CSE 154: Web Programming

Final Exam "Cheat Sheet"

HTML

Tags Used in the `head` Section

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

Tags Used in the `body` Section

Tag	Description
`<p>` **text** `</p>`	paragraph

`<h1>` **text** `</h1>`
`<h2>` **text** `</h2>`
...
`<h6>` **text** `</h6>`
	(h1 for largest to
	h6 for smallest)

`<hr />`	horizontal rule (line)
`<br />`	line break
`<a href="`**url**`">` **text** `</a>`	anchor (link)
`<img src="`**url**`" alt="`**description**`" />`	image
`<em>` **text** `</em>`	emphasis (italic)
`<strong>` **text** `</strong>`	strong emphasis (bold)

`<ol>`
  `<li>` **text** `</li>`
  `<li>` **text** `</li>`
  `<li>`
    `<ul>`
      `<li>` **nested item** `</li>`
      `<li>` **nested item** `</li>`
    `</ul>`
  `</li>`
`</ol>`
	ordered (`ol`) and unordered (`ul`) list;
	list item (`li`)

Tags Used in the `body` Section (Continued)

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

Content-Grouping Tags

Tag	Display	Description
`<header>`	Block	Container for a header of a document
`<footer>`	Block	Container for a footer of a document
`<article>`	Block	A standalone piece of content (e.g., entire blog post including title, author, etc.)
`<section>`	Block	A piece of content that is part of another (e.g., a chapter section of a reading)
`<aside>`	Block	Defines some content aside from the content it is placed in (e.g., a sidebar in an article)
`<main>`	Block	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.)

HTML Input Tags

Tag	Description
`<input type="`**type**`" name="`**name**`">` **content** `</input>`	form input tag type can be `text`, `submit`, `reset`, `checkbox`, `radio`, `file`
`<textarea rows="`**num**`">` **initial text** `</textarea>`	multi-line **text** input box
`<label>` **text** `</label>`	clickable **text** label around a form control

`<select>`
`<option>` **text** `</option>`
`<option>`
`<optgroup label="text">`
`<option>` **text** `</option>`
`<option>` **text** `</option>`
`</optgroup>`
...
`</select>`
	drop-down selection box (`select`);
	each option within the box (`option`);
	a labeled group of options (`optgroup`);

`<fieldset>`
`<legend>` **text** `</legend>` content
`</fieldset>`
	a grouped set of form fields

HTML Entities Reference

Result	Description	Entity Name
	non-breaking space	`&nbsp;`
<	less than	`&lt;`
>	greater than	`&gt;`
&	ampersand	`&amp;`
©	copyright	`&copy;`

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`).

Background Styles

Property	Values
`background-attachment`	`scroll | fixed`
`background-color`	*`color`* `| transparent`
`background-image`	*`url`* `| 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`
	`[`*`x-% y-%`*`] | [`*`x-pos y-pos`*`]`

`background-size`	*`length`* `|` *`%`* `| 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 | [`*`h-shadow v-shadow blur spread color`*`]`
`border-radius`	*`length`*

Box Model

Property	Values
`float`	`left | right | none`
`height`, `width`	`auto |` *length* `|` *%*

`min-height`, `max-height`
 `min-width`, `max-width`
	`none |` *length* `|` *%*

`margin`, `margin-*`	`auto |` *length* `|` *%*
`padding`, `padding-*`	*length* `|` *%*

`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'
`justify-content`	`flex-end | flex-start | center | space-around | space-between`	Flex container	Indicates how to position the flex-items when the parent container
`flex-direction`	`row | row-reverse | column | column-reverse`	Flex container	Indicates whether the container flows horizontally (`row`) or vertically (`column`)
`align-items`	`flex-end | flex-start | center | baseline | stretch` (default)	Flex container	Indicates how to space the items inside the container along the cross axis
`flex-basis`	`auto` (default) `|` *length* `|` *%*	Both	Specifies the default size of an element before the extra space is distributed among the flex-items
`order`	*number*	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)
`align-self`	`flex-end | flex-start | center | baseline | stretch` (default)	Flex item	Indicates where to place this specific item along the cross axis

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 | [*color length*]
letter-spacing, word-spacing	normal | *length* | *%*
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
list-style-image	none | *url*

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

Selector Types

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

JavaScript

DOM Methods and Properties

Method/Property	Description
`children`	Returns a collection of an element's child elements
`parentNode`	Returns the parent node of an element
`classList`	Returns the class name(s) of an element
`className`	Sets or returns the value of the class attribute of an element
`appendChild(child)`	Adds a new child node, to an element as the last child node
`addEventListener(event, fn)`	Attaches an event handler to the specified element
`getAttribute(attr)`	Returns the specified attribute value `attr` of an element node
`innerHTML`	Sets or returns the content of an element
`id`	Sets or returns the value of the id attribute of an element
`removeChild(child)`	Removes a child node from an element
`querySelector(selector)`	Returns the first child node that matches a specified CSS selector(s) of an element
`querySelectorAll(selector)`	Returns all child nodes that match a specified CSS selector(s) of an element
`getElementsByClassName(name)`	Returns a `NodeList` containing all elements with the specified class name
`getElementById(id)`	Returns the element that has the ID attribute with the specified value
`getElementsByTagName(tagName)`	Returns a `NodeList` containing all elements with the specified tag name
`createElement(elType)`	Creates and returns an Element node
`createTextNode`	Creates and returns a Text node

Event Object Methods and Properties

Method/Property	Description
`target`	Returns the element that triggered the event
`type`	Returns the name of the event
`offsetX`	Returns the horizontal coordinate of the mouse pointer, relative to the DOM element clicked
`offsetY`	Returns the vertical coordinate of the mouse pointer, relative to the DOM element clicked
`stopPropagation`	Prevents further propagation of an event during event flow

Event Types

`click`	`mousemove`	`keydown`	`change`
`dblclick`	`mouseout`	`error`	`focus`
`mouseenter`	`mouseover`	`success`	`submit`
`mouseleave`	`mouseup`	`load`	`select`
`mousedown`	`keyup`	`unload`	`resize`

JavaScript JSON Methods

Function	Description
`parse(string)`	Returns the given string of JSON data as the equivalent JavaScript object
`stringify(object)`	Returns the given object as a string of JSON data

JavaScript Array Methods and Properties

Method/Property	Description
`length`	Sets or returns the number of elements in an array
`push(el)`	Adds new elements to the end of an array and returns the new length
`pop()`	Removes and returns the last element of an array
`unshift(el)`	Adds new elements to the beginning of an array and returns the new length
`shift()`	Removes and returns the first element in an array
`sort()`	Sorts the elements of an array
`slice(start, end)`	Selects a part of an array and returns the new array
`join()`	Joins all elements of an array into a string
`concat(list2, ...)`	Joins two or more arrays and returns a copy of the joined arrays
`toString()`	Converts an array to a string and returns the result
`indexOf(el)`	Returns the index of the element in the array, or -1 if not found

JavaScript String Methods and Properties

Method/Property	Description
`length`	Returns the length of a string
`charAt(index)`	Returns the character at the specified index
`indexOf(string)`	Returns the position of the first found occurrence of a specified value in a string
`split(delimiter)`	Splits a string into an array of substrings
`substring(start, end)`	Extracts the characters from a string between two specified indices
`trim()`	Removes whitespace from both ends of a string
`toLowerCase()`	Returns a lowercase version of a string
`toUpperCase()`	Returns an uppercase version of a string
`concat(str2, ...)`	Joins two or more strings and returns a new joined string

JavaScript Timer Functions

Method	Description
`setTimeout(fn, ms)`	Executes a function after waiting a specified number of milliseconds
`setInterval(fn, ms)`	Repeats a given function at every given time-interval
`clearTimeout(id)`	Stops the execution of the function specified by id
`clearInterval(id)`	Stops the execution of the functions specified by id

JavaScript Math Functions

Method	Description
`Math.random()`	Returns a double between 0 (inclusive) and 1 (exclusive)
`Math.abs(n)`	Returns the absolute value of `n`
`Math.min(a, b, ...)`	Returns the smallest of 0 or more numbers
`Math.max(a, b, ...)`	Returns the largest of 0 or more numbers
`Math.round(n)`	Returns the value of `n` rounded to the nearest integer
`Math.ceil(n)`	Returns the smallest integer greater than or equal to `n`
`Math.floor(n)`	Returns the largest integer less than or equal to `n`
`Math.pow(n, e)`	Returns the base `n` to the exponent `e` power, that is, n^e
`Math.sqrt(n)`	Returns the square root of `n` (`NaN` if `n` is negative)

The Module Pattern

Whenever writing JavaScript, you should use the module pattern, wrapping the content of the code (`window.onload` handler and other functions) in an anonymous function. Below is a template for reference:

```
(function() {
  // any module-globals (limit the use of these when possible)

  window.onload = function() {
    ...
  };

  // other functions
})();
```

Javascript Ajax fetch skeletons

```
//you can assume checkStatus is already included
function checkStatus(response) {
    if (response.status >= 200 && response.status < 300) {
        return response.text();
    } else {
        return Promise.reject(new Error(response.status+": "+response.statusText));
    }
}

function callAjaxGET(){
    let url = ..... // put url string here
    fetch(url) // don't worry about cloud9 credentials
       .then(checkStatus)
       .then(JSON.parse) //optional line for processing json
       .then(function(responseJSON) {
            //success: do something with the responseJSON
        })
       .catch(function(error) {
           //error: do something with error
       });
}
```

```
function callAjaxPOST(){
    let data = new FormData();
    data.append("key", value);

    let url = ..... // put url string here

    fetch(url, {method: "POST", body: data}) // don't worry about cloud9 credentials
       .then(checkStatus)
       .then(JSON.parse) //optional line for processing json
       .then(function(responseJSON) {
            //success: do something with the responseJSON
        })
       .catch(function(error) {
           //error: do something with error
       });
}
```

PHP

PHP Array Functions

Function	Description
`count(arr)`	Returns the length of an array `arr`
`print_r(arr)`	Prints the `arr`'s contents
`array_pop(arr)`	Pops (removes) an element off the end of the array `arr`
`array_shift(arr)`	Shifts (removes) an element off the beginning of the array `arr`
`array_push(arr, el)`	Pushes (adds) one or more elements onto the end of the array `arr`
`array_unshift(arr, el)`	Prepends one or more elements to the beginning of the array `arr`
`sort(arr)`	Sorts the array `arr`
`array_reverse(arr)`	Returns an array with elements of `arr` in reverse order
`in_array(el, arr)`	Returns whether a value `el` exists in an array `arr`
`list(a, b, ...)`	Assigns variables as if they were an array
`implode(glue, pieces)`	Joins array elements (`pieces`) with a string (`glue`)

PHP String Functions

Function	Description
`strlen(s)`	Returns the length of a string `s`
`strpos(str, substr)`	Returns the position of the first occurrence of `substr` in `str`, or -1 if not found
`substr(s, start, length)`	Returns a substring of `s` starting at `start` and up to `length` characters in length
`trim(s)`	Strips whitespace (or other characters) from both ends of a string
`strtolower(s)`	Returns a lowercase version of `s`
`strtoupper(s)`	Returns an uppercase version of `s`
`explode(delimiter, s)`	Returns an array of substrings of `s` split by `delimiter`
`join(glue, pieces)`	Joins `pieces` using `glue`

PHP Standard Functions

Function	Description
`isset(el)`	Returns whether `el` is `NULL`
`print(el)`	Prints `el`
`time()`	Returns the current time in seconds
`date(format, time)`	Converts an optional `time` in seconds to a date based on `format`
`header(string)`	Sends a raw HTTP header (e.g., “`Content-type: text/plain`” or “`Content-type: application/json`”)
`die(message)`	Ends execution and sends back optional `message`
`include(path)`	Includes and evalutes the specified file `path`

PHP JSON Functions

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

PHP File Functions

Function	Description
`file(path)`	Reads entire file `path` into an array
`file_exists(path)`	Returns whether a file or directory `path` exists
`file_get_contents(path)`	Reads entire file `path` into a string
`file_put_contents(path, data)`	Writes a string `data` to a file `path`
`scandir(path)`	Lists files and directories inside the specified `path`
`glob(pattern)`	Lists path names matching `pattern`

PHP Session and Cookie Functions

Function	Description
`setcookie(name, val, expiration)`	Sends a cookie with given name and value to the user's browser, with optional expiration time given in seconds
`session_start()`	Loads existing session data or starts a new session if one doesn't exist
`session_destroy()`	Destroys old session data
`session_regenerate_id(delID)`	Regenerates session id for next session, and optionally also deletes old `delID`

PHP Regex Functions

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

PHP PDO Functions (with `mysql`)

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

Function	Description
`new PDO('mysql:dbname=database;host=yourhost', username, password')`	Constructor, connecting to the `database` using the given `yourhost` host value
`setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION)`	Sets PDO error-handling properties
`query(sqlquery)`	Returns all result rows as a `PDOStatement` (associative array of `[column -> value]`) after executing `sqlquery` in the PDO's connected database
`exec(sqlquery)`	Executes a SQL statement and returns the number ofaffected rows
`quote(var)`	Escapes any illegal characters and surrounds them with ' quotes

PDOStatement Functions

Used by the returned `$rows` variable returned by PDO's `query` function. These functions are also used with `$rows->fxn(...)` syntax.

Function	Description
`columnCount()`	Returns the number of columns in the result set.
`fetch()`	Returns the next row from the result set.
`fetchColumn(number)`	Returns the next column from the result set.
`rowCount`	Returns the number of rows in the result set.

HTTP Status Code Reference

Code	Description
200	`OK`
400	`Bad Request`
401	`Unauthorized`
404	`Not Found`
410	`Gone`
500	`Internal Server Error`

PHP Superglobals Reference

Variable	Description
`$_GET`	Superglobal array which contains query parameters passed in via a `GET` request
`$_POST`	Superglobal array which contains `POST` parameters passed in via a `POST` request

Regex Reference

`[abc]`	A single character of: a, b, or c	`.`	Any single character	`(...)`	Capture everything enclosed
`[^abc]`	Any single character except: a, b, or c	`\s`	Any whitespace character	`(a|b)`	a or b
`[a-z]`	Any single character in the range a-z	`\S`	Any non-whitespace character	`a?`	Zero or one of a
`[a-zA-Z]`	Any single character in the range a-z or A-Z	`\d`	Any digit	`a*`	Zero or more of a
`^`	Start of line	`\D`	Any non-digit	`a+`	One or more of a
`$`	End of line	`\w`	Any word character (letter, number, underscore)	`a{3}`	Exactly 3 of a
`\A`	Start of string	`\W`	Any non-word character	`a{3,}`	3 or more of a
`\z`	End of string	`\b`	Any word boundary	`a{3,6}`	Between 3 and 6 of a

options: **i** case insensitive **m** make dot match newlines **x** ignore whitespace in regex **o** perform #{...} substitutions only once

SQL

`SELECT`

Description: Used to select data from a database table. If `DITINCT` is used, no duplicate rows are returned.

Syntax (without `DISTINCT`):

```
SELECT column(s)
FROM table;
```

Syntax (with `DISTINCT`):

```
SELECT DISTINCT column(s)
FROM table;
```

`WHERE`

Description: Used to filter records, returning only those which meet provided conditions.

Syntax:

```
SELECT column(s)
FROM table
WHERE condition(s);
```

Condition types:

- `=, >, >=, <, <=`
- `<>` (not equal)
- `BETWEEN min AND max`
- `LIKE %pattern` (where % is a wildcard)
- `LIKE pattern%`
- `LIKE %pattern%`

ORDER BY

Description: Used to sort the result set in ascending (default) or descending order.

Syntax:

```
SELECT column(s)
FROM table
ORDER BY column(s) ASC|DESC;
```

LIMIT

Description: Used to give the top-n elements of a given category.

Syntax:

```
SELECT column(s)
FROM table
LIMIT number;
```

CREATE TABLE

Description: Used to create a new table.

Syntax:

```
CREATE TABLE table_name(
  column1 datatype,
  column2 datatype,
  ...
  columnN datatype,
  PRIMARY KEY (one or more columns)
);
```

Record Data Types:

- `VARCHAR(N)` - strings of up to N characters (e.g., 'Whitaker')
- `INTEGER` - integers (e.g., 10)
- `FLOAT` - floats (e.g., 1.54)
- `DATETIME` - date/time representation (e.g., '2017-05-25 18:20:32')

INSERT INTO

Description: Used to insert a new record (row) into an existing table, where the listed values correspond to the listed columns.

Syntax:

```
INSERT INTO table_name (column1, column2, ..., columnN)
VALUES (value1, value2, ..., valueN);
```

DELETE

Description: Used to remove a record (row) which matches condition(s) from an existing table.

Syntax:

```
DELETE FROM table_name
WHERE condition;
```

UPDATE

Description: Used to modify the existing records in a table.

Syntax:

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition(s);
```

JOIN

Description: Used to select values from more than one table.

Syntax:

```
SELECT col(s)
FROM table1, table2, ...
WHERE table1.a = table2.b
AND table2.c > '42';
```

OR

```
SELECT col(s)
FROM table1
JOIN table2 on table1.a = table2.b
JOIN ...
AND table2.c > '42';
```