Javascript // Cheatsheet

Comments

```
// this is a single line comment

/*
*   This is a multi line comment
*   This is a multi line comment
*   This is a multi line comment
*/
```

Variable, arrays, objects

```


/*
*
*   Javascript variables are not statically
*   typed, so you can declare a variable
*   without stating what type of data it is.
*
*/

var a = 25; // a number variable
var name = 'Renz' // a stringed variable
```

```
var aChar = '*'; // a character
var pie = 3.14; // float number

/*
*   Arrays concists of a set of data
*   or ordered lists of value
*/

var anArray = [25, 30, "Hello world"];
console.log(anArray[1]); // 30
```

You can also manipulate the data inside of an array

by using the pop(), push(), join(), shift()

The **push()** method lets you **add data** of an array.

```
var anArray = [1, 2, 3, 4, 5];
anArray.push(6); // anArray = [1, 2, 3, 4, 5, 6];
```

The **join()** method lets you join the data inside of an array

into a string by inserting a certain character.

```
var anArray = [1, 2, 3, 4, 5];
anArray.join('~'); // anArray = [1~2~3~4~5];
```

The **shift()** method lets you remove the **first** array element.

```
var anArray = [1, 2, 3, 4, 5];
anArray.shift(); // anArray = [2, 3, 4, 5];



/*
*   You can also unshift an array
*   by using the unshift() method
*/
```

and the **pop()** method lets you remove the **last** array element.

```
var anArray = [1, 2, 3, 4, 5];
anArray.pop(); // anArray = [1, 2, 3, 4];
```

```
/*
*
*   Objects are a declared by a variable
*   and inside that variable has a number
*   of sets of variables
*
*/

var theObjects = {
    fname: "Renz",
```

```
    lname: "Pulvira",
    age: 18
}

// outputting a certain data in an object
console.log(theObjects.fname);
```

Operators

You can do basic **arithmetic** in javascript such as using +,-,/,* in manipulating or calculating data.

```
// You can do basic arithmetic in javascript
// and can use variables to manipulate data

var a = 20;
var b = 30;
var result = a + b; // (a)20 + (b)30 = result
console.log(result); // result = 50

// Addition
var result = a + b; // result = 50
// Subtraction
var result = a - b; // result = -20
// Division
var result = a / b; // result = 1
```

```
// Multiplication
var result = a * b; // result = 600
// Modulo division
var result = a % b; // 0.66
```

Regular expressions

Regular expressions is very useful when your trying to find a certain *digit, non-word, word* or a *letter/character* on a **value**.

There are many regular expressions to use from that's why Regular expressions can be very useful.

```
let theValue = 'Name: Renz, Age: 18';
let reg = /\d/g;
let age = theValue.match(reg);
console.log(age); // 18
```

Event handling

Event handlers are very useful especially when you wan't your web page to be interactive to the user. event handlers is very powerful.

the very basic of event handling is the **'onclick'** command.

you basically put it on a html element so that the event will be recorded when the user **clicked** the element and run a javascript action.

```
<html>
    <body>
        <button onclick='alert('Hello world!');'>Click me
!</button>
    </body>
</html>

<!--
       the browser will pop-up a message to the user sayi
ng
            'Hello world!'
-->
```

You can also put actions inside onclick command and run a certain **function**. i.e

```
<html>
    <body>
    <button onclick="runThis();"></button>
    <script type="text/javascript">
        function runThis(){
            alert("This function ran");
        }
```

```
    </script>
    </body>
</html>


<!--
       the browser will pop-up a message to the user sayi
ng
            'This function ran'
-->
```

Another way to do event handling is ‘onload’. onload will run a function/action when the **page loads**.

```
<html>
    <body onload="alert("Alert!!");">
    </body>
</html>

<!--
       the browser will pop-up a message to the user sayi
ng
            'Alert!!'
-->
```

there are also **onMouseover** and **onMouseout** that will perform and action when the mouse hovers in/out to a certain element.

Console commands

```
// outputting data to the console
console.log("Hello world"); // >Hello world

// An object
var object = {
    fname: "Renz",
    lname: "Pulvira"
};

// Outputting data in a data tree view
console.dir(object);

// Show an alert message to the webpage
alert("This is an alert message");
```

Loops

```
/* FOR LOOP
*   A for loop needs 3 arguments
*   initialization, condition, expression
*/
for(int x = 0; x < 5; x++){
```

```
    console.log(x);
}


/* WHILE LOOP
*   A while loop, loops through a block of code
*   and will not stop depending on the condition given.
*/

while(x != 25){
    console.log(x); // Runs through 1 to 24 then stops to
 25
    x++;
}


/* DO WHILE LOOP
*   A do while loop, has a little resemblance to the whil
e
*   loop. do while loop runs the code atleast once. then
loops
*   through.
*/

x = 0;
do {
    console.log(x); // Runs through 1 to 24 then stops to
 25
```

```
    x++;
} while(x != 25);
```

Functions

```
// A simple function with no arguments
function aFunction(){
    console.log("Hello world");
}

// A function with arguments
function aFunction(a, b){
    var result = a + b;
    console.log(result);
}

// An anonymous function

/*
*   Anonymous functions consists
*   are activated/run automatically
*   when the page loads.
*/
(function(){
    console.log("hello world");
});
```

```
/*
*
*   There are already
*   Premade functions
*   i.e,
*/

function runThis(){
    console.log("HELLO");
}

/*
*   This will run every
*   3(3000) seconds
*/
setInterval(runThis, 3000);
```

Setters and getters

```
/*
*   Setters and getters are very good practice
*   Especially on OO programming. setters
*/

// get
```

```
var obj = {
    fname: "Renz",
    lname: "Pulvira",
    get fullName(){
        return this.fname + ' ' + this.lname;
    }
}

console.log(obj.fullName); // Renz Pulvira



// set
var obj = {
    age: null,
    set herAge(age){
        this.age = age;
    }
}

obj.herAge = 25;
console.log(obj.age); // 25
```

Classes

You can declare/create a class using the ‘class’ then the name of the class in javascript. A class has a **constructor**. you can create a

constructor

by using the **‘constructor()’** method.

```
    var name = null;

    class dog {
        constructor(){
            this.name = 'Marco';
        }

        sayName(){
            console.log("My name is " + this.name);
        }
    }

/*
*   and create an object to
*   output/use that class
*/

    let animal = new dog();
    console.log(animal.name); // Marco
```

You can declare a static method inside of a class. static methods are called without initiating their class and cannot be used.

```
class Pacman {
```

```
    constructor(color, height, width) {
        this.name = 'Pacman';
        this.color = color;
        this.height = height;
        this.width = width;
    }

    static setDimension(){
        this.y = height;
        this.x = width;
    }

    showDimensions(){
        console.log("x: " + this.width + "\ny: " + this.h
eight);
    }
}


let theMap = new Pacman('Yellow', 300, 500);
console.log(theMap.showDimensions()); // x: 500
                                      // y: 300
```

You can also extends a class by using that class functions and methods.

```
class Cat {
```

```
    constructor(theAction){
        this.theAction = theAction;
    }

    doSomething(){
        console.log("The Cat " + this.theAction + 's');
    }
}

class Dog extends Cat{
    doSomething(){
        console.log("The Dog " + this.theAction + 's');
    }
}

let animal = new Cat('Jump');
console.log(animal.doSomething()); // This cat Jumps
```

You can also achieve a *class-like* using a function in javascript like so.

```
function student(firstname, lastname, theDom){
    this.fname = firstname,
    this.lname = lastname,
    this.emailDom = theDom;
    this.theEmail = emailAdd;
```

```
}

function emailAdd(){
    return this.fname + '.' + this.lname + this.emailDom;
}

var theStudent = new student('Renz','Pulvira','@gmail.com
');
alert(theStudent.theEmail()); // Renz.Pulvira@gmail.com
```