

COMS 210 Internet Programming

Exam Reference Sheet

HTML

Tags Used in the `<head>` Section

Tag	Description
<code><title> text </title></code>	title shown on page tab
<code><meta attribute="value" ... /></code>	page metadata
<code><link href="url" rel="stylesheet" /></code>	links to a CSS style sheet
<code><script src="url"></script></code>	link to JavaScript code
<code><!-- comments --></code>	comment (can appear in head or body)

Tags Used in the `<body>` Section

Tag	Display	Description
<code><p>text </p></code>	Block	paragraph
<code><h1>text </h1></code> <code><h2>text </h2></code> ... <code><h6>text </h6></code>	Block	(h1 for largest to h6 for smallest)
<code><hr /></code>	Block	horizontal rule (line)
<code>
</code>	Inline	line break
<code>text </code>	Block	anchor (link)
<code></code>	Inline-block	image
<code>text</code>	Inline	emphasis (italic)
<code>text </code>	Inline	strong emphasis (bold)
<code></code> <code>text </code> <code>text </code> <code></code> <code></code> <code>nested item text</code> <code>nested item text</code> <code></code> <code></code> <code></code>	Block	ordered (<code>ol</code>) and unordered (<code>ul</code>) list; list item (<code>li</code>)

Tags Used in the `<body>` Section (Continued)

Tag	Display	Description
<code><dl></code> <code><dt>term 1 </dt></code> <code><dd>description 1 </dd></code> <code><dt>term 2 </dt></code> <code><dd>description 2 </dd></code> <code></dl></code>	Block	definition list (<code>dl</code>); term (<code>dt</code>), and its description (<code>dd</code>)
<code><blockquote></code> <code><p>text </p> ...</code> <code></blockquote></code>	Block	block-level quotation
<code><q>text </q></code>	Inline	inline-level quotation
<code><code>text </code></code>	Inline	computer code (monospace)
<code><pre>text </pre></code>	Inline	pre-formatted text (preserves whitespace)
<code><table></code> <code><caption>text </caption></code> <code><tr></code> <code><th>heading 1 </th></code> <code><th>heading 2 </th></code> <code></tr></code> <code><tr></code> <code><td> cell 1 </td></code> <code><td> cell 2 </td></code> <code></tr></code> <code>...</code> <code></table></code>	Block	table of data (<code>table</code>) description of table (<code>caption</code>) table row (<code>tr</code>) table heading cell (<code>th</code>) normal table cell (<code>td</code>)
<code><div> ... </div></code>	Block	block-level section of a page
<code> ... </code>	Inline	inline-level section of a page

HTML5 Semantic Grouping Tags (all block elements)

Tag	Description
<code><header></code>	Container for a header of a document
<code><main></code>	Specifies the main content of a document. The content inside should be unique to the document and not contain content that is repeated across pages (e.g., sidebars, nav links, search bars, etc.)
<code><footer></code>	Container for a footer of a document
<code><article></code>	A standalone piece of content (e.g., entire blog post including title, author, etc.)
<code><section></code>	A piece of content that is part of another (e.g., a chapter section of a reading)
<code><aside></code>	Defines some content aside from the content it is placed in (e.g., a sidebar in an article)
<code><nav></code>	Defines content in a navigation bar

HTML Input Tags

Tag	Display	Description
<code><button></code> content <code></button></code>	Inline	clickable button type can be submit, reset, button
<code><input type="type" name="name"></code> content <code></input></code>	Inline	form element input tag type can be text, number, checkbox, radio, file, etc.
<code><textarea rows="num" cols="num"></code> initial text <code></textarea></code>	Inline	multi-line text input box
<code><label>text </label></code>	Inline	clickable text label around a form control
<code><select</code> <code>></code> <code><option>text </option></code> <code><option></code> <code><optgroup label="text"></code> <code><option> text </option></code> <code><option> text </option></code> <code></optgroup></code> <code>...</code> <code></select></code>	Inline	drop-down selection box (select); each option within the box (option); a labeled group of option (optgroup);
<code><fieldset></code> <code><legend> text </legend></code> content <code></fieldset></code>	Block	a grouped set of form fields with a legend

HTML Entities Reference

Result	Description	Entity Name
	non-breaking space	<code>&nbsp;</code>
<code><</code>	less than	<code>&lt;</code>
<code>@</code>	at symbol	<code>&commat;</code>
<code>></code>	greater than	<code>&gt;</code>
<code>&</code>	ampersand	<code>&amp;</code>
<code>©</code>	copyright	<code>&copy;</code>

CSS

For the following property and value tables, anything *emphasized* represents values that should be replaced with specific units (e.g., *length* should be replaced with a px, pt, or em for many properties, and *color* should be replaced with a valid color value such as a hex or rgb code).

A use of | refers to separation of possible values (where you cannot provide two of these possible values for one property) and [value value value] refers to a grouping of possible values that can optionally be used together (e.g., [*h-shadow v-shadow blur spread color*] for box-shadow).

Selector Types

Name	Description	Example(s)
Universal	Any element	<code>.foo * { font: 10pt Arial; }</code>
Element	Any element of a given type	<code>h1 { text-decoration: underline; }</code>
Grouping	Multiple elements of different types	<code>h1, h2, h3 { color: purple; }</code>
Class	Elements with the given class name	<code>.example { text-decoration: underline; }</code>
Id	Single element with the given id	<code>#example { text-decoration: overline; }</code>
Descendant	Elements that are children at any level of another specified element	<code>#example h1 { text-decoration: underline; }</code>
Child	Elements that are direct children of another specified element	<code>#example > p { font-weight: bold; }</code>
Attribute	Elements that have the specified attribute	<code>input[selected] - inputs that have the selected attribute</code> <code>input[name='test'] - inputs that have a name 'test'</code>

Background Styles

Property	Values
background-color	<i>color</i> transparent
background-image	<i>url</i> none
background-origin	border-box padding-box content-box
background-position	top left top center top right center left center center center right bottom left bottom center bottom right [<i>x-% y-%</i>] [<i>x-pos y-pos</i>]
background-size	<i>length</i> % auto cover contain
background-repeat	repeat repeat-x repeat-y no-repeat
background-attachment	scroll fixed

Border Styles

Note: Replace '*' with any side of the border (top, right, left, bottom) for the desired effect.

Example style: 'border: 2px solid red' applies a solid red border with a width of 2px to all four sides of the element, while 'border-left: 2px solid red' only applies that border to the left border'.

Property	Values
border, border-* (shorthand)	border-width, border-*-width border-style, border-*-style border-color, border-*-color
border-width, border-*-width	thin medium thick length
border-style, border-*-style	none hidden dotted dashed solid double groove rigid inset outset
border-color, border-*-color	color
box-shadow	none inset [<i>h-shadow v-shadow blur spread color</i>]
border-radius	length

Font and Text Styles

Property	Values
font-style	normal italic oblique inherit
font-family	fontname
font-size	length %
font-weight	normal bold inherit
text-align	left right center justify
text-decoration	none [underline overline line-through blink]
text-shadow	none [<i>color length</i>]
text-indent	length %
text-transform	none capitalize uppercase lowercase
list-style-type	none asterisks box check diamond disc hyphen square decimal lower-roman upper-roman lower-alpha upper-alpha lower-greek upper-greek lower-latin upper-latin footnotes

Color Values

Value	Description
colorname	Standard name of color, such as red, blue, purple, etc.
rgb(redvalue, greenvalue, bluevalue)	Example: red = rgb(255, 0, 0) or red = rgb(100%, 0, 0)
#RRGGBB	Example: red = #FF0000

Box Model

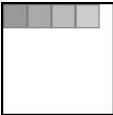
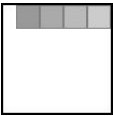
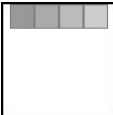
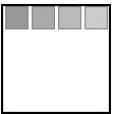
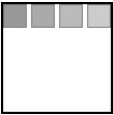
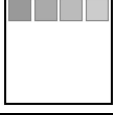
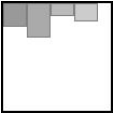
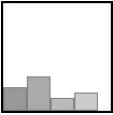
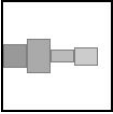
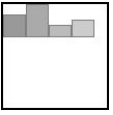
Property	Values
float	left right none
height, width	auto <i>length</i> %
min-height, max-height min-width, max-width	none <i>length</i> %
margin, margin-*	auto <i>length</i> %
padding, padding-*	<i>length</i> %
display	none inline block inline-block flex list-item compact table inline-table
overflow, overflow-x, overflow-y	visible hidden scroll auto no-display no-content
clear	left right both none

Flex Box

Property	Values	Element Type	Description
display	flex	Flex container	Sets all children to become 'flex-items'
flex-direction	row row-reverse column column-reverse	Flex container	Indicates if the container flows horizontally (<code>row</code>) or vertically (<code>column</code>)

(Flex Box continued on next page)

(Flex Box continued from previous page)

Property	Values	Element Type	Description
<code>justify-content</code>	<code>flex-start</code> <code>flex-end</code> <code>center</code> <code>space-around</code> <code>space-between</code> <code>space-evenly</code>	Flex container	<p>Indicates how to position the flex-items in the parent container</p>      
<code>align-items</code>	<code>stretch</code> (default) <code>flex-start</code> <code>flex-end</code> <code>center</code> <code>baseline</code>	Flex container	<p>Indicates how to space the items inside the container along the cross axis</p>    
<code>order</code>	<i>number</i>	Flex item	Specifies the order in which the element appears in the flex container (by default, flex items are laid out in the source order)
<code>align-self</code>	<code>flex-end</code> <code>flex-start</code> <code>center</code> <code>baseline</code> <code>stretch</code> (default)	Flex item	Indicates where to place this specific item along the cross axis

PHP

PHP Standard Functions

Function	Description
<code>isset(<i>el</i>)</code>	Will return false if <i>el</i> has been assigned the constant NULL, <i>el</i> has not been set to any value yet (undefined) <i>el</i> has been deleted using the unset function
<code>print <i>str</i></code> <code>echo <i>str</i></code>	Prints <i>str</i>
<code>time()</code>	Returns the current time in seconds
<code>date(<i>format</i>, <i>time</i>)</code>	Converts an optional <i>time</i> in seconds to a date based on <i>format</i>
<code>mt_rand(<i>min</i>, <i>max</i>)</code>	Returns a random integer between <i>min</i> and <i>max</i> (inclusive)
<code>header(<i>string</i>)</code>	Sends a raw HTTP header. Examples include: <code>header("HTTP/1.1 400 Invalid Request");</code> <code>header("HTTP/1.1 503 Service Unavailable");</code> <code>header("Content-type: text/plain");</code> <code>header("Content-type: application/json");</code>
<code>die(<i>message</i>)</code>	Ends execution and sends back optional <i>message</i>
<code>include "path"</code>	Includes and evaluates the specified file <i>path</i> such as "hidden/config.php"

PHP Array Functions

Function	Description
<code>count(<i>arr</i>)</code>	Returns the length of an array <i>arr</i>
<code>print_r(<i>arr</i>)</code>	Prints the <i>arr</i> 's contents
<code>array_pop(<i>arr</i>)</code>	Pops (removes) an element off the end of the array <i>arr</i>
<code>array_shift(<i>arr</i>)</code>	Shifts (removes) an element off the beginning of the array <i>arr</i>
<code>array_push(<i>arr</i>, <i>el</i>)</code>	Pushes (adds) one or more elements onto the end of the array <i>arr</i>
<code>array_unshift(<i>arr</i>, <i>el</i>)</code>	Prepends one or more elements to the beginning of the array <i>arr</i>
<code>sort(<i>arr</i>)</code>	Sorts the array <i>arr</i>
<code>array_reverse(<i>arr</i>)</code>	Returns an array with elements of <i>arr</i> in reverse order
<code>in_array(<i>el</i>, <i>arr</i>)</code>	Returns whether a value <i>el</i> exists in an array <i>arr</i>
<code>list(<i>a</i>, <i>b</i>, ...)</code>	Assigns variables as if they were an array
<code>implode(<i>glue</i>, <i>pieces</i>)</code>	Joins array elements (<i>pieces</i>) with a string (<i>glue</i>)
<code>array_rand(<i>arr</i>)</code>	Randomly selects a random entry from the array and returns the key (or keys) of the random entries.

PHP JSON Functions

Function	Description
<code>json_encode(<i>obj</i>)</code>	Returns JSON encoding for the given object/array/value
<code>json_decode(<i>string</i>)</code>	Parse the given JSON data string and returns an equivalent associative array object

PHP String Functions

Function	Description
<code>strlen(s)</code>	Returns the length of a string <code>s</code>
<code>strpos(str, substr)</code>	Returns the position of the first occurrence of <code>substr</code> in <code>str</code> , or <code>FALSE</code> if not found
<code>substr(s, start, len)</code>	Returns a substring of <code>s</code> starting at <code>start</code> and up to <code>len</code> characters in length. If <code>s</code> is less than <code>start</code> characters long, <code>FALSE</code> will be returned
<code>trim(s)</code>	Strips whitespace characters from both ends of a string <code>s</code>
<code>strtolower(s)</code>	Returns a lowercase version of <code>s</code>
<code>strtoupper(s)</code>	Returns an uppercase version of <code>s</code>
<code>explode(delimiter, s)</code>	Returns an array of substrings of <code>s</code> split by <code>delimiter</code>

PHP File Functions

Function	Description
<code>file(path)</code> <code>file(path, FILE_IGNORE_NEW_LINES)</code> <code>file(path, SKIP_EMPTY_LINES)</code>	Reads entire file <code>path</code> into an array. Optional <code>flags</code> parameter can be passed in such as <code>FILE_IGNORE_NEW_LINES</code> or <code>FILE_SKIP_EMPTY_LINES</code>
<code>file_exists(path)</code>	Returns whether a file or directory <code>path</code> exists
<code>file_get_contents(path)</code>	Reads entire file <code>path</code> into a string
<code>file_put_contents(path, data)</code>	Writes a string <code>data</code> to a file <code>path</code>
<code>scandir(path)</code>	Returns an array of all files and directories inside the specified <code>path</code> including <code>.</code> and <code>..</code>
<code>glob(pattern)</code>	Returns an array of path names matching <code>pattern</code>
<code>basename(path)</code>	Given a filename <code>path</code> , this function will strip any leading directory from a file path and return just the filename

PHP Superglobals Reference

Variable	Description
<code>\$_GET</code>	Superglobal array which contains query parameters passed in via a <code>GET</code> request
<code>\$_POST</code>	Superglobal array which contains <code>POST</code> parameters passed in via a <code>POST</code> request

PHP PDO Functions (with mysql)

Note that for some PDO object `$db`, you can call some function `fxn` using `$db->fxn(...)`.

Function	Description
<code>new PDO('mysql:dbname=database;host=yourhost', username, password)</code>	Constructor, connecting to the database using the given <code>yourhost</code> host value, username, and password
<code>setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION)</code>	Sets PDO error-handling properties
<code>query(sqlquery)</code>	Returns a <code>PDOStatement</code> (that contains a result set) after executing <code>sqlquery</code> in the PDO's connected database
<code>exec(sqlquery)</code>	Executes a SQL statement. Returns the number of affected rows.
<code>prepare(statement)</code>	Prepares a SQL statement to be executed by the <code>execute(arr)</code> method. The SQL statement can contain zero or more named (<code>:name</code>) parameter markers for which real values will be substituted when the statement is executed.

PDOStatement Functions

A `PDOStatement` represents a prepared statement and, after the statement is executed, an associated result set. You can retrieve the rows using a `foreach` loops, `fetch()`, or `fetchAll()`. These functions are also used with `$stmt->fxn(...)` syntax.

Function	Description
<code>execute(arr)</code>	Executes the prepared statement, filling in the named or question mark parameters with real values from the associative array. Returns TRUE if database was changed as a result, otherwise FALSE.
<code>fetch()</code>	Returns the next row from the result set. Can provide <code>FETCH::ASSOC</code> for associative array, <code>FETCH::NUMBER</code> for index-based array (default is both)
<code>fetchAll()</code>	Returns an array containing all rows in a <code>PDOStatement</code> , where each row is represented as an array (can also use <code>FETCH::ASSOC</code> and <code>FETCH::NUMBER</code> similar to <code>fetch()</code>).
<code>rowCount()</code>	Returns the number of rows in the result set.

PHP Regex Functions

Function	Description
<code>preg_match(/regex/, str)</code>	Returns whether <code>str</code> matches <code>regex</code> pattern
<code>preg_replace(/regex/, repl, str)</code>	Returns a new string with all substrings of <code>str</code> that match <code>regex</code> replaced by <code>repl</code>
<code>preg_split(/regex/, str)</code>	Returns an array of strings from given <code>str</code> split apart using given <code>regex</code> as delimiter

Regex Reference

<code>[abc]</code>	A single character of: a, b, or c	<code>.</code>	Any single character	<code>(...)</code>	Capture everything enclosed
<code>[^abc]</code>	Any single character except: a, b, or c	<code>\s</code>	Any whitespace character	<code>(a b)</code>	a or b
<code>[a-z]</code>	Any single character in the range a-z	<code>\S</code>	Any non-whitespace character	<code>a?</code>	Zero or one of a
<code>[a-zA-Z]</code>	Any single character in the range a-z or A-Z	<code>\d</code>	Any digit	<code>a*</code>	Zero or more of a
<code>^</code>	Start of line	<code>\D</code>	Any non-digit	<code>a+</code>	One or more of a
<code>\$</code>	End of line	<code>\w</code>	Any word character (letter, number, underscore)	<code>a{3}</code>	Exactly 3 of a
<code>\A</code>	Start of string	<code>\W</code>	Any non-word character	<code>a{3,}</code>	3 or more of a
<code>\Z</code>	End of string	<code>\b</code>	Any word boundary	<code>a{3,6}</code>	Between 3 and 6 of a
options: <code>i</code> case insensitive <code>m</code> make dot match newlines <code>x</code> ignore whitespace in regex <code>o</code> perform <code>#{...}</code> substitutions only once					

Special characters that need to be escaped to match as literals: `[] ^ $. | ? * + () { } \`