Is case sensitv

Every Statement ends with a smilikolon ;

/\* comment \*/

// one Line comment

Runs from Top to button

Has dynamic types = can changes types

console.log(age3); // outputs only in the console

document.write(age3 + " is youre new age"); //outputs on the page

Ein Bild, das Text, Screenshot, Software, Schrift enthält.

Automatisch generierte Beschreibung

Wird in der Console angezeigt

5 davor zeigt das es 5 mal angezeigt wird

Is a 0 bases language (starts counting from index for example at 0)

var links2 = document.getElementsByTagName("a");

for (i = 0 ; i < links2.length; i++){

    links[i].className= "link-"+ i;

}

Addes to every link a class with the name link-<number of i>

But not sure if its in the index.html file on the local thing too

Ein Bild, das Text, Elektronik, Screenshot, Software enthält.

Automatisch generierte Beschreibung

# Function

function group code in logical staff together

function getAverage(a, b){  //give the function the variable a and b

    var average = (a + b) / 2;

    console.log(average);

}

//called the function out used it

getAverage(5,10); //gives out 7.5

getAverage(); //gives out an Nan because its a+b is not a number

getAverage(5,10,8); // give out 7.5  too, because everything that is to many in the paramenter will be ignort

function with return type

function getPrice (priceOne, amount){

    var price = priceOne \* amount;

    return price;

}

var myResult = getPrice(2, 15);

console.log(myResult +  "is the price");

# Variable Scope

var foo = 20; Global variable

function myFunction (){

var bar = 10; local variable

}

# Numbers NaN

typeof <variablename>; give the datatype

typeof (a+ b);

if(isNaN(d)){   //checkes if d is Not a Number (yes it is not a number so true)

    console.log (d + " is not a datatyp number");

}

if (!isNaN(d)) {   //checkes if d is Not NOT a Number (double negativ)

    console.log(d + " is a number")

}else {

    console.log (d + " is not a datatyp number");

}

# Strings

//lower case letter is always greater than a uppercase Letter

console.log("a" < "b"); //true

console.log("a" < "B"); //false

console.log("A" < "b"); //true

console.log("A" < "B"); //true

var String1 = "abc";

var String2 = "cba";

var String3 = "ABC";

if (String1 === String2){ //false

    console.log("it is true");

}else {

    console.log("it is false");

}

if (String1 === String3){ // false because it is case senstiv

    console.log("it is true");

}else {

    console.log("it is false");

}

var myString = "I am an String";

if (myString.indexOf("ninja") === -1) {

    console.log("the word ninja is not in that string");

}

var test3 = "0123456789";

var Strg3 = test3.slice(2, 9);

document.write(Strg3); //2345678

var test4 = test3.slice(4); // von 4 till the end

console.log(test4);

var tags = "kiwi, apple, cherry, orange";

var tagsArray = tags.split(","); //splits the tags at every  comma and puts it into an array

console.log(tagsArray);

[String - JavaScript | MDN (mozilla.org)](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String?retiredLocale=de)

# Array

var myArray = [];

myArray[0] = 5;

myArray[1] = "house";

myArray[2] = true;  //can store different values in a array

console.log(myArray[0]);

console.log(myArray);

myArray[2]= false; //change the value von array 2 to false

var newArray = [23, 20, "mouse" , true, false]; // declaire and initlaize a array

var myArray3 = new Array(); //another way to make a array

var myArray3 = new Array(5); //defines how many slots the array hat

console.log(newArray.length); //give out how many inputs are in the array

console.log(newArray);

console.log(newArray.sort()); //sorts the array and kepts it sortet in the array

console.log(newArray); //still sorted

console.log(newArray.reverse()); //reverse from what it was before

console.log(newArray); //and kept it reversed

# Objects

Strings, numbers , array, ….. are objects

Obejct is a container which enclosed data and behaviour together

Bsp

var myCar = new Car();

myCar.drive(); //calles a methode

myCar.topSpeed; //calles a parameter(variable)

neues Objekt erstellen mit variablen & methods

var myCar = new Object();

myCar.maxSpeed = 50; //give the object car a new variable named maxSpeed

myCar.driver = "janine";

console.log(myCar.driver);

myCar.drive = function(){console.log("now Driving");}; //give the Object car an method

myCar.drive(); // is calling the method

var myCar2 ={

    maxSpeed: 100,

    driver: "shaun",

    drive: function(){console.log("now Driving again");}

}; //shorthand for creating a object

console.log(myCar2.maxSpeed);

myCar2.drive();

## this

var myCar3 ={

    maxSpeed: 100,

    driver: "shaun",

    drive: function(){console.log("now Driving again");},

    test: function(){console.log(this);}

    };

 //shorthand for creating a object

myCar3.test();

this gives out what is in that object (it is the same if would write **myCar3** Where **this** ist

↓

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

var myCar3 ={

    maxSpeed: 100,

    driver: "shaun",

    logDriver: function(){console.log("now Driving again "+ this.driver);},

    test: function(){console.log(this);}

    };

 //shorthand for creating a object

myCar3.logDriver();



## Constructor Function

Constructor Function starting with a Capital Letter

var Car = function(maxSpeed, driver){ //creates a Constructor

    this.maxSpeed = maxSpeed;

    //myCar2.maxSpeed = maxSpeed; is the same

    this.driver = driver;

    this.drive = function(speed,time){

        console.log(speed\*time); };

    this.logDriver = function(){

        console.log("Driver name " + this.driver);

    };

}

var myCar5 = new Car(10, "janine"); //creates new Car object from Constructor

var myCar6 = new Car(20, "Herbert");

myCar5.driver;

console.log(myCar6.maxSpeed);

## Date Object

var myDate = new Date(); // gives out the current time (change everytime when it is freshed)

console.log(myDate);

var myBirthday = new Date(1993,7,13); // month is a number from 0 - 11 (11 = december)  day is from 1 - 31

console.log(myBirthday);

var myDateTime = new Date(1231, 8, 21, 10, 12,50); //year Month, day, hour, minutes, seconds

var myDateTime2 = new Date(1231, 8, 21, 10, 12,50);

console.log(myDateTime);

console.log(myDateTime.getMonth()); // M

console.log(myDateTime.getFullYear()); //YYYY

console.log(myDateTime.getDate()); //D

console.log(myDateTime.getDay()); //gets the day of the week 0 -6 0= Sunday 6 = Saturday

console.log(myDateTime.getHours()); // get the hour

console.log(myDateTime.getTime()); // get the number ob milliseconds since 1st Jan 1970

if (myDateTime == myDateTime2){  // says it not equal even if it is, because JS is looking at the object and there are two diffent objects

    console.log("yes");

}else {

    console.log("no");

}

if (myDateTime.getTime == myDateTime2.getTime){  //now it says yes beause the same amount of miliseconds have pasted at the both objects

    console.log("yes");

}else {

    console.log("no");

}

# DOM (Document Object Model)

* Is an “application programming interface”
* Use the DOM when we interact with web pages
  + Add content to a HTML document
  + Delete conten from an HTML document
  + Change content on a HTML document
* The **Document** is just the web page
* Every html element in the document is an **object**
  + <head></head>
  + <p></p>
  + <ul></ul>
* The **Model**
  + Descriptes how the objecta are lay out
  + Ein Bild, das Text, Screenshot, Schrift, Uhr enthält.

    Automatisch generierte Beschreibung
* **Nodes**
  + Everything we can change in the document is a node
    - Elements
    - Text within in elements
    - HTML attributes

**What can we do with the DOM ?**

Examples:

* Change the <h1> text node at the top of the page
* Change the background color of an element node
* Animate the logo node from left to right
* Expand the height of an element node when you click on it
* …

## Traversing the DOM

Document Object Model

* Can reach in at any node
* Once we have that node, we can traverse the DOM to access other nodes

Gets the content tag and the h2 in the first content tag and change the inside of the tag

var myContent = document.getElementsByClassName("content");

console.log(myContent);

var myh2InContent = myContent[0].getElementsByTagName("h2");

console.log(myh2InContent);

myh2InContent[0].innerHTML= "Yo Ninjas";

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Ein Bild, das Schrift, Grafiken, Text, weiß enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Schrift, Screenshot enthält.

Automatisch generierte Beschreibung but is in the html document still the old h2

To get element with id (is not in an array because every id is unique)

var myContent = document.getElementsByClassName("content");

console.log(myh2ID);

var myContent = document.getElementsByClassName("content");

var myContent = document.getElementsByClassName("content");

where it says … Elements with s at the end its plural and give back an array

without s it only brings back one single element

## Change page content

var myBody = document.getElementsByTagName("body");

myBody[0].innerHTML; //would just give back all of the body tag

console.log(myBody[0].innerHTML);

var newP = myBody[0].innerHTML = "<p>I am a paragraph tag</p>"; //changes the whole webside, so only this tag is showen

console.log(newP);

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

That was before in the inner html body tag

That is now showen in the inner html body tag (html document stays the same)

Ein Bild, das Screenshot, Design enthält.

Automatisch generierte Beschreibung

## Accessing and changing attributes

var link = document.getElementById("linkOne");

//................Get.................

var linkAtrribute = link.getAttribute("href"); //gives out what the href from the link with id linkOne says

console.log(linkAtrribute);

var linkAtrributeClass = link.getAttribute("class");

console.log(linkAtrributeClass);

//..............Set.................

link.setAttribute("class", "pie"); //change the attribute class to pie

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

//...........add a new attribute ........

link.setAttribute("alt", "hello ");



//.....properites to change value

link.className; //show the current class name

link.className= "ninja"; //change the classname

console.log(link.href); // give the entire link back not only the insite the ahref

var linkAtrribute = link.getAttribute("href");

console.log(linkAtrribute);

console.log(link.style);

## Change styles

var title = document.getElementById("page-title");

title.setAttribute("style", "background: green;");

Ein Bild, das Text, Grün, Schrift, Grafiken enthält.

Automatisch generierte Beschreibung

var title = document.getElementById("page-title");

title.setAttribute("style", "background: green;");

title.setAttribute("style", "font-size: 100px;"); //overwrites the first style attrbute

Ein Bild, das Schrift, Typografie, Text, Grafiken enthält.

Automatisch generierte Beschreibung

Both new attributes in one, than it works

title.setAttribute("style", "font-size: 100px; background: green;"); //both in one than it works

Ein Bild, das Schrift, Text, Grün, Grafiken enthält.

Automatisch generierte Beschreibung

But better is the style property (that updates just the style and don´t overwrite stuff

title.style.color = "red";

Ein Bild, das Grafiken, Schrift, Grün, Logo enthält.

Automatisch generierte Beschreibung

**Style attributes out of Two words must be written in Camsel Case not with an –**

**font-family = fontFamily**

title.style.fontFamily = "fantasy";

Ein Bild, das Text, Schrift, Grafiken, Grün enthält.

Automatisch generierte Beschreibung

## Adding Elements to the DOM

Ein Bild, das Text, Schrift, Screenshot, Zahl enthält.

Automatisch generierte Beschreibung

var newList = document.createElement("li"); //creates new <li></li> but put is nowhere now

var newA = document.createElement("a"); //creates new <a></a> but put is nowhere now

var menu = document.getElementById("main-nav").getElementsByTagName("ul")[0]; //get where the new element should go

menu.appendChild(newList); //adds the new <li></li> element at the end of the ul in ne main-nav

newList.appendChild(newA); // add the a tag in the new list tag

newA.innerHTML = "new List tag"; // creates new content in the a tag

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Schrift, weiß, Screenshot enthält.

Automatisch generierte Beschreibung

Insert new tag before an another one

menu.insertBefore(newList, menu.getElementsByTagName("li")[0]); //creates a new Element, before the first li tag

newList.innerHTML= "new List tag before the first Li tag";

Ein Bild, das Text, Schrift, Screenshot, weiß enthält.

Automatisch generierte Beschreibung

## Removing Element from the DOM

Ein Bild, das Text, Schrift, Screenshot, Zahl enthält.

Automatisch generierte Beschreibung

Ein Bild, das Schrift, Text, weiß, Screenshot enthält.

Automatisch generierte Beschreibung

var parent = document.getElementById("main-nav").getElementsByTagName("ul")[0]; // specifies where to remove

var child = parent.getElementsByTagName("li")[0]; //specifies what to remove

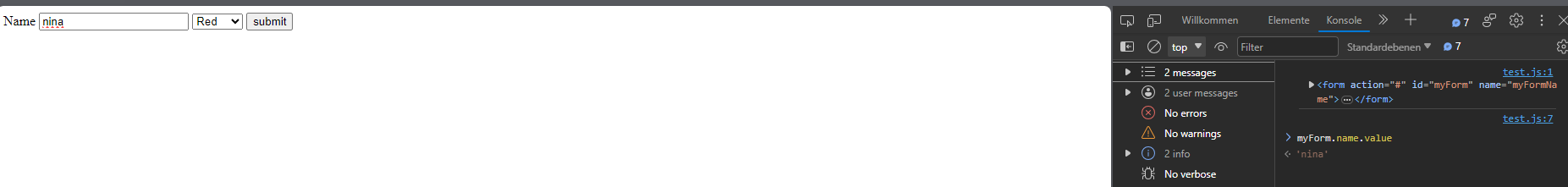
var remove = parent.removeChild(child); //stores the removed variable in the variable to it can be later be back insert if needed

console.log(remove);

menu.appendChild(remove); //put the removed Element back to the end of the list

## Events

## Accessing Form Elements



# NEW DOM

const para = document.querySelector("p"); //gets the first p

const error = document.querySelector(".error"); //gets the first class error

para.innerHTML = "hello";

error. innerHTML= "error";

const para = document.querySelector("p"); //gets the first p

const error = document.querySelector(".error"); //gets the first error class

const errorDiv = document.querySelector("div.error"); //gets the first div with a error class

const heading = document.querySelector("body > h1"); //gets first h1 in body

const heading2 = document.querySelectorAll("h2");//give out all h2 in Nodlist (kind of a array)

para.innerHTML = "hello";

error. innerHTML= "error";

errorDiv.innerHTML = "div Error";

heading.innerHTML= "heading 1 in body";

console.log(heading2); //give out all h2 in kind of a array

heading2[0].innerHTML = "erste h2 heading";

heading2.forEach(head => { // gets every h2 from the heading 2

    head.innerHTML=" new heading 2 in for each";

});

const allError = document.querySelectorAll(".error");

allError.forEach(zaehler => { //gives out all error classes in the console

    console.log(zaehler); //zaehler is ever element in the Nodlist

});

### // ------------ids ----------------

const mainheading = document.getElementById("mainheading"); //kann ids so auswählen

const mainheading2 = document.querySelector("#mainheading2"); //oder so

console.log(mainheading);

console.log(mainheading2);

mainheading.innerHTML = "new Überschrift mit ID";

mainheading2.innerHTML =" überschrift mit ID 2 ";

### //--------------------class-----------------

const errosClass = document.getElementsByClassName("error"); //gives out a HTMLCollection of all erros

const errosClass2 = document.querySelectorAll(".error"); //gives out a NodeList

console.log(errosClass);

console.log(errosClass2);

console.log(errosClass[0]); //gives out the first error class

console.log(errosClass2[0]); //gives out the first error class

//cant use for each on HTML Collection

// errosClass.forEach(eins => {

// console.log(eins);

// });

errosClass2.forEach(eins => {

    console.log(eins);

});

### //----------- Tags----------------------

const pa =  document.getElementsByTagName("p");

const pa2 = document.querySelectorAll("p");

## //------- changing inner value of tags ------------

console.log(pa[0].innerHTML);

console.log(pa[0].innerText);

### //change text

pa[0].innerText= "new p with inner Text";

pa[1].innerHTML= "new p with innerHTML";

### //adding text

pa[2].innerText += "adding new test on  p with inner Text +=";

pa[3].innerHTML += "adding new text on  p with innerHTML +=";

### //change all  p in ones

const allP = document.querySelectorAll("p");

allP.forEach(p => {

    p.innerText = "all new P";

});

## //-------------change html tag ---------------

const content = document.querySelector(".content");

console.log(content.innerHTML); //shows the p  html tags too

console.log(content.innerText); //just shows what is in the p tags in the content class

content.innerHTML= "<h2> THIS IS A NEW H2 WAS P BEFORE </h2>"; //change p to h2

// content.innerText= "<h2> THIS IS A NEW H2 WAS P BEFORE </h2>"; // would write was is in the "" on the screen

content.innerHTML += "<h2> THIS IS A NEW H2 WAS P BEFORE </h2>"; //leaves the old p tag and adding an h2 tag at the end of it

## //------------ new p list for array --------------

const people = ["mario" , "yoshi", "browser"];

people.forEach(person => {

    content.innerHTML +=   `<p>${person}</p>`;

});

### //--------------getting a attribute -----------

//attribute = class href src,....

const link = document.querySelector("a");

console.log(link.getAttribute("href"));

//changing attribute

link.setAttribute('href', "https://www.google.de/?hl=de");

link.innerText =" new link";

const mssg = document.querySelector("p");

console.log(mssg.getAttribute("class"));

mssg.setAttribute("class", "newClass"); //change the class to newClass

console.log(mssg.getAttribute("class"));

### //adding new attribute

mssg.setAttribute("style", "color:green;"); //addes a new attribute to the p tag

### //style attribute

const h1 =  document.querySelector("h1");

// h1.setAttribute("style", "font-size:70px;"); //overwrites the already existing style attribute on the h1

### //add new style(NO overwriting)

console.log(h1.style.color); //gives out the color of the attrbute

h1.style.border = "solid black 2px"; // adding the styles

h1.style.padding = "20px";

//2 Letter words are written in CamaleCase font-size = fontSize

h1.style.fontSize ="10px";

h1.style.borderLeftColor= "green";

### //deleting style attribute;

h1.style.padding= ""; //deleting the padding

h1.style = ""; //deleting all style attribute on h1

## //------------adding and deleting attributes (classes) ----------

const content2 = document.querySelector("span");

console.log(content2.classList);//gives out all classes from that element

content2.classList.add("newClass"); //adding a new class

console.log(content2.classList);

content2.classList.remove("success"); //deleting classes

# Libraries

* General purpose
* Animation
* Form enhancement
* Video

# Global Scope

Ein Bild, das Text, Screenshot, Schrift, Software enthält.

Automatisch generierte Beschreibung

Kann gleiche variable deklarin und initalzieren in verschieden Scopes

# Function

greet();

//**function declaration** (can call function before it is declarared)

function greet(){

    console.log("hello");

}

// speak(); is not working

//**function expression** (can only be called after the function is declared)

const speak = function(){ //the variable is now the function name.

console.log("hello two")

}; //has smilikon at the end of the code clode, because it is a expression (probelbly only time with semilkon end of code block)

greet(); //calling function /invoking function

speak();

function expression kann man **nicht vor der code** der function aufrufen

function declaration kann man vor den code der function aufrufen(javascrip is hoisting function in top of the file (pretend the function are coming first)is not doing that with function expression)

## default value

function greet2(name = "luigi", time= "night" ){ //name = function parameter

    // luigi & night are default values, will fired if the function calling does´nt have parameters

    console.log(`good ${time} ${name}`);

}

greet2("hildi", "morning");

greet2(); //used defail value

greet2("joshi");//uses only the second default value

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Kann auch array übergeben

## return values

const calcArea = function(radius){

    let area = 3.14 \* radius \*\* 2;

    return area;

    // return 3.14 \*radius \*\*2; kanns auch direkt in return schreiben

};

console.log(calcArea(5));

const area = calcArea(8);

console.log(area);

## arrow function

//shorter way of a function (with only one paramter the () are not need for the parameter, but with zero parameter there are needed)

const calArea2 = (radius) => {

    return 3.14 \*radius \*\*2;

};

//shoter way (only one return) then a **return and {} is not needed**

const calArea3 = radius => 3.14 \*radius \*\*2;

Primitve Types & Reference Types

Primitive Types

* Numbers
* Strings
* Boolean
* Null
* Undefined
* Symbols

Reference Types

* All types of objects
  + Objects literals
  + Arrays
  + Functions
  + Dates
  + All other objects