# CSS Optimizations:

## CSS Image Sprites

Image Sprites

- An image sprite is a collection of images put into a single image.
- A web page with many images can take a long time to load and generates multiple server requests.
- Using image sprites will reduce the number of server requests and save bandwidth.

Index.html

```html
<html>
   <head>
      <title>CSS Image Sprites</title>
      <link href="style.css" type="text/css" rel="stylesheet" />
   </head>
   <body>
      <ul class="social-media-icons">
         <li><a class="twitter" href="https://twitter.com/"></a></li>
         <li><a class="facebook" href="https://www.facebook.com/"></a></li>
         <li><a class="pinterest" href="https://www.pinterest.com/"></a></li>
      </ul>
   </body>
</html>
```

style.css

```css
/* Resets default styling */
.social-media-icons {
   list-style: none;
   margin 0;
   padding: 0;
}
```

```css
.social-media-icons li {
    display: inline-block;
}

.social-media-icons a {
    background-image: url(./assets/img/social-icons.png);
    background-image: url(./assets/img/social-icons.svg), none;
    background-repeat: no-repeat;
    display: block;
    height: 58px;
    width: 50px;
}

/* Move the background to the right position for each social network */
.twitter {
    background-position: 0 0;
}

.facebook {
    background-position: -53px 0;
}

.pinterest {
    background-position: -159px 0;
}
```

## WebP Vs PNG Vs Jpeg:

| Feature | WebP | PNG | JPEG |
|---|---|---|---|
| File Size | Smallest (lossy & lossless) | Larger (lossless) | Smaller (lossy) |
| Compression | Lossy and Lossless | Lossless | Lossy |
| Image Quality | Maintains high quality (lossy & lossless) | Maintains perfect quality | Loses some quality (gradual degradation) |
| Transparency | Supported | Supported | Not Supported |
| Animation | Supported (WebP animation format) | Not Supported | Not Supported |
| Best Use Cases | Photos, illustrations, icons (lossy & lossless) | Graphics, logos, screenshots, images with text (lossless) | Photographs, portraits, complex images (lossy) |
| Browser Support | Modern browsers (good) | All browsers | All browsers |

- WebP generally offers the best balance of file size and image quality.
- PNG is ideal for images with sharp details and transparency.
- JPEG is suitable for photos where a small loss in quality is acceptable for a significant reduction in file size

**Lossless Compression:**

- Preserves all the original data in a compressed format.
- Achieves smaller file sizes by identifying and removing redundant or unnecessary data without affecting the core information.

**Lossy Compression:**

- Sacrifices some of the original data to achieve a significantly smaller file size.
- Focuses on identifying and discarding data that is less important or imperceptible to the human eye/ear.

- This introduces some quality loss in the decompressed file compared to the original.
- Like converting a high-resolution image to a lower resolution one - you lose some detail, but the image is still recognizable

## CSS Expressions

- CSS expressions degrade rendering performance; replacing them with alternatives will improve browser rendering for end users.
- Application of CSS rules will depend on JavaScript execution in expression. Which will cause delay
- **Performance Issues:** The browser needs to constantly re-evaluate the expression whenever the window is resized. This can cause performance issues, especially on slower devices or with frequent resizes.
- **Limited Browser Support:** Modern browsers no longer support CSS expressions. While this specific code might work in older versions of Internet Explorer, it won't work in most current browsers.

css_expression.html

```
<!DOCTYPE html>
<html>
<head>
<title> CSS Expression</title>
<style>
 #text-size {
   font-size: expression((document.body.clientWidth / 5) + "px");
 }
</style>
</head>
<body>
 <p id="text-size">This text size adjusts based on window width (bad
practice).</p>
</body>
</html>
```

css_expression.html

```
<head>
<title>Better Example: Using Media Queries</title>
<style>
 #text-size {
   font-size: 1.2rem;  /* Set a base font size */
 }

 @media (min-width: 768px) {
   #text-size {
     font-size: 1.5rem;  /* Adjust for larger screens */
   }
 }
</style>
</head>
<body>
 <p id="text-size">This text size adjusts based on window width (better
approach).</p>
</body>
</html>
```

## Preload Placeholders

- preload :tells the browser to download and cache a resource (like a script or a stylesheet) as soon as possible.It's helpful when you need that resource a few seconds after loading the page, and you want to speed it up.
- prefetch :asks the browser to download and cache a resource (like, a script or a stylesheet) in the background.
- The download happens with a low priority, so it doesn't interfere with more important resources. It's helpful when you know you'll need that resource on a subsequent page, and you want to cache it ahead of time.

placeholder.html

```
<!doctype html>
<html lang="en">
 <head>
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <title>JS and CSS preload example</title>

   <link rel="preload" as="stylesheet" href="style.css" as="style">
   <link rel="preload" as="script" href="main.js" as="script">
   <link rel="prefetch" as="image" type="image/jpg" href="./img/car1.jpg">

   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6h
W+ALEwIH" crossorigin="anonymous">
 </head>
 <body>
   <ul class="nav">
     <li class="nav-item">
       <a class="nav-link active" aria-current="page" href="./placeholder.html">Home</a>
     </li>
     <li class="nav-item">
       <a class="nav-link" href="./cars.html">Cars</a>
     </li>

   </ul>
   <script src="main.js"></script>
   <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7
N6jIeHz" crossorigin="anonymous"></script>
 </body>
</html>
```

Cars.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Document</title>
</head>
<body>
   <img src="./img/car1.jpg" alt="">

</body>
</html>
```

## <picture> Tag

- The <picture> tag gives web developers more flexibility in specifying image resources.
- The most common use of the <picture> element will be for art direction in responsive designs. Instead of having one image that is scaled up or down based on the viewport width, multiple images can be designed to more nicely fill the browser viewport.
- The <picture> element contains two tags: one or more <source> tags and one <img> tag.

index.html

```html
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<h1>The picture element</h1>
<p>Resize the browser window to load different images.</p>
```

```
<picture>
  <source media="(min-width:650px)" srcset="img_pink_flowers.jpg">
  <source media="(min-width:465px)" srcset="img_white_flower.jpg">
  <img src="img_orange_flowers.jpg" alt="Flowers" style="width:auto;">
</picture>
</body>
</html>
```

## Lazy Loading:

- According to the HTTP Archive, images are the most-requested asset type for most websites, and they usually take up more bandwidth than any other resource. At the 90th percentile, sites send over 5 MB of images on desktop and mobile.
- The loading attribute specifies whether a browser should load an image immediately or to defer loading of off-screen images until for example the user scrolls near them.
- Add loading="lazy" only to images which are positioned below the fold

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
     img {
       max-width: 50%;
       height: auto;
     }
  </style>
</head>
<body>
  <h1>Lazy Loading</h1>

  <img src="./img/car1.jpg" loading="lazy">
  <img src="./img/car2.jpg" loading="lazy">
  <img src="./img/sunrise.jpg" loading="lazy">
  <img src="./img/sunset.jpg" loading="lazy">
  <img src="./img/mountain.jpg" loading="lazy">

</body>
</html>
```

## JS Optimizations

### Defer vs Async:

- If the defer attribute is set, it specifies that the script is downloaded in parallel to parsing the page, and executed after the page has finished parsing
- If async is present: The script is downloaded in parallel to parsing the page, and executed as soon as it is available (before parsing completes)
- If defer is present (and not async): The script is downloaded in parallel to parsing the page, and executed after the page has finished parsing
- If neither async or defer is present: The script is downloaded and executed immediately, blocking parsing until the script is completed

```
<!DOCTYPE html>
<html>
<body>
<h1>The script defer attribute</h1>
<script src="script.js" defer></script>
<p>The script above requests information from the paragraph below. Normally,
this is not possible, because the script is executed before the paragraph
exists.</p>
<p id="p1">Hello World!</p>
<p>However, the defer attribute specifies that the script should be executed later.
This way the script can request information from the paragraph.</p>

</body>
</html>
```

## Make AJAX Cacheable

index.php

```
<html>

<head>
  <title> AJAX SIMPLE CALL USING JAVASCRIPT </title>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
</head>

<body>
  <select onchange='getResponseVal()' id='genre_list'>
     <option value="Fantasy" selected='selected'>Fantasy</option>
     <option value="Mystery">Mystery</option>
     <option value="Horror">Horror</option>
  </select>

  <div id="demo"></div>
</body>



<script>
$(document).ready(function() {
  // Create a cache object to store genre data
  var cache = {};

  // Set a function to check for cache expiration
  function isCacheValid(genre) {
     if (!cache[genre]) {
        return false; // Cache doesn't exist
     }

     // Check if the cache has an expiration date set
```

```
        if (!cache[genre].hasOwnProperty('expiration')) {
            return false; // No expiration set, treat as invalid
        }
        // Check if the expiration date has passed
        return cache[genre].expiration > new Date().getTime();
    }
    $("select").change(function() {
        var genre = $("#genre_list").find('option:selected').val();

        // Check if data for the selected genre is already in cache and valid
        if (isCacheValid(genre)) {
            $("#demo").html(cache[genre].data); // Use data property from cache
        } else {
            // If not cached or expired, make the AJAX request
            $.ajax({
                type: "GET",
                url: "data.php",
                data: 'genre=' + genre,
                success: function(data) {
                    $("#demo").html(data);

                    // Store the fetched data with an expiration date
                    cache[genre] = {
                        data: data,
                        expiration: new Date().getTime() + (1000 *
                            60) // Set expiration in 1 hour
                    };
                }
            });
        }
    });
});
</script>

</html>
```

connection.php

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$database = "demo";
// Create connection
$conn = mysqli_connect($servername, $username, $password, $database );
// Check connection
if (!$conn) {
die("Connection failed: " . mysqli_connect_error());
}
// echo "Connected successfully";
?>
```

data.php

```
<?php
include './connection.php';
if (isset($_GET['genre'])) {
  $genre = $_GET['genre'];
  // echo "genre: " . $genre;
}
$sql = "SELECT * FROM Book where genre='$genre'";
// echo $sql;


$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
  while ($row = mysqli_fetch_assoc($result)) {
        echo "author : ".$row['author'].", Title : ".$title =  $row['title']."</br>";
        }
 }
?>
```

## Optimizing PHP:

### Lazy Loading:

Index.php

```
<?php
   include 'connection.php';
   $limit = isset($_POST["limit-records"]) ? $_POST["limit-records"]
: 10;
   $page = isset($_GET['page']) ? $_GET['page'] : 1;
   $start = ($page - 1) * $limit;
   $result = $conn->query("SELECT * FROM book LIMIT $start, $limit");
   $books = $result->fetch_all(MYSQLI_ASSOC);

   $result1 = $conn->query("SELECT count(id) AS id FROM book");
   $custCount = $result1->fetch_all(MYSQLI_ASSOC);
   $total = $custCount[0]['id'];
   $pages = ceil( $total / $limit );

$Previous = $page - 1;
$Next = $page + 1;

?>
<!doctype html>
<html lang="en">
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap
.min.css" rel="stylesheet"

integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JM
hjY6hW+ALEwIH" crossorigin="anonymous">
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js
"></script>
</head>
```

```
<body>

   <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.b
undle.min.js"

integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIds
lK1eN7N6jIeHz" crossorigin="anonymous">
   </script>
</body>

</html>

<div class="container">
   <h1 class="text-center"></h1> PHP and MySQL Pagination</h1>
   <div class="row">
       <div class="col-md-10">
           <nav aria-label="Page navigation">
               <ul class="pagination">

                   <li class="page-item">
                       <a class="page-link" href="index.php?page=<?=
$Previous; ?>" aria-label="Previous">
                           <span aria-hidden="true">&laquo;
Previous</span>
                       </a>
                   </li>
                   <?php
                   $max = 3;
                   for($i = 1; $i<= $max; $i++) : ?>
                   <li class="page-item"><a class="page-link"
href="index.php?page=<?= $i; ?>"><?= $i; ?></a></li>

                   <?php endfor; ?>
                   <li class="page-item">
                       <a class="page-link" href="index.php?page=<?=
$Next; ?>" aria-label="Next">
                           <span aria-hidden="true">Next
```

```
&raquo;</span>
                       </a>
                   </li>
               </ul>
           </nav>
       </div>
       <div class="text-center" style="margin-top: 20px; "
class="col-md-2">
           <form method="post" action="#">
               <select name="limit-records" id="limit-records">
                   <option disabled="disabled"
selected="selected">---Limit Records---</option>
                   <?php foreach([10,20,50] as $limit): ?>
                   <option
                       <?php if( isset($_POST["limit-records"]) &&
$_POST["limit-records"] == $limit) echo "selected" ?>
                       value="<?= $limit; ?>"><?= $limit; ?></option>
                   <?php endforeach; ?>
               </select>
           </form>
       </div>
   </div>
   <div style="height: 600px; overflow-y: auto;">
       <table id="" class="table table-striped table-bordered">
           <thead>
               <tr>
                   <th>Id</th>
                   <th>Author</th>
                   <th>Title</th>
                   <th>Genre</th>
               </tr>
           </thead>
           <tbody>
               <?php foreach($books as $book) :  ?>
               <tr>
                   <td><?= $book['id']; ?></td>
                   <td><?= $book['author']; ?></td>
                   <td><?= $book['title']; ?></td>
                   <td><?= $book['genre']; ?></td>
               </tr>
               <?php endforeach; ?>
           </tbody>
       </table>
   </div>

   <script type="text/javascript">
   $(document).ready(function() {
       $("#limit-records").change(function() {
           $('form').submit();
       })
   })
   </script>
   </body>

   </html>
```

Database Decoupling by Memcached:

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Document</title>
  <script>
    function submitOnGenreChange() {
      // Get the genre select element
      const genreSelect = document.getElementById('genre');

      // Check if a genre is selected (value is not empty)
      if (genreSelect.value !== '') {
        // Submit the form programmatically
        document.getElementById('myForm').submit();
      }
    }
  </script>
</head>
<body>
  <form id="myForm" action="./data.php" method="post">
    <label for="genre">Book Genre:</label>
    <select id="genre" name="genre" onchange="submitOnGenreChange()">
      <option value="">Select Genre</option>
      <option value="Fantasy">Fantasy</option>
      <option value="Fiction">Fiction</option>
      <option value="Mystery">Mystery</option>
      <option value="Romance">Romance</option>
      <option value="Science Fiction">Science Fiction</option>
    </select>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

## Data.php

```
<?php

$choice = $_POST['genre'];
// Memcached server details (replace with your own)
$memcached_host = "localhost";
// $memcached_port = 11211;
$memcached_port = 11213;

// Database connection details (replace with your own)
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "demo";
```

```
// Create Memcached connection
$memcache = new Memcached();
$memcache->addServer($memcached_host, $memcached_port);

// Function to retrieve data (checks Memcached first, then database)
function get_data($key) {
  global $memcache, $servername, $username, $password, $dbname;

  // Check Memcached for data
  $data = $memcache->get($key);


  if (!$data) {
    // Data not found in Memcached, fetch from database
    $conn = mysqli_connect($servername, $username, $password,
$dbname);
    if (!$conn) {
      die("Connection failed: " . mysqli_connect_error());
    }

    $sql = "SELECT * FROM book WHERE genre="."'$key'"; // Adjust
query based on your table structure
    $result = mysqli_query($conn, $sql);

    if (mysqli_num_rows($result) > 0) {

        while ($row = mysqli_fetch_assoc($result)) {
          $data .= $row["title"]." by ".$row['author']."</br>";
          $memcache->set($key, $data, 3600); // Cache data for 1 hour
            }
    } else {
      $data = "No data found";
    }

    mysqli_close($conn);
  }

  return $data;
}

// Get data using the function (replace 'Fantasy' with your desired
key)
$data = get_data($choice);

// Display the data
echo $data;

?>
```

## Use memory for sessions:

```
<?php

// Memcached Server Configuration (replace with your actual settings)
$memcache_servers = array(
   array('host' => 'localhost', 'port' => 11211), // Replace with your Memcached server
host and port
);

// Session Configuration
ini_set('session.save_handler', 'memcache');
ini_set('session.save_path', implode(',', array_map(function ($server) {
   return sprintf('tcp://%s:%d', $server['host'], $server['port']);
}, $memcache_servers)));

// Database Connection (replace with your actual credentials)
$db_host = 'localhost';
$db_username = 'your_username';
$db_password = 'your_password';
$db_name = 'your_database_name';

// Start the session
session_start();

// Example: Fetch user data from database
function get_user_data($user_id) {
   global $db_host, $db_username, $db_password, $db_name;

   $conn = mysqli_connect($db_host, $db_username, $db_password, $db_name);
   if (!$conn) {
       die("Connection failed: " . mysqli_connect_error());
   }

   $sql = "SELECT username, email FROM users WHERE email = ?";
   $stmt = mysqli_prepare($conn, $sql);
   mysqli_stmt_bind_param($stmt, "i", $user_id);
   mysqli_stmt_execute($stmt);
   $result = mysqli_stmt_get_result($stmt);

   if (mysqli_num_rows($result) > 0) {
       $row = mysqli_fetch_assoc($result);
       return $row;
   } else {
       return null;
   }

   mysqli_close($conn);
}

// Check if user data exists in Memcached session
$user_data = null;
if (isset($_SESSION['user_data'])) {
   $user_data = $_SESSION['user_data'];
}

// If user data is not in session or expired (optional), fetch from database and store in
Memcached
if (!$user_data) {
   $user_id = 1; // Replace with actual user ID
   $user_data = get_user_data($user_id);

   if ($user_data) {
       $_SESSION['user_data'] = $user_data;
   }
}

// Example usage:
if ($user_data) {
   echo "Welcome, " . $user_data['username'] . "!";
   echo "<br>Email: " . $user_data['email'];
} else {
   echo "No user data found.";
}

// Destroy session (optional)
// session_destroy();

?>
```

## Php store session using memcached:

Search for : Memcheched php version/8.1 windows

How to install memchached php 8

How to install memcached on php 7

Download thread safe version.

Move php_memchached.dll file to php/ext folder

Move other dll file to C>windows folder

Now open php.ini file:

Search for extension and add

extension = memcached

Now search for session and add the following configuration :

```
session.save_handler = memcached
session.save_path = "tcp://localhost:11211"
```

**1. Download the Thread Safe Version:**

- Search online for "Memcached php version 8.1 windows".
- Download the **Thread Safe** version (likely named php_memcached.dll) compatible with your system architecture (x64 or x86).

**2. Move the DLLs:**

- Copy the downloaded php_memcached.dll file to your PHP extensions directory. This is typically located at C:\php\ext (adjust the path based on your PHP installation).
- Copy the additional DLL file (likely named libmemcached.dll) to your system's Windows folder (usually C:\windows).

**3. Configure php.ini:**

- Open your php.ini file located in your PHP installation directory (e.g., C:\php).
- **Enable the Memcached extension:**
  - Find the section containing lines that start with extension=.

Add a new line with the following text:

```
extension=memcached
```

- ○
- **Configure Memcached for Session Handling:**
  - Find the section containing lines that start with session..

Add the following lines (adjust the port if your Memcached server uses a different one):

```
session.save_handler = memcached
session.save_path = "tcp://localhost:11211"
```

- ○

**4. Restart your web server** for the changes to take effect.

```php
<?php

// Enable Memcached session storage
ini_set('session.save_handler', 'memcache');
ini_set('session.save_path', 'localhost:11211');

// Start the session
session_start();

// Check for Memcached connection errors
if (extension_loaded('memcache') === false) {
  die('Memcache extension is not loaded. Please install and enable
it.');
}

// Session data management examples
$_SESSION['username'] = 'john_doe';
$username = isset($_SESSION['username']) ? $_SESSION['username'] :
null;

// ... Your application logic here ...

// Destroy the session
// session_destroy();

?>

```

## Lab Work:

## Lab Session Implementation in Group Project

**Preparation:**

1. **Identify Application:** Discuss how the lab session's topic can be applied to your group project. Brainstorm potential functionalities or improvements related to the covered concept.
2. **Divide Responsibilities:** Based on the chosen application, assign tasks within your group. Consider individual strengths and expertise when delegating tasks

**Implementation:**

1. **Individual Contribution:** Each member completes their assigned tasks. This might involve coding specific functionalities, creating necessary configurations etc.
2. **Group Integration:** Discuss challenges, and ensure all parts are coming together cohesively.
3. **Testing and Refinement:** Test the implementation of the lab session topic within your group project. Identify any bugs, errors, or areas for improvement. Collaboratively refine your work based on testing results.