# LECTURE08: ERROR HANDLING AND CSS

CS418/518: WEB PROGRAMMING

BY DR. JIAN WU

COURTESY: DR. JUSTIN BRUNELLE

# ERROR HANDLING AND INPUT VALIDATION

CS418/518

# WHY DO WE NEED INPUT VALIDATION?

- Typos
- Malicious input, e.g., XSS vulnerability
- We'll look at validating:
  - Simple string values
  - Integer values
  - Formatted text input

# SERVER VS. CLIENT

- Client-side security == no security
  - Clients have full control of the browser
  - Malicious or broken clients
  - Do not rely on client-side validation
- JavaScript (client-side) might be disabled
- Place a server-side validator for all input

# VARIABLE HANDLING FUNCTIONS

- `isset()`: determine if a variable is declared and is not NULL
- `empty()`: determine whether a variable is empty
- `is_numeric()`: finds whether a variable is a number of a numeric string
- `is_bool()`: finds out whether a variable is a boolean
- `is_array()`: finds whether a variable is an array
- `is_object()`: finds whether a variable is an object
- More at: https://www.php.net/manual/en/ref.var.php

# DIE() FUNCTION

The die() function prints a message and exits the current script.

```php
<?php
    if(!file_exists("/tmp/test.txt")) {
        die("File not found");
    }else {
        $file = fopen("/tmp/test.txt","r");
        print "Opend file sucessfully";
    }
    // Test of the code here.
?>
```

diefunc.php

The argument is optional.
You can just call die();

ordiefunc.php

```php
<?php
    $url = "https://www.example.com/";
    fopen($url,"r")
    or die("Can't connect!");
?>
```

# CUSTOM ERROR FUNCTION

- Syntax
  - `error_function(error_level,error_message, error_file,error_line, error_context);`

| **Required — The following parameters are required** | |
|---|---|
| error_level | Specifies the level of the error, as an integer. This corresponds to the appropriate error level constant ( E_ERROR, E_WARNING, and so on) |
| error_message | Specifies the error message as a string |
| **Optional — The following parameters are optional** | |
| error_file | Specifies the filename of the script file in which the error occurred, as a string |
| error_line | Specifies the line number on which the error occurred, as a string |
| error_context | Specifies an array containing all the variables and their values that existed at the time the error occurred. Useful for debugging |

# POSSIBLE ERROR LEVELS

most important error levels

| Error level | Value | Description |
|---|---|---|
| E_ERROR | 1 | A fatal run-time error, that can't be recovered from. The execution of the script is stopped immediately. |
| E_WARNING | 2 | A run-time warning. It is non-fatal and most errors tend to fall into this category. The execution of the script is not stopped. |
| E_PARSE | 4 | The compile-time parse error. Parse errors should only be generated by the parser. |
| E_NOTICE | 8 | A run-time notice indicating that the script encountered something that could possibly an error, although the situation could also occur when running a script normally. |
| E_CORE_ERROR | 16 | Fatal errors that occur during PHP's initial start-up. |
| For other errors, see: https://www.tutorialspoint.com/php/php_error_handling.htm | | |

# CUSTOM ERROR FUNCTION EXAMPLE

```php
<?php
// Error handler function
function customError($errno, $errstr){
    echo "<b>custom Error:</b> [$errno] $errstr";
}

// Set error handler
set_error_handler("customError");

// Trigger error
echo($test);
?>
```

We must tell PHP to use this function to handle errors.
Here, we set customError() to handle all errors
But you can set it to handle specific types of errors

handleerror.php

# TRIGGER AN ERROR

- In a script where users can input data, it is useful to trigger errors when an illegal input occurs. In PHP, this is done by the trigger_error(*message, type*) function.
- trigger_error() can be used anywhere you wish in a script, and by adding the optional *type* parameter, you can specify what error level is triggered.
-

```php
<?php
$test=2;
if ($test>=1) {
  trigger_error("Value must be 1 or below");
}
?>
```

triggererror.php

Note: this is not an echo statement, so the errors are not shown on the webpage. The errors are in web server logs.

# TRIGGER ERROR EXAMPLE

triggererror2.php

```php
<?php
//error handler function
function customError($errno, $errstr) {
  echo "<b>Error:</b> [$errno] $errstr<br>";
  echo "Ending Script";
  die();
}

//set error handler
set_error_handler("customError",E_USER_WARNING);

//trigger error
$test=2;
if ($test>=1) {
  trigger_error("Value must be 1 or below",E_USER_WARNING);
}
?>
```

Use `customError` to handle `E_USER_WARNING` errors only.
Change this to `E_USER_ERROR` and see what happens

Other types of errors

- `E_USER_ERROR` - Fatal user-generated run-time error. Execution of the script is halted
- `E_USER_WARNING` - Non-fatal user-generated run-time warning.
- `E_USER_NOTICE`

# SEND ERROR MESSAGES BY EMAIL

```php
<?php
//error handler function
function customError($errno, $errstr) {
  echo "<b>Error:</b> [$errno] $errstr<br>";
  echo "Webmaster has been notified";
  error_log("Error: [$errno] $errstr",1,
  "someone@example.com","From: webmaster@example.com");
}

//set error handler
set_error_handler("customError",E_USER_WARNING);

//trigger error
$test=2;
if ($test>=1) {
  trigger_error("Value must be 1 or below",E_USER_WARNING);
}
?>
```

`error_log()` function can send error logs to a specified file or a remote destination

This should not be used with all errors. Regular errors should be logged on the server using the default PHP logging system.

# GENERATE ERRORS ON THE SAME PAGE

Instead of quitting the application instantly, a common practice is to concatenate all errors and print them together

```php
<?php
if (empty($var1)) {
    $errors .= "var1 should not be empty<br>";
}
if (!is_numeric($var2)) {
    $errors .= "var2 should be a number<br>";
}
// check all anticipated error conditions ...
if (empty($errors)){
    echo "No errors found!";
} else {
    internal_error_function($errors);
}

function internal_error_function ($errors) {
    echo "Errors found: ". $errors;
    // provide link to start over
}
?>
```

samepageerror.php

# URI WITH ERROR IN AN ARGUMENT

```php
<?php
if (empty($var1)) {
  $errors .= "$var1 should not be empty";
}
if (!is_numeric($var2)) {
  $errors .= "$var2 should be a number";
}
// check all anticipated error conditions
if (empty($errors)) {
   echo "I am error free!";
} else {
  $errors = urlencode($errors);
  header("Location: https://www.google.com");
}
?>
```

- If an error is made, direct me to Google.

urlencode: see next page

# ENCODE URLS

- RFC1738 (replaced by RFC3986) requires “unsafe” and “reserved” characters to be encoded in URLs
  - Reserved examples: “`/`”, “`:`”, and “`?`”
  - Unsafe examples: [space], “%”, “$”
  - Why are they called “unsafe characters”? Because they may lead to wrongly parsed URLs. See more at https://perishablepress.com/stop-using-unsafe-characters-in-urls/
- PHP `urlencode()`, `urldecode()`
- More information, examples, and rationales: http://www.blooberry.com/indexdot/html/topics/urlencoding.htm

# INPUT VALIDATION EXAMPLES: REGULAR EXPRESSIONS

- To do formatted text (e.g., date) validation, use regular expressions. Example:
  - DD-MM-YYYY format for date
    - ([0-9]{2})-([0-9]{2})-([0-9]{4})
    - 2 digits between 0-9
    - “-” character
    - 2 digits between 0-9
    - “-” character
    - 4 digits between 0-9

# REGULAR EXPRESSIONS (REGEX)

```
if ( !preg_match("/([0-9]{2})-([0-9]{2})-([0-9]{4})/",
$_POST['movie_release'], $reldatepart) )
{
    $error .= "Please enter a date with the dd-mm-yyyy format";
}
```

pseudo-code:

- if $_POST[‘movie_release’] is 31-05-1969 then:
  - $reldatepart[0] = 31-05-1969
  - $reldatepart[1] = 31
  - $reldatepart[2] = 05
  - $reldatepart[3] = 1969

# DATE TIME

- Change the date string into a timestamp (to store in the database) using `mktime(`*hour, min, seconds, month, day, year*`)`
  - If valid date, returns the number of seconds since January 1, 1970
  - If not a valid date (e.g., 99-99-9999), return -1

continued from the last example:

```
$movie_release = mktime (0, 0, 0,
$reldatepart['2'], $reldatepart['1'],
$reldatepart['3']);
```

# WORKING WITH MYSQL

- UNIX_TIMESTAMP()
  - Takes YYYY-MM-DD HH:MM:SS format string
  - Creates a timestamp
  - Continued from the last example:

```
$reldate = $reldatepart['3'] . "-" . $reldatepart['2'] . "-" .
$reldatepart['1'] . " 00:00:00";
$sql = "INSERT INTO movie_main (release_date) " . "VALUES
(UNIX_TIMESTAMP('$reldate'))";
```

# XSS VULNERABILITY

- **Cross Site Scripting (XSS)**
- Malicious scripts are injected into otherwise benign and trusted websites.
- Watch this youtube video: https://www.youtube.com/watch?v=EhOcAZJp81s

# HTML IMAGES

CS418/518

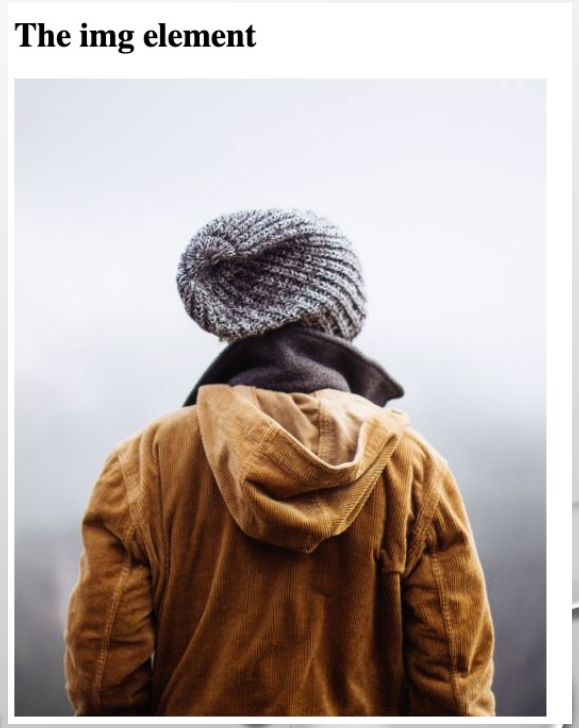
# HTML <IMG> TAG

- How to insert an image

```
<img src="img_girl.jpg" alt="Girl in a jacket" width="500" height="600">
```

- The images can be jpg, jpeg, gif, png, etc. Most browsers do not support TIFF. https://developer.mozilla.org/en-US/docs/Web/HTML/Element/img
- Images are not technically inserted into a web page; images are linked to web pages. The <img> tag creates a holding space for the referenced image.
- src - the path to the image
- alt - an alternate text for the image, if the image cannot be displayed

Example: showimage.html
Change the path to other places and see if it works.



# SPECIFY IMAGE SIZES IN PIXELS

```
<img src=”img_girl.jpg" alt=”Girl in a Jacket" width="42" height="42”>
```

- The width and height attribute specifies the width of an image, in pixels.
- Always specify both the height and width attributes for images for the browser to reserve the space.
- Rescale the image with a program before using it on a page.
  - Image resizer for Windows: https://www.bricelam.net/ImageResizer/
    Image resizer for Mac: https://apps.apple.com/us/app/image-resizer-resize-photos/id1188274404?mt=12
  - Image resizers for Linux: ImageMagick: https://imagemagick.org/index.php

# ADD A HYPERLINK TO AN IMAGE

```
<a href=" https://www.cs.odu.edu/~jwu/ ">
<img src="jianwu.png" alt="Jian Wu's Homepage" width="100" height="100">
</a>
```

- Example: imagehyperlink.html

# ALIGN IMAGES

```
<img src="smiley.gif" alt="Smiley face" width="42" height="42" style="vertical-align:bottom">
<img src="smiley.gif" alt="Smiley face" width="42" height="42" style="vertical-align:middle">
<img src="smiley.gif" alt="Smiley face" width="42" height="42" style="vertical-align:top">
<img src="smiley.gif" alt="Smiley face" width="42" height="42" style="float:right">
<img src="smiley.gif" alt="Smiley face" width="42" height="42" style="float:left">
```

- Example: imagealign.html
- Note: HTML inline-block by default.

# ADD AN IMAGE BORDER

```
<img src="smiley.gif" alt="Smiley face" width="42" height="42" style="border:5px solid black">
```

https://www.html.am/html-codes/image-codes/html-image-borders.cfm

- Example: imageborder.html

# ADD MARGINS TO IMAGES

```
<img src="smiley.gif" alt="Smiley face" width="42" height="42" style="vertical-
align:middle;margin:0px 50px">

<img src="smiley.gif" alt="Smiley face" width="42" height="42" style=”vertical-
align:middle;margin:50px 0px">
```

- Example: htmlmargin.html

# INSERT IMAGES FROM ANOTHER FOLDER OR A URL

```
<img src="/images/stickman.gif" alt="Stickman" width="24" height="39">
<img src="https://upload.wikimedia.org/wikipedia/commons/thumb/f/fc/Two_Adult_Guinea_Pigs_%28cropped%29.jpg/220px-Two_Adult_Guinea_Pigs_%28cropped%29.jpg" alt=”Guinea Pigs" width="32" height="32">
```

- Example: imageexternal.html

# CREATE AN IMAGE MAP

```
<img src="workplace.jpg" alt="Workplace" usemap="#workmap" width="400" height="379">

<map name="workmap">
  <area shape="rect" coords="34,44,270,350" alt="Computer" href="computer.htm">
  <area shape="rect" coords="290,172,333,250" alt="Phone" href="phone.htm">
  <area shape="circle" coords="337,300,44" alt="Cup of coffee" href="coffee.htm">
</map>
```

- Example: imagemap.html

# PHP GD AND IMAGE FUNCTIONS

- https://www.php.net/manual/en/ref.image.php
- GD image library
- getimagesize(): Get the size of an image
- imagejpeg(): Output image to browser or file

# CREATE AN IMAGE GRID

- https://www.w3schools.com/howto/tryit.asp?filename=tryhow_js_image_grid


# CSS (CASCADING STYLE SHEETS)

CS418/518

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>My Web Page</title>
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>
    <body>
        <header>
            <h1>My Web Page</h1> <h2>Home to my stuff</h2>
        </header>
        <nav>
            <ul>
                <li><a href="#section1">Section 1</a></li>
                <li><a href="#section2">Section 2</a></li>
                <li><a href="#section3">Section 3</a></li>
            </ul>
        </nav>
        <section id="section1">
            <h2>Welcome to Section 1!</h2>
            <p>A section of content!</p>
        </section>
    </body>
</html>
```

import a style.css file that specifies the formats of text inside certain blocks, such as text inside certain HTML tags.

# HTML HEAD

```
<head>
    <meta charset="utf-8">
    <title>My Web Page</title>
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
```

more about the link tag

- Reference to stylesheets (CSS), external or inline
- The HTML <meta> tag is used for declaring metadata for the HTML document.
- `charset` specifies the character encoding used by the document. This is called a character encoding declaration.

# THE MOTIVATION OF USING CSS IS TO SEPARATE CONTENT AND STYLE

- Use HTML for content/structure
  - Align the content/structure with an appropriate tag
    - E.g., <nav> for navigational items
- Use CSS for styles, such as position, color, etc.
  - Only tabular data in tables!
- Use PHP or JavaScript to define behaviors

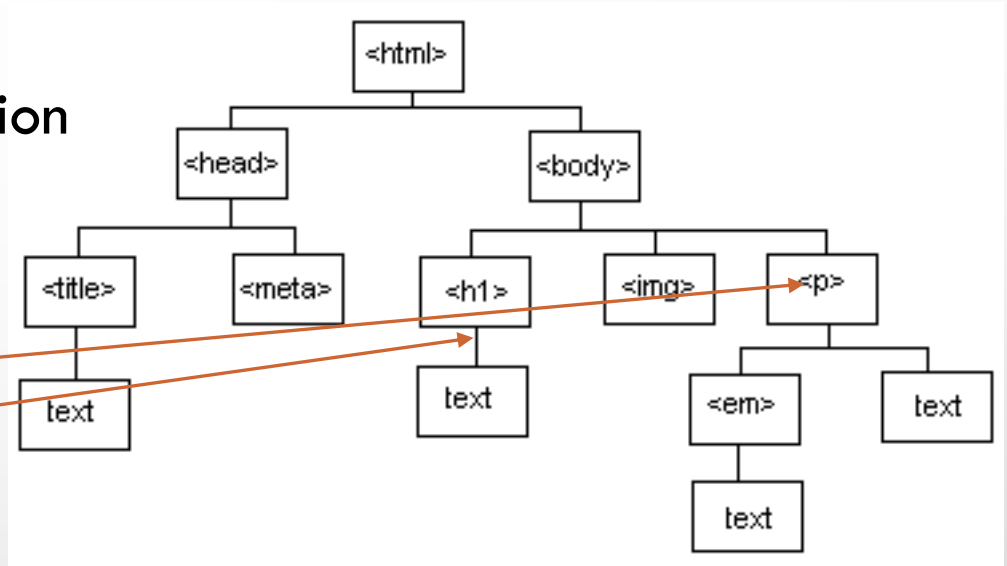
# DOM

- DOM: Document object model
  - Defines STRUCTURE of an HTML representation
  - Tree-based
  - Basis for CSS and Javascript
- CSS:

```
p {background-color: blue;}
section > h1 {font-size: 24px;}
```

- JavaScript:

```
var navs = document.getElementsByTagName("nav")
```



# DOM ATTRIBUTES – ID

- HTML definition – ID

```
<h1 id="myHeader">My Header</h1>
```

- CSS selector
  - The syntax for id selector: write a hash character (#) character, followed by an id name. Then, define the CSS properties within curly braces {}.
  - `#myHeader {color:red;}`
- JavaScript

```
var myP = document.getElementById('myHeader');
```

- The HTML id attribute is used to specify a unique id for an HTML element.
- You cannot have more than one element with the same id in an HTML document.
- The id attribute is used to point to a specific style declaration in a style sheet. It is also used by JavaScript to access and manipulate the element with the specific id.

```
<!DOCTYPE html>
<html>
<head>
<style>
#myHeader {
  background-color: lightblue;
  color: black;
  padding: 40px;
  text-align: center;
}
</style>
</head>
<body>

<h1 id="myHeader">My Header</h1>

</body>
</html>
```

# DOM ATTRIBUTES – CLASS

- HTML definition

```
<p class="specialElements">Lorem Ipsum</p>
<div class="anotherGroup">Another group's contents</div>
<span class="specialElements">Dolor Sit Amet</span>
```

- CSS selector

```
.specialElements{text-decoration: underline;}
```

- JavaScript reference

```
var mySpecialElements=document.getElementsByClassName('specialElements');
var secondSpecialElement=specialElements[1]; //note the array syntax
```

- The HTML class attribute is used for specifying a class for an HTML element.
- Multiple HTML elements can share the same class.
- The class attribute is often used to point to a class name in a style sheet. It can also be used by a JavaScript to access and manipulate elements with the specific class name.

# CSS SELECTORS

- A CSS rule is made up of a selector and several declarations. A declaration consists of property and value:

```
selector{
    property: value;
    property: value;
    ……
    }
```

  - A selector is often an element of HTML identifier, such as an ID or a class.
  - Properties and values tell an HTML element how to display.
  - For more CSS selectors, see http://www.w3schools.com/cssref/css_selectors.asp

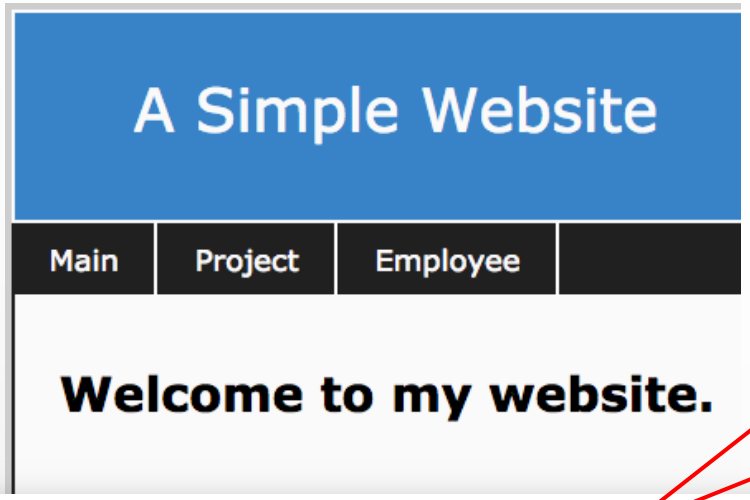
# HOW TO INTERPRET CSS FILES

More about

[CSS border property]

[CSS border-top]

[CSS border-bottom]



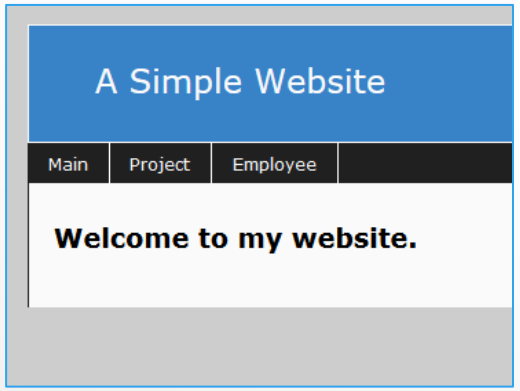

styles.css

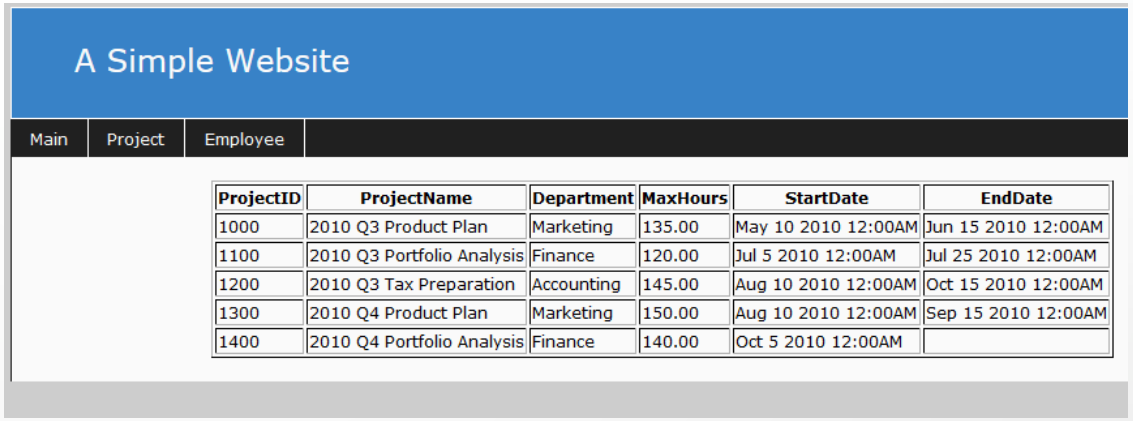
```css
#centre {
border:1px solid #202020;
border-bottom:0;
border-top:0;
color:#000;
padding:1.5em;
}

#conteneur {
background-color:#fafafa;
margin:1em 10%;
min-width:60em;
position:absolute;
width:80%;
}

#haut {
background-color:#202020;
height:2.4em;
max-height:2.4em;
}

#header {
background-color:#3882C7;
border:1px solid #fafafa;
color:#fafafa;
font-size:2em;
height:2.5em;
padding-left:2em;
padding-top:1em;
}
```

```php
<html>
    <head>
        <title>A Simple Website</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-15" />
        <link rel="stylesheet" href="styles.css" />
    </head>
        <div id="conteneur">
          <div id="header">A Simple Website</div>

        <?php include 'navbar.php'; ?>

          <div id="centre">
            <h1>Welcome to my website.</h1>
            </div>
        </div>
</html>
```

main.php

# A SIMPLE EXAMPLE




| ProjectID | ProjectName | Department | MaxHours | StartDate | EndDate |
|---|---|---|---|---|---|
| 1000 | 2010 Q3 Product Plan | Marketing | 135.00 | May 10 2010 12:00AM | Jun 15 2010 12:00AM |
| 1100 | 2010 Q3 Portfolio Analysis | Finance | 120.00 | Jul 5 2010 12:00AM | Jul 25 2010 12:00AM |
| 1200 | 2010 Q3 Tax Preparation | Accounting | 145.00 | Aug 10 2010 12:00AM | Oct 15 2010 12:00AM |
| 1300 | 2010 Q4 Product Plan | Marketing | 150.00 | Aug 10 2010 12:00AM | Sep 15 2010 12:00AM |
| 1400 | 2010 Q4 Portfolio Analysis | Finance | 140.00 | Oct 5 2010 12:00AM | |

main.php

project.php

shared

shared

```
#centre {
border:1px solid #202020;
border-bottom:0;
border-top:0;
color:#000;
padding:1.5em;
}

#conteneur {
background-color:#fafafa;
margin:1em 10%;
min-width:60em;
position:absolute;
width:80%;
}

#haut {
```

styles.css

checkout the folder website/ for source code

# YOU CAN REDEFINE THE STYLE OF AN HTML TAG WITH CSS

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  color: blue;
  font-family: verdana;
  font-size: 300%;
}
p  {
  color: red;
  font-family: courier;
  font-size: 160%;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

# PRECEDENCE OF CSS OPERATIONS

- Inline style > id > class > element

| HTML Code | CSS Code | Effect |
|---|---|---|
| Hello | p {color: red;} | Hello |
| Hello | p {color: red;} [Element level]<br>p.test {color blue; background-color: yellow;} [class level] | Hello |
| Hello | p {color: red;} [Element level]<br>p.test {color blue; background-color: yellow;} [class level]<br>p#myP {background-color: orange;} [id level] | Hello |
| Hello [inline level] | p {color: red;} [Element level]<br>p.test {color blue; background-color: yellow;} [class level]<br>p#myP {background-color: orange;} [id level] | Hello |
| <p style="color: white; id="myP"> Hello World! | p#myP {background-color: orange;} [id level]<br>p span {background-color: purple} [element level] | Hello World! |

# BACKUP SLIDES BEYOND THIS POINT

CS418/518