CSE 154: Web Programming

Midterm 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)
` `	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	` `
<	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 skeleton

```
//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 callAjax(){
    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
       });
}
```