JavaScript Cheat Sheet

|  |  |  |
| --- | --- | --- |
| **Serial (video number)** | **Code/Topic** | **Explanation** |
|  | Datatypes | Javascript has 5 data types.   * Numbers (whole/fractional/negative) * Strings ( “ ” ) * Booleans (True/ false) * Null (explicitly nothing) * Undefined (doesn’t have a value yet) |
|  | jS escape characters | \ (backslash)  Ex: “Hello I am \“ maruf\” ” |
|  | Variable creation | To make a variable we use  *var* VariableName = value  However, we have something new that was introduced in 2015. Let and const.  Var – global scope  LET and CONST – block scope   * Const is more like final variable in java. For example,   If we have, const a = 25;  Then a = 24; Would give an error.   * The location is defined it cant be changed however when It comes to reference we cant change it. * Let can be updated. Similar to var however but it’s a block object. * Var and let can be undefined const cant. * Const and let cant be redeclared. |
|  | Variable hoisting | Variable are created first then they are initialized. That is why  Console.log(cat); creates an error where as,  Console.log(cat);  Var cat = “abc”  This doesn’t throw an error. Just prints undefined. |
|  | alert(value) | Pop up msg |
|  | Promt(“value”) | Users can give input |
|  | == | Compare values |
|  | === | Compare values and types |
|  | Function | Function giveName (arguemnts){  }  giveName(argument);  ex: function area (len, width){  console.log(len\*width);  }  Area(5,6);   * 30   We can also return same as java..  Function declaration: normal creation of function  Fucntion expression: assigning the function to a variable. Thus reassigning the variable would end delete the function. |
| 1. 153 | SetInterval(method, time in ms) | Set interval calls the corresponding method. Here we aren’t the ones calling the method that’s why we are not putting any parenthesis here.  This is called a higher order function.  If we write a function inside a higher order function like,  setInterval(function(){  },2000); |
| 1. 155 | Arrays | Same as java. However,  It supports any type of data inside.  it supports dynamic addition. 2 legnth array. Want to add a new item?  Array[2] = “mango”;  Array is now of length 3.  For more info go to MDN |
| 1. 156 | Array addition | Array.push(“content”) <= same as list  Array.pop(“content”) <= same as list  Array.shift(); <= pops the leftmost element and returns it.  Array.unshift(“value”); <= pushes the value in the leftmost element (array[0]) |
| 1. 163 | forEach(parameters) | If you only give one argument, then it's the element, every time.  If you give two, then it's the element, then the index, in that order.  If you give three, then it's element, index, and array, in that order.  Ex:  Array.forEach(a){  Console.log(a);  } |
| 1. 166 | Function name VS function name () | Just calling the function wouldn’t execte the code however when we say function()then the code is ececuted. |
| 1. 166 | Creating our own method on array | Array.prototype.functionName = function(parameter){  Function content  //this refers to the Array on which the method/function is called on.  } |
| 1. 167 | JS objects | Array isn’t the best option all the time. We can use JS objects which stores values in a key-value pair.  Var person ={  name: “”,  id: 21,  city: “NYC”  }  Getting the value:  Person.name  OR  Person[“name”]  We can also create objects using this =>  Var person = {}  Person.name=”asdsad”  Objects can hold data pf any type. |
| 1. 172 | Methods of an object | Var person = {  Add: function(arguments){  }  }  Person.add(); |
| DOM | | |
| 1. 176 | DOM | Stands for document object model. Its called a document object model because everything(objects) stays inside the document.  Document is the root node or the top level object |
| 1. 177 | Selecting items using JS | Variable = document.querySelector(“item name ex: h1”)  document.getElementById(“id name”) //singular  document.getElementsByClass(“class name”) //return HTML collection (more like array). Cant run for on this. The contents are individual objects.  document.getElementsByTagName(“tag name”) // returns an HTML collection  document.querySelector(“”)//css like selector  document.querySelector(“.header”) OR querySelector(“#quote”) //gives the first match  document.querySelectorAll(“”)//returns all |
| 1. 182 | Styling by element | Var tag = document.getElementById(“head”)  Tag.style (this is a huge collection of possible styles)  Tag.style.color=”Blue”  However, one of the problems is that when we select a element and want to put multiple styles on it we would have to write multiple lines of the same thing. That would make a WET CODE. So to solve that in CSS we use the class property and we can add this to that element we choosen using JS.  .class{  All the CSS here  }  Add the class to the element  Tag.classList.add(“class name”)  Toggle on / off  Tag.classList.toggle(“class name”)  Remove  Tag.classList.remove(“class name”) |
| 1. 183 | Manipulating text using JS | Tag.textContent //Gives you all the text content inside the tag. Removes if any tags are inside the content  We can manipulate this using  Tag.textContent = “bla bla”  Tag,innerHTML // same thing but doesn’t remove tags |
| 1. 184 | Get and set attribute | getAttribute(“ ”) //gets the value of the specific attribute.  setAttribute(“ attribute”, “with what”)  ex: tag = document.getElementByTagName(“img”)  tag.setAttribute(“src”, “banana.img”) |
| 1. 187 | Event listener | Element.addEventListerner(“type”, function) |
| 1. 193 | Other events | There are many events. Go to MDN for more.  Mouseover,mouseout etc  <https://developer.mozilla.org/en-US/docs/Web/Events> |
| jQuery | | |
| 1. 210 | Selecting with jQuery | jQuery follows the css style selectors.  $(“id/class/tag ”)  Advanced : $(“li a”) => All a in list  ALWAYS RETURNS A LIST |
|  | Using CSS | $(“h1”).css(“property”,”value”)  We can also add multiple styles by creating a style object.  Var object = { “property”: “value”,  “property”: “value”, }  $(“h1”).css(object)  OR  $(“h1”).css({“property”,”value”,  “property”,”value” }) |
| 1. 211 | Getting the first elemet | $(“div).first()  $(“div:nth-of-type(number)) |
| 1. 212 | More Methods | $(“tags”).html() // Returns all the content with tags  Chaninging attributes  $("div").attr(“attribute”,”value”)  $("img").attr(“src”,”unsplash”) //always selects the first element of the list  .last() // last element of the selected.  .val() // gets the value of it. Ex: for input  .addClass(“class name”) // can add multiple class  .removeClass(“class name”)  .toggleClass(“className”)  .first() // selects the first |
| 1. 216 | Events | .click(function(){  })  Click works for current elements and onclick works for future events too.  This doesn’t work $(this) works  Keypress(function(){  }) EX: finding which key is pressed.  Keypress(function(event){ if(event.which ===13){  //13 is for enter } })  On() //similar to add event listener  On(“type ”, function(){  })  .fadeOut()  .slideToggle(time,function(){ //function is optional }) |
| 1. 226 | Stop bubble effect | $("span").click(function(event){  event.stoppropagation();  }) |
| 1. 226 | Getting the parent element | .parent() |
| 1. 226\*\* | Click vs on (“click”) | This is a tricky part. So normal click works on items that are already created not on the items that are dynamically created after the page has been laoded.  On() on the other hand works on all future items too.  $("ul").on("click", "li", function () {      $(this).toggleClass("completed");  });  Here we have to change the code a bit and a 3 parameters its saying listen when an li is clicked inside the ul then run this code. Ul definitely there when the page loaded. That’s why the parent is used. |