

## EECS 1012: JavaScript Cheat sheet (3 pages)

Link to HTML (in <head> </head> section)

```
<script src="file.js"></script>
```

### Data Types

Number, Boolean, Array, String, Object

### Declaring a Variable

```
var variableName = <value>;
```

### Initialize and Accessing an Array

```
var arrayname = [ item1, item2, item3 ];  
e.x: var a = ["one", "two", "three"];  
var b = a[1];           returns "two"
```

### Arithmetic Operators (for Numbers)

+ (addition), - (subtraction), \* (multiply), / (division)

### Relational Operators

== (Equal), != (Not Equal), < (Less than), > (Greater than), <= (Less than or equal), >= (Greater than or equal)

### Array length method

```
var a = ["one", "two", "three", ... "last"];  
var x = a.length; (returns size of the array)
```

### String operator and length method

```
+ (concatenate operator)  
var s = "hello" + " world"; // "hello world"  
var x = s.length; (returns length of the string)
```

### Function declaration

```
function name() {  
    statements;  
}  
  
function () {           /* anonymous function */  
    statements;  
}
```

### Global variables

```
var x=0; /* declared outside any function */  
function func1()  
{  
    x can be used here (or any other  
    function).  
    x keeps its value after function call.  
}
```

### Alert Popup (useful for debugging)

```
alert("text"); Creates an alert popup
```

### Console output (for debugging)

```
console.log("output string");
```

### if-else

```
if (condition1) {  
    statements;  
}  
else {  
    statements;  
}
```

### for-loop

```
for (<initialize>; <condition>; <update>)  
{  
    statements;  
}
```

Ex:

```
for(var i=0; i < 4; i++)  
{  
    ...  
}
```

### while-loop

```
while (<condition>)  
{  
    statements;  
}
```

### DOM Tree manipulation

```
var a = document.getElementById("id");  
returns single element object of id="id"  
returns null if "id" is not found
```

```
var a = document.getElementsByTagName("tag");  
returns an array of elements of type "tag" (e.g. p, h1)  
returns null if "tag" is not found
```

*Above methods above can also be called by elements*

```
var mydiv = document.getElementById("mydiv");  
var allP = mydiv.getElementsByTagName("tag");  
returns an array of elements of type "tag" that are  
descendant of element mydiv.
```

### Elements siblings

```
var myElement = document.getElementById("id");  
var b = myElement.nextElementSibling();  
returns the next sibling of myElement
```

### Children

```
var mydiv = document.getElementById("mydiv");  
var allC = mydiv.children;  
returns an array of all children nodes for mydiv
```

(continues on next page)

## DOM Tree manipulation continues

### Creating/Deleting Elements in the DOM Tree

```
var x = document.createElement("tag");
    creates a new element
x.innerHTML = "...";
    can be used to modify non-void element
var mydiv = document.getElementById("mydiv");
mydiv.appendChild(x); adds the element
mydiv.removeChild(mydiv.children[0]);
    removes an element
```

### DOM Object properties

```
var a = document.getElementById("id");
a.innerHTML /* accesses innerHTML of a */
```

Modifying style

```
a.style.backgroundColor="yellow";
a.style.color="red";
a.style.fontFamily = "serif";
a.style.fontSize = "2em";
```

For checkboxes, radiobuttons, disabled elements

```
a.checked /* bool if checkbox or radio is checked */
a.disabled /* bool if element is disabled or not */
```

For images

```
a.src = "image.jpg"; (changes img element src)
```

For form input

```
a.value (access input form's value, including <select>)
a.value = ""; (value can also be set)
(note: input fields have no innerHTML)
```

### Setting click event handler

```
a.onclick = functionName;
    Assigns onclick event to call function functionName.
    Remember do not to use () after function name.
```

### Window onload

```
window.onload = functionName;
    calls function after document loads
```

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

anonymous function version of onload

### Form submit

```
var a = document.getElementById("formElement");
a.submit(); /* $("a").submit() also works */
    submits the form, note: var a must be a form element
    in the HTML page
```

## Timers

```
setTimeout(function, delay in millisecs);
setInterval(function, delay in millisecs);
    Timeout is called only 1 time
    Interval is called repeatedly
    both calls return ids, e.g. id = setInterval(myfunc, 200);
```

```
clearTimeout(id); removes timeout timer
```

```
clearInterval(id); removes interval timer
```

## PROTOTYPE LIBRARY

### linking to HTML

```
<script src="prototype.js"></script>
```

### The dollar sign (\$) function

```
$("#id") is equivalent to
    document.getElementById("id");
```

Example uses:

```
$("#id").innerHTML = "";
    sets the innerHTML of "id" to empty
$("#id").onclick = functionName;
$("#id").style.backgroundColor = "yellow";
$("#id").getElementsByTagName("p");
```

### Prototype event handler

```
$("#id").observe("event", functionName);
```

Event should be passed as a string.

Mouse events

- click - when mouse is clicked
- mousedown – mouse button press down
- mouseover – mouse moves over element
- mouseout – mouse exits element
- mousemove – mouse moves within element

Keyboard events (typically used with form elements)

- keydown – key is pressed down
- keyup - key is released
- keypress – key is pressed (down and up)

Ex. usage: `$("#id").observe("mousedown", myFunc);`

### Event Object

Prototype allows functions that pass an *event* object:

```
function myFunction(event) {
    if (event.type == "click") ....
}
```

## Event methods and properties

event.type - type of event, e.g. "click" (see list above)  
event.stopObserving(); – removes an event handler

## Mouse event positions

event.clientX /\* coords within browser window \*/  
event.clientY  
event.offsetX /\* coords within element \*/  
event.offsetY  
event.pageX() /\* coords on entire webpage \*/  
event.pageY()

## Regular Expression

```
var re=/pattern/;  
re.test(string); /* returns true or false */
```

### Example character ranges

[ADC]	– matches the letter A or D or C
[9bC]	– matches a 9, b, or C
[a-z]	– matches any lowercase letter
[A-Z]	– matches any uppercase letter
[0-9]	– matches any digit (a number)
[a-zA-Z]	– matches upper or lower case letter
[a-zA-Z0-9]	– matches any letter or digit
[a-zA-Z0-9\ ]	-- matches any letter, digit or space
[\\$A-Z0-9]	-- matches \$, any letter, of number

## Regex Flag

```
var re=/pattern/i; /* i means ignore case */
```

## Regex Qualifiers

### \* means 0 or more occurrences

```
/abc*/ matches "ab", "abc", "abcc", "abccc", ...  
/a*b*c*/ matches "a", "aabc", "bbcc", "aaabbbccc", ...
```

### + means 1 or more occurrences

```
/ab+c+/ matches "abc", "abbbcc", "abccc", ...  
/Goo+gle/ matches "Google", "Goooogle", "Goooooogle", ...
```

### ? means 0 or 1 occurrences

```
/Martina?/ matches "Martin" or "Martina"  
/A?B?C?/ matches "AC", "BC", "ABC", "A", "B", "C", ...
```

## Regex anchors ^, \$

```
var re = /^pattern/;  
^ pattern must be at the beginning of the string  
var re = /pattern$/;  
$ pattern must be at the end of the string  
var re = /^pattern$/;  
means beginning and end must match the pattern  
(forces the string has the same number of chars as the  
pattern)
```

## Additional String Manipulation

```
String s1 = "hello";  
String s2 = null;  
s2 = s1.toLowerCase(); /* returns s1 as  
lowercase */  
s2 = s1.toUpperCase(); /* returns s2 as  
uppercase */  
  
/* converts code to string */  
var checkmark = String.fromCharCode(###);  
/* Example character codes */  
&#10004; ✓                      &#10006; ✗
```

## Convert String to Number

```
var s1="10";  
var s2="10.50";  
var num1 = parseInt( s1 ); /* returns an Integer */  
var num2 = parseFloat( s2 ); /* returns a Float */
```

## Array search

```
var myArray = ["MB", "SK", "QC", "ON"];  
  
myArray.includes(someString);  
returns true if someString is an element in the array,  
false otherwise.
```