CS11 JS "Cheat Sheet"

This reference summarizes the most useful methods/properties used in the CS11 JavaScript Track. It is not an exhaustive reference for everything in JavaScript (for example, there exist many more `window` methods/properties than are shown below), but provide most functions/properties you will be using in this class.

The Module Pattern

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

```
"use strict";
(function() {

  // any module-globals (limit the use of these when possible)
  window.addEventListener("load", init);

  function init() {
    ...
  }

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

Handy Alias Functions

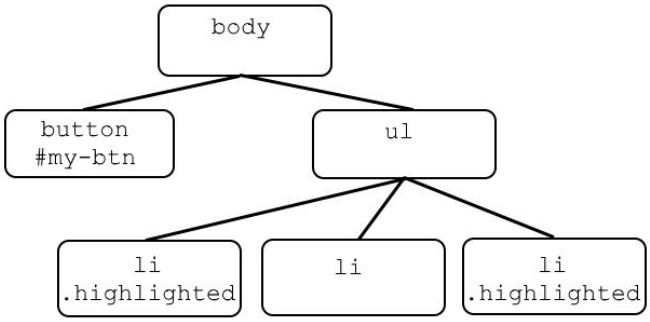
The following four shorthand functions will be used frequently in the class.

```
function id(idName) {
  return document.getElementById(idName);
}

function qs(selector) {
  return document.querySelector(selector);
}

function qsa(selector) {
  return document.querySelectorAll(selector);
}

function gen(elType) {
  return document.createElement(elType);
}
```

Accessing DOM elements from the `document`

These are methods/properties than can be accessed from the global `document` object, for example:

```
document.getElementById("my-btn");
```



Method/Property and Example	Description
`getElementById`(*`idName`*) `getElementById("my-btn");`	Returns a DOM object whose id property matches the specified string. If no matches are found, null is returned.
`querySelectorAll`(*`selector`*) `querySelectorAll("li.highlighted");`	Returns a list of the document's elements that match the specified group of selectors. If no matches are found, null is returned.
`querySelector`(*`selector`*) `querySelector("li.highlighted");`	Returns the first DOM element that matches the specified selector, or group of selectors. If no matches are found, null is returned.

DOM Element Methods and Properties

Recall that if you have an HTML element on your page that has attributes, you can set those properties through JavaScript as well. For instance if your

```
<img id="dog-tag" src="img/doggie.jpg" alt="My Cute Dog" />
```

Your could do the following in your JavaScript code (using the `id` alias for `document.getElementById`):

```
id("dog-tag").alt = "My really cute dog";
```

Example DOM Element attributes include (other than `src`, and `alt` above) are:

Property	Description
`el.`**`disabled`**	Whether or not the DOM element (e.g. a button or input) is disabled
`el.`**`value`**	The value attribute of form elements (`input`, `textarea`, `checkbox` `radio`, `select`, etc.)
`el.`**`name`**	The value of the name attribute of a form element
`el.`**`href`**	The `href` attribute for <`link`> or <`a`> element
`el.`**`id`**	The id attribute of an element

These are methods/properties for DOM elements. For example:

```
let ol = document.getElementById("my-list");
ol.children; // collection of all DOM elements as direct children in <ol id="my-list">...</li>
```

DOM Method/Property	Description
`el.`**`textContent`**	Sets or returns the text content of the specified node
`el.`**`innerHTML`**	Sets or returns the HTML content of an element
`el.`**`getAttribute`**`(`*`attr`*`)`	Returns the specified attribute value `attr` of `el`
`el.`**`children`**	Returns a collection of the child elements of `el`
`el.`**`parentNode`**	Returns the parent node of `el`
`el.`**`classList`**	Returns the class name(s) of `el`
`el.`**`className`**	Sets or returns the value of the class attribute of `el`

DOM Manipulation Methods

Method/Property and Example	Description

`document.createElement(tagname)`
 	
`let li = document.createElement("li");`
	Creates and returns an Element node. Note that this method is used on **document** not a DOM node.

`el.appendChild(child)`
 	
`ol.appendChild(li);`
	Adds a new child node to `el` as the last child node

`el.removeChild(child)`
 	
`ol.removeChild(li);`
	Removes a child node from an element

`el.insertBefore(newNode, refNode);`
 	
`ol.insertBefore(newLi, existingLi);`
	Adds `newNode` to parent `el` before `el`'s child `refNode` position

DOM and Events

DOM Method/Property	Description
`el.addEventListener(event, fn)`	Attaches an event handler function `fn` to the specified element `el` to listen to `event`
`el.removeEventListener(event, fn)`	Removes the event handler `fn` to the specified `el` listening to `event`

Other DOM Element Properties

Recall that if you have an HTML element on your page that has attributes, you can set those properties through JavaScript as well. For instance if you have an image element:

```
<img id="dog-tag" src="img/doggie.jpg" alt="My Cute Dog" />
```

Your could do the following in your JavaScript code (using the `id` alias for `document.getElementById`):

```
id("dog-tag").alt = "My really cute dog";
```

Example DOM Element attributes include (other than `src`, and `alt` above) are:

Property	Description
`el.`**`disabled`**	Whether or not the DOM element (e.g. a button or input) is disabled
`el.`**`value`**	The value attribute of form elements (`input`, `textarea`, `checkbox` `radio`, `select`, etc.)
`el.`**`name`**	The value of the name attribute of a form element
`el.`**`href`**	The `href` attribute for <`link`> or <`a`> element
`el.`**`id`**	The id attribute of an element

DOM Element `.classList` Methods

Method	Description

`el.classList.add``(class)`
 	
`div.classList.add("skittle")`
`div.classList.add("skittle", "green");`
	Adds specified class values. These values are ignored if they already exist in the list.

`el.`**`classList.remove`**`(class)`
 	
`div.classList.remove("green");`
	Removes the specified class value

`el.`**`classList.toggle`**`(class)`
 	
`div.classList.toggle("hidden");`
	Toggles the listed class value. If the class exists, then removes it and returns false, if it did not exist in the list add it and return true

`el.`**`classList.contains`**`(class)`
 	
`div.classList.contains("highlighted");`
	Returns true if the specified class value is exists in the classList for this element

Useful Event Object Methods and Properties

Any function assigned in an `addEventListener` can accept an optional argument, which will be the Event object. These are some things you can do with that object.

```
function init() {
  id("my-btn").addEventListener("click", handleClick);
}

function handleClick(evt) {
  console.log(evt.target); // #my-btn
  console.log(this);       // #same as above
}
```

Method	Description
`evt.target`	Returns the element that triggered the event
`evt.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
`timestamp`	Timestamp (in ms) the event object was created.

Event Types

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

JavaScript `string` Methods and Properties

Method	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

JavaScript Array Methods and Properties

Method	**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
`join()`	Returns a string concatenating all elements of an array (maintaining order)
`indexOf(el)`	Returns the index of the element in the array, or -1 if not found

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

JavaScript Timer Functions

Method	Description
`setTimeout(fn, ms)`	Executes a function `fn` after a delay of `ms` milliseconds. Returns a value representing the ID of the timeout being set.
`setInterval(fn, ms)`	Executes a function `fn` at every given time-interval (in milliseconds). Returns a value representing the ID of the interval being set.
`clearTimeout(id)`	Stops the execution of the delay timer specified by id
`clearInterval(id)`	Stops the execution of the interval timer specified by id

JavaScript JSON Functions

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

Other Handy JavaScript Functions

Function	Description
`parseInt(data, [radix])`	Parses an argument and returns an integer of the specified (optional) radix (the base in mathematical numeral systems). If no radix is passed, returns the integer as base-10. Returns `NaN` if data cannot be parsed as an integer.
`console.log(data)`	Prints the `data` to the JavaScript console

`window` Methods and Properties

Method/Property	Description
`getComputedStyle(element)`	Returns an object that reports the values of all CSS properties of an element after applying active stylesheets and resolving any basic computation those values may contain

Javascript AJAX Fetch Skeleton

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

// This is an example template for how to make an AJAX fetch request
function makeRequest(){
  let url = ..... // put url string here
  fetch(url)
    .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
    });
}
```