JAVASCRIPT CHEAT SHEET

CORE JAVASCRIPT

ARRAY METHODS

array.length: returns the number of elements in the array
array.concat(v1, v2, …, vn): joins two-or more arrays into a single array.
array.join(delimiter): returns the elements as a string where the elements are separated by a delimiter.
array.pop(): removes the last item from the array and returns the value
array.push(item): adds the item as the last element in the array and returns the new length.
array.reverse(): reverses the order of the elements in the array
array.slice(start, end): returns a sub-array that includes elements at indices start through end-1.
array.sort(); sorts the array elements

STRING METHODS

s.length: returns the number of characters in the string
s.charAt(n): returns the character having index n
s.indexOf(s2): returns the index of substring s2 in s or -1 if s2 is not contained in s.
s.substr(start, end): returns the substring of s including all characters whose indices are in start to end-1.
s.toLowerCase(): returns a copy of s in lower-case characters
s.toUpperCase(): returns a copy of s in upper-case characters.

JAVASCRIPT DOM

OBTAINING DOCUMENT ELEMENTS

document.getElementById('id'): Retrieves the element with the given id as an object
document.getElementsByTagName('tagname'): Retrieves all elements with the tag name tagname and stores them in an arraylike list

READING ELEMENT ATTRIBUTES, NODE VALUES AND OTHER DATA

node.getAttribute('attribute'): Retrieves the value of the attribute with the name attribute
node.setAttribute('attribute', 'value'): Sets the value of the attribute with the name attribute to value

NAVIGATING BETWEEN NODES

node.previousSibling: Retrieves the previous sibling node and stores it as an object.
node.nextSibling: Retrieves the next sibling node and stores it as an object.
node.childNodes: Retrieves all child nodes of the object and stores them in a list.
node.firstChild: Retrieves the first child node
node.lastChild: Retrieves the last child node
node.parentNode: Retrieves the node containing node.

CREATING NEW NODES

document.createElement(elemName): Creates a new element node with the given name (provided as a string).
document.createTextNode(string): Creates a new text node with the node value of string.
node.cloneNode(bool): Creates a copy (clone) of node. If bool is true, the clone is a deep copy otherwise shallow.
node.appendChild(newNode): Adds newNode as a new (last) child node to node.
node.insertBefore(newNode,oldNode): Inserts newNode as a new child node of node before oldNode.
node.removeChild(oldNode): Removes the child oldNode from node.
node.replaceChild(newNode, oldNode): Replaces the child node oldNode of node with newNode.

element.innerHTML: Reads or writes the HTML content of the given element as a string—including all child nodes with their attributes and text content.

JAVASCRIPT REGULAR EXPRESSIONS

REGULAR EXPRESSION FLAGS/MODIFIERS

g	global match
i	ignore case
m	multiline
y	sticky
gi	both global match and ignore case (demonstrates the use of multiple flags)

METHODS THAT USE REGULAR EXPRESSIONS

RegExp.exec

var results = objRegex.exec("this is a string");

Executes a search for a match in a string. It returns an array of information.

RegExp.test

var exists = objRegex.test("this is a string");

Tests for a match in a string. It returns true or false.

String.match

var matches = "my string".match(objRegex);

Executes a search for a match in a string. It returns an array of information or null on a mismatch.

String.search

var intPosition = "my string".search(objRegex);

Tests for a match in a string. It returns the index of the match, or -1 if the search fails.

String.replace

var newString = "my string".replace(objRegex, "replacement");

Searches for a match in a string, and replaces the matched substring with a replacement substring.

String.split

var results = "my string".replace(objRegex);

Uses a regular expression or a fixed string to break a string into an array of substrings.

REGULAR EXPRESSION SPECIAL CHARACTERS

\	Escape character.
^	Matches beginning of input or line.
$	Matches end of input or line.
*	Matches 0 or more instances of preceding character.
+	Matches 1 or more instances of preceding character.
?	Matches 0 or 1 instances of preceding character.
.	Matches any single character other than the newline / carriage return character.
(x)	Matches x and remembers the match.
(?:x)	Matches x but does not remember the match (? represents non-capturing parentheses).
x(?=y)	Matches x only if followed by y (but y is not part of match results).
x(?!y)	Matches x only if not followed by y.
x|y	Matches either x or y.
{n}	Matches exactly n instances of preceding character (where n is an integer).
{n,}	Matches at least n instances of preceding character (where n is an integer).
{n,m}	Matches at least n and at most m instances of preceding character (where n and m are integers).
[xyz]	Matches any one of enclosed characters (specify range using hyphen, such as [0-9].
[^xyz]	Matches any character not enclosed (specify range using hyphen, such as [^0-9].