



JS

Javascript

Grundlagen

Was ist JavaScript

- Objektorientierte Skript-Sprache: kein kompilieren
- Wird im Browser (Client) ausgeführt
- Seit neustem auch auf dem Server möglich: Node.js
- Es gibt viele Frameworks: jQuery, Angular.js, React.js, Vue.js, ...

Javascript AutoSubmit Form Example

Form will automatically submit in **17 seconds**.

Name :

Email :

Gender :

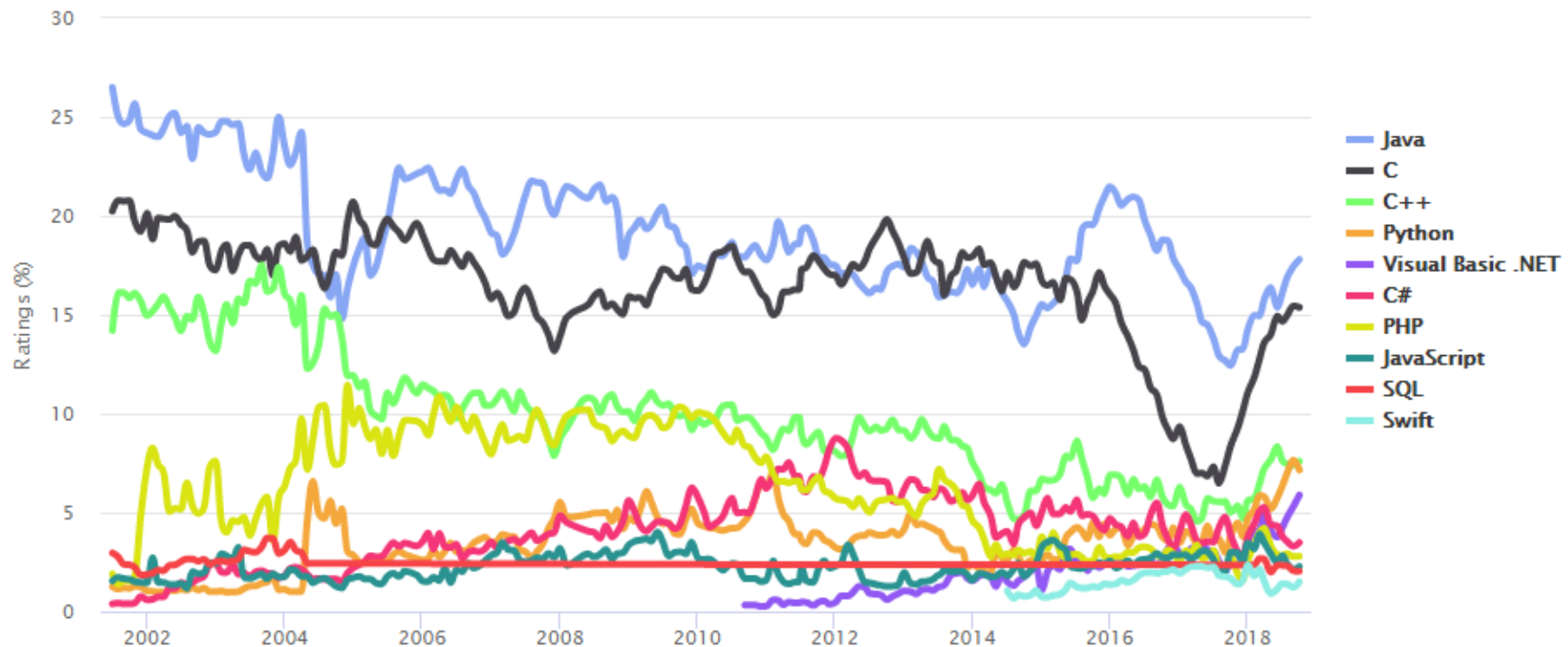
☒ Male ☐ Female

Contact No. :

Welche Programmiersprache ist am populärsten?

TIOBE Programming Community Index

Source: www.tiobe.com



Methodology: Based on the number of queries in popular search engines such as Google, Bing, Yahoo, Wikipedia, Amazon, YouTube

Welche Programmiersprache ist am populärsten?

StackOverflow Developer Survey

Frequency: Annually.

Methodology: 56,033 coders in 173 countries [surveyed for 2016](#).

Rankings:

1. JavaScript
2. SQL (eliminated because it's not a conventional PL)
3. Java
4. C#
5. PHP
6. Python
7. C++

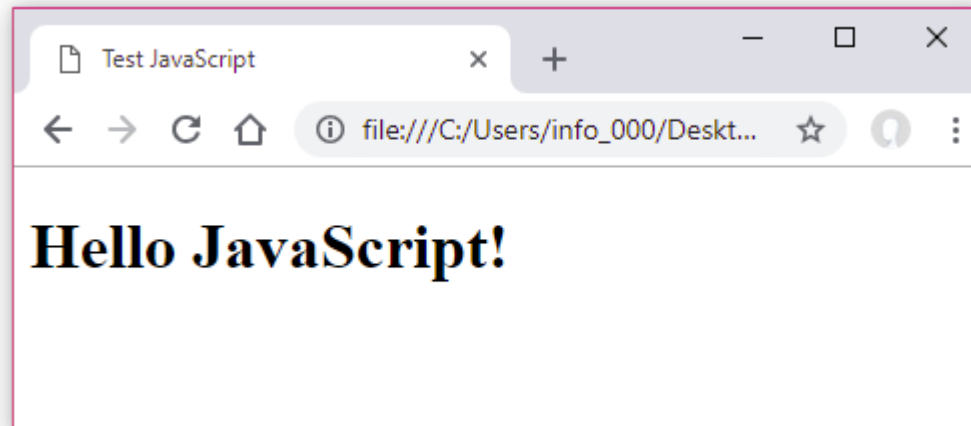


JS

Erste Beispiele

Beispiel hello1.html

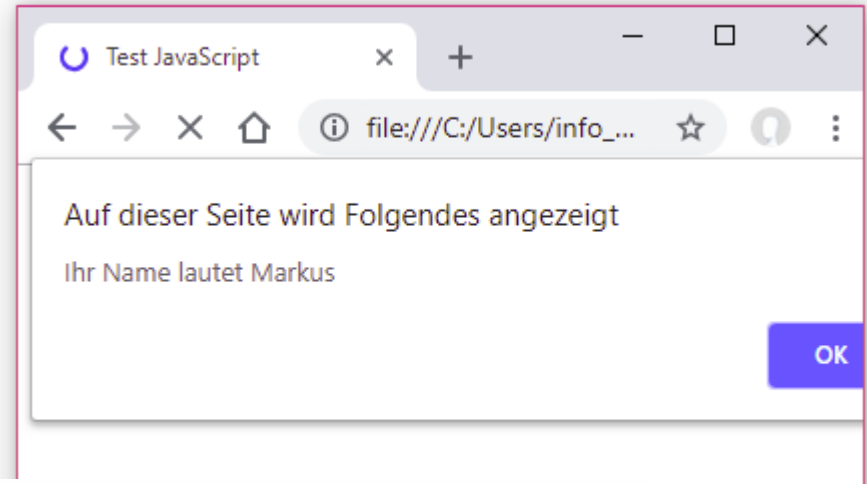
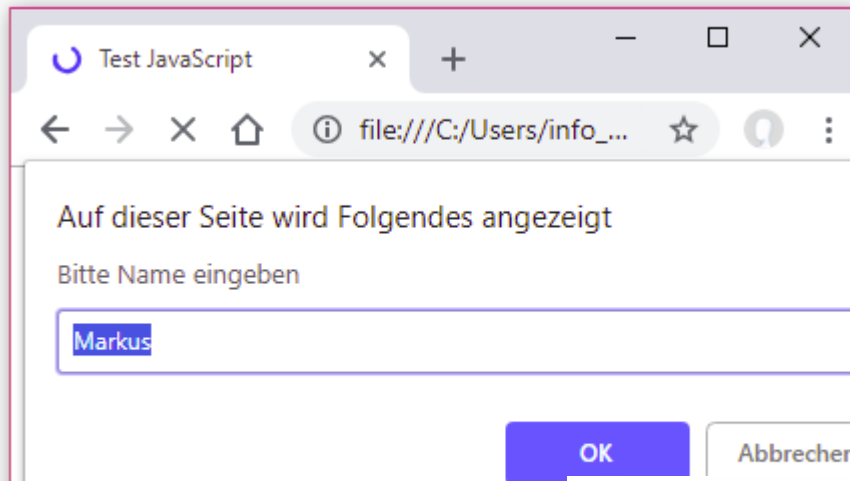
```
<!doctype HTML>
<head>
  <meta charset="utf-8">
  <title>Test JavaScript</title>
</head>
<body>
  <script>
    document.write("<h1>Hello JavaScript!</h1>");
  </script>
</body>
</html>
```



Beispiel hello2.html

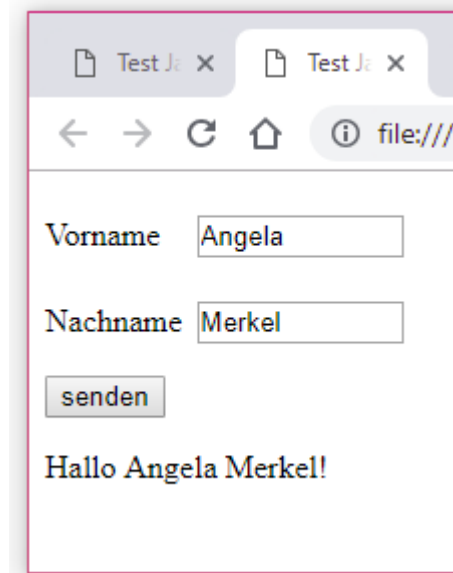
```
<!doctype HTML>
<head>
  <meta charset="utf-8">
  <title>Test JavaScript</title>
</head>
<body>
  <script>
    var ihrName="Tom";
    document.write("<h1>Hello JavaScript!</h1>");
    document.write("<p>Variable ihrName="+ihrName+"<p>");
    var ihrName=prompt("Bitte Name eingeben", "Markus");
    alert("Ihr Name lautet "+ihrName);
    document.write("<p>Variable ihrName="+ihrName+"<p>");
  </script>
</body>
</html>
```

Beispiel hello2.html



Beispiel hello3.html

```
<body>
<p><label>Vorname</label> <input id="idVorname" size="10"> </p>
<p><label>Nachname</label> <input id="idNachname" size="10">
</p>
<p><input id="idSenden" type="button" value="senden"></p>
<p id="idBegruessung"></p>
<script src="util.js"></script>
<script>
    handleEvent("idSenden", "click", senden);
    handleEvent("idVorname", "returnKey", senden);
    handleEvent("idNachname", "returnKey", senden);
    function senden() {
        document.getElementById("idBegruessung")
            .innerHTML="Hallo"
            +document.getElementById("idVorname").value
            +" "
            +document.getElementById("idNachname").value
            +"!"; }
</script>
</body>
```



JS

Unterschied Java und Javascript

Bus



Quelle: [Wikimedia Commons](#)

und

Bussard



Quelle: [Wikimedia Commons](#)

Quelle: <http://selfhtml.apsel-mv.de/java-javascript/>

Unterschied Java - Javascript 1/7

- Variablen: keine Datentypen, keyword `var`

```
var intValue = 1;  
var floatValue = 3.0;  
var stringValue = "This is a string";  
var sqString = 'This is also a string';
```

Primitive Datentypen: number, string, boolean, undefined, null

```
typeof text // number, string, boolean, undefined
```

Unterschied Java - Javascript 2/7

• Arrays

```
var zahlArray = new Array(7);  
var personArray = ["Peter", "Monika", "Hans"];  
var tabArray = [[22.3, 18.5], [21.6, 19.7], [24.6, 20.1]];  
var dict = []; dict["house"] = "haus"; dict["cat"] = "katze";  
// dict = [house: "Haus", cat: "Katze"] (Alternative Objekte dict={})
```

• Besonderheiten

- `personArray[3]` // undefined, kein Fehler
- `for(var x of personArray)`
- `//int[][] pyramid = new int[3][4];
pyramid = Array.from(Array(3), () => new Array(4));`

Unterschied Java - Javascript 3/7

- ArrayList

- Definieren

```
var ar=[];
```

- Hinzufügen

```
ar.push(1); ar.push(2); ar.push(3); // [1, 2, 3]  
ar.unshift(-1); // [-1, 1, 2, 3]
```

- Entfernen

```
ar.pop() // [1, 2]  
ar.shift(); // [2]  
ar = [1,2,3,4]; ar.splice(2, 1) ; //index=2, numEl=1 //  
[1, 2, 4]  
  
function arrayRemove(arr, value) {  
    return arr.filter(function(el){ return el != value; }); }  
var ar = arrayRemove(ar, 4); // [1, 2]
```

- Enthält

```
var isEl = ar.includes(2); // true
```

Unterschied Java - Javascript 4/7

- Funktionen

```
function addiere (a, b, c=6) {  
  var sum = a + b +c;  
  return sum; }
```

```
var erg = addiere (1,2,3);
```

- Funktionen an Variablen binden

```
var summeFkt = function(a,b) {return a+b;}  
// var summeFkt = (a,b) => {return a+b;}
```

- Funktionen als Wert übergeben

```
function meineSumFkt(summeFkt,a,b) {return summeFkt(a,b);}
```

- Überladen / Defaultwerte für Parameter

```
function mult(a=0,b=1) {...}
```

Vorsicht: Überladen `function mult(a){}`, `function mult(a,b){}` funktioniert nicht (Unterschied zu Java).

Unterschied Java - Javascript 5/7

- Objekte ohne Klassen

```
var peter = {name: "Peter", alter: 21,  
             sage: function(text) {  
                 alert(this.name+" sagt:"+text);  
             }};
```

```
//Auf Attribute/Methoden zugreifen  
alert(peter.name); //alternativ peter["name"]  
peter.name="Pitt";  
peter.sage("Hallo");
```

```
// Attribute/Methoden aufzählen  
for(x in peter) {alert(peter[x]);}
```

```
// Attribute/Methoden löschen  
delete peter.alter;
```

```
// Auf Keys zugreifen  
Object.keys(peter) //["name", "alter", "sage"]
```

Unterschied Java - Javascript 6/7

- Klassen: als Funktionen realisiert

```
function Fahrzeug(f,g) {  
    // Attribute  
    this.farbe=f;  
    this.geschwindigkeit=g;  
    // Methoden  
    this.beschleunigen=fahrzeugBeschleunigen;  
    this.toString = fahrzeugAusgeben; }  
  
function fahrzeugBeschleunigen(wert) {  
    this.geschwindigkeit+=wert;}  
function fahrzeugAusgeben() {return "fahrzeug("+this.farbe+  
    ", "+this.geschwindigkeit+")";}  
  
// Objekt erzeugen  
var dacia = new Fahrzeug("rot",50);  
alert(dacia.farbe);  
dacia.geschwindigkeit=52  
dacia.beschleunigen(35);
```


Unterschied Java - Javascript 7/7

- Ausgabe

```
alert("Ihr Name lautet "+ihrName);  
console.log("Ich bin hier");  
document.write("<h1>Hello JavaScript!</h1");
```

- Eingabe

```
var ihrName = prompt("Bitte Name eingeben", "Markus");  
var ok = confirm("Wirklich?");
```

- HTML verändern

```
document.getElementById("demo").innerHTML = "Hello";  
document.getElementById("demo").firstChild.nodeValue="Hello";  
document.getElementById('myImage').src='pic_bulbon.gif';  
document.getElementsByTagName('h1').style.display='none';
```

Auch noch nützlich

- Datentypen

```
parseFloat("3.14"); parseInt("3.14"); isNaN(a) // keine Zahl  
Math.random(); Math.floor(i) //Nachkommastellen abschneiden.  
eval(3+5)
```

- Webseite überschreiben

```
document.open();  
document.write("...");  
document.close();
```

- Bei Fehler abbrechen

```
function fehler(fehler, datei, zeile) {  
    alert("Fehler: "+fehler+"\nDatei: "+datei+  
        "\nZeile: "+zeile); }  
window.onerror=fehler;
```

Spezielle Aufgaben

- Länge eines Strings

```
var someString = 'the cat looks like a cat';  
var l = someString.length; // in Java: .length()
```

- ReplaceAll in Strings

```
var someString = 'the cat looks like a cat';  
var anotherString = someString.replace(/cat/g, 'dog');
```

Konkrete Übersetzungen

Java

- `List[] col = new ArrayList[cands.length];`
- `int num = (int) cans[s].get(0);`
- `cands[s].size()`
- `cands[y][x].add(obj);`
- `Stack<String> stack=new Stack<String>();`
- `stackCands.isEmpty()`

JavaScript

- `var col = new Array(cands.length);`
- `var num = parseInt(cands[s][0]);`
- `cands[s].length`
- `cands[y][x].push(obj);`
- `var stack = [];`
- `stackCands.length==0`



JS

Ereignisse

Auf Ereignisse reagieren

- Im HTML-Tag

```
<input type="button" value="ok" onclick="geklickt('ok');">  
  
function geklickt(text) {alert("Button "+text+" gedrückt");}
```

- Über ID

```
<input id="okBtn" type="button" value="ok">  
<a id="link1" href="#">Link</a>  
  
document.getElementById("okBtn").onclick = geklickt("okButton");  
document.getElementById("link1").onclick = geklickt("link1");  
  
function geklickt(text) {  
    alert("Button/Link "+text+" gedrückt"); }
```

Auf Ereignisse reagieren

- Hyperlinks

```
<a href="javascript:geklickt('link1')">Link</a>
function geklickt(text) { alert("Link "+text+" gedrückt"); }
```

- Event Listener

```
<input id="okButton" type="button" value="ok">
handleEvent("okButton", "click", geklicktOk);
function geklicktOk(objekt) { alert("Ok Button gedrückt"); }
```

```
//util function
function handleEvent(id, ereignis, funktion) {
if(window.addEventListener)
    document.getElementById(id)
        .addEventListener(ereignis, funktion, false);
else if(window.attachEvent)
    document.getElementById(id).attachEvent("on" + ereignis,
funktion);
}
```

Arten von Events 1/2

- **Laden**
 - onload
- **Formulare**
 - onsubmit / onreset: Der Submit/Reset-Knopf wird gedrückt.
 - onchange: input, textarea, select wird verändert.
 - onkeyup: input, textarea, select wird eine Taste gedrückt.
- **Maus**
 - onclick
 - onmouseover, onmouseout, onmousemove
 - onmousedown, onmouseup

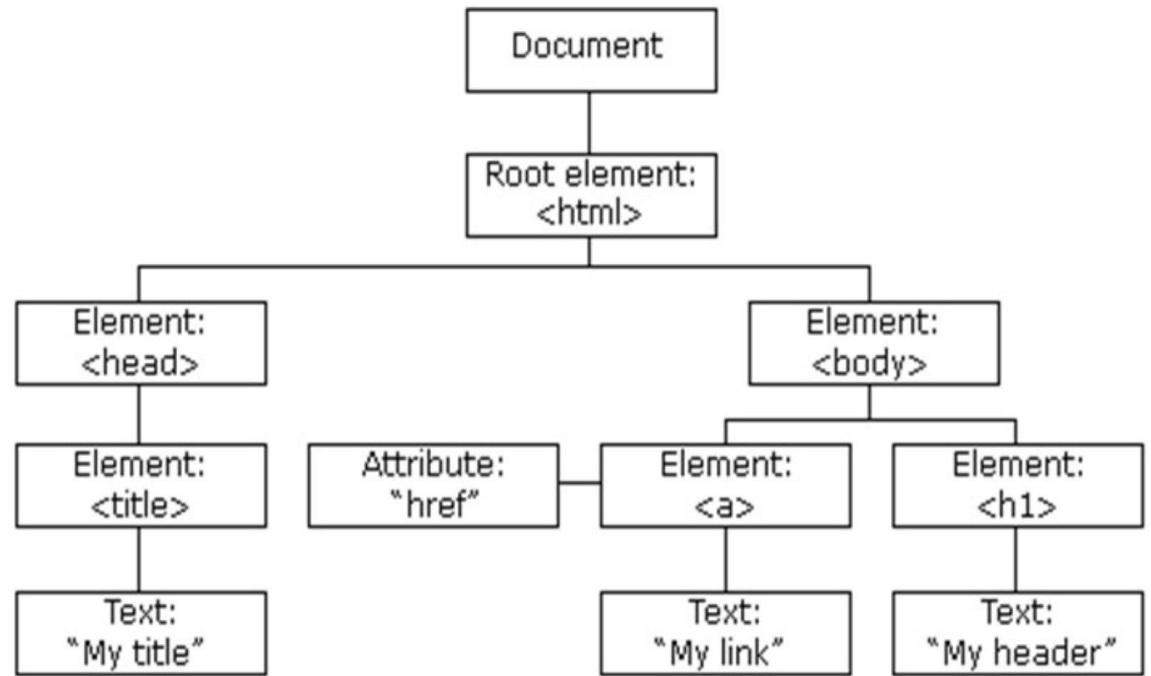
Arten von Events 2/2

- Tastatur
 - onkeypress
 - onkeyup, onkeydown

```
handleEvent("search", "keypress", searchByKey);  
  
function searchByKey(e) {  
    if (e.keyCode == 13) { search(); return false; }  
}
```

Beim EventListener wird der Präfix "on" weggelassen.

JS



DOM

(Document Object Model)

DOM: Zugriff / Einfache Änderungen

- Zugriff auf Elemente

```
var absatz1 = document.getElementById("a1");  
var absatz2 = document.getElementsByTagName("p")[0];
```

```
firstChild, lastChild, childNodes[i]  
document.body.lastChild
```

- Einfache Änderungen

```
document.getElementById("demo").innerHTML = "Hello";  
document.getElementById("demo").firstChild.textContent =  
"Hello";  
document.getElementById("demo").firstChild.nodeValue="Hello";  
document.getElementById('myImage').src='pic_bulbon.gif';  
document.getElementsByTagName('h1').style.display='none';
```

DOM: neue Knoten hinzufügen

- Parent bestimmen

```
var parentNode=document.getElementById("page");
```

- Neuen Knoten erstellen

```
var nodeDiv = document.createElement("div");
```

- Attribute setzen / Klassen hinzufügen

```
nodeDiv.setAttribute("id", cellName+"Num");  
nodeDiv.setAttribute("onclick",  
    "javascript:clickCell('"+cellName+"')");  
nodeDiv.classList.add("divFarbig");  
//cssAddClass(nodeDiv, "divFarbig");
```

- HTML-Text hinzufügen

```
nodeDiv.innerHTML="Das ist ein Text";
```

- Neuen Knoten Parent anhängen / davor einfügen

```
parentNode.appendChild(nodeDiv);
```

DOM Modell: children, childnodes

- Kinder ansprechen: `children`, `childNodes`

```
var histUl=document.getElementById("hist").children[2];  
histUl.removeChild(histUl.childNodes[i]);
```

- Erstes, Letzes Kind: `firstChild`, `lastChild`

```
histUl.lastChild.innerHTML="last";
```

- ```
document.write("Hat Kinder: " + absatz.hasChildNodes() + "
");
document.write("Anzahl Kinder: "+ absatz.childNodes.length +
"

");
for(var i=0; i<absatz.childNodes.length; i++){
 if(absatz.childNodes[i].nodeType == 3) //3=Textknoten
 document.write("Kind " + i + ": Typ=Text, Wert:"
 + absatz.childNodes[i].nodeValue + "
");
 else if(absatz.childNodes[i].nodeType == 1) //1=Element-Knoten
 document.write("Kind " + i + ": Typ=Element, Name:"
 + absatz.childNodes[i].nodeName + "
");
}
```

## DOM: insertBefore, sibling, removeChild, removeAttribute

- **insertBefore** anstatt appendChild:  

```
el.insertBefore(document.createTextNode("3"),
 el.childNodes[0]); //als erstes einfügen
```
- **Geschwister einfügen**  

```
el.parentNode.insertBefore(
 document.createTextNode("3"), el.nextSibling)
```
- **Kinder löschen: removeChild, removeAttribute**  

```
var absatz = document.getElementById("a2");
document.body.removeChild(absatz);
node.removeChild(node.childNodes[1]);
node.removeAttribute("style");
```

## DOM: createTextNode, cloneNode, replaceNode

```
function textAendern(id) {
 var absatz = document.getElementById(id);
 if(absatz.hasChildNodes()) {
 absatz.firstChild.nodeValue = "Geänderter Inhalt";
 } else {
 var text = document.createTextNode("Erzeugter Inhalt");
 absatz.appendChild(text);
 }
}

function umgeben() {
 var absatz = document.getElementById("a3");
 var kursiv = document.createElement("i");
 var ersetzt = absatz.replaceChild(kursiv, absatz.firstChild);
 kursiv.appendChild(ersetzt);
}

function klonen() {
 var absatz = document.getElementById("a3");
 var kopie = absatz.cloneNode(true); //true = with children
 document.body.appendChild(kopie);
}
```



**JS**

***Maps***



# Objekte als Maps

```
var o={};
o[1]="1"; o[2]="2"; o[3]="3"; //{1=>"1", 2=>"2", 3=>"3"}
console.log(o[1]); // "1"
o[1]="-1";
Object.keys(o).length //3
for(i in o) { console.log(i+" "+o[i]);}
delete o[1]; //{2=>"2", 3=>"3"}
```

# Map als Maps

```
var m=new Map();
m.set(1,"1"); m.set(2,"2"); m.set(3,"3");
//Map(3) {1 => "1", 2 => "2", 3 => "3"}
m.get(1); // "1"
```

```
var map = new Map([[1, '1'],[2, '2'],[3, '3'],]);
```

**Weitere:** delete(key), has(key), clear(), keys(), values(),  
foreach(), size()

```
map.forEach((value, key) => {
 console.log(`${key}: ${value}`)});
```

```
for (const [key, value] of map) {
 console.log(`${key}: ${value}`)};
```



**JS**

***Ende***



**JS**

# ***Ergänzungen***