INFO 151
Web Systems and Services

Week #9 Exam Revision (1)

Dr Philip Moore

Dr Zhili Zhao

Overview

- The following slides show:
  - The subject areas we have covered in Info 151 (Web Systems and Services)
  - The slides show the majority of the important topics introduced within each subject area
    - However: there are topics which are not covered in these slides such as the JavaScript ES6 Language Specification
  - In is important to note that you must refer to the weekly lecture slides for the full details on the subject areas and topics we have covered in this course

Course Overview

Weeks 1 – 3

- Introduction to Web Systems and Services
- Creating Web-Pages and Web-Sites with a Markup Language
- Introductory HTML 4 and HTML 5 with CSS

Weeks 4 – 6

- Client-Side Web Programming
- Object Oriented Programming
- Introductory and further JavaScript

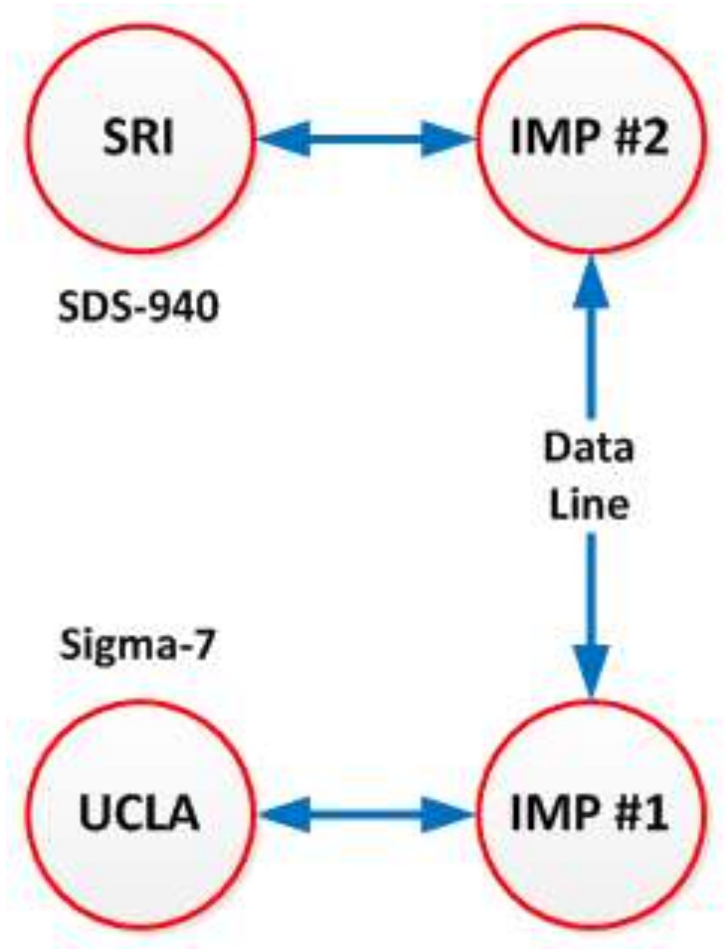
Weeks 7 – 9

- Server-side Programming
- Introductory PHP
- Introduction to Database, SQL, and MySQL

The Internet (or WWW)

Interface Message Processors (IMP)

- Shown in the figure is the layout of the first network between Stanford Research Institute (SRI) and the University of California (UCLA)
- The IMP No. 2 is an interface to the mainframe computer at SRI and the data from IMP No. 1 which is an interface to the mainframe computer at UCLA
- The IMP changes the data into a format the mainframe computer can access



Internet Technologies

The Internet – *TCP* and IP

- *Transport Control Protocol* (TCP) and the IP merged in 1978 to form the TCP/IP protocol
  - The *IP* suite is a set of rules and procedures (generally referred to as TCP/IP)
- The *TCP*
  - defines how applications create channels of communication across a network
  - manages how messages are assembled into smaller *packets* before they are transmitted over the internet and reassembled (in the right order) at the destination address.
- The *IP*
  - defines how to *address* and *route* each packet to reach the right destination
  - Each *gateway* computer (on a network) checks this *IP* address to determine the destination address

Uniform Resource Locator (URL)

- A feature of *Internet* (and *intranet*) is the address used to identify the location of information and resources on the www.
  - The address is termed the ***Uniform Resource Locator*** (***url***)
  - Requests are made to servers from web-browsers using the url for the resource required
- An example or a url is:
  - https://www.quanzhanketang.com/
  - All url addresses follow these syntax rules without spaces

The Internet – HTTP

- In the early 1990's the *Hypertext Transfer Protocol* (HTTP)
- HTTP is the language of the Internet and enables:
  - Information on the Internet to be accessed from *anywhere* by *anyone* (with a computer and Internet connection)
  - Users create linked web-pages using *hypertext* links
  - HTTP can be seen in web-page URL addresses:
    - https://en.wikipedia.org/wiki/ARPANET

HTTP *vs* HTTPS

- In the url:
  - https://www.quanzhanketang.com/ (links are often underlined)
  - Note: the https://...
- *HTTPS*
  - Is an extension of the HTTP
  - It provides secure communication over a computer network
  - It is widely used on the Internet.
  - The communication protocol is encrypted using *Transport Layer Security* (TLS) (or its predecessor *Secure Sockets Layer* (SSL)

The Internet and intranet

- In practice the *Internet* can be:
- The ***Internet*** (also known as the *World Wide Web* (www))
  - Is a system which allows linked documents (and parts of documents) to be connected using hypertext links.
- An ***intranet***
  - Is a local (or restricted) communications network, an example is a private network
  - A company intranet can provide a single starting point to access internal and external resources
- An ***intranet*** is established in *local-area-networks* (LAN) and *wide-area-networks* (WAN)

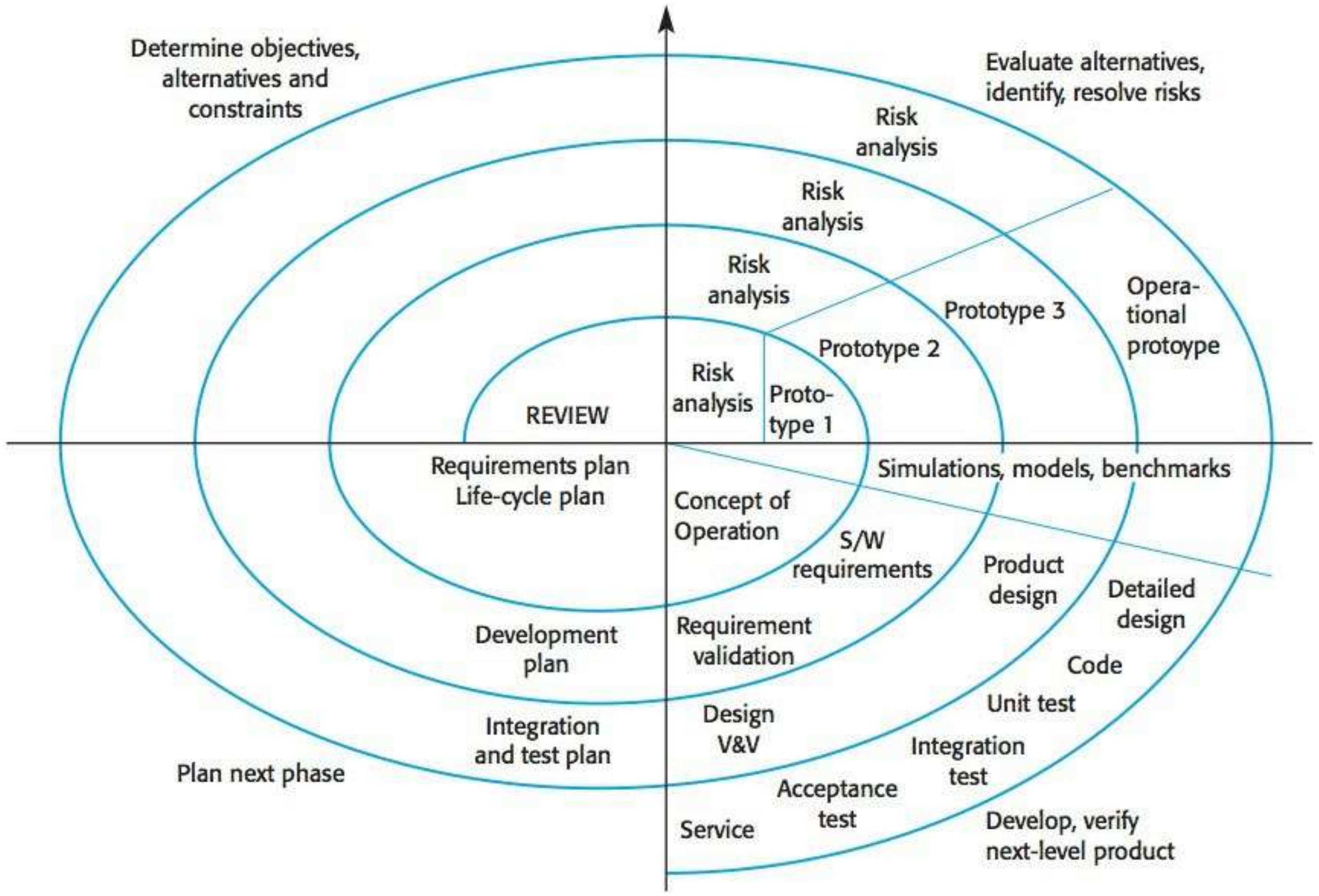
Web Systems

Web Systems

- Web systems architectural models
  - Are conceptual models that describe and define the configuration of a system addressing a systems:
    - Structure
    - Hardware and software
    - Behaviour
    - System design
  - A conceptual model is
    - A formal description and representation of a system
    - A model is organized in a way that supports reasoning about the structures and behaviours of a system

Web Systems Software Design

The Spiral Software Development Model

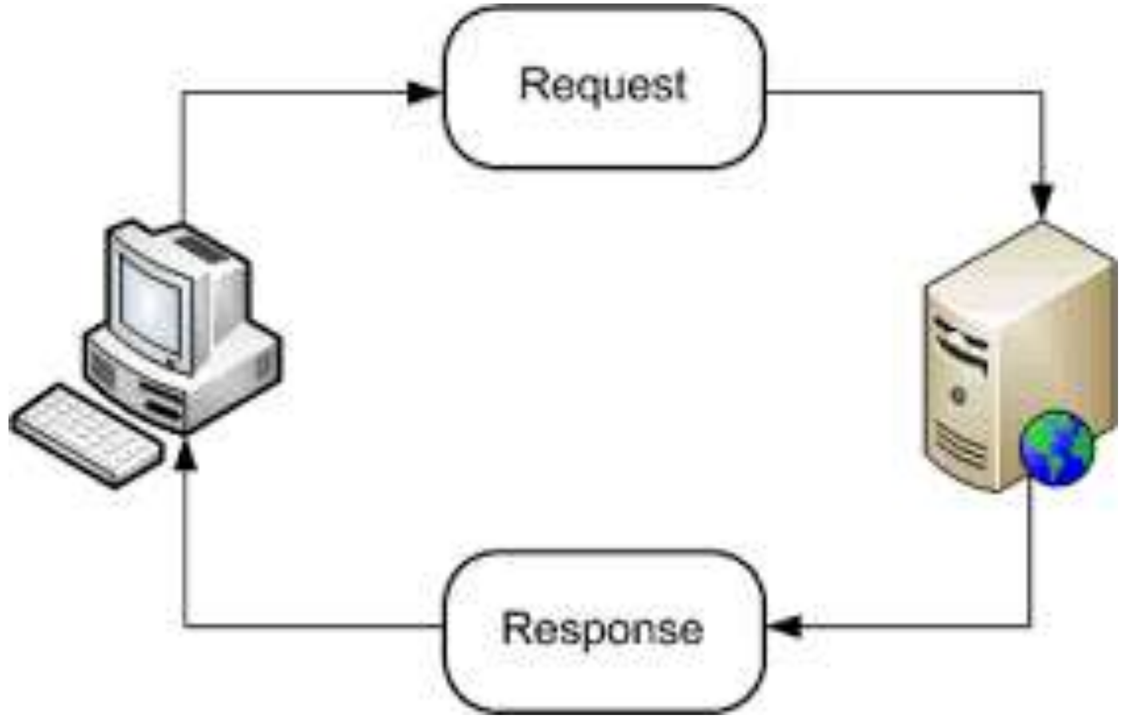


Web Systems Architectures

- There are three types of architecture found in web-systems
- ***two-tier*** *architectures*
- ***three-tier*** architectures
- ***three-tier*** architectural model for a database application
- We will briefly introduce the architectures with conceptual models showing their configuration

Web-Systems Request / Response Interactions

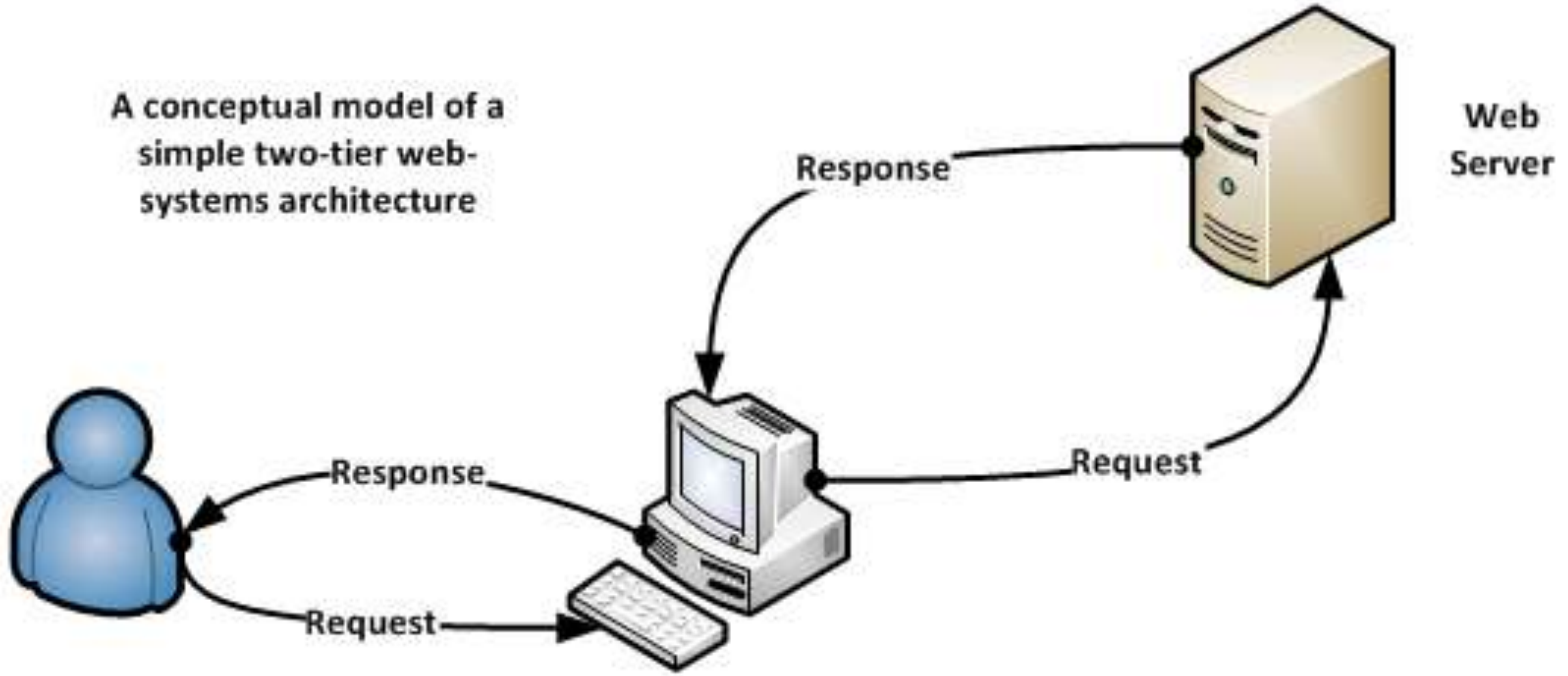
- When a web-page is opened
  - There are a series of interactive requests and responses
  - The interactions are between the client (a web-browser) and the web server that hosts the requested resources
  - Each request tells the server that the client wants a specific resource(s)
  - The response to the request delivers the resource content.
- The figure shows this interaction in a 2 layer web systems architecture



Two-Tier Architecture

- ***two-tier** architectures*
  - A client tier (client computer systems and web browsers)
  - A second tier where the web server is located
- Interactions
  - A user requests a web-page or resource
  - The local web-browser requests a resource from a web server
  - A web-server sends a response to the web browser
  - The web browser shows the web-page or processes the resource

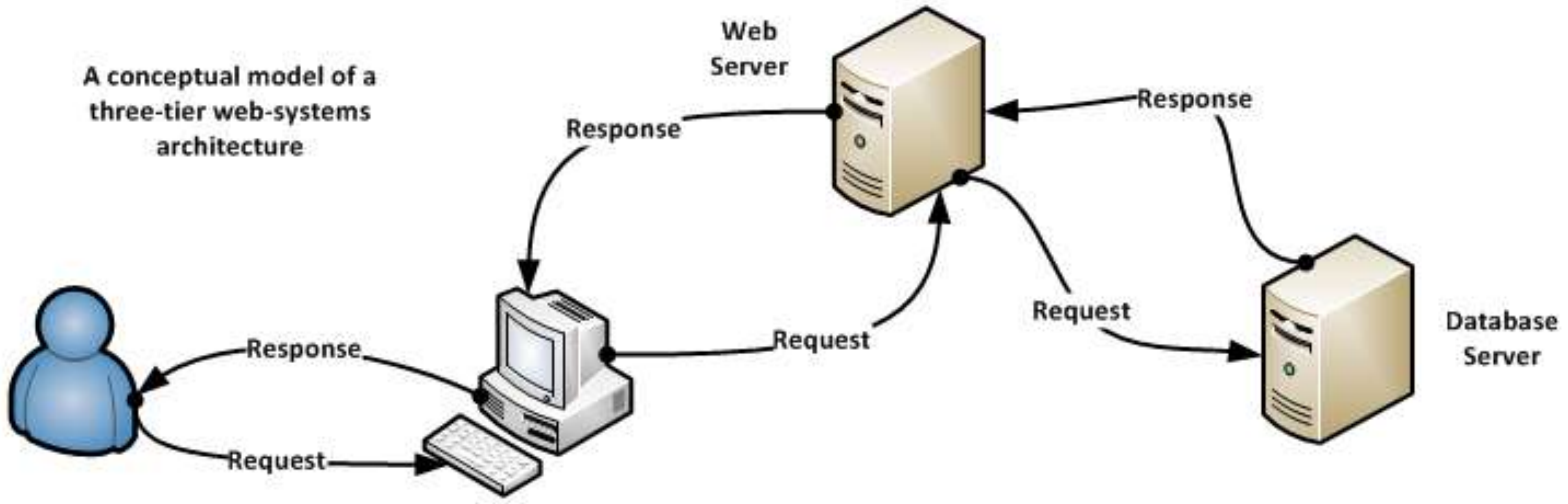
Two-Tier Web-Systems Architecture


A conceptual model of a simple two-tier web-systems architecture
Response
Web Server
Response
Request
Request

Three-Tier Architectures

- ***three-tier*** architectures
  - A client tier (client computer systems and web browsers)
  - A middle tier where the web server is located
  - A third tier where a database server is located
- Interactions
  - A user requests a web-page or resource
  - The local web-browser requests a resource from a web server
  - The web-server sends a request to the database server
  - The database server sends a response to the web-server
  - A web-server sends a response to the web browser
  - The web browser shows the web-page or processes the resource

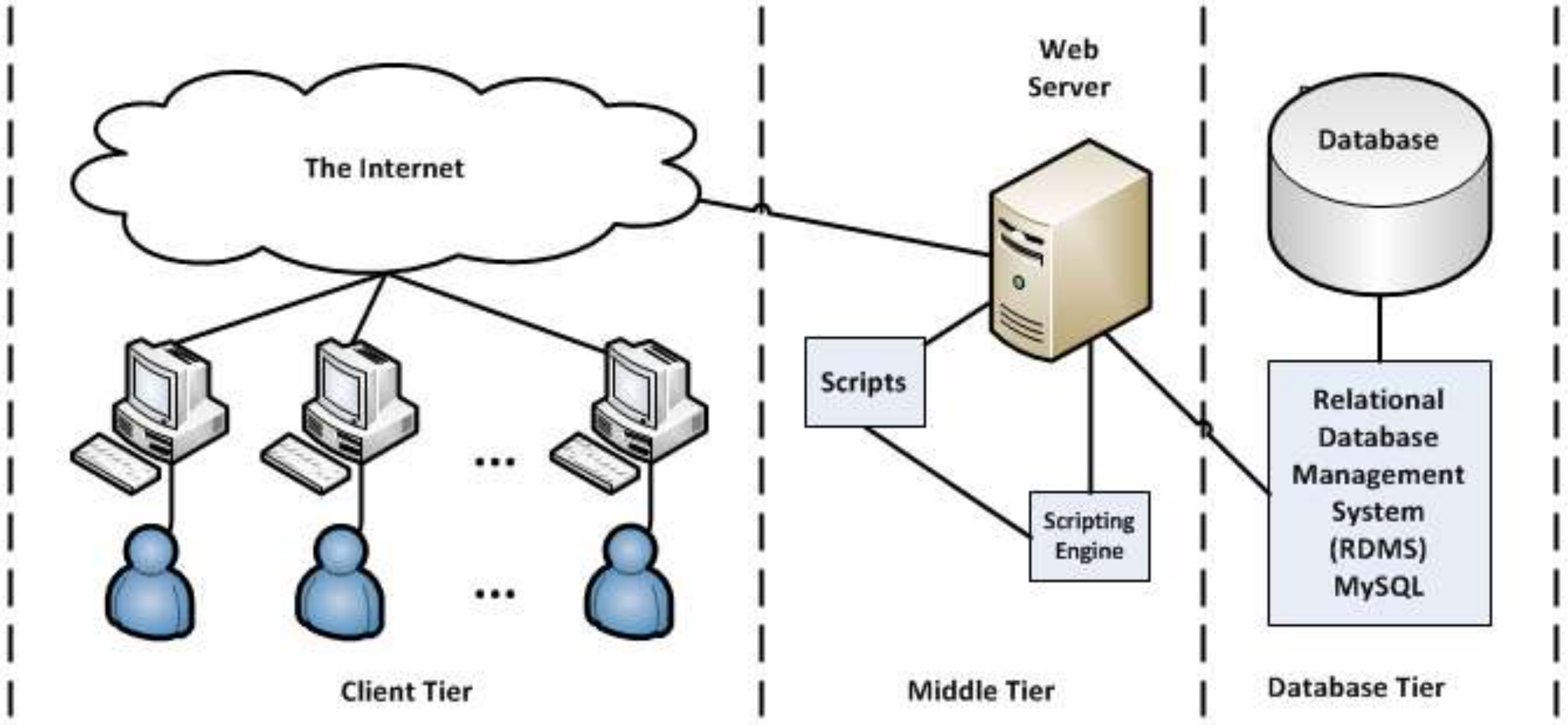
Three-Tier Web-Systems Architecture



Web Systems Database Applications

- In developing web-systems database applications
  - In the client tier
    - Clients computer systems
    - The web browsers
- The client tier connects to the middle tier
  - In the middle tier
    - The web server
    - The scripting engine
    - The scripts (PHP)
- The middle tier connects to the database server

Three-Tier database Application Architecture



A three-tier architectural model of a web-systems database application

Web System Architectures

- The three web-systems architectures are design models of typical architectures
- In *'real-world'* web systems
  - The configuration of physical servers will vary
  - There are physical servers in the second and third layers
  - There is a separate database server (Holding for example the MySQL server)
  - All the physical components are created within a single physical server using dedicated partitions (*virtual servers*)
- The design of a web system is based on
  - The size of a web-site measured in terms of the anticipated number of 'hits'
  - The database function will be designed based on the number of user records

Database Technologies

- There are a number of database technologies in *'real-world'* Internet systems
  - *Relational Database Management Systems* (RDMS) (*structured* data)
  - *NoSQL* database systems (*unstructured* data)
- From a web systems design perspective
  - A RDMS database is located and run from a single server (or *virtual server*)
  - A NoSQL database is generally implemented using *horizontal* and *vertical* scaling

Types of Web-Page

- There are two types of web-page:
  - A *static* web page
    - Termed ‘*stateless*’
  - A *dynamic* web page
    - Termed ‘*stateful*’
- A simple HTML web page is ‘*stateless*’
- A simple HTML web page when extended using JavaScript, PHP, and MySQL becomes a ‘*stateful*’ web page
- While a ‘*stateless*’ web page may use a *two-tier* architecture
- A ‘*stateful*’ web page requires a *three-tier* architecture

Introduction to Web-Systems Programming
网络系统编程导论
Introduction to HTML5.2
HTML5.2 简介

Computer Programming

- There are only three operations in any computer program
- ***Sequential*** operations:
  - Run one-after another without repetition
- ***Selection*** operations (conditional actions):
  - Select a particular course of action from a number of available alternatives
- ***Iteration*** operations (repeating actions):
  - Repeat until a condition is satisfied or termination criteria is reached
- A computer program mixes the three processes to achieve a desired result
- In this course we will learn how to use the three operations in JavaScript and PHP

Web Systems Programming

- When we consider web systems programming we will consider two approaches
  - ***Client-side*** programming
    - The program code is located and run within a web browser
    - The results are displayed by the web browser
    - A web-browser is termed a thin client
  - ***Server-side*** programming
    - The program code is located and run in a web server
    - The results are returned and displayed by the web browser
- *Client-side* programming uses HTML and JavaScript
- *Server-side* programming uses HTML and PHP
- HTML, JavaScript, and PHP may be used in a single program

Web Systems Programming

- JavaScript is a *client-side* programming language
- PHP is a *server-side* programming language
- JavaScript can not connect directly to MySQL server
  - There are exceptions which require *'bridging'* such as
    - Node.js (or)
    - RESTful application programming interface (API)
- To connect to the database server (and MySQL)
  - We must use PHP
- In this course we will connect to MySQL server using PHP

Documenting Program Code

- In preparing an HTML file with JavaScript, PHP, and MySQL
  - It is important that the program code is documented
- By documentation we refer to *inserting comments* in the code
- Proper documentation is important because:
  - Comments provide information to understand the purpose and reason for the lines in the program code
  - Understanding is important for multiple programmers
  - Understanding is also important to remind the programmer of program logic and purpose
- Proper documentation is very important in the software life cycle
  - To enable maintenance and updating of the software

The Internet
Basic HTML

Hypertext and HTML

- The term *Hypertext*
  - Originally referred to text stored in electronic form with cross-referenced links between pages.
  - It has developed into a broader term that refers to: objects (text, images, files, etc) that can be linked to other objects.
  - HTTP is a language for describing how text, graphics, and files containing other information are organized and linked.
- The *Hypertext Markup Language* (HTML) is a standard *'markup language'* for the creation of web pages and web-systems applications.
- When used with *Cascading Style Sheets* (CSS) and scripting languages it forms the main technologies of the *Internet*

HTML 5

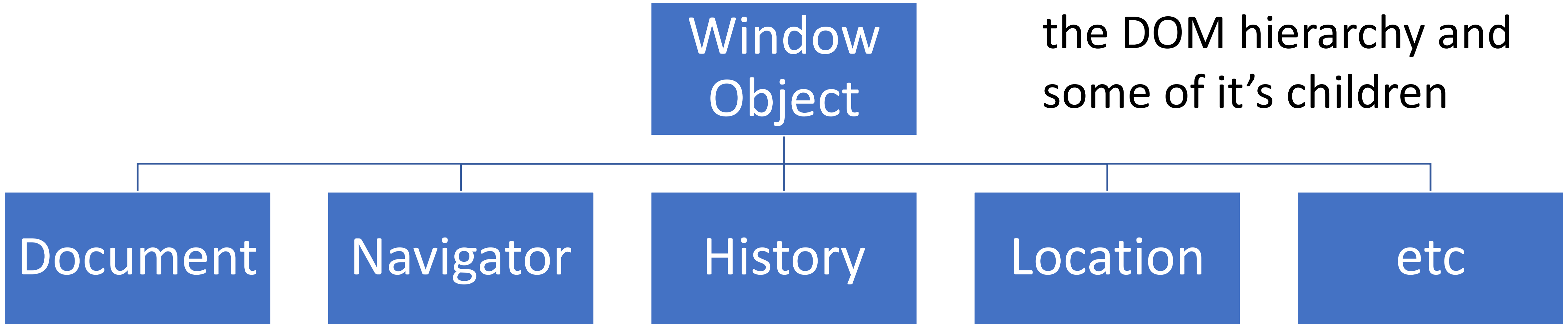
- HTML 5 is currently
- The latest HTML standard (2018) is HTML5.2
- HTML5.2 is generally supported by the major web-browsers
  - We will consider this later as it relates to web browsers and devices
- Improved integration of multimedia
- Easy to work with
  - No XML directive and lengthy DOCTYPE declaration
- Similar (nearly identical) to XHTML

```
<!DOCTYPE html>
<html>
…
</html
```

Document Object Model (DOM)

- The HTML DOM is a standard object model and programming interface for HTML. It defines:
  - The HTML elements as objects
  - The properties of all HTML elements
  - The methods to access all HTML elements
  - The events for all HTML elements
- In simple terms: the HTML DOM is a standard for how to get, change, add, or delete HTML elements
- We will introduce the HTML DOM Object in Week 3
- The DOM and JavaScript will be introduced in later weeks

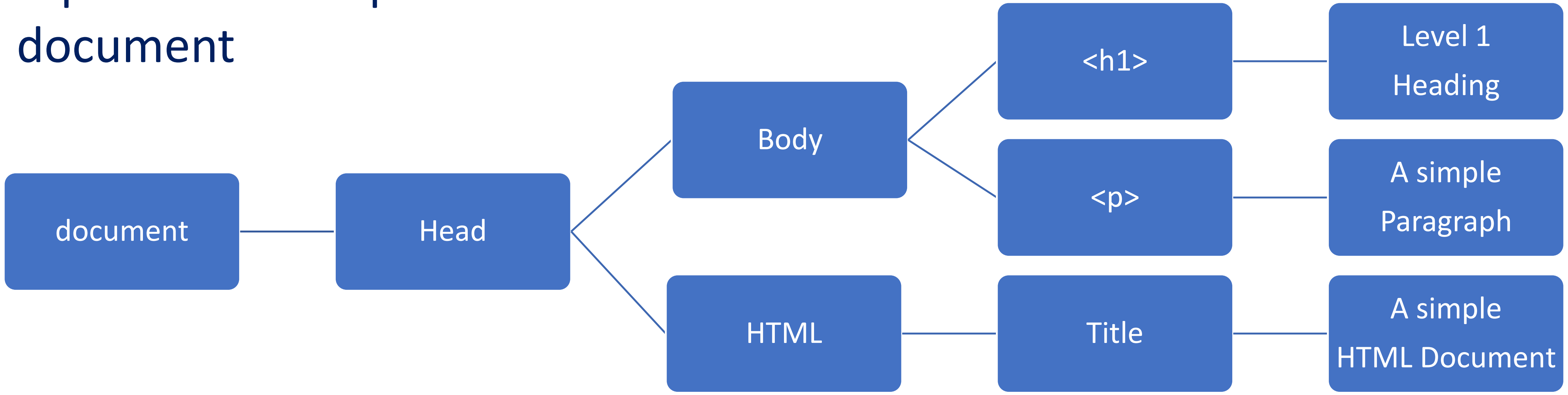
Document Object Model Hierarchy



The window section of the DOM hierarchy and some of it's children

Document Object Model Example

How the DOM hierarchy represents a simple HTML document



First Web Page

- Creating a Sample HTML File
- This simple HTML code prints
  - “My first web page!”
- You will learn about HTML
- You will learn how to add style to web-pages using
  - Cascading Style Sheets (CSS)

```
<!- this is a comment -->
<!DOCTYPE html>
<html>
<head>
<title>First Web Page</title>
</head>
<body>
<p>
My first web page!
</p>
</body>
</html>
```

Web Systems and Web-Browser Support

- Web-based systems must accommodate
  - A range of browsers
  - Implemented in a range of devices
  - A range of screen resolutions
- The support for HTML4 is achieved in all major browsers
- The support for HTML5.2 is not universal
  - Web-pages and web-sites must
    - Incorporate HTML that enables the web-pages web-sites to be viewed in a range of current and older browsers and devices

Basic functions in HTML

HTML Tags

```
<!– this is a comment -->
<!DOCTYPE html>
<html>
<head>
<title>First Web Page</title>
</head>
<body>
<p>
My first web page!
I want the following text on a new
line
</p>
</body>
</html>
```

```
<!– this is a comment -->
<!DOCTYPE html>
<html>
<head>
<title>Second Web Page</title>
</head>
<body>
<p>
This is my first line of text
<br>
This is my second line of text
</>
</p>
</body>
</html>
```

The <meta> Tag Elements

- In the `<meta>` tag
  - Located within the `<head>…</head>` tags
  - Information is specified describing web document
  - Provide information for search engine catalogs
- The <meta> tag provides information such as:

```
<meta name="keywords" content=”web page, design, education,..." />
<meta name="description" content=”This is the first web page of ..." />
```

HTML

Language Selection

- English is the language of HTML
- We may however change the language the text is presented in the web-page
- The language used in the web-page may be selected.
- For example:
  - The general tag for Chinese:
    - `<html lang=”zh-Hans"> … </html>`
  - The tag for Chinese (Simplified):
    - `<html lang=”zh-Hant"> … </html>`
  - The tag for Chinese (Traditional):
    - `<html lang=”zh"> … </html>`

Adding and Formatting Text

- Adding text is an essential part of the formatting of a web-page
- Text may be added using:
  - HTML
  - JavaScript
  - PHP
  - In this lecture we focus on HTML (JavaScript and PHP will be covered later)
- Inserting text is simply a matter of typing into the HTML file the required text
  - The language shown in the web-page is the text inserted
  - However, ‘white space’ is ignored in the processing of an HTML file
- We must insert tags to:
  - Place text on a new line and create paragraphs
  - Format text (*italic* or *emphasised* or **Bold** or in **colour**)

Adding Colour

- Adding colour uses a number of codes:
  - Hexadecimal code (#FFFFF0) (letters and numbers)
  - Red Green Blue (rgb) decimal scale – 0,0,0 (black) to 255,255,255 (white)
- HTML Cascading Style Sheets support 140 standard colours
- For a detailed description of colour information see:
  - https://www.colorcodehex.com/fffff0/
- For a detailed description of colour in HTML see:
  - https://www.quanzhanketang.com/html/html_colors.html
- The following slides show examples of HTML <tags> and colours

Changing a Font Colour

- Text is shown in the default black font
- You will want to change the colour in web pages
- There are two ways to do this:
  - Inline changes
  - Using Cascading Style Sheets (covered in another lecture)
- Inline changes are suitable for single web pages
- Cascading style sheets are suitable for web sites with multiple web pages or where a consistent style is required
- The following slides show how to change font colour using the inline method

Inline change for a Font Colour

- The structure of the HTML to use inline font colour:
  - <h3 style="insert the colour">Your text here</h3>
- There are 140 colours in HTML5.2 which can be seen at the URL;

  https://www.quanzhanketang.com/colors/colors_names.html
- There are three ways to specify the colour:
  - <h3 style="color:blue">Your text is now blue</h3>
  - <h3 style="color:rgb(0,0,255)> Your text is now blue </h3>
  - <h3 style="color:0000FF;"> Your text is now blue </h3>
    - Remember: the hexadecimal reference is made up of numbers and letters
- The colour Blue may in the red-green-blue (rgb) scale:
  - 0,0,0 (black) to 255,255,255 (white)

Document Linking

- In week I we introduced *hypertext* which is central feature of HTML
- The HTML tags for document linking can be found at w3shools.com
- Using hypertext linking we can:
  - Create an external link in a web-page (the source) to another web-page (the destination)
    - `<a href="http://www.w3schools.com/html/">Visit our HTML tutorial</a>`
  - Create an local (internal) link in a section of a web-page to another section within the same web-page
    - `<a href="html_images.asp">HTML Images</a>`
  - Create a link to enable users viewing your web-page to send an email to the creator of the web-page (or any other legitimate email address (this is termed a *mailto*)
    - `<a href= "mailto: abc@example.com">Send Email</a>`

Creating an Ordered List

- An ordered list starts with the **`<ol>`** tag.
- Each list item starts with the **`<li>`** tag.
- Uppercase letters
  - **`<ol type="A">`**
- Lowercase letters
  - **`<ol type="a">`**
- Numbers
  - **`<ol type="1">`**

```
<ol type="1">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

```
<ol type="A">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

```
<ol type="I">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Creating an Unordered List

- An unordered list starts with the
  - **`<ul>`** tag
- Each list item starts with the
  - **`<li>`** tag
- The list items will be marked with bullets (small black circles) by default

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Creating a Description List

- HTML supports description lists.
- A description list is a list of terms, with a description of each term.
- The **`<dl>`** tag defines the description list
- The **`<dt>`** tag defines the term (name)
- The **`<dd>`** tag describes a description list with terms and descriptions

```
<dl>
  <dt>Coffee</dt>
  <dd>- black hot drink</dd>
  <dt>Milk</dt>
  <dd>- white cold drink</dd>
</dl>
```

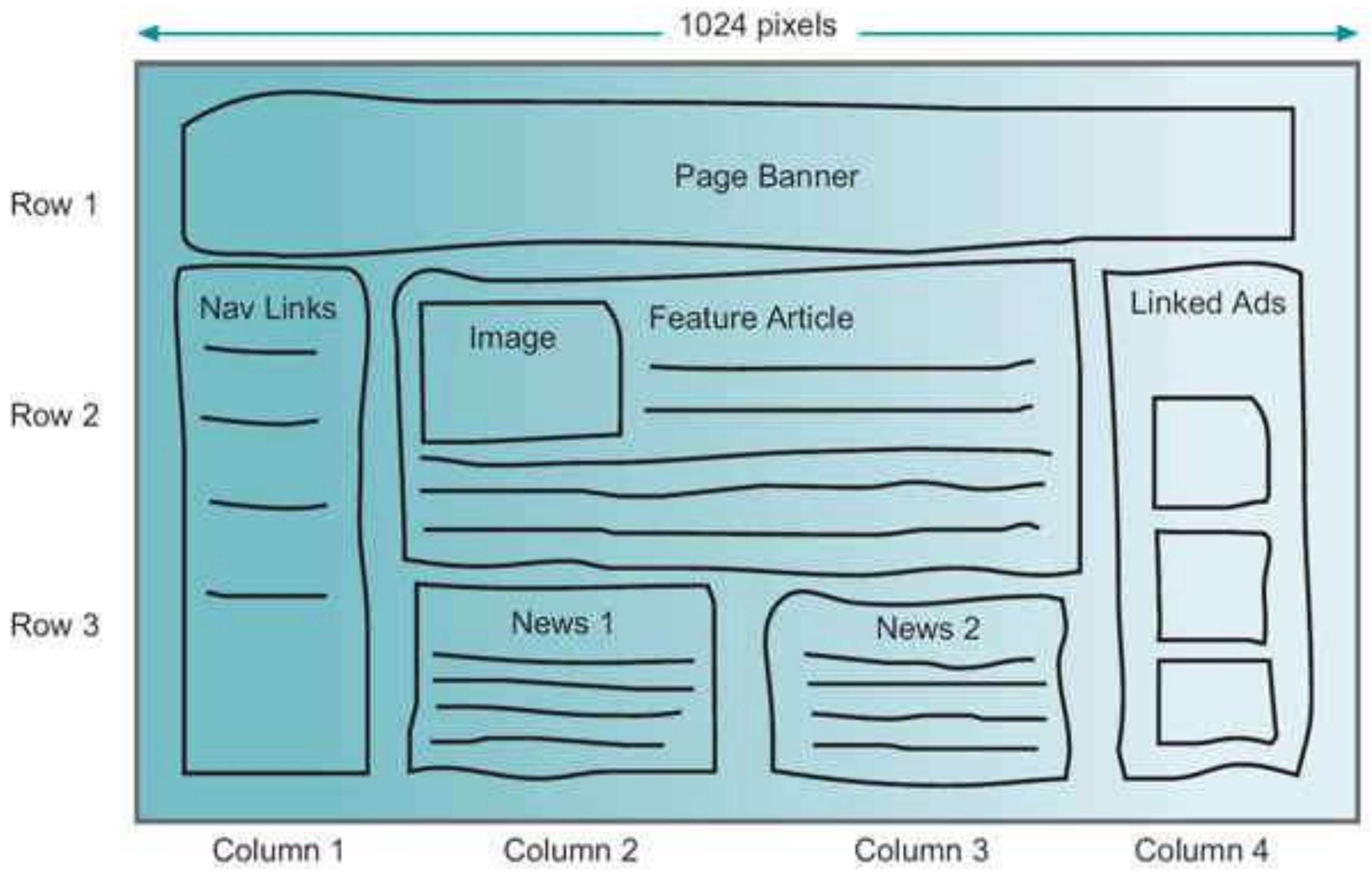
Creating Tables

- An HTML table is defined with the **`<table>`** tag
- Each table row is defined with the **`<tr>`** tag
- A table header is defined with the **`<th>`** tag
- By default, table headings are bold and centered
- A table data/cell is defined with the **`<td>`** tag

```
table style="width:100%">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

Tables in Web-Page Layout

- Web-page creation requires the page layout to be designed
- As shown in the figure areas of a web-page can be allocated for specific uses
- Tables (and nested tables) can be used) to build a web-page layout and navigation
- We will consider web-page layout, frames, inline frames <iframe> and windows in Week 3



Adding Images (1)

- Adding images
  - Requires the location of the image which is stored in a local folder
  - The path to the image will be known and used to access the image
  - When the image is changed (in the local folder) the image in the web-page will be changed when the web-page is created
- In HTML
  - Images are defined with the **`<img>`** tag.
  - Note: the **`<img>`** tag is empty, it contains attributes only, and does not have a closing tag.
  - The **`<src>`** attribute specifies the URL (web address) of the image

Adding Images (2)

- The **`<alt>`** attribute provides an alternate text for an image
- In the event of a slow connection (or) if there is an error in the **`<src>`** attribute (or) if the user is uses a screen reader:
  - The value of the alt attribute should describe the image (the image will not be shown – only the **`<alt>`** attribute
- Using the HTML:
  - **`<img src="img_chania.jpg" alt="Flowers in Chania">`**
  - The **`<alt>`** attribute will show on the web-page:
  - "Flowers in China"

Adding Images as a Hypertext Link

- We may use an image as a hypertext link
  - Just put the **`<img>`** tag inside the <a> tag:
- For example:

```
<a href="default.asp">
      <img src="smiley.gif" alt="HTML tutorial”
      style="width:42px;height:42px;border:0;">
</a>
```

- The HTML shows the **`<a>`** … **`</a>`** tags and adds style with the size of the image.
  - We will introduce style and *Cascading Style Sheets* (CSS) in week 3

Image Mapping (1)

- We have learned that an image can be a link to:
  - Another web-page
  - A section in the same page
- We can also subdivide a single image and link parts of the image to another web-page
  - This type of subdivided image is termed: An *image map*
- The following slide illustrates the concept of an image map
  - The map (a coincidence) shows the states of the USA
  - Each state has a link to a separate dedicated web-site for each state

Image Mapping

- The <map> tag is used to define a client-side image-map.
- An image-map is an image with clickable areas.
- The required name attribute of the <map> element is associated with the <img> usemap attribute to create a link between the image and the map.
- The <map> element contains a number of <area> elements
- The <area> element defines the clickable areas in the image map.
- In Week 3 we will consider
  - Frames and inline-frames <iframe>
  - Windows
  - Web-page layout using frames

Image Mapping

- Image maps may be created on using both *client-side* (or) *server-side* web programming
- In both cases the image map is stored on the web-server
- In client-side implementation
  - The web-browser does the work to deliver the new location
  - The web-browser selects the specified link in the activated region and follows it
- In server-side implementation
  - The user clicks on the server-side image map
  - The server receives the request
  - The server delivers the response (the linked resource)

Frames

<frameset>

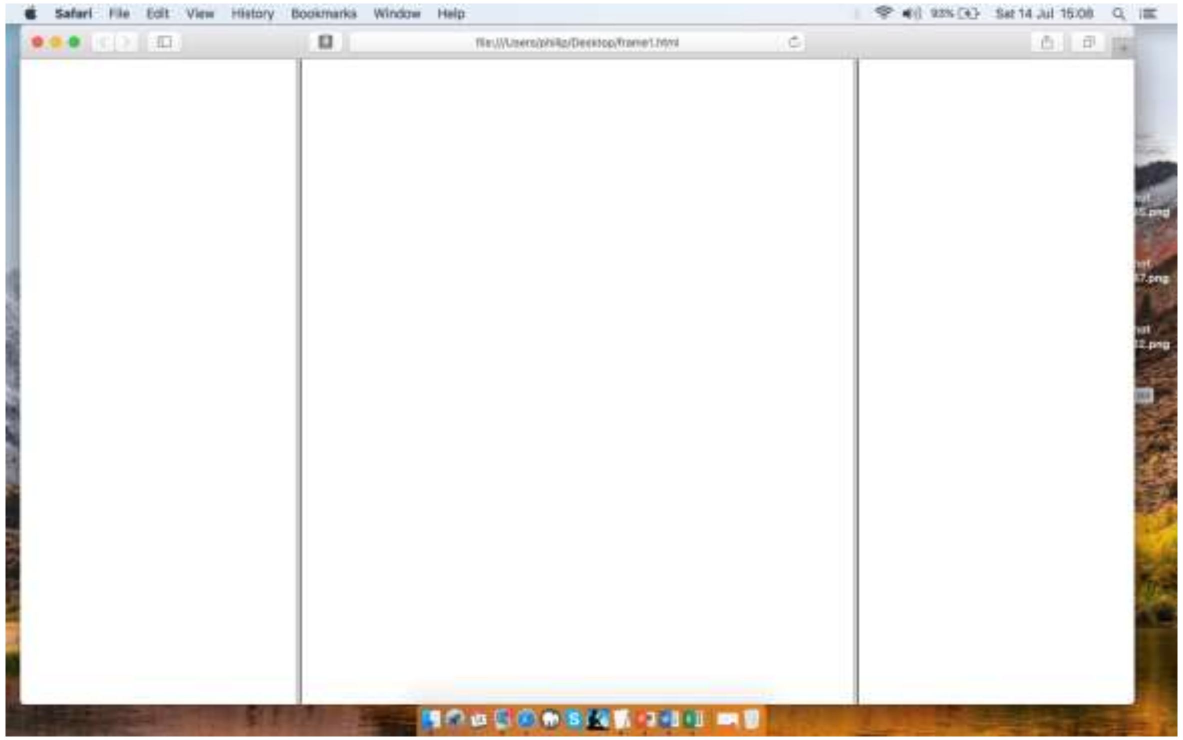
- The <frameset> element
  - Holds one or more <frame> element(s)
  - Each <frame> element can hold a separate HTML document
  - Specifies the number of columns and / or rows in a <frameset>
  - Specifies the size (the amount of space occupied by the frame)
  - The size is represented as either
    - A percentage (%) of the width of the browser window
      - The actual size varies according to the browser and the screen resolution
    - In pixels
      - This will produce a fixed size – the browser will introduce scroll bars

<frame>

- The <frame> tag defines one particular window (or frame) within a <frameset> and each <frame> in a <frameset> can have different attributes:
  - frameborder: *specifies whether or not to display a border around a frame*
  - longdesc: *specifies a page that contains a long description of the content of a frame*
  - marginheight: *specifies the top and bottom margins of a frame*
  - marginwidth: *specifies the left and right margins of a frame*
  - name: *specifies the name of a frame*
  - noresize: *specifies that a frame is not resizable*
  - scrolling: *specifies whether or not to display scrollbars in a frame*
  - src: *specifies the URL of the document to show in a frame*

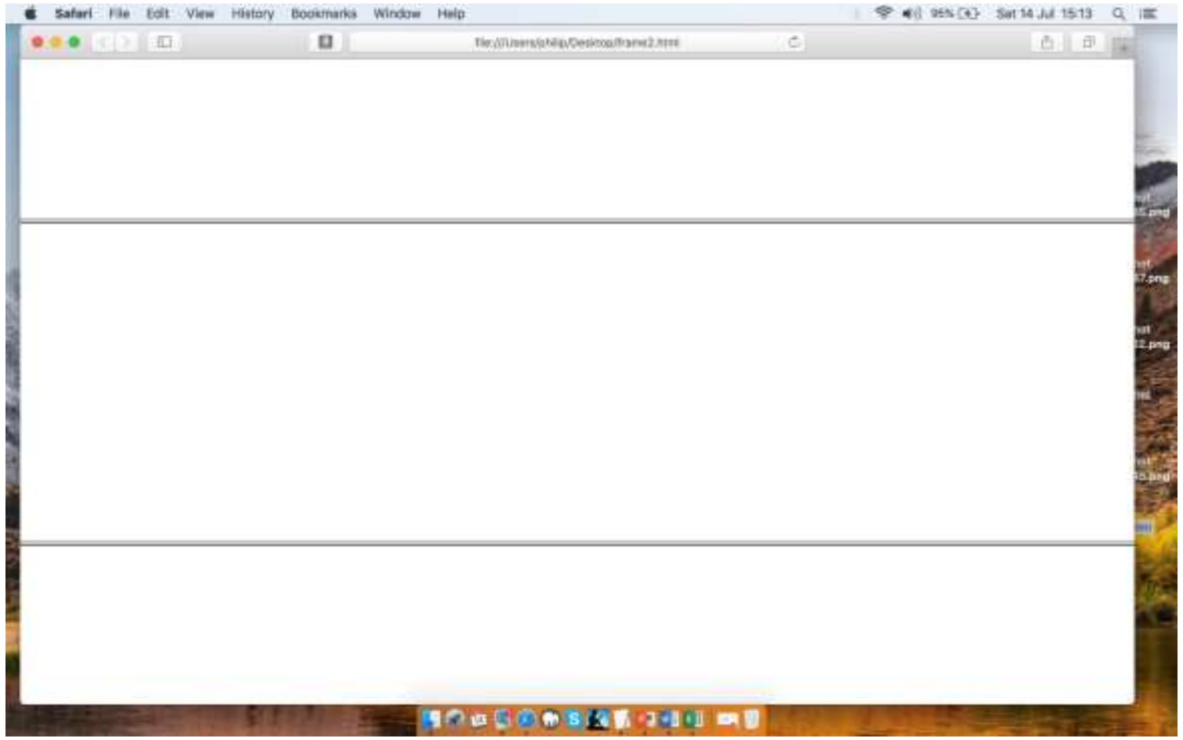
Horizontal Frames

```
<!DOCTYPE html>
<html>
<frameset cols="25%,*,25%">
  <frame src="frame_a.htm">
  <frame src="frame_b.htm">
  <frame src="frame_c.htm">
</frameset>
</html>
```



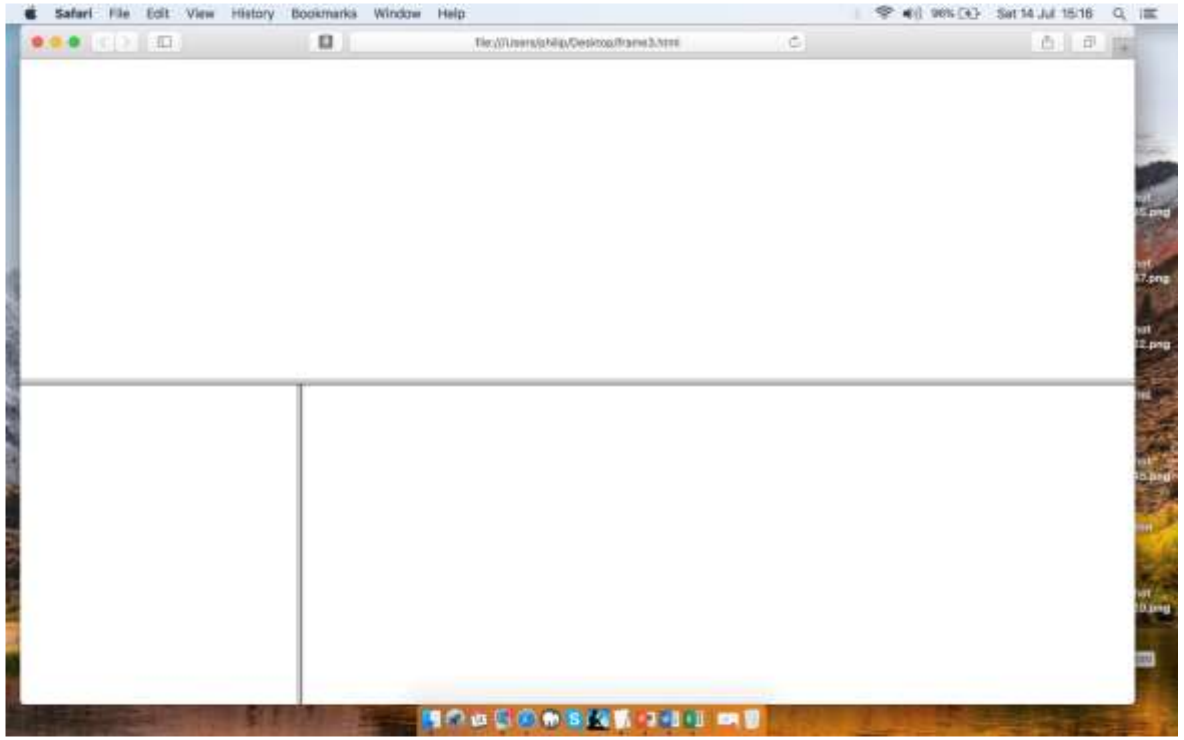
Vertical Frames

```
<!DOCTYPE html>
<html>
<frameset rows="25%,*,25%">
  <frame src="frame_a.htm">
  <frame src="frame_b.htm">
  <frame src="frame_c.htm">
</frameset>
</html>
```



Nested Frames

```
<!DOCTYPE html>
<html>
<frameset rows="50%,50%">
  <frame src="frame_a.htm">
  <frameset cols="25%,75%">
    <frame src="frame_b.htm">
    <frame src="frame_c.htm">
  </frameset>
</frameset>
</html>
```



Inline frames

Inline Frames (<iframe>)

- The <iframe> tag specifies an inline frame which is used to embed another document within the current HTML document
  - The same function achieved using the <frameset> and <frame> tags
- To work with browsers that do not support <iframe>
  - add a text statement between the opening <iframe> tag and the closing </iframe> tag
- While style may be embedded in an HTML document
  - It is recommended that a CSS is used to style <iframe> (including scrollbars)
- There are differences Between HTML 4 and HTML5
  - In HTML5 there are additional new attributes
  - Several HTML 4.01 attributes are removed from HTML5
- There are differences Between HTML and XHTML
  - In XHTML the <name> attribute is ***deprecated*** and will be removed.
  - Use the *global id attributes* (w3schools.com)

```html
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Nested iframe example</title>
</head>
<body>
<iframe width="560" height="315"
src="inset.html" frameborder="0"
allowfullscreen>
</iframe>
</body>
</html>
</html>
```

- The index.html file:
  http://localhost/Nested_iframe_snu/

```html
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>nested iframe</title>
</head>
<body>
<iframe width="560" height="315"
src="SNU.mp4" frameborder="0"
allowfullscreen>
</iframe>
</body>
</html>
```

- This file is the “inset.html” file called from the ”index.html” file

Frames and Inline Frames

- Frames may be used to structure an HTML document
- Inline frames must **NOT** to be used to structure an HTML document
  - In the example shown the <iframe> is used to hold a media file within an HTML document
- In HTML5 structuring a document uses:
  - Layout managers
  - Cascading Style Sheets
    - This will be covered in later sessions

What is Localhost?

Localhost

- "Localhost" refers to the local (your) computer that a program is running on
- For example:
  - If you are running a Web browser (such as firefox) on your computer
  - Your computer is the "localhost”
- Two non-local computers must be defined by their IP addresses
  - For example: 210.26.118.25 (and) 210.26.110.24
- The local machine is defined as "localhost"
  - The IP address is 127.0.0.1

Forms and Data Input

Forms and Data Input

- When considering the creation of forms and data input and processing we must consider:
  - Usability
  - Security
  - We will explore the usability / security ‘trade-off’ later
- Forms are used to collect data and information from users
- The data which may be collected includes:
  - email, personal information, text input
  - The processing of the information (requires JavaScript and PHP)
- In considering data input and processing there are 2 methods:
  - There are 2 methods for managing data input: ***GET*** and ***POST***

Why use GET (or) POST

- ***GET*** and ***POST*** are HTTP methods
- The motivation for selecting GET method (or) POST method
  - Essentially relates to Internet security
- The default input method is ***GET***
  - However there are security implications for users if GET input method is used
- The alternative input method is ***POST***
  - Where data security and sensitive personal information is entered and processed
  - The ***POST*** input method achieves improved security
- An overview of HTTP request and input methods can be found in the course resources

The GET Input Method

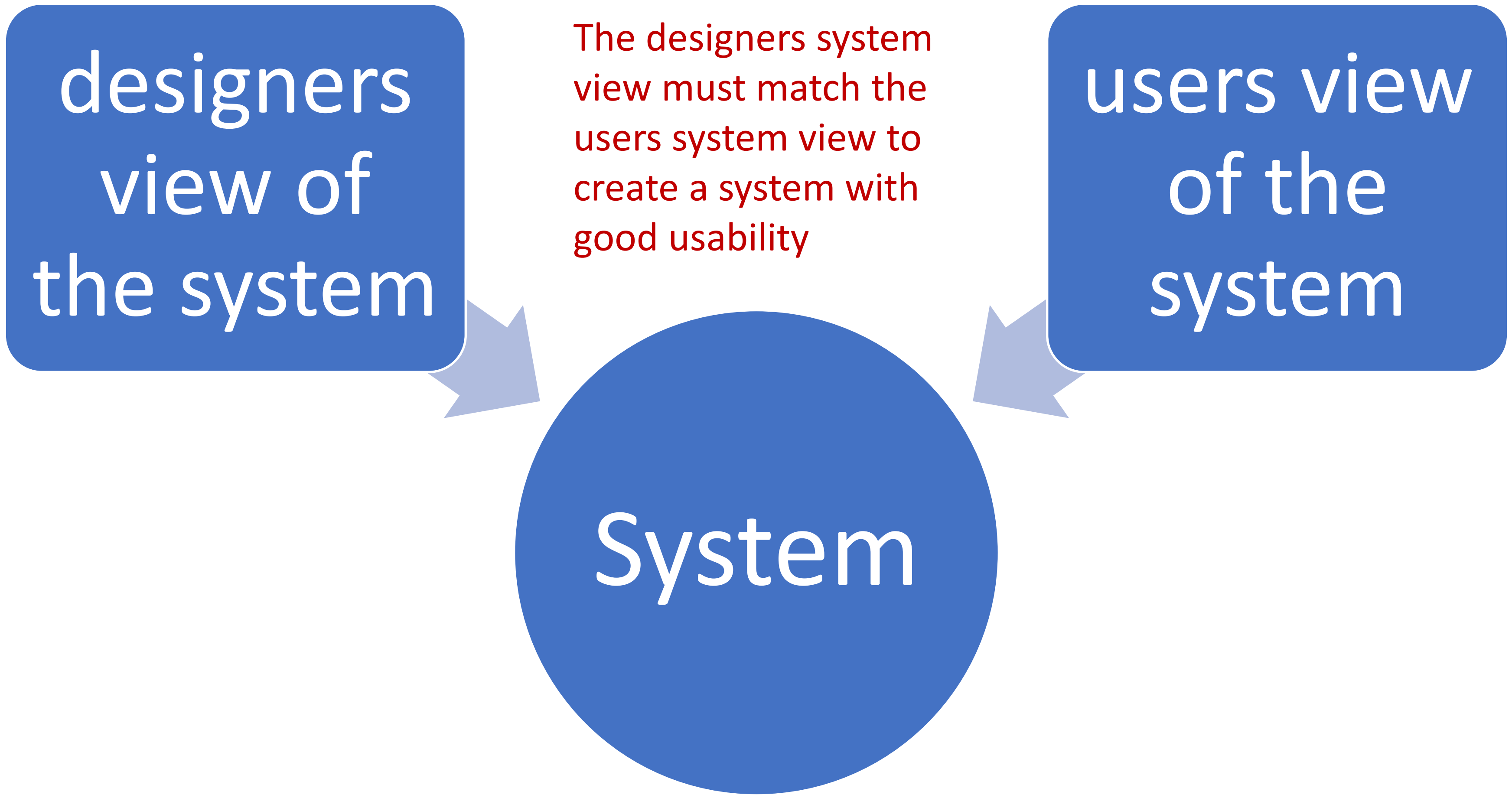
- The ***GET*** input method is the default method for data submission to an HTML 5 form
- When the ***GET*** method is used:
  - The form data submitted form data will be visible in the web-page address field
  - The form-data is appended to the URL in name/value pairs
  - The length of a URL is limited (approximately 3000 characters)
- In summary the ***GET*** method it is:
  - Never used for sensitive data (it will be visible in the URL)
  - Useful for form submission for bookmarking
  - Restricted to non-secure data (such as query strings)

The POST Input Method

- The ***POST*** input method is the:
  - Preferred input method where data and information security is important
  - The form data contains sensitive or personal information.
- The ***POST*** input method:
  - Does not display the submitted form data in the page address field
  - Can be used to send large amounts of data
  - Form submissions using ***POST*** cannot be bookmarked with certainty

Web-Site Design
Design Principles
Web Site Building

A System Design Conceptual Model



The designers system view must match the users system view to create a system with good usability

Coffee Pot (users system view)

Coffee Pot (designers system view)

Donald Norman – Design Principles

- Provide rich, complex, and natural signals
- Be predictable
- Provide a good conceptual model
- Make the output understandable
- Provide continual awareness, without annoyance
- Exploit natural mapping to make interaction understandable and effective

Web Design Principles (Donald Norman)

- Visibility
  - Users need to know what all the options are, and know straight away how to access them. In the case of websites, this is an easy win.
- Feedback
  - Every action needs a reaction. There needs to be some indication, like a sound, a moving dial, a spinning rainbow wheel, that the user’s action caused something.
- Affordance
  - Affordance is the relationship between what something looks like and how it's used.

Six Web Design Principles (Donald Norman)

- Mapping
  - Mapping is the relationship between control and effect. The idea is that with good design, the controls to something will closely resemble what they affect.
- Constraints
  - Constraints are the limits to an interaction or an interface. Some are really obvious and physical, for example the screen size on a phone.
- Consistency
  - The same action has to cause the same reaction, every time.
- For more information on these web design principles with some examples see:
  - https://enginess.io/insights/6-principles-design-la-donald-norman

Web Systems Design

- From the previous slides we can see that web site design:
  - Is driven by the requirements of the web site owner
  - There is no one-way to design a web site
    - While there are correct design layouts (good design)
    - There are also incorrect design layouts (bad design)
- The design principles set out by Donald Norman must be applied to all web site design
- The following slides show what not to do!

Web-Page Layout Semantic Elements Web page Design and Build

12/10/2020

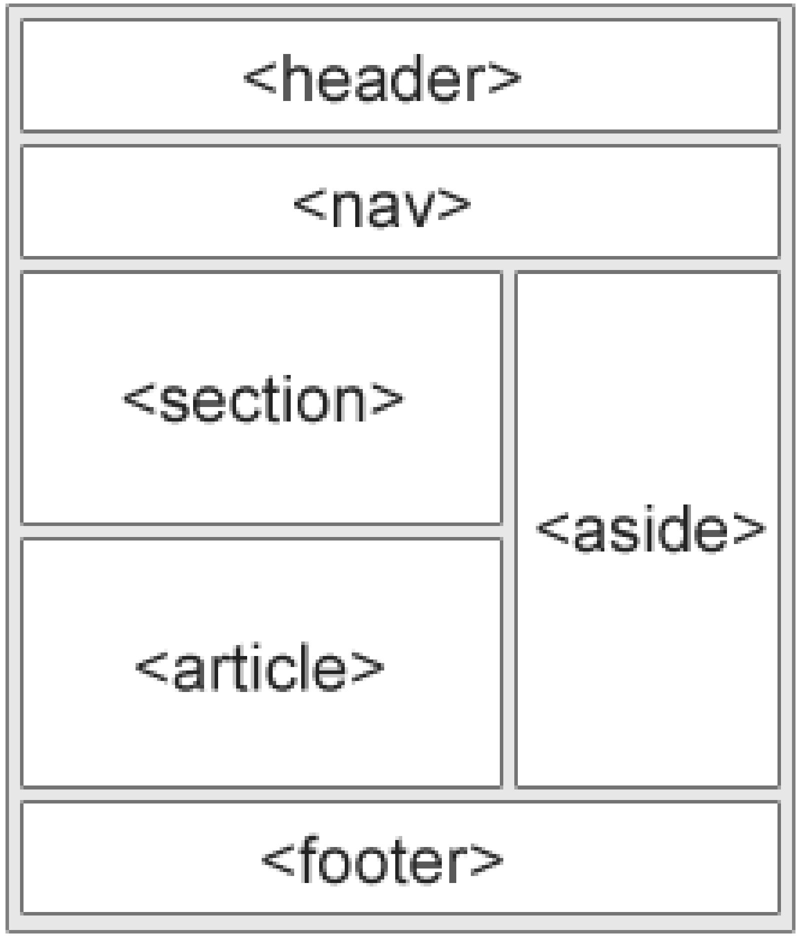
HTML4 Semantic Elements

- What are Semantic Elements?
- A semantic element describes the meaning and purpose to both the *web browser* and to *web designers* and *developers*
- In HTML 4 there were *non-semantic elements* such as:
  - <div> <span> <p> <br>
  - These elements define and specify nothing about the content within the tag
- In HTML 4 there were limited *semantic elements* such as:
  - <form> <table> <img>
  - These elements define and specify the content within the tag

HTML5 Semantic Elements

- Many web sites contain HTML code such as:
  - <div id="nav"> <div class="header"> <div id="footer">
  - To indicate navigation, header, and footer
- HTML5 introduced new *semantic elements* to define different parts of a web page
- There are 13 *semantic elements* in HTML5
  - <article> <aside> <details> <figcaption> <figure> <footer> <header>
  - <main> <mark> <nav> <section> <summary> <time>
- Details of semantic elements may be found at:
  - The course text book (see pages 38 to 50)
  - The W3schools web site at:
    - https://www.quanzhanketang.com/html/html5_semantic_elements.html

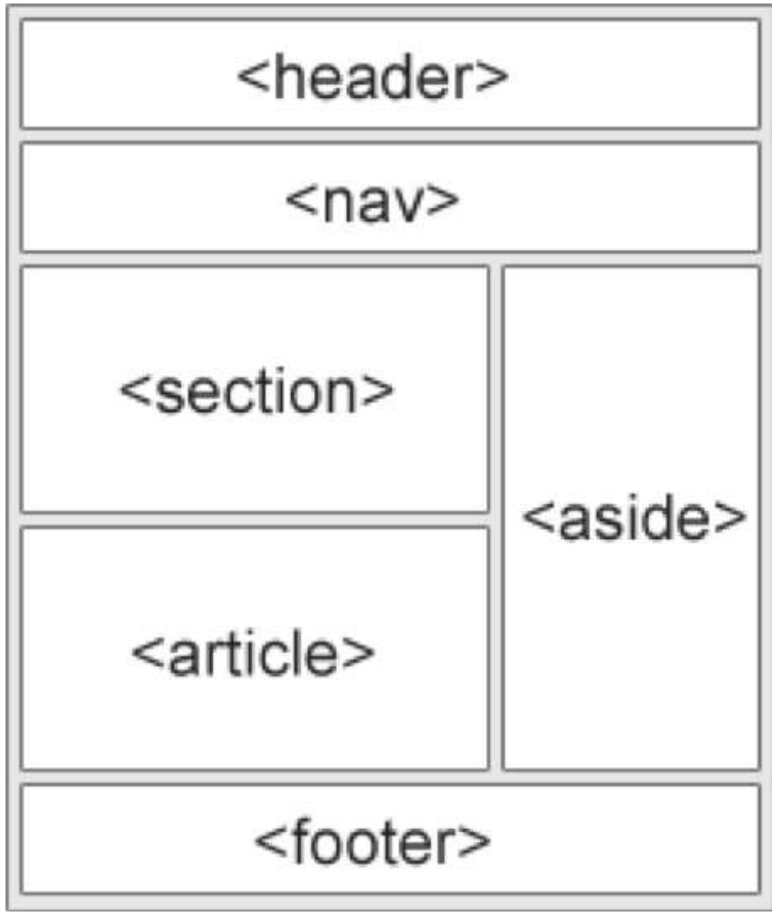
HTML5 Semantic Elements



- The figure shows an example of a web page structure defined using *semantic elements*
- The layout is only an example
- The actual layout of a web page will be defined by the web designer
- In general the location of the <header> <nav> <footer> elements are as shown in the figure
  - However: as shown in the previous examples the layout may change to suit the web site requirements

HTML Layout Elements

- <header>
  - Defines a header for a document or a section
- <nav>
  - Defines a container for navigation links
- <section>
  - Defines a section in a document
- <article>
  - Defines an independent self-contained article
- <aside>
  - Defines content aside from the content (like a sidebar)
- <footer>
  - Defines a footer for a document or a section
- <details>
  - Defines additional details
- <summary>
  - Defines a heading for the <details> element

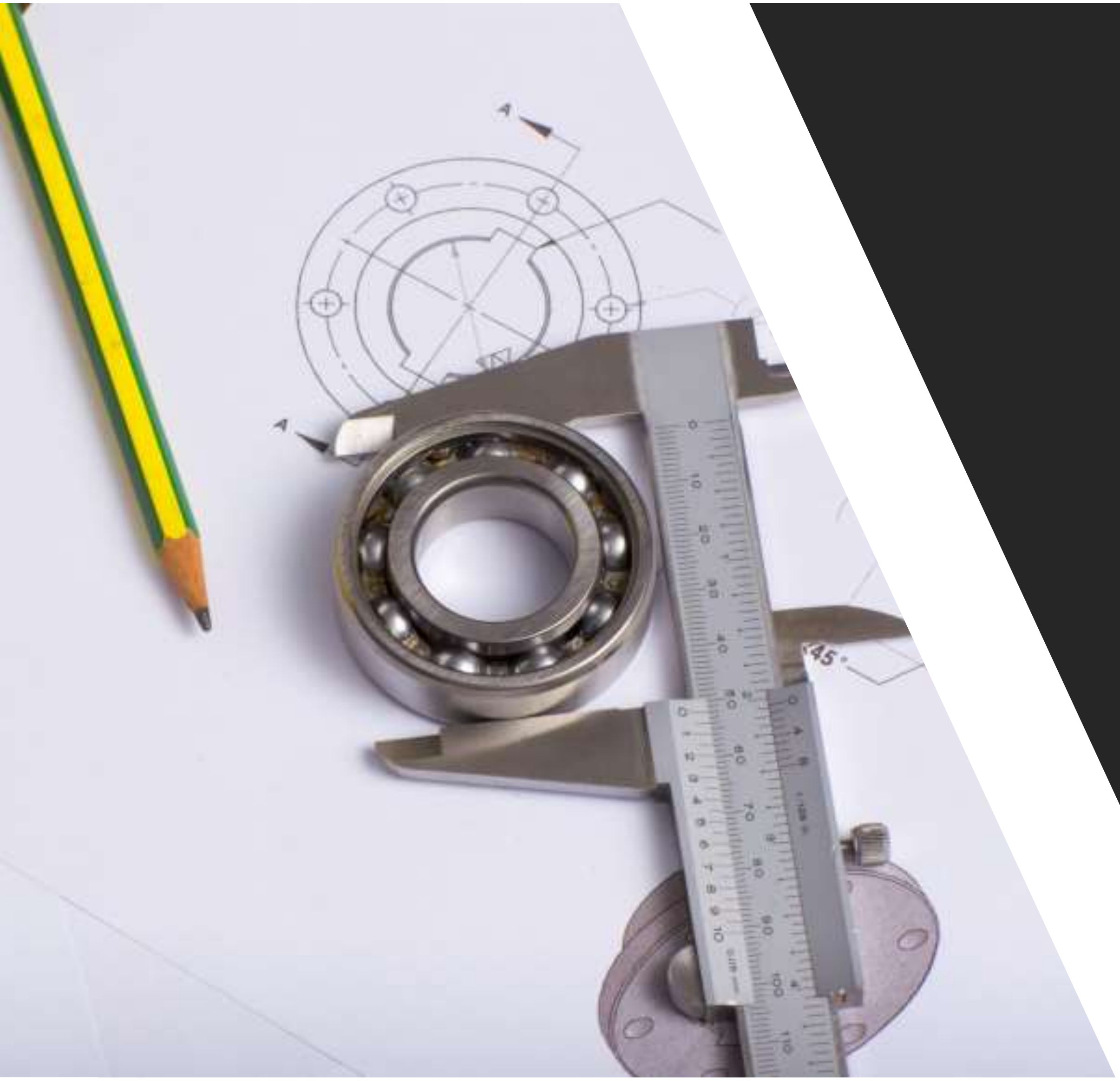


bringing it all together

- Knowing how to create HTML files and run web-pages / web-sites does not address the need for good web-site design
- Bringing it all together is the process of design where
  - The requirements are investigated and identified
  - The appropriate features are selected
  - The features selected are combined into a web-page / web-site that is
    - Attractive / easy to use / effective in presenting the information / provides user interactive feedback
- It is the task of the designer to
  - Meet the user requirements specification
  - Provide a multi-page web-site with clear navigation that users will want to visit

Usability and Security

- Web-based systems must address a two important factors:
  - *Usability* (and)
  - *Security*
- Usability:
  - Relates to how easy users can access a web-site and use the functions
  - Usability applies to both *'stateless'* (static) and *'stateful'* (dynamic) web sites
- Security:
  - While we must consider security for both *'stateless'* and *'stateful'* web sites
  - Security is very important for *'stateful'* web sites
    - For example: security is essential in an on-line banking web-site / mobile application
- There is a *'trade-off'* between *usability* and *security*



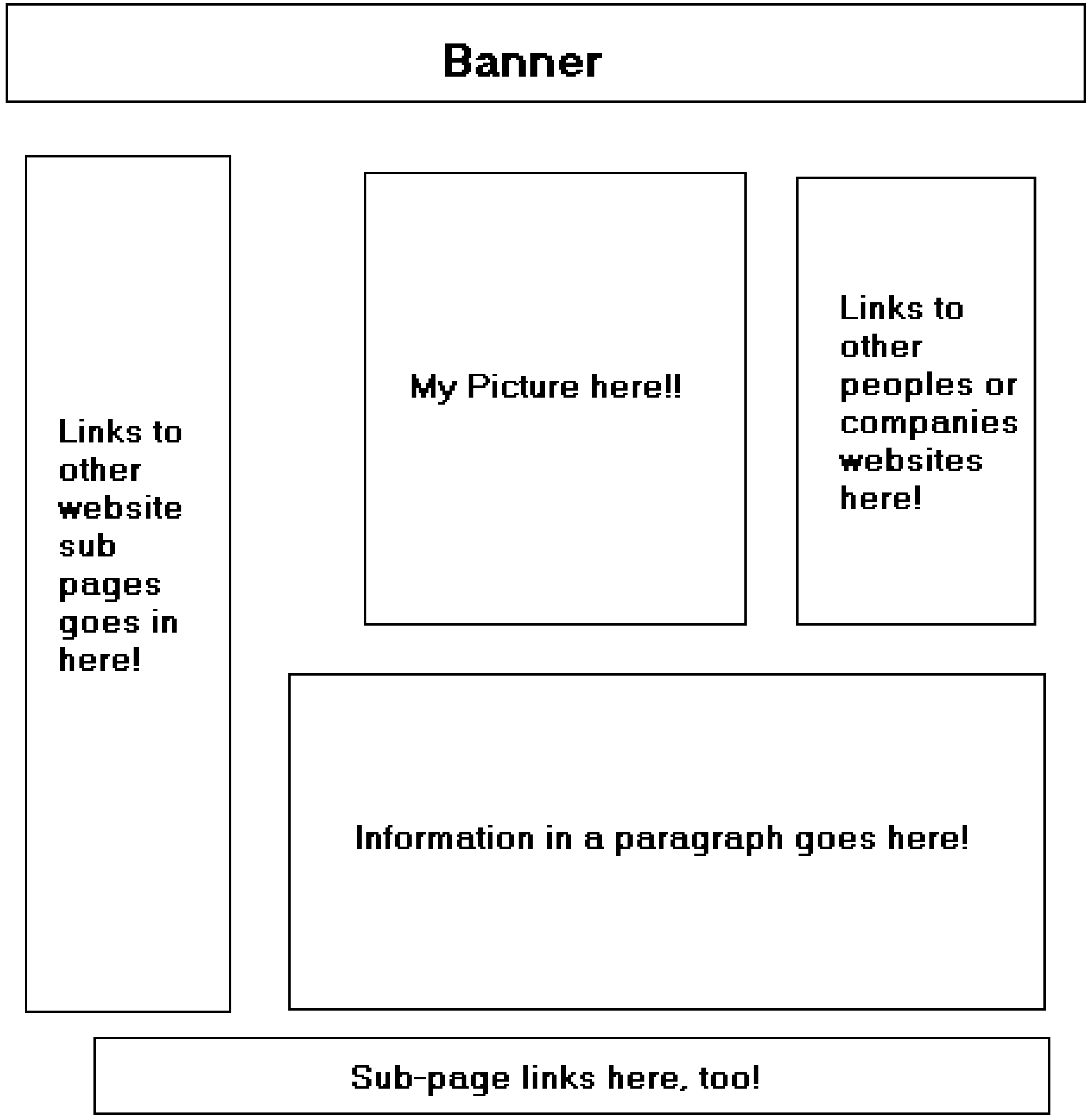
The Design Process

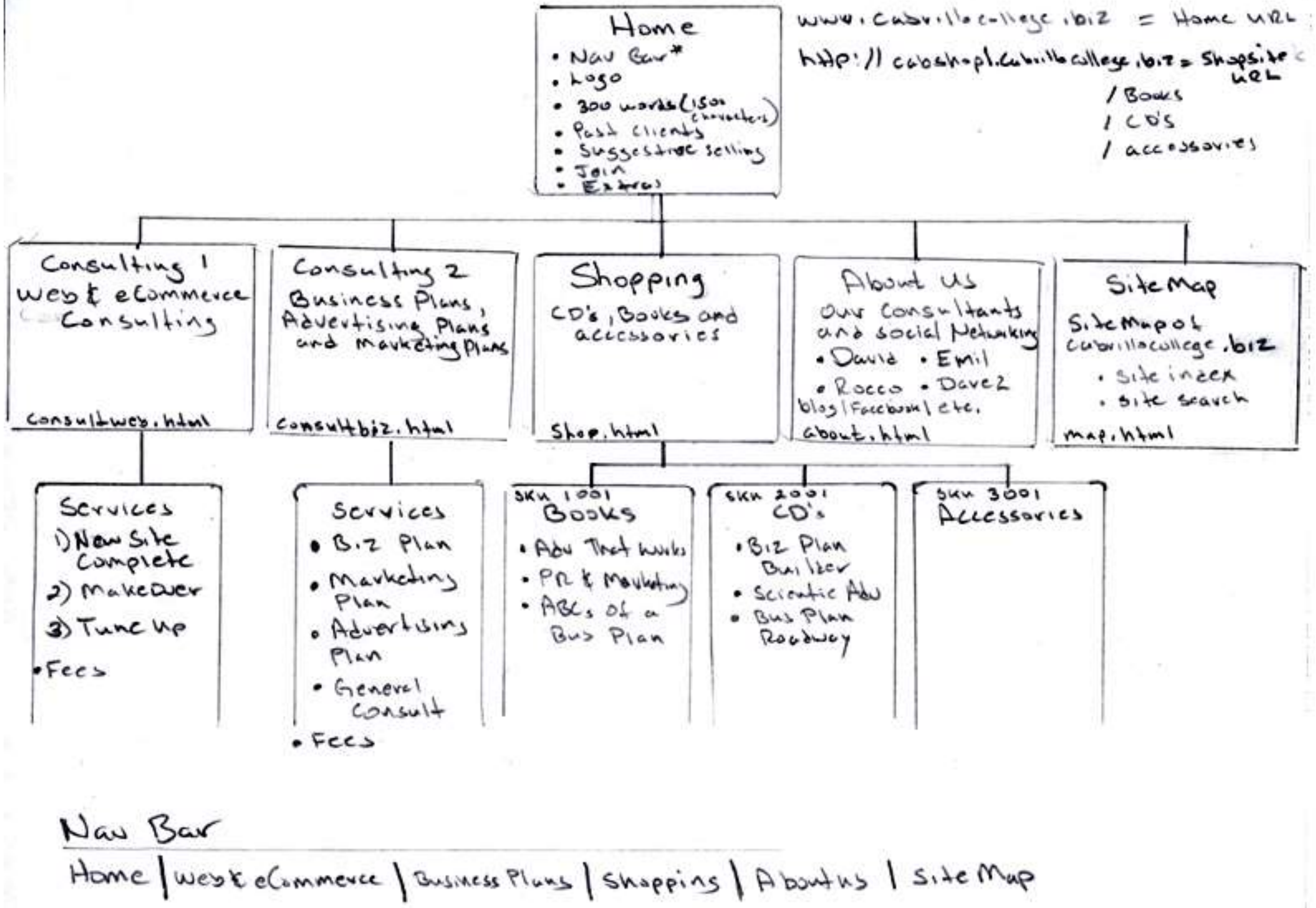
To Design a Web Site

- Carry out the following steps:
  - Identify and document the requirements and uses for the web site
  - Design your web site on paper (termed the LoFi design process)
  - Create a 'storyboard'
    - A storyboard is used in software development to identify the specifications for a particular web site
    - During the specification phase, screens that the web site pages will display are drawn, either on paper or using other specialized modelling software, (to illustrate the important steps of the user experience)
    - The storyboard is then modified to address the specific needs of the web site

Storyboarding

- A storyboard shows how the web site (and the related links) will look and function
- It is cheaper to make changes to a storyboard than a built web site
  - The following slides show three typical examples of 'storyboarding'
  - The use of specialized software is shown in the first two examples
  - The use of paper design is shown in the third example


Banner
Links to other website sub pages goes in here!
My Picture here!!
Links to other peoples or companies websites here!
Information in a paragraph goes here!
Sub-page links here, too!


Home
• Nav Bar*
• Logo
• 300 words (1500 characters)
• Past clients
• Suggestive selling
• Join
• Extras
www.cabrillocollege.biz = Home URL
http://cabshop1.cabrillocollege.biz = Shopsite URL
/ Books
/ CD's
/ accessories
Consulting 1
Web & eCommerce Consulting
consultweb.html
Consulting 2
Business Plans, Advertising Plans and Marketing Plans
consultbiz.html
Shopping
CD's, Books and accessories
Shop.html
About Us
Our Consultants and social Networking
• David • Emil
• Rocco • Dave2
blog | Facebook | etc.
about.html
Site Map
SiteMap of cabrillocollege.biz
• site index
• site search
map.html
Services
1) New Site Complete
2) MakeOver
3) Tune Up
• Fees
Services
• Biz Plan
• Marketing Plan
• Advertising Plan
• General Consult
• Fees
SKU 1001
Books
• Adv That Works
• PR & Marketing
• ABC's of a Bus Plan
SKU 2001
CD's
• Biz Plan Builder
• Scientific Adv
• Bus Plan Roadway
SKU 3001
Accessories
Nav Bar
Home | Web & eCommerce | Business Plans | Shopping | About us | Site Map

Storyboarding

- In designing your own personal web site:
  - Prepare a storyboard showing the home web page and the linked HTML web pages (showing information about you)
  - On the storyboard show the links with links to any related files (e.g., a separate file holding your images)

The Build Process

Web Site Building (1)

- When you are happy with your storyboard design
  - Create the home web page (index.html) structure (the layout) including the navigation links to the related web pages
- The structure for all the web pages (the layout) can be defined in the home web page (index.html)
- The index.html web page may be used as a template and copied (using a different page specific name - ***.html**) and used throughout the web site
  - This ensures consistency, saves time, and reduces errors

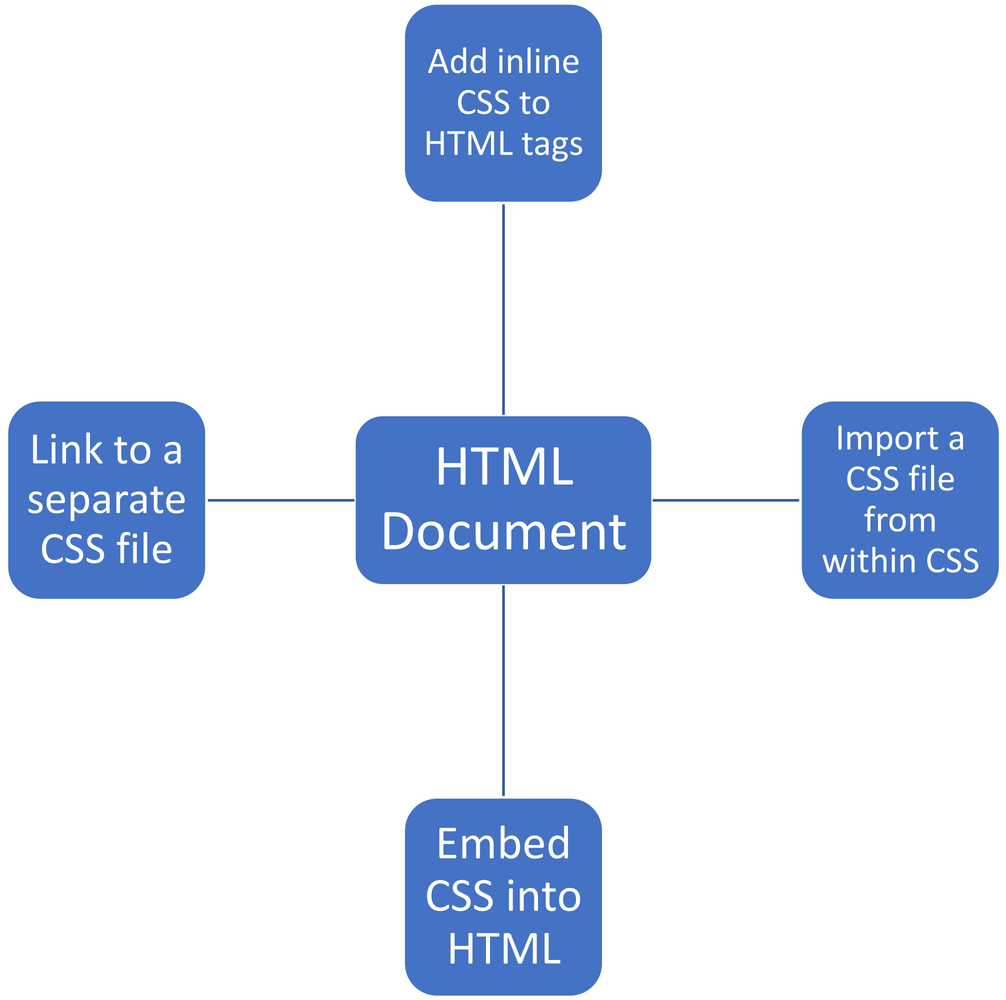
Web Site Building (2)

- When the home web page (index.html) structure and layout (including the navigation links) is complete:
  - Make renamed copies for each of the related web pages
- All the html web pages and related files plus the images must be in the same project folder
- You can then proceed to complete the HTML content in the index.html file and the related html web pages.
- As the build continues periodically test the work in progress
- On completion of the build test and validate the final web site
- Remember to place two links in the footer of each web page to
  - Return to the top of the web page
  - Return to the home web page

Adding Style

Adding Style to HTML

- There are 4 methods of adding style to an HTML document
  1. Adding inline CSS to individual HTML tags
  2. Linking to a separate CSS file
  3. Embedding CSS into the HTML
  4. Import a CSS file from within CSS
- In this course we will focus on methods 1, 2, and 3


Add inline CSS to HTML tags
Link to a separate CSS file
HTML Document
Import a CSS file from within CSS
Embed CSS into HTML

Embedded CSS

- CSS 'style rules' may be embedded directly into any HTML web-page
- To embed CSS rules into an HTML document add the following code to the **<head>** of an HTML document:

```
<style media="screen" type="text/css">
   Add style rules here
</style>
```

- All CSS rules are placed between the **<style>** … **</style>** tags.
- The media can be "screen" for your computer screen or "print" for printing
- The disadvantage with embedding is the styles must be downloaded every time someone visits the page
- However there are a advantages
  - Because the CSS is part of the HTML document the whole web-page exists as just one file
  - This can be useful if the web-page is a template

Inline CSS (1)

- Style rules can also be added directly to any HTML element
  - Simply add a style parameter to the element and enter your style rules as the value.
  - Here is an example of a heading with red text and a black background:

```
<h2 style="color:red;background:black;">
    This is a red heading with a black background
</h2>
```

  - This is not a very good method: it will cause 'bloat' in the HTML
  - More importantly: inline CSS makes web-site updating and maintenance difficult and error prone

Inline CSS (2)

- However inline CSS can be useful in some situations
  - An example may be where a system is used which has no access to the CSS file
  - In such a case add the 'style rules' directly to the elements
- Other potential uses for inline CSS include
  - An internal style sheet may be used if one single page has a unique style
  - Internal styles are defined within the <style> … </style> element, inside the <head> … </head> section of an HTML page
  - An inline style may be used to apply a unique style for a single element
  - To use inline styles, add the style attribute to the relevant element – the style attribute can contain any CSS property

Linking to a Separate (external) CSS

- With this method all of your style rules are contained in a single text file that is saved with the *.CSS extension.
- This file is saved on the server and is linked to each HTML file
- The link is a line of HTML in the <head> … </head> section of an HTML document as follows:

```
<link rel="stylesheet" type="text/css"
href="mystyles.css" media="screen" />
```

Running the External CSS Method

- We have seen the creation of
  - The “index.html” file
  - The simple “styles.css” file
- To run these files
  - The “index.html” file will be found by the browser and run in the normal way
  - The “styles.css” file will be stored in a known location where the path is known to the index.html file
- It is the usual (and best) practice
  - To store the “styles.css” file in the same directory (file) as the “index.html” file

HTML Layout Techniques

A multi-column web-page layout may be created using 4 methods:

- HTML tables
- CSS float property
- CSS flexbox
- CSS framework

The CSS float and flexbox frameworks are similar to the *GridBagLayout* and *FlowLayout* page layout methods used in Java

Details of the layout techniques can be found at w3schools.com

CSS Float Properties

Full details of the CSS float property can be found at [w3schools.com]

Property	Description
box-sizing	Defines how the width and height of an element are calculated: should they include padding and borders, or not
clear	Specifies what elements can float beside the cleared element and on which side
float	Specifies how an element should float
overflow	Specifies what happens if content overflows an element's box
overflow-x	Specifies what to do with the left/right edges of the content if it overflows the element's content area
overflow-y	Specifies what to do with the top/bottom edges of the content if it overflows the element's content area

HTML Layout using CSS flexbox

- CSS flexbox is a new layout mode in CSS3
- Advantages
  - Use of flexbox ensures that elements behave predictably when
    - The web-page layout must accommodate different interfaces and screen sizes
    - The web-page layout must present the optimal layout on different devices
- Disadvantages
  - CSS flexbox is not supported by older browsers (IE10 and earlier)

CSS Flexbox Properties

Full details of the CSS flexbox properties can be found at [w3schools.com]()

Value	Description
flex-grow	A number specifying how much the item will grow relative to the rest of the flexible items
flex-shrink	A number specifying how much the item will shrink relative to the rest of the flexible items
flex-basis	The length of the item. Legal values: "auto", "inherit", or a number followed by "%", "px", "em" or any other length unit
auto	Same as 1 1 auto.
initial	Same as 0 1 auto. Read about *initial*
none	Same as 0 0 auto.
inherit	Inherits this property from its parent element. Read about *inherit*

CSS Properties used in flexbox

Property	Description
display	Specifies the type of box used for an HTML element
flex-direction	Specifies the direction of the flexible items inside a flex container
justify-content	Horizontally aligns the flex items when the items do not use all available space on the main-axis
align-items	Vertically aligns the flex items when the items do not use all available space on the cross-axis
flex-wrap	Specifies whether the flex items should wrap or not, if there is not enough room for them on one flex line
align-content	Modifies the behavior of the flex-wrap property. It is similar to align-items, but instead of aligning flex items, it aligns flex lines
flex-flow	A shorthand propert for flex-direction and flex-wrap
order	Specifies the order of a flexible item relative to the rest of the flex items inside the same container
align-self	Used on flex items. Overrides the container's align-items property
flex	Specifies the length of a flex item, relative to the rest of the flex items inside the same container

HTML Layout using CSS flexbox

- From the following slides we can see that
  - The profile (the width) of the web-page dynamically changes
  - Changing the profile of the web-page also redefines the sections defined in the web-page
  - This is achieved by ‘dragging’ the edges of the web-page to a desired size
  - The height of the web-page will also change dynamically
- As for the *CSS flexbox* the capability to change profile has a number of benefits
  - The user may change the profile ‘on-the-fly’ when viewing the web-page
  - Again: the layout of the webpage can be changed to suit different devices and interfaces

HTML Layout using CSS float

- From the following slides we can see that
  - The profile (the width) of the web-page dynamically changes
  - Changing the profile of the web-page also re-locates the sections in the web-page
  - This is achieved by ‘dragging’ the edges of the web-page to a desired size
  - The height of the web-page will also change – as the width is reduced the height also changes

HTML Layout using a CSS framework

- What is a CSS framework?
  - CSS frameworks provide a basic structure for designing consistent solutions
  - Addresses common recurring issues across front end web development.
  - CSS frameworks provide generic functionality which can be overridden to create domain-specific applications
  - There are many existing CSS Frameworks that offer Responsive Design.
- A typical CSS framework is the W3.CSS framework
- The W3.CSS enables development of web-sites across a range of devices and interfaces

W3.CSS Examples

- W3.CSS cover the majority of HTML elements – examples include
  - W3.CSS Navigation
  - W3.CSS Sidebar
  - W3.CSS Panels
  - W3.CSS Text
  - W3.CSS Round
  - W3.CSS Buttons
  - W3.CSS Tables
  - W3.CSS Lists
- The following slides demonstrate the implementation of these elements using W3.CSS

Multimedia

- Multimedia forms a central role in modern Internet applications
- In the early days of the Internet users were restricted to viewing text and static images
- In the current Internet users may
  - Access multimedia (video and sound files)
  - Interact with web sites and add / change / delete content
    - These developments are often termed Web 2.0
- Multimedia has many formats as it can be almost anything you can hear or see.
  - Examples include sound, videos, and animations, etc
- In this course we will limit our study to adding and viewing multimedia files

Web Page Design

12/10/2020

Feedback

- When designing and developing a web-page (or web-site) we have in previous sessions introduced
  - Usability and security considerations and the 'trade-off'
- A further important consideration is 'feedback' to users
- We have seen in this session a CSS float example where a navigation bar was created with links to other web-pages
  - In this example we saw that when the cursor *hovered* over the home link (*mousover*) the colour changed from red to black
  - This is an example of feedback which informs the user of an action
- When developing web-pages and web-sites user feedback is important

Scripting Languages

Scripting Languages

- A *scripting language* is a programming language that supports *scripts*
  - *Scripts* are (generally) small programs written to function in *'real-time'*
  - *Scripts* are (generally) written to control and automate the execution of tasks and events
- *Scripting* languages (such as *JavaScript* and *php*) are *interpreted*
- High-level programming languages (such as 'Java' and 'C') are *compiled*
- A *scripting language* enables (generally) small programs to be combined into more complex programs

Scripting Languages

- JavaScript (client-side) and PHP (server side) are frequently used together
    - PHP can naturally work with MySQL whereas JavaScript can not
- Environments to which a *scripting* can be applied include:
    - Software applications
    - Web-systems (web-pages and web-sites) running in a web-browser
    - Shell programming (in a Unix – or Linux operating system)
    - Embedded systems
    - Online games

Object-Orientated Programming

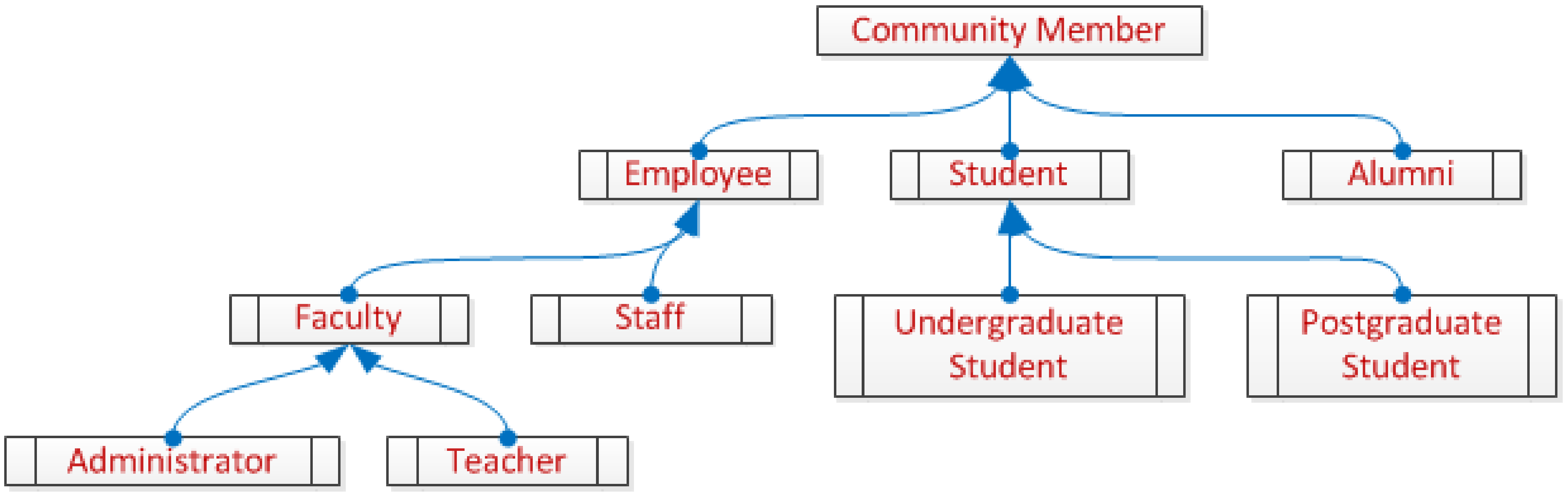
Object-Oriented Programming

- Object-oriented programming (OOP) is a programming paradigm based on the concept of *objects*
  - Objects hold data in the form of attributes
  - Objects hold program code in the form methods
- A feature of an object is the capability for the methods to access and modify the data and properties of the object with which they are associated
  - Objects have a notion of **`this`** (or "self")

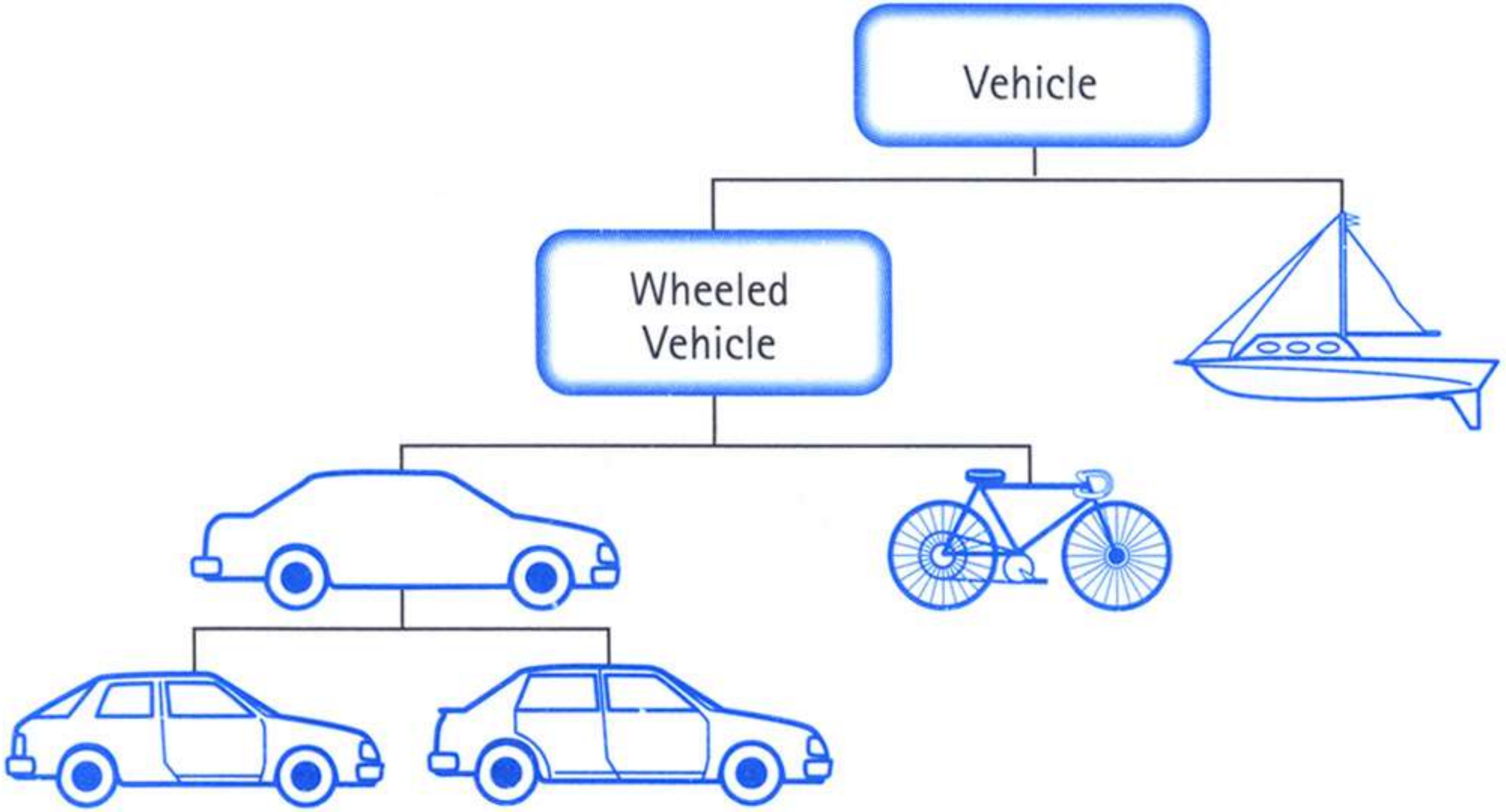
Object-Oriented Programming

- In OOP computer programs are designed around the concept of creating interactions between objects
- In JavaScript there is the concept of inheritance where a *child* (object) inherits the attributes of the *parent* and adds *additional attributes*
  - When an inherited function is executed the value of **`this`** points to the *inheriting* object

Inheritance



Objects – Transport



JavaScript

JavaScript Identifiers

- Identifiers
  - Variables, functions, and label names are JavaScript *identifiers*
  - Identifiers are composed of any number of letters and digits and $ characters
- The first character of an identifier must not be a digit
- The following are legal identifiers:
  - **v**
  - **my_variable_name** //no spaces allowed use underscore (**_**)
  - **V13**
  - **$str**

JavaScript Variables

- In computer programming the aim is to process and manipulate data to achieve a desired result
- A computer program uses variables to hold data
- We have identified two variable types: *local* and *global*
  - In this part of the course we are focusing on *global* variables
  - In later sessions we will introduce *local* variables with local function scope

Variable Typing

- In JavaScript global variables are part of a *global object*
  - This is a good program design feature as it simplifies the code writing
  - It is also a very bad program design feature as variables used in functions can be modified from outside the function (there is no native block scope)
  - As we shall see later in the course we can create *static* variables within functions which have *local* scope
- In JavaScript variables are *untyped*
  - This means that a variable can contain values of any datatype
    - A numerical (n = 10) can be changed to (n = “string”): this is a logic problem
  - Java, C, and C++ has variables that are *strongly typed*

Datatypes

- JavaScript supports three *primitive* data types:
  - numbers
  - booleans (*true* or *false*)
  - strings
- In JavaScript are two *compound* data types
  - objects
  - arrays
- JavaScript defines *specialised* types of objects to represent
  - functions
  - dates
  - regular expressions (not included in this course)

Numbers

- In JavaScript numbers are represented in 64 bit floating point format and makes no distinction between integers and floating point numbers
  - The 64 bit format is the same as a *'double'* in Java
- Numeric literals appear in JavaScript using the usual syntax of digits with an optional decimal point. For example:
  - 1
  - 3.14
  - 0001
  - 6.02e23
- Errors in numeric data processing will result in a value that is either '*not-a-number*' (NaN)

Booleans

- In JavaScript we may need to represent if a statement is *true* or *false*
- The truth of falsity is represented by *Boolean* values
- The Boolean values can measure
  - `Truth` (or) `on`
  - `False` (or) `off`
- Boolean values are measured numerically using an integer value
  - `Truth is [1]`
  - `False by [0]`
- There will be more details for Boolean values in a later session

Literals

- There are two types of *literal*
  - String *literals*
    - In JavaScript string literals appear between single quotes (‘ … ‘) or double quotes (“ … “)
    - There is no difference between the two approaches
    - This is not always the case with other programming languages (e.g., PHP)
  - Object *literals*
    - Are used to specify new objects
    - Objects and object literals will be covered in later sessions

Concatenation

- In JavaScript when two strings are joined
  - We say that the two strings have been "concatenated"
- We may also concatenate strings and variables into a string output
- The following slides show the NetBeans IDE and demonstrate
  - The string concatenation process program code
  - Strings are concatenated into a single string
  - A string and a variable are concatenated into a single string
  - The resulting output strings

Boolean Values

- A Boolean value represents one of two values: **true** or **false**
- Programming requires a data type that can only have one of two values
  - **YES** / **NO**
  - **ON** / **OFF**
  - **TRUE** / **FALSE**
- JavaScript has a **Boolean** data type which can only take the values **true** or **false**
- For example:

• **(10 > 9)**	(returns **true**
• **(10 >= 10)**	(returns **true**)
• **(10 >= 11)**	(returns **false**)
• **(10 < 9)**	(returns **false**)
• **(10 <= 10)**	(returns **true**)

Comparison Operators

- Comparison operators are used in logical statements to determine equality or difference between variables or values
  - The table below explains the comparison operators
  - The following worked examples show the JavaScript and the result

Operator	Description	Comparing	Returns
==	equal to	x == 8	false
		x == 5	true
		x == "5"	true
===	equal value and equal type	x == 8	false
		x == 5	true
		x == "5"	true

Conditional Operators

- Comparisons and Conditions
  - The range of JavaScript comparison and conditional operators many be found in the course resources
  - Below we show some examples:

Operator	Description	Example
==	equal to	if (day == "Monday")
>	greater than	if (salary > 9000)
<	less than	if (age < 18)
>=	greater than or equal to	if(age >= 18)
<=	less than or equal to	if(age <=18)

Logical Operators

- Logical operators are used to determine the logic between variables or values
- Given the table below explains the logical operators

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x == 5 || y == 5) is false
!	not	!(x == y) is true

Conditional (Ternary) Operator

- JavaScript contains a conditional operator that assigns a value to a variable based on some condition
- The syntax

```
variablename = (condition) ? value1:value2
```

- For example

```
var voteable = (age < 18) ? "Too young":"Old
enough";
```

Comparing Different Types

- Comparing data of different types may give unexpected results.
- When comparing a string with a number, JavaScript will convert the string to a number when doing the comparison.
- An empty string converts to 0 – a non-numeric string converts to `NaN` which is always false.
- For example

```
2 < 12 = true
"2" > "12"  = true
```

- When comparing two strings, "2" will be greater than "12" because (alphabetically) 1 is less than 2
- To secure a correct result variables should be converted to the proper type before comparison

Creating Date Objects

- There are two ways to create a Date Object:
  - Construct an empty date object with the new Date() method
  - Construct a date object with parameters
    - new Date(*year, month, day, hours, minutes, seconds, milliseconds*)
    - new Date(*milliseconds*)
- To create a date object and convert it to a string
  - Var d = new Date();
  - Var n = d.toString();
  - Note: This method is automatically called by JavaScript whenever a Date object needs to be displayed as a string

Operators

Operator	Syntax	Example	Definition
addition	+	x + y	Sum of x and y
subtraction	-	x - y	Difference of x and y
multiplication	*	x * y	Product of x and y
division	/	x / y	Quotient of x and y
modulo	%	x % y	Remainder of x / y
exponent	**	x ** y	x to the y power
increment	++	x++	x plus one
decrement	--	x--	x minus one

Expressions and Operators

- A JavaScript expression is formed by combining values which can be any combination of the following:
  - Strings / string literals / variables / object properties / array elements / function invocations
  - Parenthesis (…) can be used to group sub-expressions
    - This can be used to change the default evaluation (processing) order
    - While the JavaScript syntax may be correct the program logic may be incorrect resulting in the wrong answer
- In JavaScript (in all programming languages) there is operator precedence

Operator Precedence Examples

- Consider the following examples:
  - `100 + 50 * 3 = 250`
    - multiplication has precedence over addition
  - `(100 + 50) * 3 = 450`
    - multiplication has precedence over addition (but) parenthesis has precedence over multiplication
  - `100 + 50 / 3 = 116.66666666666667`
    - division has precedence over addition
  - `(100 + 50) / 3 = 50`
    - division has precedence over addition (but) parenthesis has precedence over division
  - `100 + 50 - 3 = 147`

JavaScript Functions

- A function is defined once and may be re-used (i.e., *invoked* (or ‘*called*’) multiple times.
  - Code re-use which improves the code by reducing errors
  - Reduces the need to write a specific operation multiple times
  - A function can be re-used multiple times with different arguments - variable values (to produce different results)

Function Invocation

- When a function is assigned to a property of an *object* it is termed a *method* of that *object*
  - Within the body of the method the keyword *this* refers to the *object*
- Within the body of a function the `arguments [] array` contain the complete set of `arguments` passed to the function
  - The JavaScript code inside the function will execute when "something" *invokes* (or *calls*) the function
- JavaScript is essentially an *event-driven* program, *events* can include:
  - When a user clicks a button in a form (or) when it is *called* from JavaScript code

Self Invoking Functions

- A Function expression(s) can be made *"self-invoking“*
- A *"self-invoking“* expression is *invoked* automatically without being called
- Function expressions will execute automatically if the expression is followed by ()
- self-invoking can not be used for a function declaration.
- Parentheses () must be added around the function to indicate that it is a function expression

Self Invoking Function Example

```
<!DOCTYPE html>
<html>
<body>
<p>Functions can be invoked automatically without being called:</p>
<p id="demo"></p>
<script>
(function () {
    document.getElementById("demo").innerHTML = “self invoked";
})();
</script>
</body>
</html>
```

Function Return

- JavaScript functions receive input parameters and process the inputs to produce an output (the result)
- To process the inputs the body of the function will implement variables and statements to compute the result
- If the function was invoked from a *statement*, JavaScript will "*return*" to execute the code after the invoking statement
- The result is returned to the caller by the return statement
  - When JavaScript reaches a *return statement* the function will stop executing
  - Functions often compute a *return value* the return value is "returned" back to the "caller":
- **A function can return only one result**

Function as Variables

- Functions can be used the same way as variables are used
  - For example in all types of:
    - Formulas
    - Assignments
    - calculations
- In the code ( ) operator Invokes the function
- Accessing a function without () will return the function definition instead of the function result
- The worked examples shown have demonstrated
  - The use of a function() as a variable
  - The assignment of a function() return result to a variable
  - The output to a web-browser with string operators

if … else

- The following **`if…else`** conditional statements execute a statement ***IFF*** an expression is **`true`**

```
if( expression )
    statement
```

- The following **`if…else`** conditional statements execute a statement IF an expression is **`false`**

```
if(expression)
        statement 1
else
        statement 2
```

if … else … else

- Any else clause may be combined with a nested **`if…else`** statement to produce an **`else…if`** statement as follows:

```
if(expression)
        statement 1
else
        statement 2
else
        statement 3
```

- The following slides show working examples of the conditional selection operators

If / if…else / if…else if… else Statements

- In all the examples block curly brackets {…} have been used
- For if statements
  - The {…} may be omitted
- For if…else and if…else..else blocks
  - The block curly brackets {…} must be used
  - If the block curly brackets {…} are omitted there will be an error
- It is good programming practice to always use {…}
  - This can improve the readability of the program code
  - Reduce the possible errors in writing the JavaScript code

The Switch Case Syntax

```
switch(expression) {
    case n:
        code block
        break;
    case n:
        code block
        break;
    default:
        code block
}
```

- The *first case* which matches the expression will be executed

The Switch Case **`break`** Operator

- A switch statement may include the **`break`** operator
- When JavaScript reaches a **`break`** keyword
  - It breaks out of the switch block and the code processing
    - proceeds to the next line of JavaScript code or
    - Or returns the result
  - This will stop the execution of more code and case testing inside the block.
- When the ***expression*** has been satisfied the evaluation is complete
  - There is no need for more switch blocks to be evaluated
- The **`switch`** statement can
  - Output text
  - Call functions and create objects

Iteration Operations (loops)

- There are four types of loop:
  - A `for` loop
    - combines the *initialisation* and *increment expressions* with the loop *conditional* expression
  - A `for…in` loop
    - loops through the properties of a specified object
  - A `while` loop
    - statement is a basic loop which repeatedly executes a *statement* while an *expression* is `true`
  - A `do while` loop
    - Repeatedly executes a statement while an expression is `true`
    - It is similar to the `while` loop except that the *loop condition* appears (and is *tested*) at the bottom of the loop
    - This means that the body of the loop is always executed at least once

`for` loop

- The syntax for a `for` loop is as follows

```
for ( initialise ; test ; update){
    statement
}
```

- The loop:
  - Repeatedly executes the ***statement*** while the ***test*** expression is `true`
  - Evaluates the ***initialise*** expression once before starting the loop run
  - Then evaluates the ***update*** expression at the termination of each iteration

for…in loop

- The syntax for a for…in loop is as follows

```
    for ( variable in object ){
        statement
    }
```

- The for…in loop executes a statement once for each property in an object
- Each time through the for…in loop is assigns the name of the current property to a specified variable
- Some properties of pre-defined JavaScript objects are not enumerated by the for…in loop
- User defined properties are always enumerated
- The following example demonstrates the for…in loop

While loop / Do…While

- The syntax for a **`while`** loop is as follows

```
while ( expression ){
    statement
}
```

- The syntax for a **`do…while`** loop is as follows

```
do{
    statement
}while ( expression );
```

- The following examples show the while and do…while loops

JavaScript Arrays

Why use an Array?

- To choose between an object and an array
  - We need to identify the purpose of each structure
  - JavaScript arrays model the way books store information
  - Objects model the way that newspapers store information
- Arrays are the best approach when a defined order (of information) is the most important factor for organizing the information
  - For example: in a book which has a chapter structure
- Objects are the approach when information is organize based on data labels
  - For example: newspaper pages may be read in a random order

Creating Arrays (1)

- There are 2 methods to create a new array
  - `var cars = ["Saab",  "Volvo",  "BMW"];`
  - `var cars = new Array("Saab", "Volvo", "BMW");`
- The general recommendation is
  - Avoid **`new Array()`**
    - There is no need to use the JavaScript's built-in array constructor **`new`** Array()
    - Use **`[…]`** instead

Creating Arrays(2)

- The following two different statements both create a new empty array named **`points`**:

```
var num = new Array(); //not recommended)
var num = [];          //recommended)
```

- The two different statements both create a new **`num`** array containing 6 numbers:

```
var num = new Array(40, 100, 1, 5, 25, 10);
var num = [40, 100, 1, 5, 25, 10];
```

The *new* keyword

- In a high-level programming language such as Java the **`new`** keyword is used to create a new object – the **`new`** keyword performs the same function in JavaScript
- However - In JavaScript the **`new`** keyword only complicates the code
  - It can produce some unexpected results such as:

    ```
    var points = new Array(40, 100);
    ```

    - Creates an array with two elements (40 and 100)
  - Removing one of the elements?

    ```
    var points = new Array(40);
    ```

    - Creates an array with 40 undefined elements

Combining Arrays and Objects

- In JavaScript strings may be stored and processed in arrays and objects
- Other basic data types (numbers and Booleans) may be stored and processed in arrays and objects
- There are other available options:
  - Arrays may be used within objects
  - Objects may be used within arrays
  - Arrays may be used within other arrays
  - Objects may be used within other objects
- As we can have seen
  - While the basic concepts of objects and arrays are very simple
  - Using the available options and combination of options is a JavaScript program can be very complex
  - In ‘real-world’ JavaScript programs you will almost always need a combination of the two to store your data in a scalable way

Arrays

- An array is a linear allocation of memory with numerical indexes for elements accessed using integers [0…n]
- Arrays can be very fast data structures
  - High-level programming languages (e.g., Java) implement arrays and array methods
  - In high-level programming languages arrays and variables are strongly typed (e.g., int / float / double / string / etc

JavaScript Arrays

- While we refer to *arrays* JavaScript does not incorporate arrays
  - For example: in JavaScript variables and arrays are NOT typed
- In JavaScript
  - An array is a special type of **`object`**
  - JavaScript automatically creates an **`associative array`** for each **`object`**
  - Note: this syntax is different in PHP!
- When investigating arrays
  - The **`typeof`** operator in JavaScript returns **`object`** for arrays

JavaScript Arrays

- In JavaScript *arrays* are *objects*
    - A specialized type of object with a length property and array methods
    - The array methods are useful when working with array elements
- Arbitrary data values are associated with arbitrary names
- For example:

```
myArray[“x”] = 1;
myArray[“y”] = 2;
myArray[“y”] = “Philip”;
```

Associative Arrays

- Many programming languages support arrays with named indexes
- Arrays with named indexes are called *associative arrays* (or hashes)
- JavaScript does not support arrays with *named indexes*
  - In JavaScript arrays always use numbered indexes
- For example

```
var person = []; //a new empty array
person[0] = "John";
person[1] = "Doe";
person[2] = 46;
var x = person.length;  //person.length will return 3
var y = person[0];      // person[0] will return John
```

JavaScript Arrays and Named Indexes

- If named indexing is used JavaScript will redefine the array as a standard object
  - The result of the redefinition will be that some array methods and properties will produce incorrect results
- For example

```
var person = [];
person["firstName"] = "John";
person["lastName"] = "Doe";
person["age"] = 46;
var x = person.length;  //person.length will return 0
var y = person[0];      //person.length will return undefined
```

JavaScript Arrays

- JavaScript arrays can hold:
  - Primitive data types
  - Functions
  - Objects
- JavaScript variables and arrays are **NOT** typed
  - JavaScript arrays can hold different data types:

```
myArray[0] = Date.now;
myArray[1] = myFunction;
myArray[2] = myCars;
myArray[3] = "Philip";
```

JavaScript Indexing

- In designing a JavaScript program
  - There is a choice to be made
    - When to Use Arrays
    - When to use Objects
- The factors to consider are:
  - JavaScript does NOT support associative arrays
  - You should use objects when you want the element names to be strings (text)
  - You should use arrays when you want the element names to be numbers

JavaScript Methods

- In considering JavaScript arrays we have introduced JavaScript:
  - Operators
  - Operands
  - Properties
  - Methods
  - Functions
- Operators / methods / functions can be used to manipulate arrays and the array data

Array Sort

- Sorting the elements of a JavaScript array is a frequent operation
- Arrays can be sorted in a number of ways
  - Numeric sort
  - Object Array sort
  - Descending sort
  - Ascending sort
  - Sorting an Array in Random Order
  - Find the Highest (or Lowest) Array [element] Value

Generating Random Numbers

- We have covered math methods in week 5
- A list of math methods may be found in the course resources
- A very useful and often used method is used to create a random number
    - **`Math.random();`** //returns a 64 bit random number
    - In fact JavaScript produces a ***pseudorandom*** number
- A *pseudorandom number generator* (as used in JavaScript it is also termed a *deterministic random bit generator*)
    - Is an algorithm for generating a sequence of numbers whose properties ***approximate*** to the properties of sequences of ***genuine random numbers***

Generating Random Integers

- To create random integers

  `Math.ceil(x)` and `Math.floor(x)`

  ```
  Math.ceil(4.4);    //returns 5 – rounded up)
  Math.floor(4.7);   //returns 4 – rounded down)
  ```

- If a random number within a certain range is required

  ```
  Math.floor(Math.random() * 10) + 1;
  ```

  - (returns a random integer in the range 1 … 10)

  ```
  Math.floor(Math.random() * 10);
  ```

  - Returns a random integer in the range 0 … 9

Scope

- Scope in a computer program controls the visibility and lifetime of variables
- In many high-level languages
  - Variables defined within a block can be released when execution of the block is finished
  - Unfortunately JavaScript does not include block scope
    - even thought the syntax suggests it does
- JavaScript does include function scope
  - This means that parameters and variables defined within a function are not visible outside the function
  - A variable defined anywhere within a function is visible everywhere within the function
- The scope is an important consideration in computer program design

Global and Local Scope

- In JavaScript there are two levels of scope:
  - Global scope
  - Local scope
- Any variable declared outside of a function has global scope
  - Global variables are accessible from anywhere in JavaScript code
  - All scripts and functions in a web-page can access global variables
- A function variable has function scope
  - Variables declared within a function has function scope
  - A variable declared within that function is only accessible from that function and any nested functions
  - Function arguments (parameters) work as local variables inside functions

Why is Variable Scope Important?

- As we have seen JavaScript is based on *a global object*
  - This can be a problem for JavaScript programming
  - Programmers must know how and when variable is modified
- Why is this important?
  - Unrestricted access to a variable (a feature of the *global object*) may be required and useful
  - However: uncontrolled changes to the variable (data) can be a significant problem
  - Uncontrolled (*global*) variables may result in unreported errors
    - Such errors are not syntax errors but logic errors

Runtime Errors

Catching Runtime Errors

- When executing JavaScript code different runtime errors can occur
- Errors include
  - Invalid input by a user
  - There are other (possibly unforeseen) runtime errors
- JavaScript provides the throw and try and catch methods to catch errors
  - The **`try`** statement tests a block of code for runtime errors
  - The **`catch`** statement handles the runtime error
  - The **`throw`** statement creates custom errors
  - The **`finally`** statement execute JavaScript program code after the try and catch methods regardless of the result

Potential Runtime Errors

- JavaScript runtime error checking methods will `catch` a range of potential errors including:
  - HTML Validation Errors
  - Range Errors
  - Reference Errors
  - Syntax Errors (note: this will not find logic errors)
  - Type Errors
  - URI (Uniform Resource Identifier) Errors
- A list of the possible errors with the syntax and examples may be found at
  - https://www.w3schools.com/js/js_errors.asp
- Examples of the syntax and examples are shown in the following slides

Non-Standard Error Object Properties

- Mozilla and Microsoft defines some non-standard error object properties:
  - *fileName* (Mozilla)
  - *lineNumber* (Mozilla)
  - *columnNumber* (Mozilla)
  - *stack* (Mozilla)
  - *description* (Microsoft)
  - *number* (Microsoft)
- Do not use these properties in public web sites as they will not work in all browsers

Try…catch block

- The **`try`** statement defines a block of code to be tested for errors while it is being executed (at ***runtime***)
- The **`catch`** statement defines a block of code to be executed if an error occurs in the **`try`** block.
- The JavaScript statements **`try`** and **`catch`** come in pairs

```
try {
    Block of code to try
}
catch(err) {
    Block of code to handle errors
}
```

try…catch…finally block

- The syntax for a **`try…catch..finally`** block

```
    try {
          statements
     }
    catch ( arguments ) {
    }
    finally {
          statements
    }
```

Server-side Programming Hypertext Preprocessor (PHP)

The Advantages of Server-Side Scripting

- In order for client side scripting to work at all
  - Browsers need plugins and scripting technologies
  - Browsers may not support JavaScript or it may be disabled
- Server-side scripting using scripting languages such as PHP
  - Server-side scripting often reduces the loading time for web pages
- Server-side scripting enhances security because:
  - Scripting takes place on the server the script itself is not sent to the browser (thus assisting in the prevention of copying / cloning/ or being scrutinised)
  - Server-side scripting offers greater protection for user privacy and is the preferred option for e-commerce / membership applications / and social media sites

PHP and Server-Side Server Systems

- Unlike JavaScript PHP is a server-side language
- PHP provides greatly enhanced functionality
  - As such it is similar to a fully functional high-level programming language such as Java or C++
- For example
  - PHP provides the capability to interact with database systems
  - Database systems and PHP will be introduced in week 8
    - In this course the focus is on MySQL
- Server-side systems
  - Provide enhanced security in validation and input checking
  - Provide less interactivity (by design)

JavaScript and PHP

- The basic approach to programming using PHP and JavaScript is very similar
- There are many functions in PHP that are not available in JavaScript
- The syntax of PHP is similar to JavaScript
- There are syntax differences – the key differences are:
  - Variables in PHP are prefixed with a **$** sign
  - Local variables must be declared in JavaScript (**not** required in PHP)
  - Different opening and closing **<tags>** are used
  - String concatenation in JavaScript use the (**+**) sign / PHP uses a period (**.**)
- In the following slides the language basics of PHP and JavaScript are compared

Syntax Comparison (1)

Language Component	PHP	JavaScript
Opening and closing tags	<?php … ?>	<script> … </script>
Block statement	{…}	{…}
Multi-line comment	/* comment*/	/* comment*/
Single-line comment	// comment	// comment
Constant declaration	define (“a”, 1);	cons a = 1;
Variable declaration	Not required	Required for local variables (var a = 1;)
Variable assignment	$a = 0;	a = 0;
Assignment shortcut style	$a += 5;	a += 5;
Variable typing	at runtime	at runtime
Statement terminator	;	; (or the end of a line)
Equality type and value testing	Double-equals, ==	Double-equals, ===

Syntax Comparison (2)

Language Component	PHP	JavaScript
Equality type and value testing	Triple-equals, ===	Triple-equals, ===
Inequality testing	!=	!=
Strings	“string” ‘String’	“string” ‘String’
String constants	\n and \t	\n and \t
String concatenation	$a = $b . $c;	the (+) operator
Boolean values	true / false	true / false
Logical AND	&&	&&
Logical OR	||	||
Logical NOT	!	!

String Literals and Database

- A common task in a PHP script is to access and operate on data in a *Relational Database Management System* (or RDMS)
- The following example Is used to construct a PHP statement in the *Structured Query Language* (or SQL)

```
    $query = "SELECT max(order_id)
              FROM orders
              WHERE cust_id = $custID";
```

  - The variable (**`$query`**) is used to output the result

  The example shows the use of a *structured string* to produce a database search query – we will introduce RDMS (MySQL) in week 8

`printf` and `sprintf` operator types

Operator Type	Function Description
%%	A literal percent character
%b	An integer formatted as a binary character (the number 5 in the binary system is: **101**)
%c	An integer formatted as an ASCII character
%d	An integer formatted as a signed decimal number (can hold numbers: **1** and **-1**)
%u	An integer formatted as an unsigned decimal number (can only hold numbers: **1**)
%o	An integer formatted as an octal number (the base 8 number system)
%x (or) **%X**	An integer formatted as a hexadecimal number (the base 16 number system / using lowercase and uppercase letters)
%f	A float formatted with a specified number of decimal places
%s	A string

Database Technologies MySQL and SQL

Database Basics

Database Basics

- A database
  - Is an organized collection of data (stored and accessed electronically)
- A database-management system (DBMS) Is a software application that enables
  - Interaction between a database and users (including other web-systems)
- A general-purpose DBMS enables
  - The definition, creation, querying, update, and administration of databases
- A database is generally stored in a DBMS-specific format which is not portable and typically uses SQL

Relational Database Management Systems

- A relational database management systems (RDMS)
  - Is a DMS based on the *relational model* (RM) which in turn is based on *relational calculus* (addressed in another course)
  - The RM is an approach to managing data using a structure and language consistent with *first-order predicate logic*
  - In a RM all data is represented in terms of *tuples* which are *grouped into relations*
- RDMS operate using the structured query language (SQL)
- RDMS has a specific terminology

What is a Tuple?

- In the context of a relational database:
  - A tuple is one record (or one row in a database table)
- The information in a database:
  - Can be thought of as a spreadsheet
  - With columns (known as fields or attributes) representing different types of information
  - For example: a person table may have data such as:
    - first_Name, surname, Initials, email, home telephone, work telephone, etc
- Tuples representing all the information from each field are associated with a single record

Database Records

Winery ID	Winery name	Address	Region ID
1	Moss Brothers	Smith Road	3
2	Hardy Brothers	Jones Street	1
3	Penfolds	Artherton Road	1
4	Lindermans	Smith Avenue	2
5	Orlando	Jones Street	1

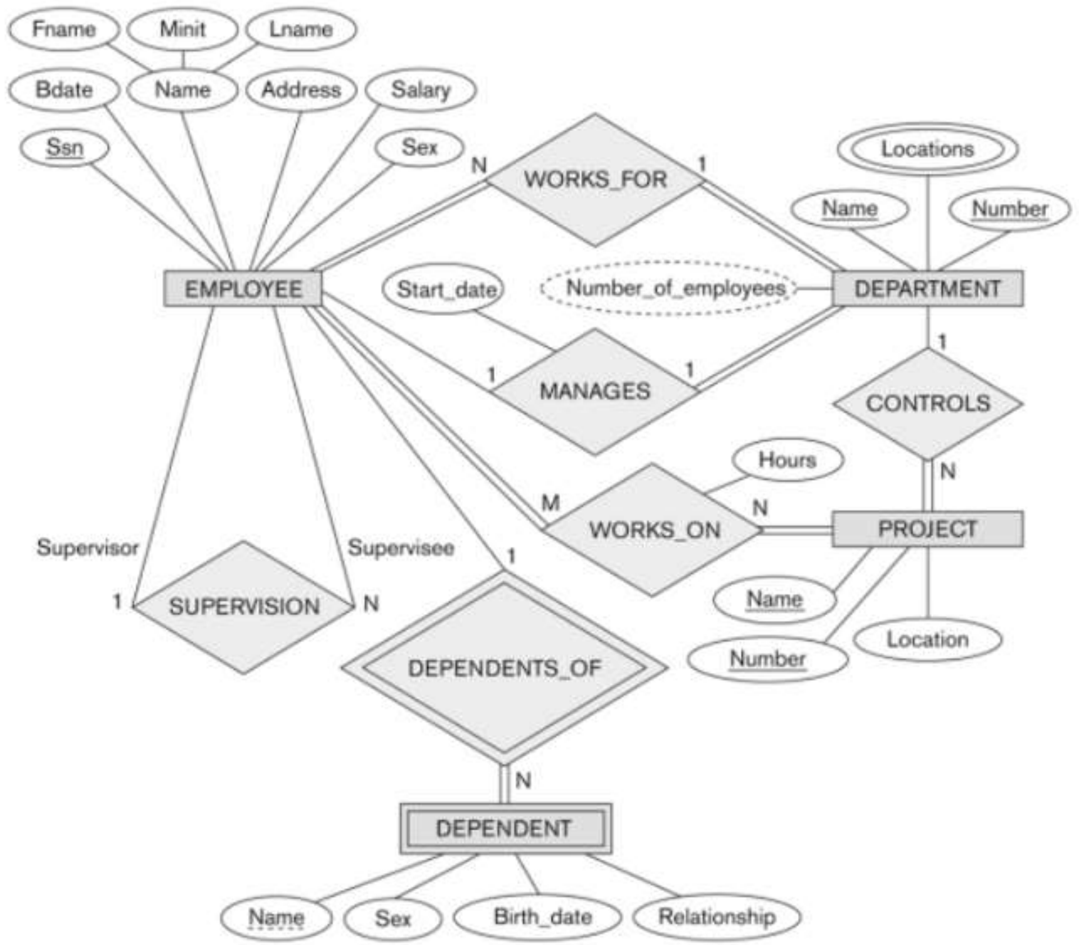
RDMS Terminology

- There is a specific terminology used for databases:
  - *Database*: a repository to store data
  - Table: a part of a database that stores data related to an entity – e.g., a customer in an on-line web-site
  - *Attribute*: for the columns in a table – all rows have the same attributes
  - *Row*: (a data record) contains values for each attribute
  - *Relational model*: a formal model that uses the database / tables / attributes to store and manage data and their relationships
  - *Relational Calculus*: is a non procedural query language using mathematical predicate calculus (instead of algebra) and it provides:
    - The description about the *query* to get the *result* (whereas) relational algebra provides the *method* to get the *result*

RDMS Terminology

- *RDMS*: a software application to manage date in a database based on the RM
  - (MySQL server is a RDMS
- *SQL*: (see later slide)
- *Constraints*: restrictions or limitations on tables and attributes)
- *Primary key*: an attribute (a row) is a unique identifier for a table
- *Index*: a data structure used for fast access to rows in a table)
- *Entity-relationship (ER) modelling*
  - This term relates to a technique used to describe 'real-world' data (a conceptual model) in terms of entities / attributes / relationships

Entity Relationship Diagram

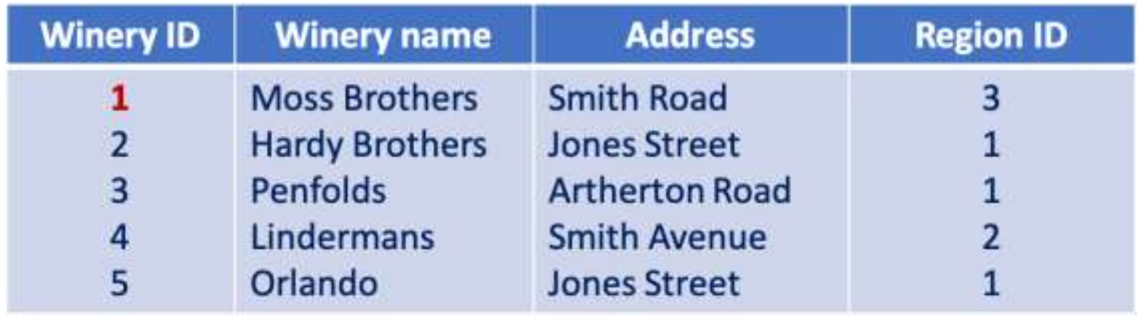


Normalisation

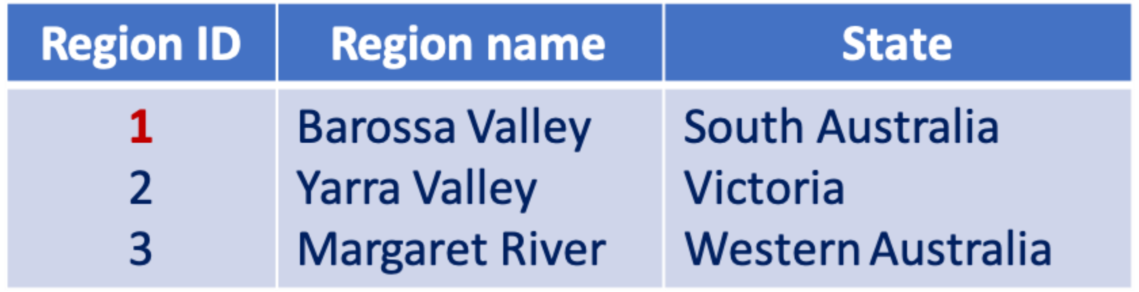
- Normalisation
  - A correctly designed database is created from the ER model
  - The initial data is generally set in a single table (all the data)
  - The database is then *defragmented* (the process of *normalisation* – there are 5 levels of normalisation (of which 3 levels are considered sufficient in most cases)
    - First normal form / second normal form / third normal form
- The process of normalization is designed to remove duplication of data records in a database
  - Record duplication (of records and data) causes issues in maintaining and updating a database with potential errors

Database Tables

- Shown are two related database tables
  - The *Winery Table*
  - The *Region Table*
- The two tables form part of a much larger database
- Each table will have a unique identifier (*primary key*)
  - In the *Winery Table* it may be the *Winery ID*
  - In the *Region Table* it may be the *Region ID*



Winery ID	Winery name	Address	Region ID
1	Moss Brothers	Smith Road	3
2	Hardy Brothers	Jones Street	1
3	Penfolds	Artherton Road	1
4	Lindermans	Smith Avenue	2
5	Orlando	Jones Street	1



Region ID	Region name	State
1	Barossa Valley	South Australia
2	Yarra Valley	Victoria
3	Margaret River	Western Australia

The Structured Query Language

- SQL is an abbreviation of *structured query language* and is pronounced either *see-kwell* (or) as separate letters *SQL*
    - The original version called *SEQUEL* (structured English query language) was designed by an IBM research center in the mid 1970's
- *SQL* is a standardised query language for
    - Accessing a *RDMS* (and)
    - Requesting / adding / deleting / and requesting (querying / searching) for information from a database

MySQL

- The following slide shows the creation of a database table in MySQL server
- The table is populated with records (data)
- In this example the table is created using the command line
- We will use the NetBeans IDE and the XAMPP server (which includes the MySQL server)
- Shown is the SQL statements and the resulting table stored in the MySQL server

Structured Query Language SQL Statements

12/10/2020

SQL Statements

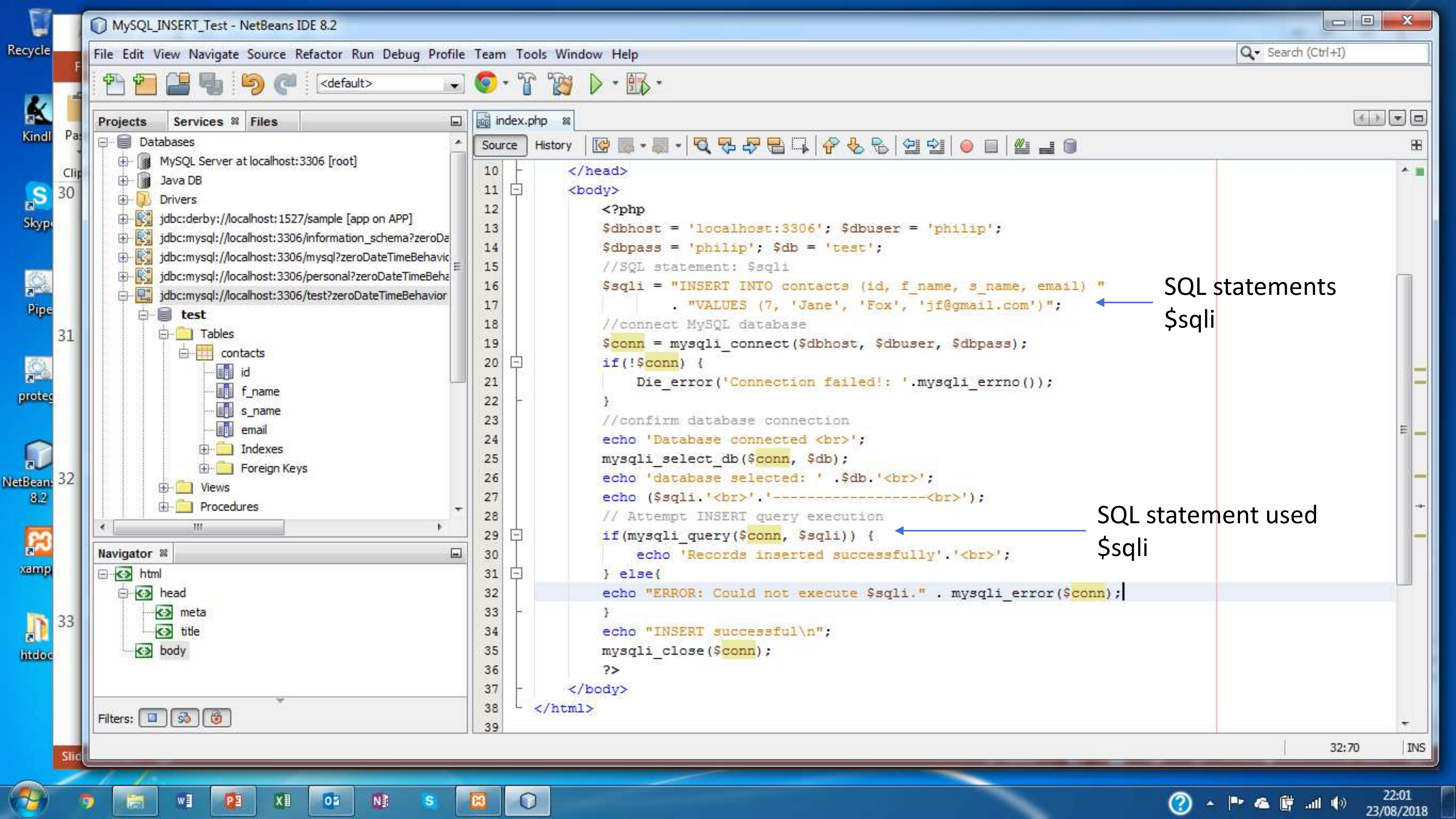
- SQL statements:
  - INSERT INTO customer VALUES (1, 'Williams', 'Lucy', 'E', '2002-07-02’);
  - INSERT INTO customer VALUES (2, 'JOnes', 'Thomas', 'R', '1993-05-23’);
  - INSERT INTO customer VALUES (3, 'Thomas', 'Philip', 'M', '1997-11-17’);
  - SELECT * FROM customer LIMIT 100;
  - SELECT * FROM customer WHERE surname='Thomas’
  - UPDATE customer SET surname = 'Jones' WHERE cust_id = 2;
- These SQL statement demonstrate the INSERT / SELECT / UPDATE statements
  - The SELECT * FROM customer selects all the records in the table
- The other SQL statements follow this pattern
- Examples of SQL statements may be found in the course resources

Advanced MySQL

- We have seen the basic (and most often used) functions and techniques to access a MySQL database including user defined variables
- MySQL provides advanced *SELECT* and *other functions* to mange the database tables and data
- In RBMS the results of a query is termed a 'view'
  - In a SELECT query we are generating a view of selected data
- A list of MySQL functions and operators can be found in:
  - The course resources book (Sams Teach Yourself PHP, MySQL & JavaScript All in One SIXTH EDITION)
  - We have provided two MySQL tutorial documents in the course resources
    - The tutorials cover every aspect of MySQL including the functions and operators

Advanced MySQL

- MySQL provides advanced functions to work with – for example:
  - Strings / dates and times / Arithmetic and comparison operators / mathematical functions / string concatenation CONCAT() / Numeric data types (signed and unsigned) / Advanced join types (e.g., SELECT from 2 or more tables in a database)
  - Using aliases
  - Nested queries
  - Other advanced clauses including:
    - IN / EXISTS / ROLLUP /GROUP BY / HAVING / IGNORE / REPLACE / LIKE / DELETE / TRIM / etc
  - Automatic (scheduled) querying
  - MySQL server and MySQL database management security and access
  - Database and MySQL server reports

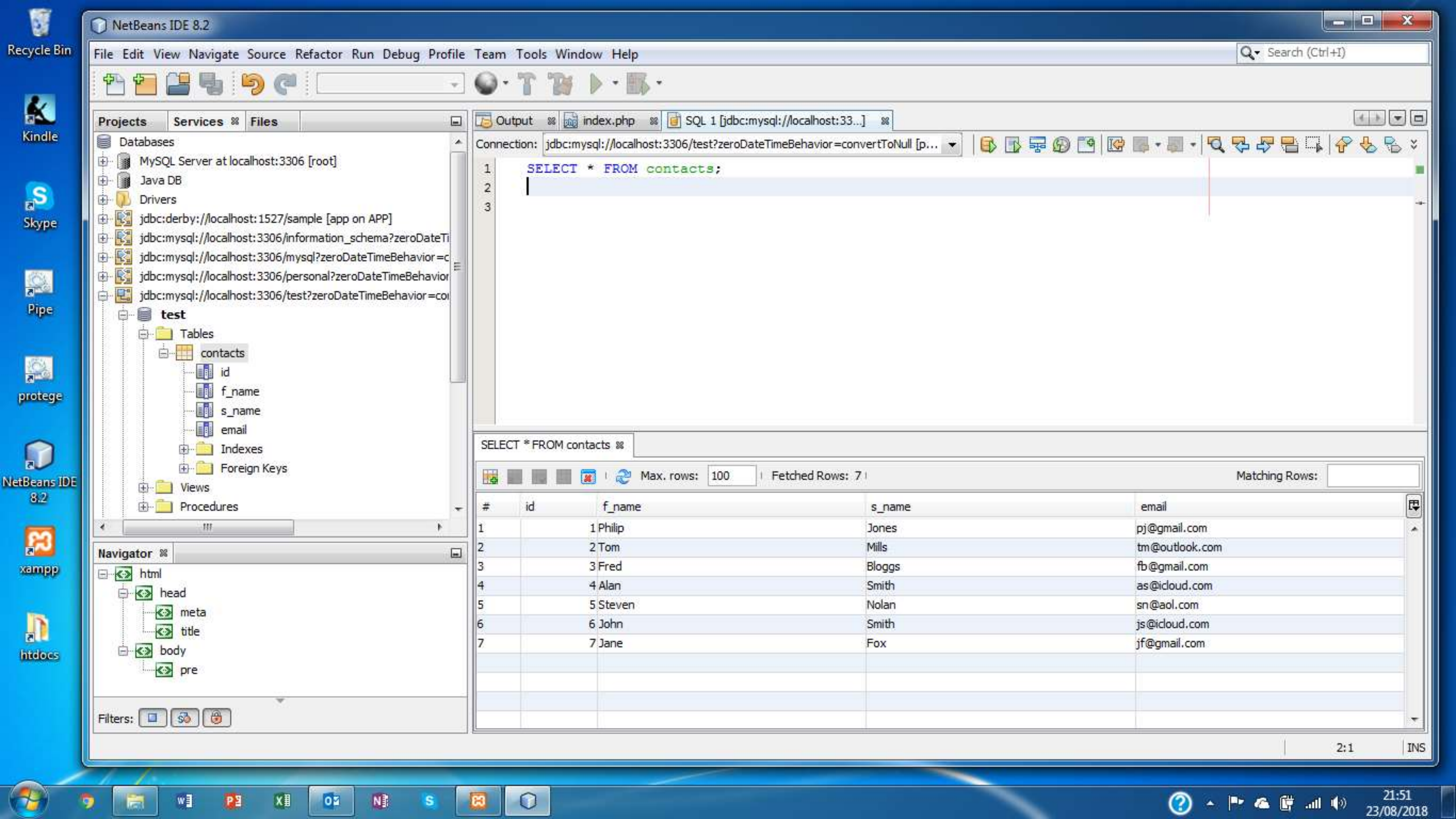


```
    </head>
    <body>
        <?php
        $dbhost = 'localhost:3306'; $dbuser = 'philip';
        $dbpass = 'philip'; $db = 'test';
        //SQL statement: $sqli
        $sqli = "INSERT INTO contacts (id, f_name, s_name, email) "
                . "VALUES (7, 'Jane', 'Fox', 'jf@gmail.com')";
        //connect MySQL database
        $conn = mysqli_connect($dbhost, $dbuser, $dbpass);
        if(!$conn) {
            Die_error('Connection failed!: '.mysqli_errno());
        }
        //confirm database connection
        echo 'Database connected <br>';
        mysqli_select_db($conn, $db);
        echo 'database selected: ' .$db.'<br>';
        echo ($sqli.'<br>'.'------------------<br>');
        // Attempt INSERT query execution
        if(mysqli_query($conn, $sqli)) {
            echo 'Records inserted successfully'.'<br>';
        } else{
        echo "ERROR: Could not execute $sqli." . mysqli_error($conn);
        }
        echo "INSERT successful\n";
        mysqli_close($conn);
        ?>
    </body>
</html>
```

SQL statements
$sqli

SQL statement used
$sqli

MySQL_INSERT_Test - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Services Files
Databases
MySQL Server at localhost:3306 [root]
Java DB
Drivers
jdbc:derby://localhost:1527/sample [app on APP]
jdbc:mysql://localhost:3306/information_schema?zeroDa
jdbc:mysql://localhost:3306/mysql?zeroDateTimeBehavio
jdbc:mysql://localhost:3306/personal?zeroDateTimeBeha
jdbc:mysql://localhost:3306/test?zeroDateTimeBehavior
test
Tables
contacts
id
f_name
s_name
email
Indexes
Foreign Keys
Views
Procedures
Navigator
html
head
meta
title
body
index.php
Source History
32:70 INS
22:01 23/08/2018


NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects
Services
Files
Databases
MySQL Server at localhost:3306 [root]
Java DB
Drivers
jdbc:derby://localhost:1527/sample [app on APP]
jdbc:mysql://localhost:3306/information_schema?zeroDateTi
jdbc:mysql://localhost:3306/mysql?zeroDateTimeBehavior=c
jdbc:mysql://localhost:3306/personal?zeroDateTimeBehavior
jdbc:mysql://localhost:3306/test?zeroDateTimeBehavior=co
test
Tables
contacts
id
f_name
s_name
email
Indexes
Foreign Keys
Views
Procedures
Navigator
html
head
meta
title
body
pre
Filters:
Output
index.php
SQL 1 [jdbc:mysql://localhost:33...]
Connection: jdbc:mysql://localhost:3306/test?zeroDateTimeBehavior=convertToNull [p...
SELECT * FROM contacts
Max. rows: 100
Fetched Rows: 7
Matching Rows:
2:1
INS
21:51
23/08/2018

```
SELECT * FROM contacts;
```

#	id	f_name	s_name	email
1	1	Philip	Jones	pj@gmail.com
2	2	Tom	Mills	tm@outlook.com
3	3	Fred	Bloggs	fb@gmail.com
4	4	Alan	Smith	as@icloud.com
5	5	Steven	Nolan	sn@aol.com
6	6	John	Smith	js@icloud.com
7	7	Jane	Fox	jf@gmail.com

Improve the PHP Script

- The PHP script as used is a simple implementation of a database connection and access to the ‘test’ database with the results sent to a web browser
- In practice the script would be improved by:
  - Using **try…catch…finally** blocks
  - These would be applied to (at least) lines 23 to 25 and 31 to 33 (see slide 30
- The motivation for using **try…catch…finally** blocks is:
  - To catch errors in connecting to and accessing the database
  - To catch attempted criminal access to the database
  - To catch any errors in user input or structured SQL statements

Program Design

Program Design

- The design process for a computer program may be as follows:
    1. Identify and document the requirements specification
        1. This will set out the required *input* (data) and the required *output* (results)
    2. Create a block layout showing the program algorithm structure
        1. This will set out the sequence of operations (sequential / selection / iteration)
        2. Write pseudo program code describing the algorithm
    3. Plot the path of the data through the program
    4. Design a test plan to:
        1. Test and verify if the data values throughout the program are correct
            1. Test to check if the result is consistent with the required output
    5. Testing may use both "black-box" and "white-box" testing