Javascript

Cheat Sheet

Javascript Basics

Including JavaScript in an HTML Page

```
<script type="text/javascript">
  //JS code goes here
</script>
```

Call an External JavaScript File

```
<script src="myscript.js"></script><code></code>
```

Including Comments

`//`

Single line comments

`/* comment here */`

Multi-line comments

Variables

var, const, let

`var`

The most common variable. Can be reassigned but only accessed within a function. Variables defined with var move to the top when code is executed.

`const`

Cannot be reassigned and not accessible before they appear within the code.

`let`

Similar to const, however, let variable can be reassigned but not re-declared.

Data Types

`var age = 23`

Numbers

`var x`

Variables

```
var a = "init"
```
Text (strings)

```
var b = 1 + 2 + 3
```
Operations

```
var c = true
```
True or false statements

```
const PI = 3.14
```
Constant numbers

```
var name = {firstName:"John", lastName:"Doe"}
```
Objects

Objects

```
var person = {
  firstName:"John",
  lastName:"Doe",
  age:20,
  nationality:"German"
};
```

Arrays

```
var fruit = ["Banana", "Apple", "Pear"];
```

Array Methods

`concat()`

Join several arrays into one

`indexOf()`

Returns the first position at which a given element appears in an array

`join()`

Combine elements of an array into a single string and return the string

`lastIndexOf()`

Gives the last position at which a given element appears in an array

`pop()`

Removes the last element of an array

`push()`

Add a new element at the end

`reverse()`

Reverse the order of the elements in an array

`shift()`

Remove the first element of an array

`slice()`

Pulls a copy of a portion of an array into a new array

`sort()`

Sorts elements alphabetically

`splice()`

Adds elements in a specified way and position

`toString()`

Converts elements to strings

`unshift()`

Adds a new element to the beginning

`valueOf()`

Returns the primitive value of the specified object

Operators

Basic Operators

```
+  - Addition  Subtraction
*  / Multiplication
(..)  Division     Grouping
% ++  operator      Modulus
--     (remainder)
       Increment     numbers
       Decrement numbers
```

Comparison Operators

```
==    Equal to
===   Equal value and equal type
!=    Not equal
!==   Not equal value or not equal type
> <   Greater than
>=    Less than
<=    Greater than or equal to
?     Less than or equal to
      Ternary operator
```

Logical Operators

```
&&    Logical and
||    Logical or
!     Logical not
```

Bitwise Operators

```
& |   AND     statement     OR
~ ^   statement NOT XOR Left
<<    shift Right shift Zero
>>    fill right shift
>>>
```

Functions

```
function name(parameter1, parameter2, parameter3) {
  // what the function does
}
```

Outputting Data

`alert()`

Output data in an alert box in the browser window

`confirm()`

Opens up a yes/no dialog and returns true/false depending on user click

`console.log()`

Writes information to the browser console, good for debugging purposes

`document.write()`

Write directly to the HTML document

`prompt()`

Creates an dialogue for user input

Global Functions

`decodeURI()`

Decodes a Uniform Resource Identifier (URI) created by encodeURI or similar

`decodeURIComponent()`

Decodes a URI component

`encodeURI()`

Encodes a URI into UTF-8

`encodeURIComponent()`

Same but for URI components

`eval()`

Evaluates JavaScript code represented as a string

`isFinite()`

Determines whether a passed value is a finite number

`isNaN()`

Determines whether a value is NaN or not

`Number()`

Returns a number converted from its argument

`parseFloat()`

Parses an argument and returns a floating point number

`parseInt()`

Parses its argument and returns an integer

Loops

```
for (before loop; condition for loop; execute after loop) {

  // what to do during the loop
}
```

`for`

The most common way to create a loop in Javascript

`while`

Sets up conditions under which a loop executes

`do while`

Similar to the while loop, however, it executes at least once and performs a check at the end to see if the condition is met to execute again

`break`

Used to stop and exit the cycle at certain conditions

`continue`

Skip parts of the cycle if certain conditions are met

If – Else Statements

```
if (condition) {

  // what to do if condition is met
} else {
  // what to do if condition is not met
}
```

Strings

```
var person = "John Doe";
```

```
\' \" \\ \b \f \n \r \t
```

Escape Characters

– Single quote – Double quote – Backslash – Backspace – Form feed – New line – Carriage return – Horizontal tabulator

`\v` — Vertical tabulator

String Methods

`charAt()`

Returns a character at a specified position inside a string

`charCodeAt()`

Gives you the unicode of character at that position

`concat()`

Concatenates (joins) two or more strings into one

`fromCharCode()`

Returns a string created from the specified sequence of UTF-16 code units

`indexOf()`

Provides the position of the first occurrence of a specified text within a string

`lastIndexOf()`

Same as indexOf() but with the last occurrence, searching backwards

`match()`

Retrieves the matches of a string against a search pattern

`replace()`

Find and replace specific text in a string

`search()`

Executes a search for a matching text and returns its position

`slice()`

Extracts a section of a string and returns it as a new string

`split()`

Splits a string object into an array of strings at a specified position

`substr()`

Similar to slice() but extracts a substring depended on a specified number of characters

`substring()`

Also similar to slice() but can't accept negative indices

`toLowerCase()`

Convert strings to lowercase

`toUpperCase()`

Convert strings to uppercase

`valueOf()`

Returns the primitive value (that has no properties or methods) of a string object

Regular Expressions

Pattern Modifiers

```
e – Evaluate replacement
i – Perform case-insensitive matching
g – Perform global matching
m – Perform multiple line matching
s – Treat strings as single line
x – Allow comments and whitespace in pattern
U – Non Greedy pattern
```

Brackets

```
[abc]      Find any of the characters between the brackets
[^abc]     Find any character not in the brackets
[0-9]      Used to find any digit from 0 to 9
[A-z]      Find any character from uppercase A to lowercase z
(a|b|c)    Find any of the alternatives separated with |
```

Metacharacters

```
.    – Find a single character, except newline or line terminator
\w   – Word character
\W   – Non-word character
\d   – A digit
\D   – A non-digit character
\s   – Whitespace character
\S   – Non-whitespace character
\b   – Find a match at the beginning/end of a word
\B   – A match not at the beginning/end of a word
\0   – NUL character
\n   – A new line character
\f   – Form feed character
\r   – Carriage return character
\t   – Tab character
\v   – Vertical tab character
```

```
\xxx – The character specified by an octal number xxx \xdd – Character
specified by a hexadecimal number dd \uxxxx – The Unicode character
specified by a hexadecimal number xxxx
```

Quantifiers

```
n+    – Matches any string that contains at least one n
n*    – Any string that contains zero or more occurrences of n
n?    – A string that contains zero or one occurrences of n
n{X} – String that contains a sequence of X n's
n{X,Y} – Strings that contains a sequence of X to Y n's
n{X,} – Matches any string that contains a sequence of at least X n's
n$    – Any string with n at the end of it – String with n
^n ?  at the beginning of it – Any string that is followed
=n    by a specific string n – String that is not followed
?!n   by a specific string n
```

Numbers and Math

Number Properties

`MAX_VALUE`

The maximum numeric value representable in JavaScript

`MIN_VALUE`

Smallest positive numeric value representable in JavaScript

`NaN`

The "Not-a-Number" value

`NEGATIVE_INFINITY`

The negative Infinity value

`POSITIVE_INFINITY`

Positive Infinity value

Number Methods

`toExponential()`

Returns a string with a rounded number written as exponential notation

`toFixed()`

Returns the string of a number with a specified number of decimals