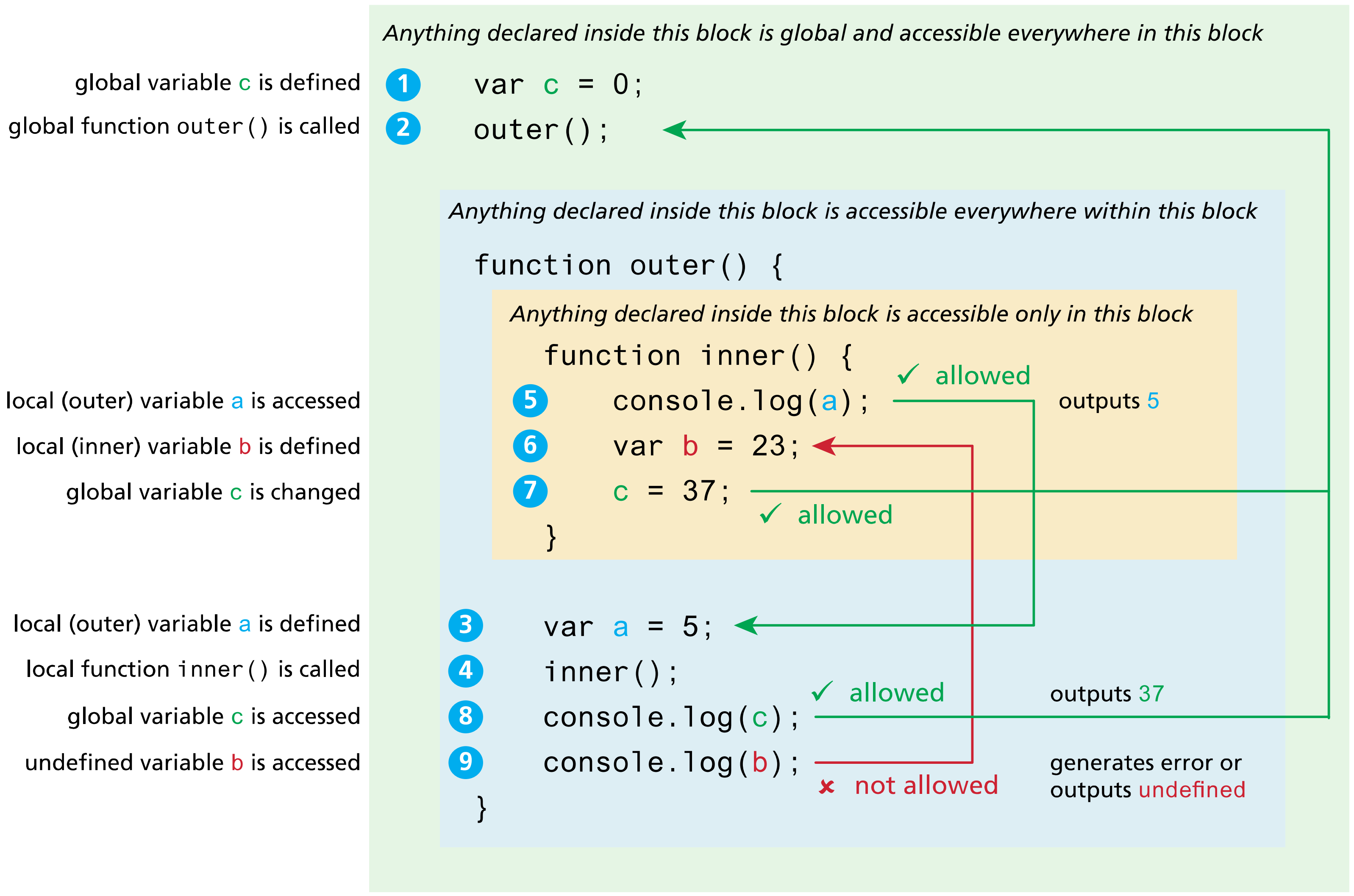


```
var c = 0 ;
outer();

function outer() {
    function inner() {
        console.log(a);
        var b = 23;
        c = 37;
    }
    var a = 5;
    inner();
    console.log(c);
    console.log(b);
}
```

- What will be logged in the console for the variable **a**?
- What will be logged in the console for the variable **b**?
- What will be logged in the console for the variable **c**?

Variable Scope - Examples



Anything declared inside this block is global and accessible everywhere in this block

Anything declared inside this block is accessible everywhere within this block

Anything declared inside this block is accessible only in this block

```
1   var c = 0;
2   outer();

    function outer() {
        function inner() {
5           console.log(a);
6           var b = 23;
7           c = 37;
        }

3       var a = 5;
4       inner();
8       console.log(c);
9       console.log(b);
    }
```

global variable c is defined	1
global function outer() is called	2
local (outer) variable a is accessed	5
local (inner) variable b is defined	6
global variable c is changed	7
local (outer) variable a is defined	3
local function inner() is called	4
global variable c is accessed	8
undefined variable b is accessed	9

JavaScript Output

- *alert()* displays content within a pop-up box
  - *alert("Hello world");*

- *console.log()* displays content in the Browser's JavaScript console

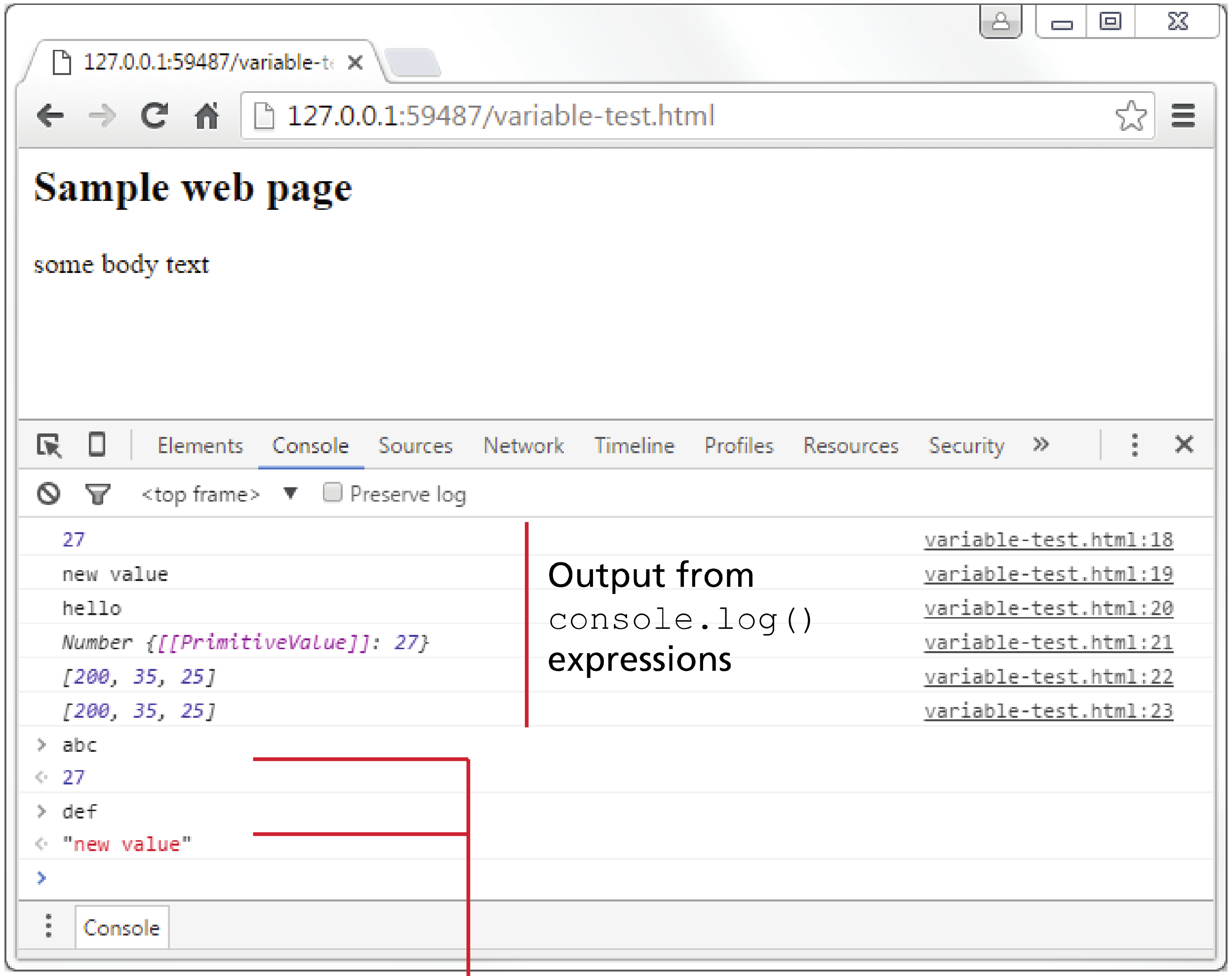
- *document.write()* outputs the content (as mark-up) directly to the HTML document

```
var name = "COMP5347";
document.write("<h1>Title</h1>");
// this uses the concatenate operator (+)
document.write("Hello " + name + " and welcome");
```

JavaScript Output

Web page content

JavaScript console



Sample web page

some body text

```
27                                          variable-test.html:18
new value                                   variable-test.html:19
hello                                       variable-test.html:20
Number {[[PrimitiveValue]]: 27}             variable-test.html:21
[200, 35, 25]                               variable-test.html:22
[200, 35, 25]                               variable-test.html:23
> abc
<· 27
> def
<· "new value"
>
```

Output from `console.log()` expressions

Using console interactively to query value of JavaScript variables

Outline

- More HTML
  - Table
    - Elements
    - Styling
  - Form
    - Controls
- JavaScript
  - Location and Basic Syntax
    - Variables, Control Structure, Function, Object, Array
    - More about functions, objects, variable scopes, passing function as parameter
  - 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 subcategories
  - 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

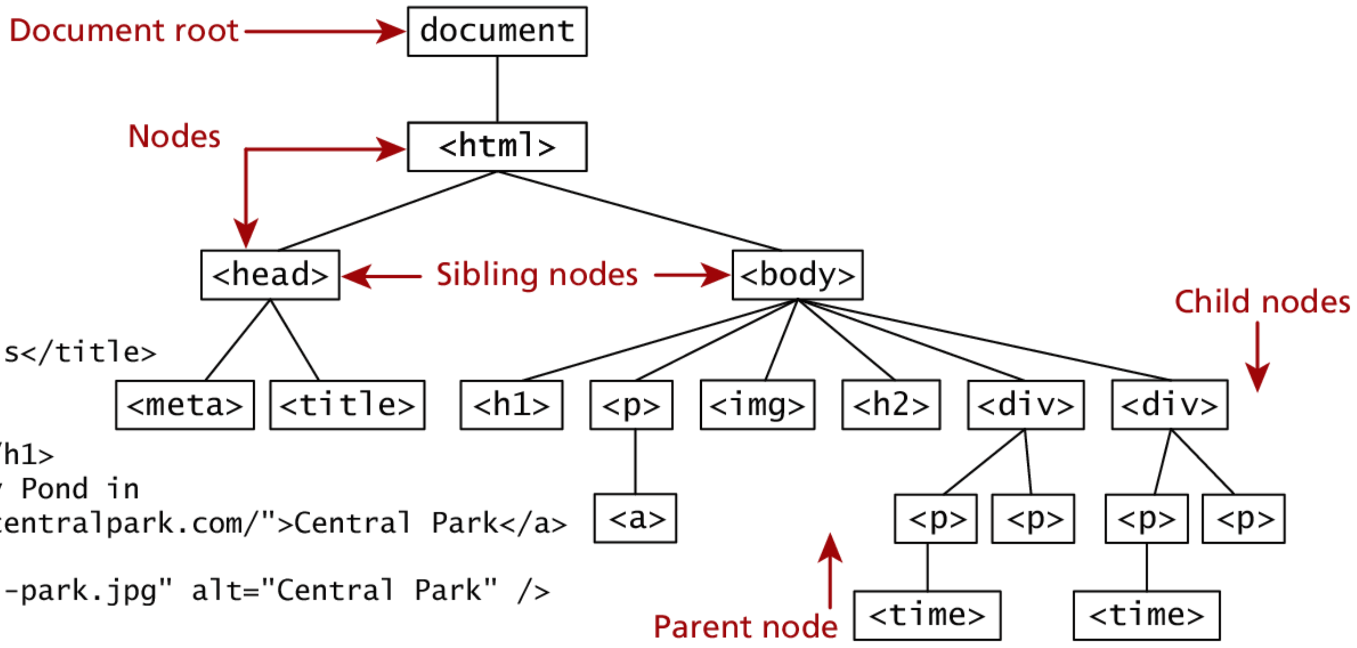
DOM Basics

- The DOM presents documents as a hierarchy of Node objects
  - Node is the most abstract concept
  - Different types of nodes
    - Document: the root of the tree
    - Element: HTML or XML element
    - Attr: attribute of an element (not considered as part of a DOM tree)
    - Comment: HTML comment
    - Text: the textual content of an Element or Attr
    - …
  - A node may have a child node
    - Element may have other element or text as child node
- DOM allows developers to access all the elements of 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
- Nodes have *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 has no parent and is called the root node

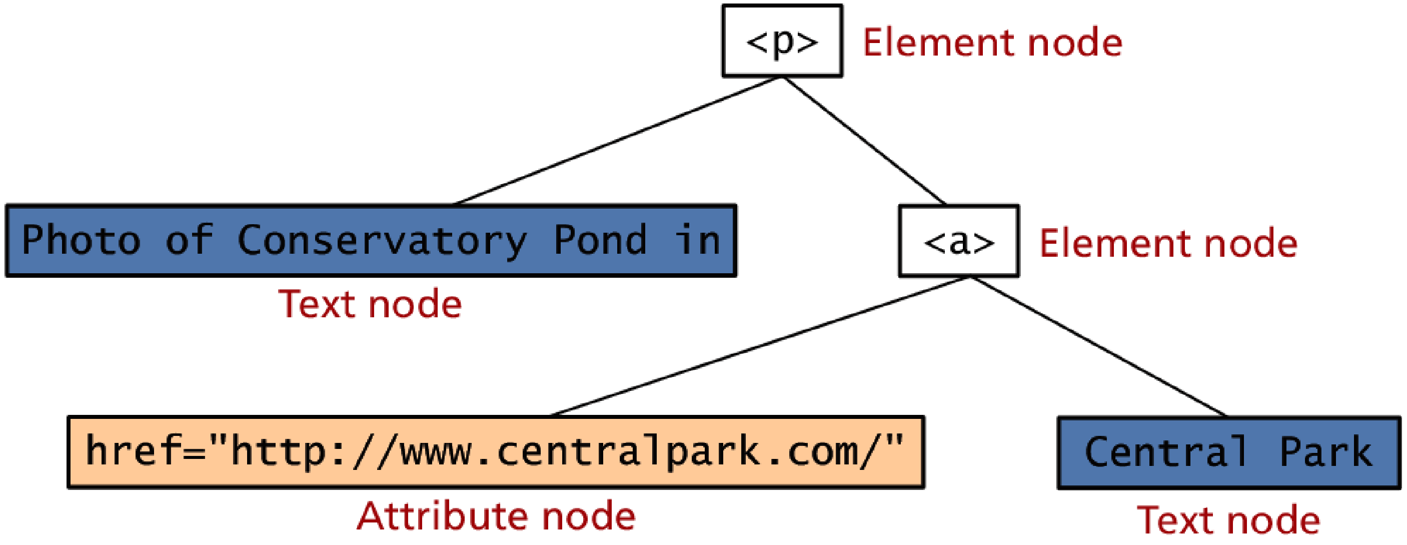
The DOM



```
<html>
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels</title>
</head>
<body>
   <h1>Share Your Travels</h1>
   <p>Photo of Conservatory Pond in
      <a href="http://www.centralpark.com/">Central Park</a>
   </p>
   <img src="images/central-park.jpg" alt="Central Park" />
   <h2>Reviews</h2>
   <div id="latestComment">
      <p>By Ricardo on <time>September 15, 2012</time></p>
      <p>Easy on the HDR buddy.</p>
   </div>
   <div>
      <p>By Susan on <time>October 1, 2012</time></p>
      <p>I love Central Park.</p>
   </div>
</body>
</html>
```

DOM Nodes

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



Essential Node Properties

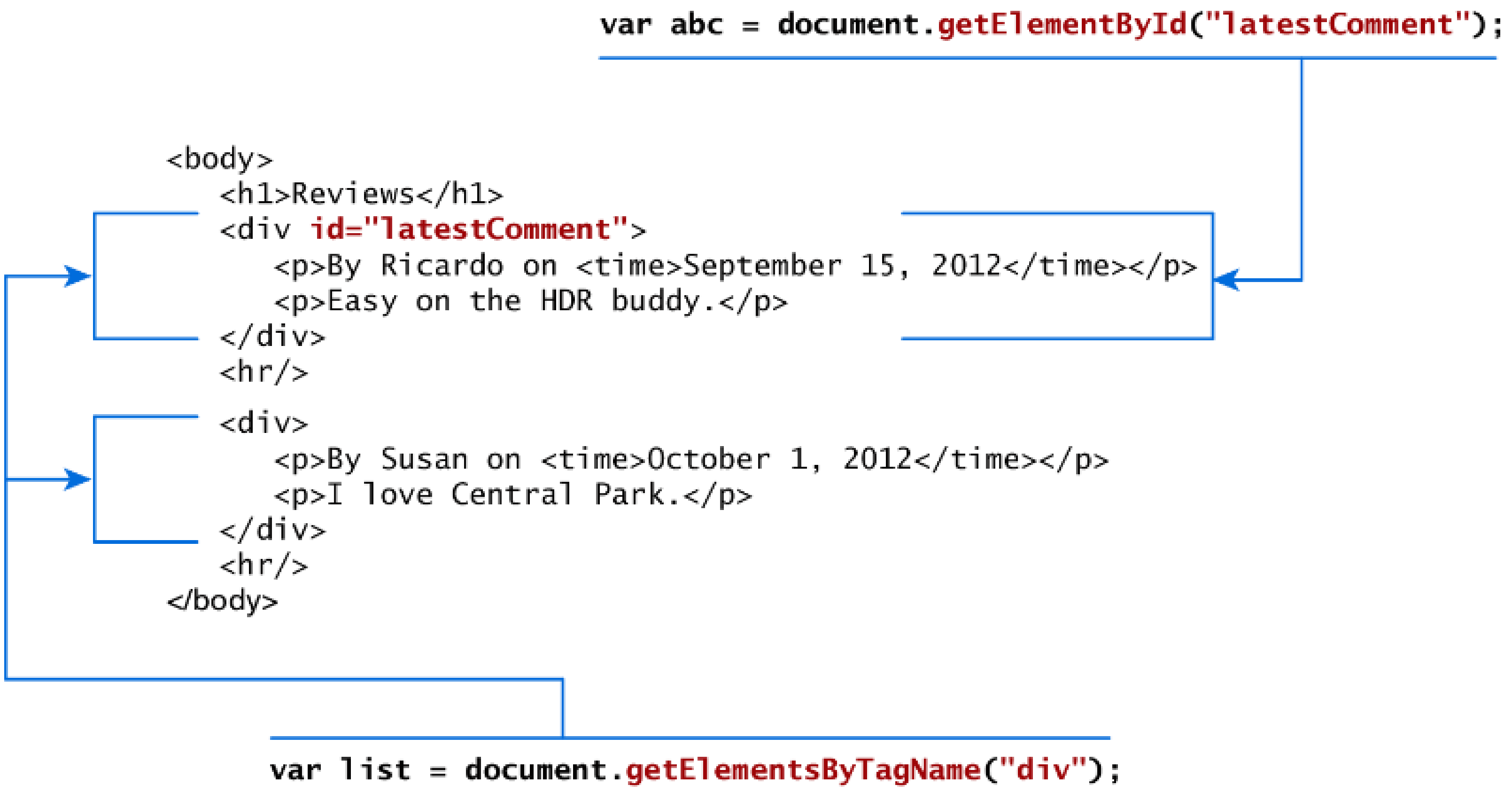
Property	Description
attributes	Collection of node attributes
childNodes	A NodeList of child nodes for this node
firstChild	First child node of this node
lastChild	Last child of this node
nextSibling	Next sibling node for this node
nodeName	Name of the node
nodeType	Type of the node
nodeValue	Value of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

Document Object

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

https://www.w3schools.com/js/js_htmldom_document.asp

Accessing Nodes – Selection Methods



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

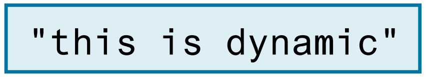
```
<body>
   <h1>Reviews</h1>
   <div id="latestComment">
      <p>By Ricardo on <time>September 15, 2012</time></p>
      <p>Easy on the HDR buddy.</p>
   </div>
   <hr/>

   <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");
```

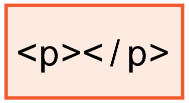
Modifying the DOM

1 Create a new text node



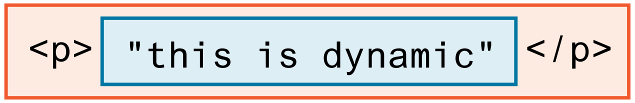
```
var text = document.createTextNode("this is dynamic");
```

2 Create a new empty <p> element



```
var p = document.createElement("p");
```

3 Add the text node to new <p> element



```
p.appendChild(text);
```

4 Add the <p> element to the <div>

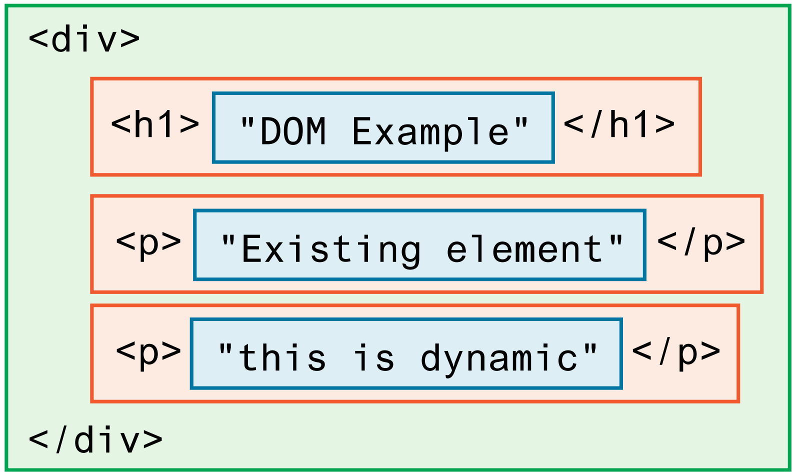
```
var first = document.getElementById("first");
first.appendChild(p);
```

Modifying the DOM

4 Add the <p> element to the <div>

```
var first = document.getElementById("first");
first.appendChild(p);
```

```
<div id="first">
    <h1>DOM Example</h1>
    <p>Existing element</p>
    <p>this is dynamic</p>
</div>
```



Modifying Element’s Style

```
var commentTag = document.getElementById("specificTag");
commentTag.style.backgroundColour = "#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
    - More about functions, objects, variable scopes, passing function as parameter
  - Windows and DOM object
  - Event model

Events

- HTML events are “things” that happen to HTML elements
- When JavaScript is used in HTML pages, it can “react” on these events
- An HTML event can be something the browser or a user does:
  - 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 – Listener Approach

```
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);
```

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

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");
  }
}
```

References

- Randy Connolly, Ricardo Hoar, Fundamentals of Web Development, Global Edition, Pearson

- W3Schools, HTML Tutorial [https://www.w3schools.com/html/default.asp]

- W3Schools, JavaScript tutorial [https://www.w3schools.com/js/default.asp]

W3 Tutorial: HTML and JavaScript

W4 Lecture: JavaScript and Browser Rendering Process

Assignment 1 - released (in W2)