

Web Application Development

COMP4347

COMP5347

HTML and Client-Side
JavaScript

Week 3

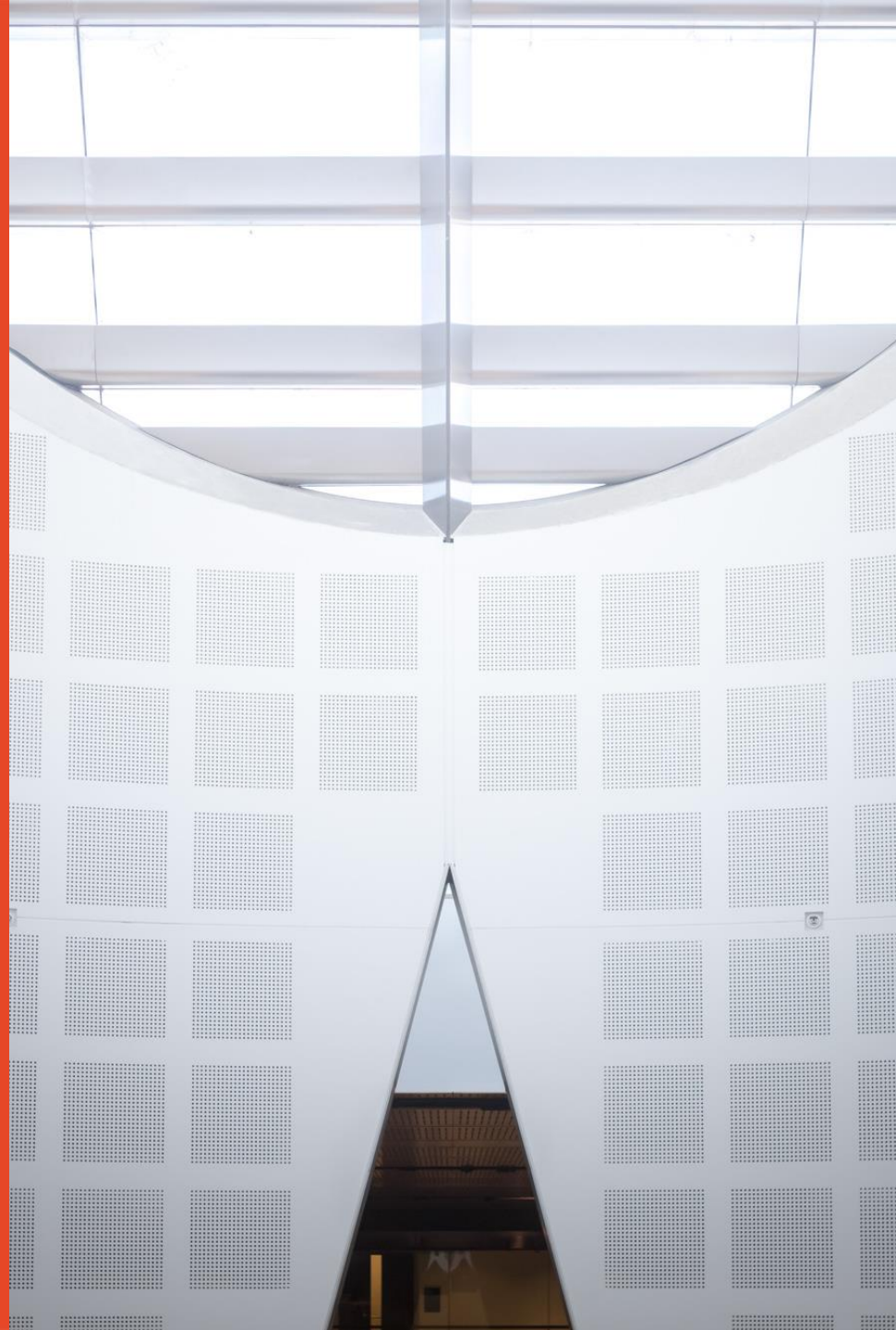
Semester 1, 2025

Dr. Mehdi Nobakht

School of Computer Science



THE UNIVERSITY OF
SYDNEY



**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.

Announcements

- Assignment 1 was released
 - Layout in mock-up as a reference
 - Use of genAI is not recommended for assignments

Outline

- **More HTML**
 - **Table**
 - **Elements**
 - **Styling**
 - **Form**
 - **Controls**
- **JavaScript**
 - **Syntax**
 - **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

The image displays four browser windows, each showcasing a different HTML table structure:

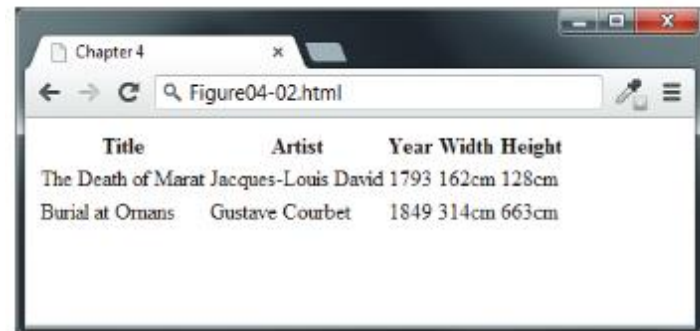
- Chapter 4 (Pricing Table):** A table with 4 columns: Feature, Free, Basic, and Premium. It lists various features like Upload Space, Daily Uploads, Total Uploads, Social Sharing, Analytics, and Price per year.
- Chapter 4 (Artist Inventory):** A table with 5 columns: Artist, Title, Year, and Home. It lists details for Jacques-Louis David's works, including 'The Death of Marat' and 'The Intervention of the Sabine Women'.
- Chapter 4 (Paintings List):** A table with 5 columns: Title, Artist, Year, Genre, and an Edit button. It lists various paintings such as 'Death of Marat', 'Lictors Bearing to Brutus the Bodies of his Sons', 'Liberty Leading the People', 'Arrangement in Grey and Black', and 'Mademoiselle Caroline Riviere'.
- Chapter 4 (Calendar):** A calendar for October 2014, showing days of the week and dates.

Basic Table Example

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

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>
```



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 colspan=2>	
<tr>		The Death of Marat	Jacques-Louis David	1793	162cm	128cm
<tr>		Burial at Ornans	Gustave Courbet	1849	314cm	663cm

```
<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>
```

Notice that this row now only has four cell elements.

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

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 Elements

<caption>

A title for the table is good for accessibility.

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

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

<col>

<colgroup>

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

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



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

<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>
```

Tables – Layout

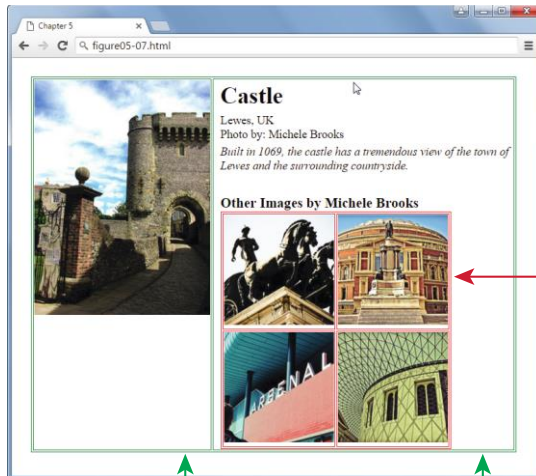
NOT recommended in practice



```
<table>
<tr>
  <td>
    
  </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>
  </td>
</tr>
</table>
```

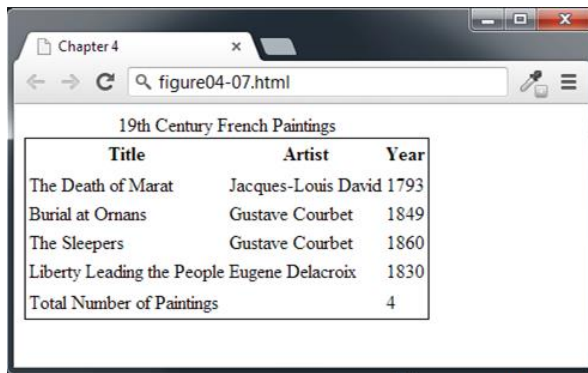
<h3>Other Images by Michele Brooks</h3>

```
<table>
<tr>
  <td></td>
  <td></td>
</tr>
<tr>
  <td></td>
  <td></td>
</tr>
</table>
```



Styling Tables

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

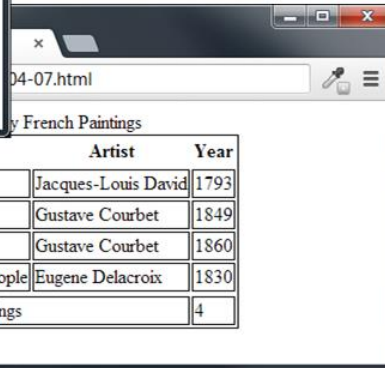


Chapter 4 x
figure04-07.html

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;  
}
```

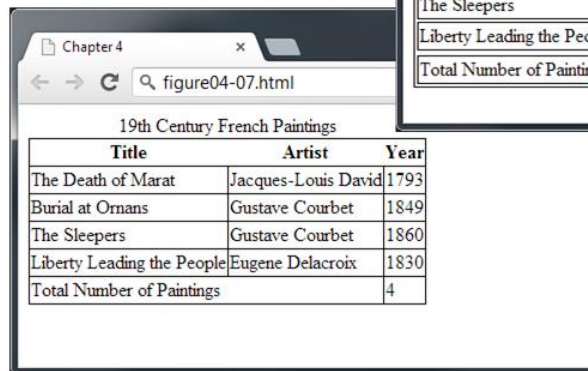


Chapter 4 x
figure04-07.html

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;  
}
```



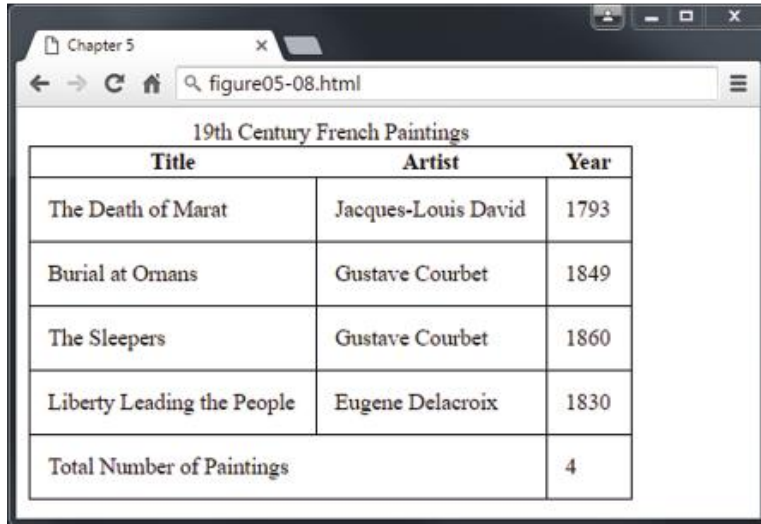
Chapter 4 x
figure04-07.html

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;  
}
```

Styling Tables



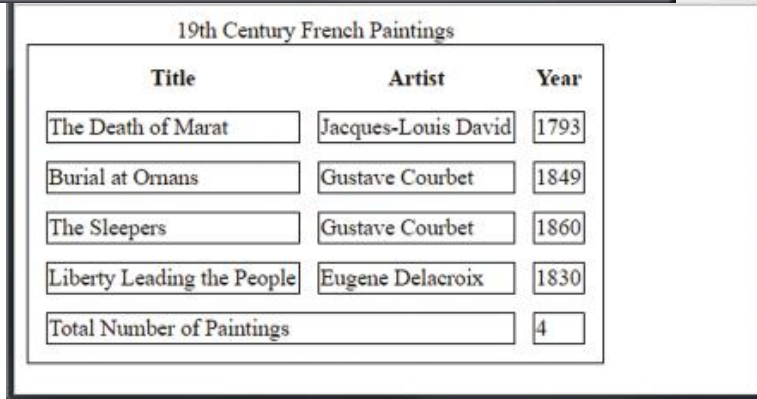
Chapter 5 x

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;  
}
```

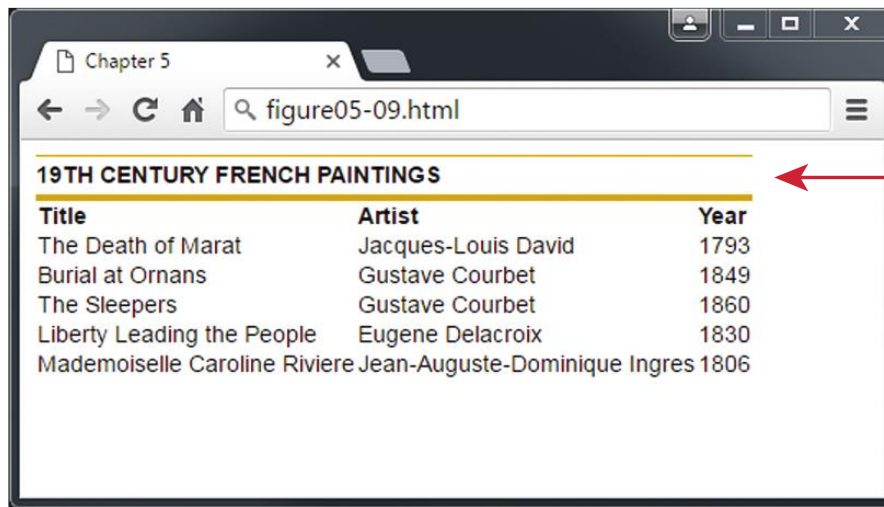


19th Century French Paintings

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



Chapter 5

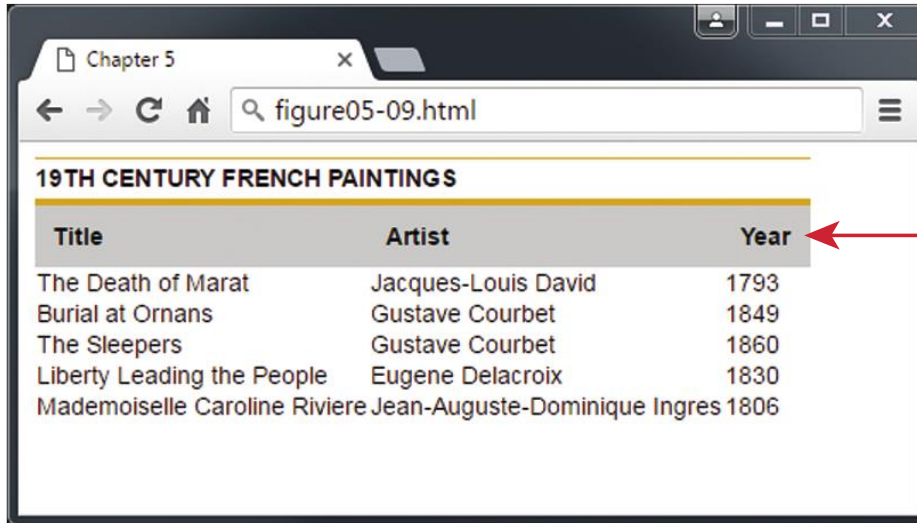
figure05-09.html

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
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



Chapter 5

figure05-09.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
Mademoiselle Caroline Riviere	Jean-Auguste-Dominique Ingres	1806

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

Styling Tables



Chapter 5

figure05-09.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
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

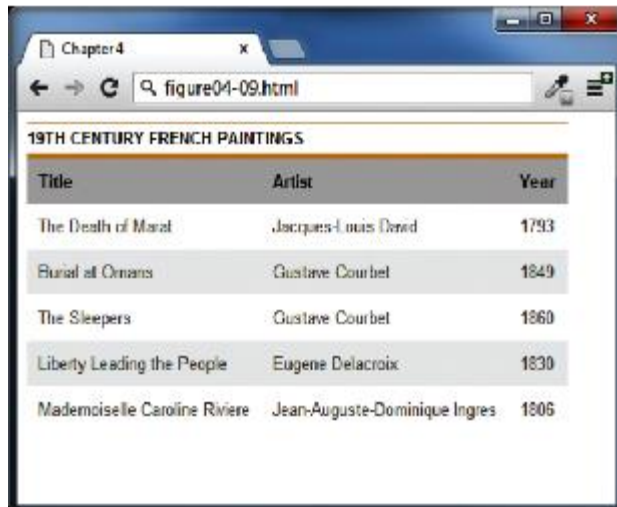


A screenshot of a web browser window titled 'Chapter4' showing a table of 19th-century French paintings. The table has three columns: Title, Artist, and Year. The row for 'The Sleepers' by Gustave Courbet is highlighted with a dark background, indicating a hover effect.

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;  
}
```



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 a light gray background and a white background, creating a zebra-stripe effect.

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

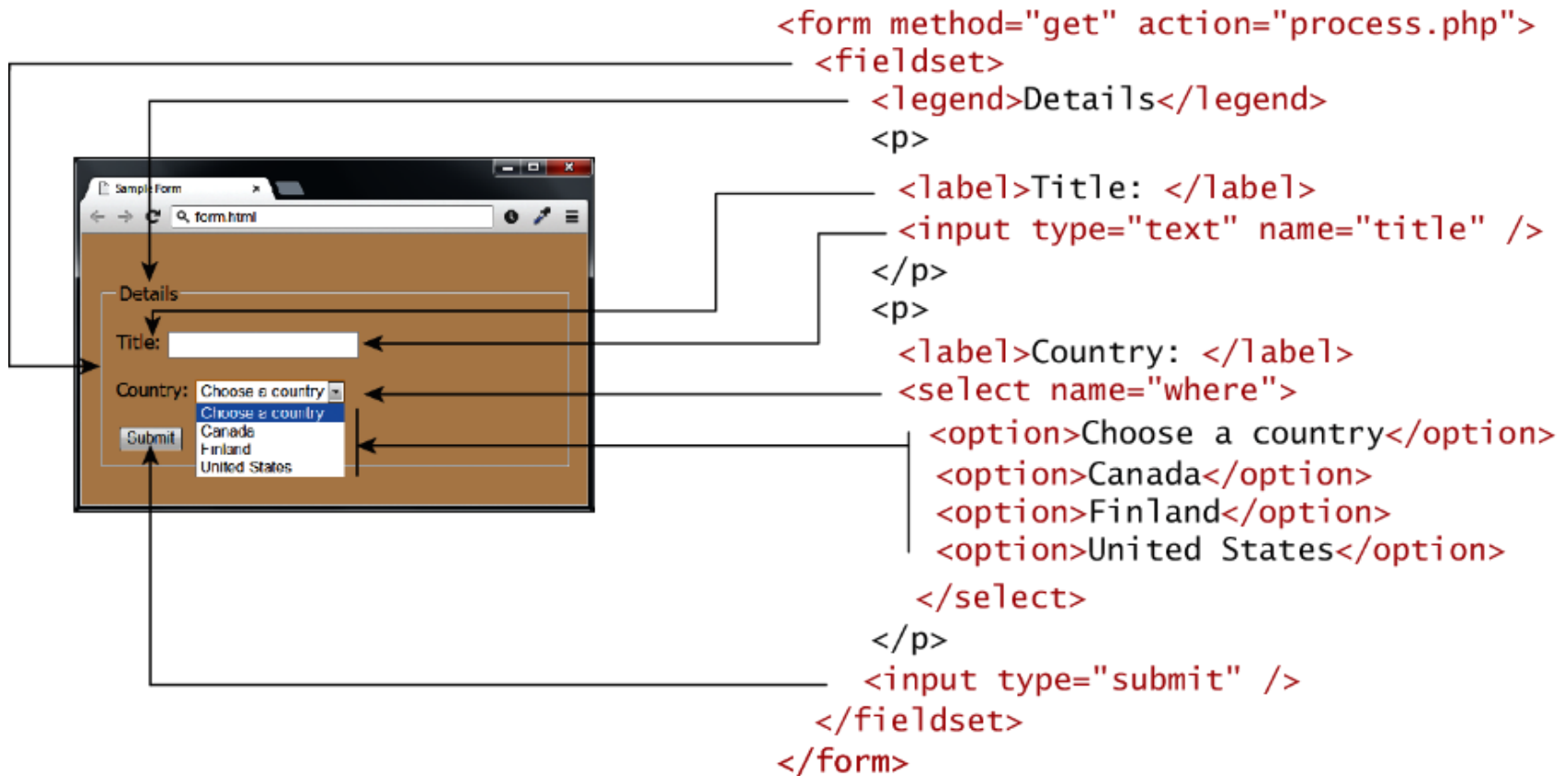
- **More HTML**
 - Table
 - Elements
 - Styling
 - **Form**
 - **Controls**
- **JavaScript**
 - Syntax
 - More about functions, objects, variable scopes, passing function as parameter
 - Windows and DOM object
 - Event model

HTML Forms

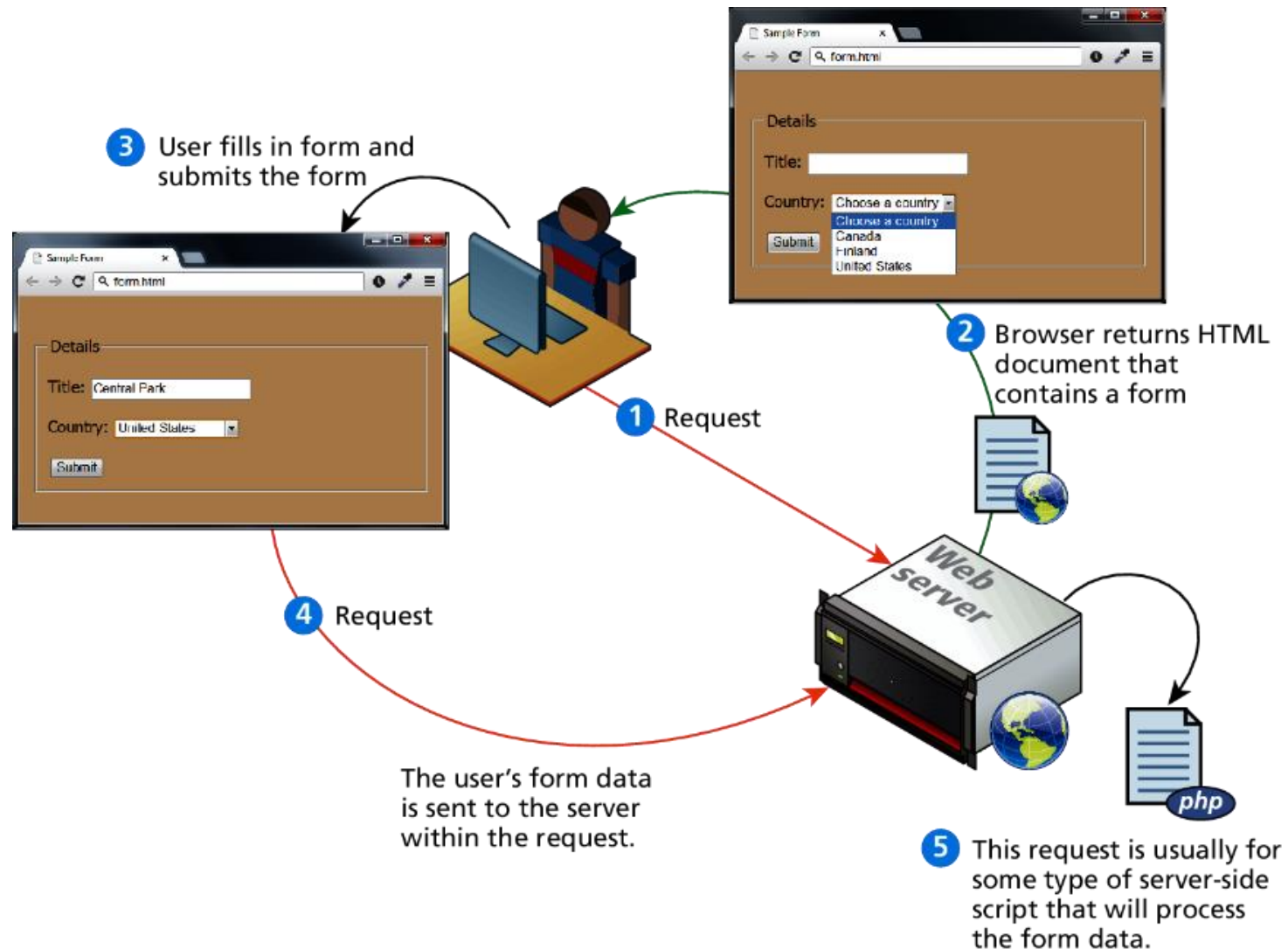
- Forms provide a way for users to interact with a web server
- Forms contain elements similar to desktop GUI
 - Plain text or password input
 - Selection
 - Radio and check boxes
 - Buttons

Form Structure

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



How Forms Work?



Form-Related HTML Elements

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

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" ... />
```

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:

Choice Controls

- Select Lists
 - Datalist element

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>
```

Choice Controls

– Select Lists

Select:

Select:
First
Second
Third

Cities:
North America
Calgary
Los Angeles
Europe
London
Paris
Prague

```
<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>
```

Choice Controls

– Radio Buttons

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
```

– Checkboxes

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
```

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

Button Controls

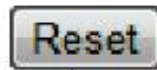
Type	Description
<code><input type="submit"></code>	Creates a button that submits the form data to the server.
<code><input type="reset"></code>	Creates a button that clears any of the user's already entered form data.
<code><input type="button"></code>	Creates a custom button. This button may require 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>

Button Controls – Example

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



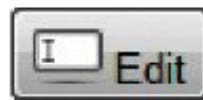
```
<input type="reset" />
```



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



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



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

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

Form Control Elements – Number and Ranges

Rate this photo:

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

Grumpy ————— Ecstatic

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

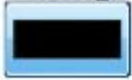
Rate this photo:

Grumpy ————— Ecstatic

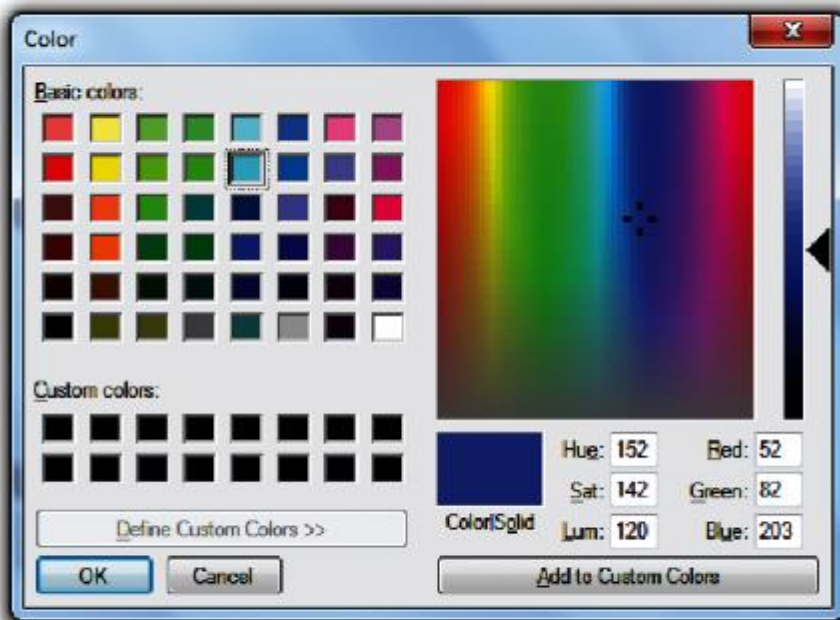
Controls as they appear in browser
that doesn't support these input types

Form Control Elements – 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

Form Control Elements – Date and Time

Date:

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:

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

DateTime:

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

DateTime Local:

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

Form Control Elements – File Upload

Upload a travel photo
 No file chosen



Upload a travel photo
 IMG_0020.JPG

```
<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

- Syntax

- More about functions, objects, variable scopes, passing function as parameter

- Windows and DOM object

- Event model

JavaScript

- Object-based, dynamically typed scripting language
 - Client-side scripting language for HTML and CSS
 - Has 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
 - Considered as not properly defined and targeting amateurs
- Integral part of web development in the mid 2000s with **AJAX**
 - Initial effort from Microsoft late 1999, get adopted by other browsers
 - Made very popular by Gmail and Google maps
 - Receives a lot more professional programming attention
- Many JavaScript frameworks have since created: 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

- Use **let** and **const** to declare variables and constant – good recommended programming practice

Program Control

– If ... else

```
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";
```

<u>Condition</u>	<u>Value</u>	<u>Value</u>
	if true	if false

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

Program Control

- switch ... case

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

Program Control

– for loop

initialization condition post-loop operation

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

– While

```
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/reference types
 - Object (reference types)
 - Array
 - Function

Primitive Types vs. Reference Types

- What is the difference between primitive types and reference types in JavaScript?
- Given below JavaScript snippet:

```
var abc = 27;  
var def = "hello";  
var foo = [45, 35, 25]  
var xyz = def;  
var bar = foo;  
bar[0] = 200;
```

What will be the value of `foo[0]` after running the script?

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)

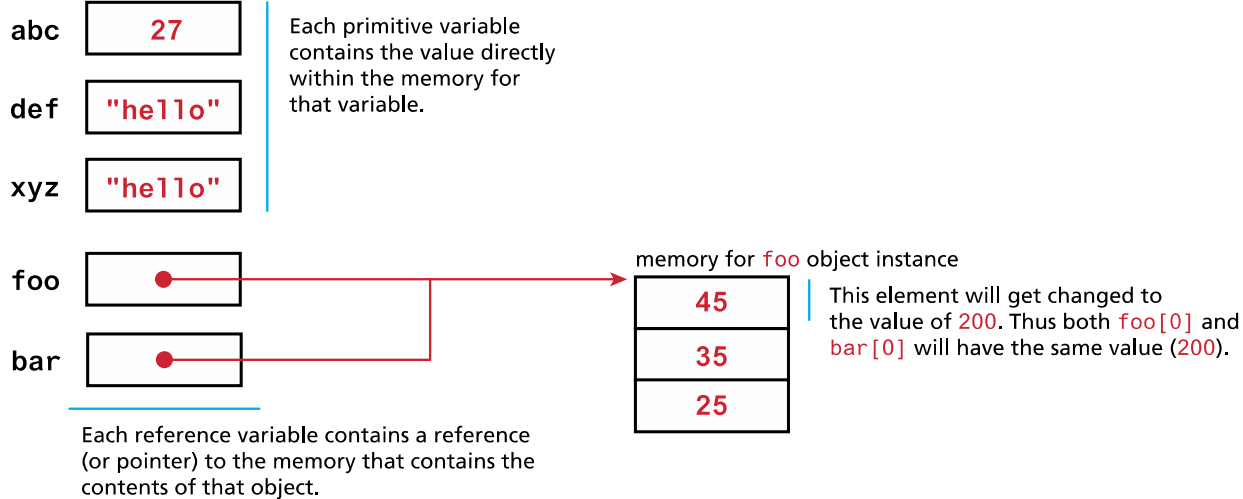
```
var xyz = def;  
var bar = foo;
```

these new variables differ in important ways
(see below)

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

changes value of the first element of array

Memory representation



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
 - Objects can have:
 - "constructor": use the "new" keyword and class name
 - 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
 - Defined by using the reserved word **function** followed by function name and (optional) parameters

Example:

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

- Call/invoke function:

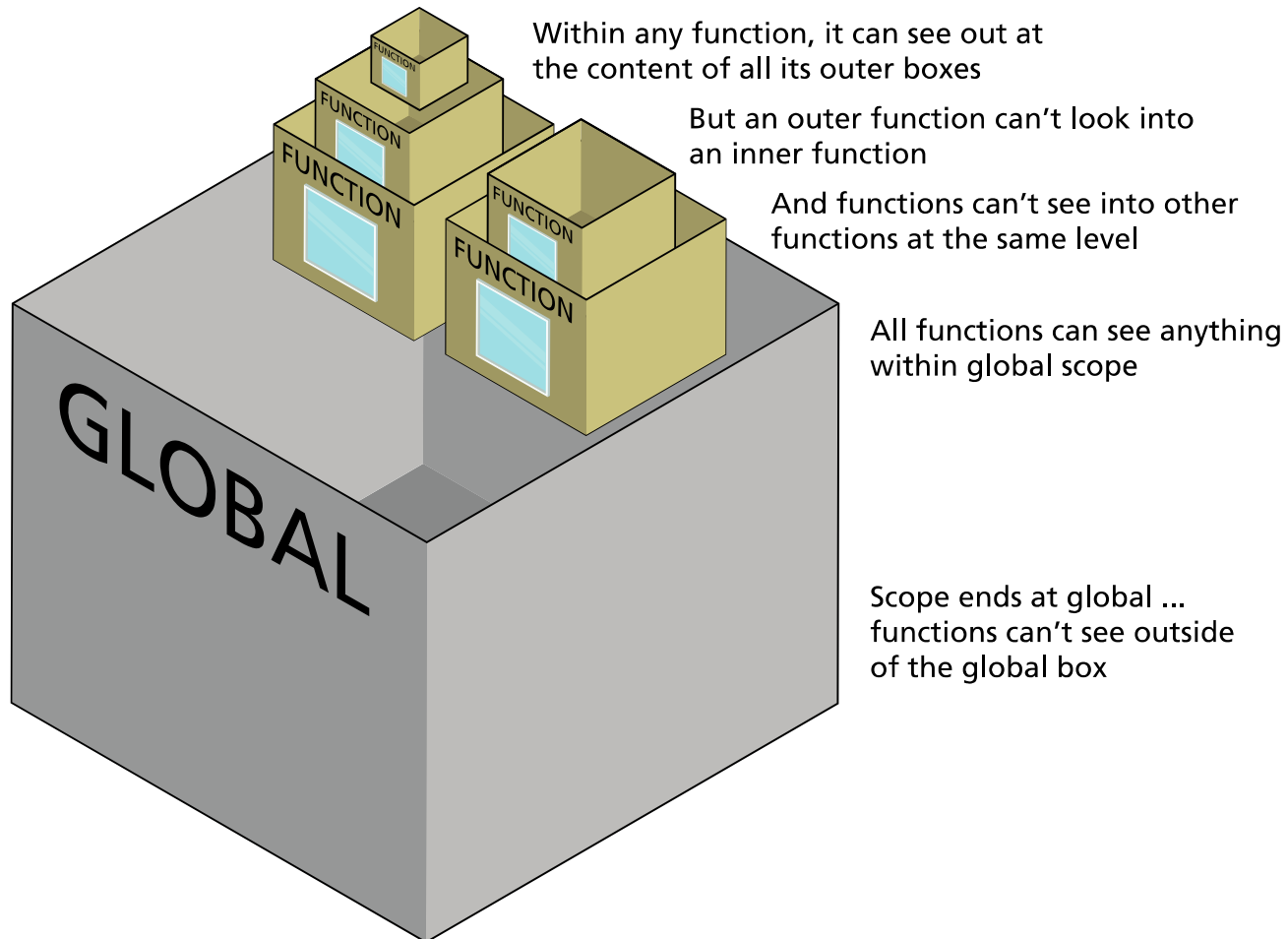
```
var result = subtotal(10,2);
```

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 webpage can access it.
- Variables declared inside a function has local scope
 - They can only be accessed within in the function

JavaScript – Variable Scope

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



Variable Scope - Examples

global variable **c** is defined
global function `outer()` is called

local (outer) variable **a** is accessed
local (inner) variable **b** is defined
global variable **c** is changed

local (outer) variable **a** is defined
local function `inner()` is called
global variable **c** is accessed
undefined variable **b** is accessed

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

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

Anything declared inside this block is accessible everywhere within this block

```
function outer() {
```

Anything declared inside this block is accessible only in this block

```
    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);  
}
```

✓ allowed

outputs 5

✓ allowed

✓ allowed

outputs 37

✗ not allowed

generates error or
outputs undefined

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

```
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;  
}
```

```
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);  
};
```

2

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

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

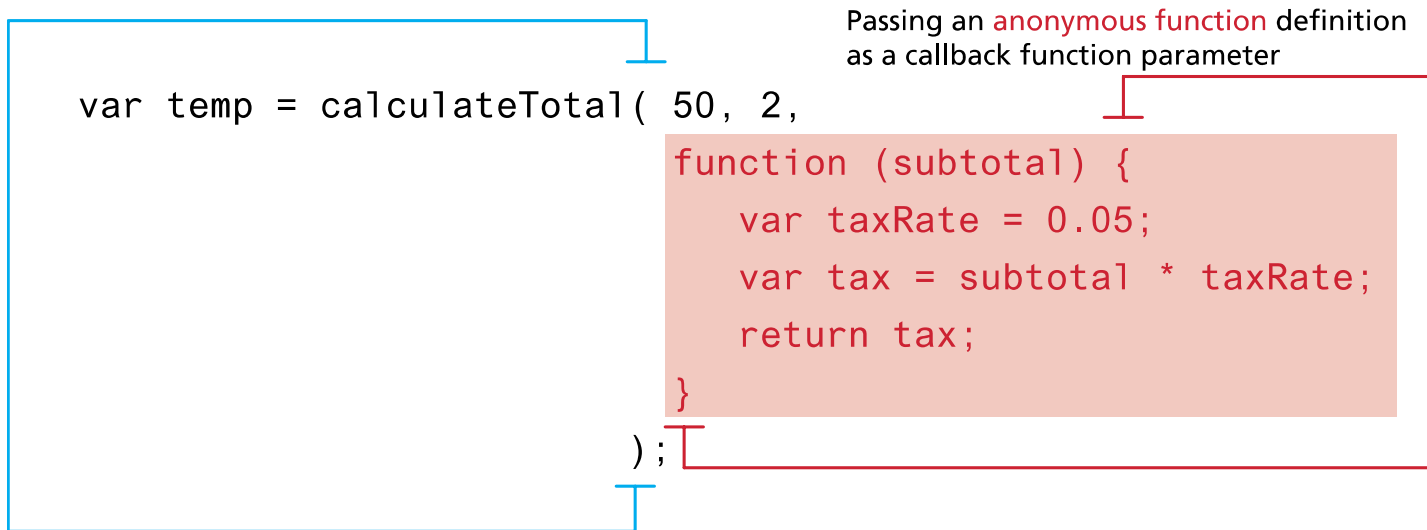
1

Passing the `calcTax()` function object as a parameter

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

We can say that `calcTax` variable here is a **callback function**

JavaScript – Callback Anonymous Function



JavaScript Output

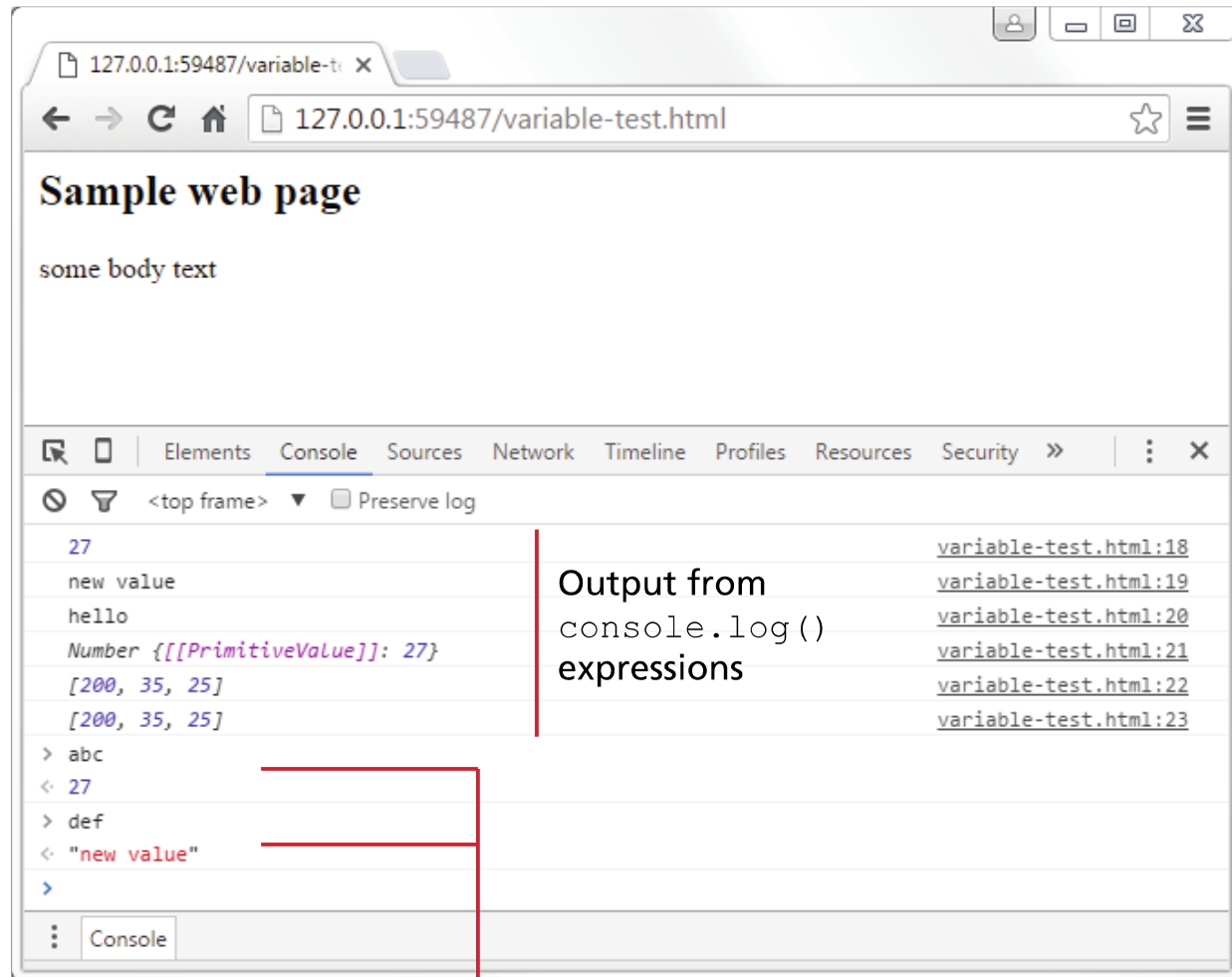
- **alert()** displays content within a pop-up box
 - `alert("Hello world");`
- **console.log()** displays content in the Browser's JavaScript console
- **document.write()** outputs the content (as mark-up) directly to the HTML document

```
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



Output from
`console.log()`
expressions

Using console interactively to query
value of JavaScript variables

Outline

- **More HTML**
 - **Table**
 - Elements
 - Styling
 - **Form**
 - Controls
- **JavaScript**
 - **Syntax**
 - 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

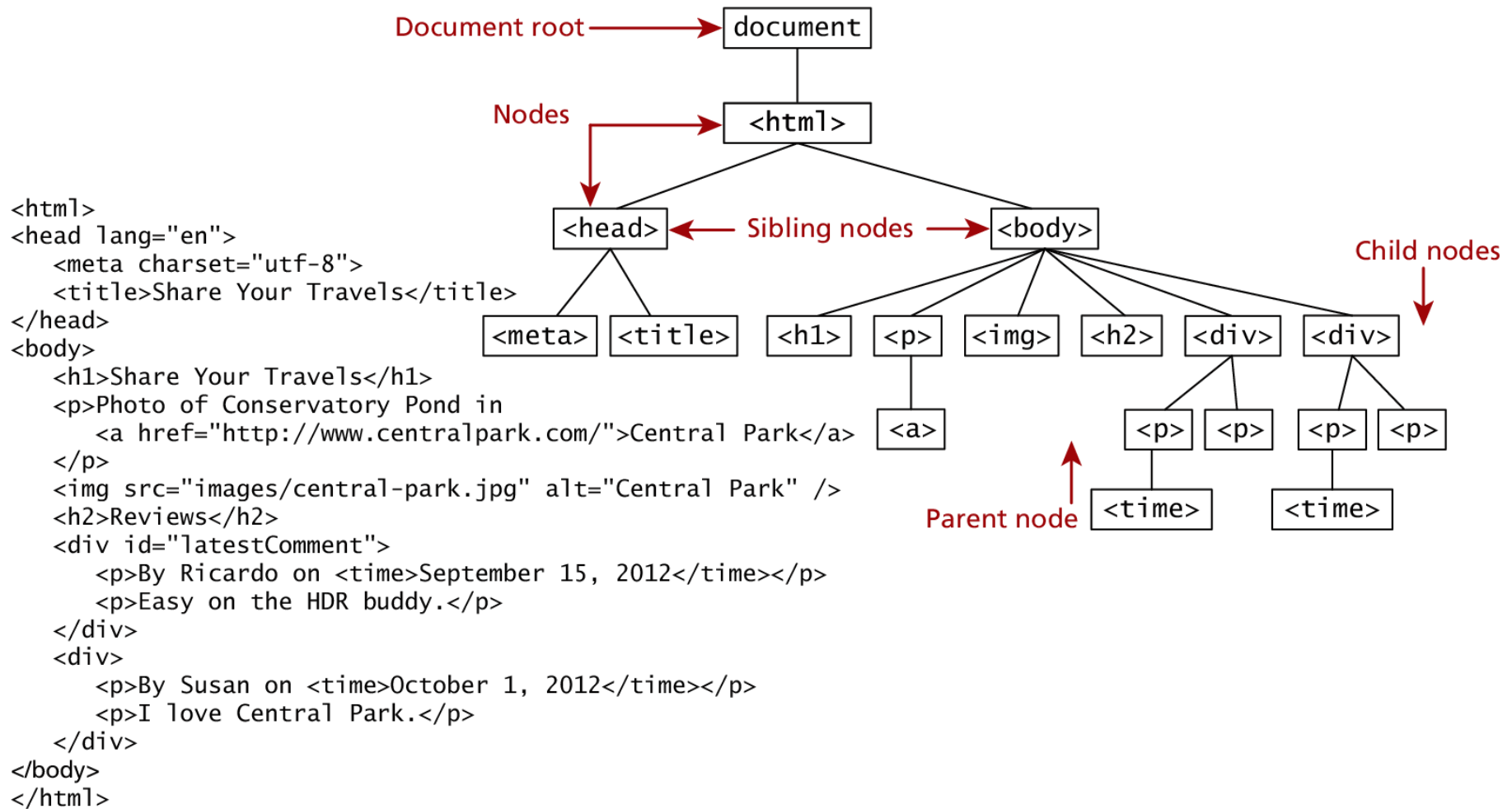
DOM Basics

- 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

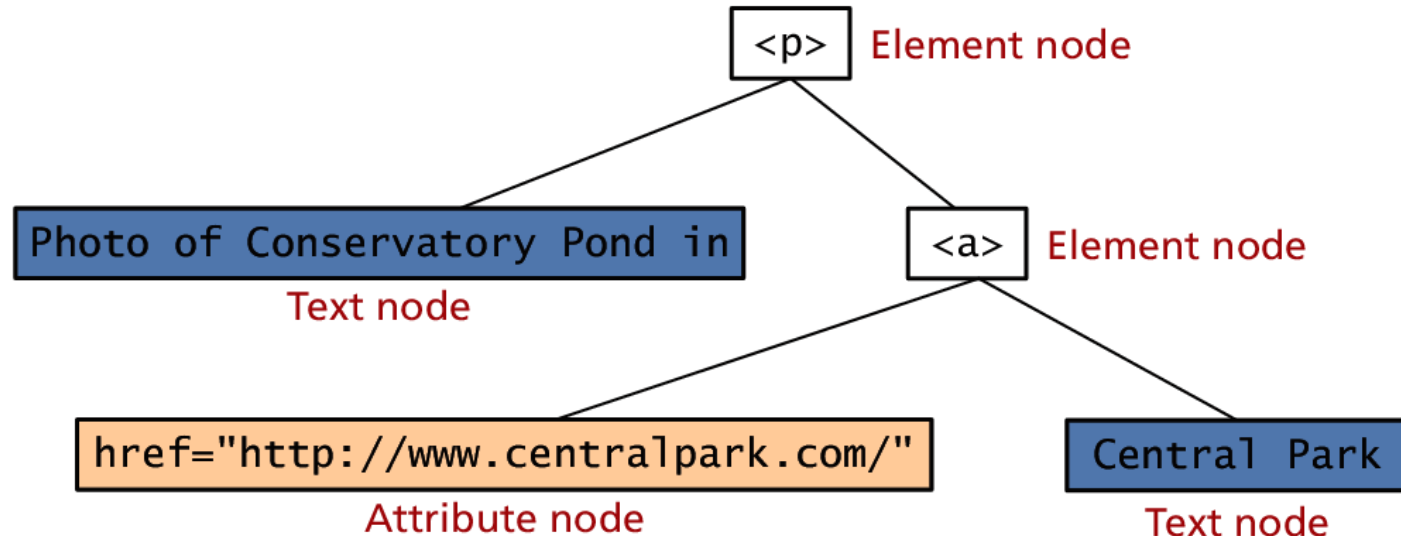
- The nodes in a document make up the page's DOM tree
- Nodes have *child-parent* relationships
- A node may have *multiple children*, but only *one parent*
- Nodes with the same parent node are referred to as *siblings*
- The document node has no parent and is called the root node

The DOM



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_html_dom_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>
```

The diagram illustrates the selection of the first `div` element with the ID `latestComment`. A blue line connects the `getElementById` method in the code above to the first `div` element in the HTML structure. Another blue line connects the `getElementsByName` method in the code below to the `div` elements in the HTML structure.

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


Modifying the DOM

- 1 Create a new text node

"this is dynamic"

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

- 2 Create a new empty <p> element

<p></p>

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

- 3 Add the text node to new <p> element

<p> "this is dynamic" </p>

```
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>
```

```
<div>  
  <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.backgroundColor = "#FFFF00";  
commentTag.style.borderWidth="3px";
```

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

Outline

- **More HTML**
 - Table
 - Elements
 - Styling
 - Form
 - Controls
- **JavaScript**
 - Syntax
 - 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

