JavaScript IF – Anweisung

Inhalt:

Kontrollstrukturen:

Kontrollstrukturen sind klassische Anwendungen von logischen oder vergleichenden Operatoren und spezielle Anweisungsschritte in einer Programmiersprache, mit denen ein Programmierer Entscheidungen über den weiteren Ablauf eines Programms oder Skripts vorgeben kann, wenn bestimmte Bedingungen eintreten. In JavaScript hat man die gleichen Kontrollstrukturen zur Verfügung, die es in fast allen Programmiersprachen in gleicher oder ähnlicher Form gibt. Es gibt drei Arten von Kontrollflussanweisungen:

1. Entscheidungsanweisungen. Diese suchen auf Grund einer Bedingung einen Programmfluss heraus.
2. Schleifen bzw. Iterationsanweisungen. Diese wiederholen eine bestimmte Anzahl an Anweisungen.
3. Sprunganweisungen. Diese verlassen eine Struktur.

1)Entscheidungsanweisungen

Ein Programm kann abhängig von bestimmten Bedingungen unterschiedliche Teile eines Programms durchlaufen. Man sagt auch: Es verzweigt sich. Verzweigungen gehören zu den wichtigen Kontrollstrukturen.

moderne und ältere Schreibweise:

a) moderne Schreibweise – Kurzschreibweise: Mit Fragezeichen und Doppelpunkt

Hat der User eine Mitgliedschaft, dann wird das Video angezeigt, ansonsten nicht.
Code:

```
<script>
    let hatMitgliedschaft = true;

    hatMitgliedschaft ? document.write("show video") : document.write("hide video");
</script>
```

Anderes Beispiel ohne genauen Code:

```
age >= 18 ? “You may enter” : “Too young”
```

Beispiel: was passiert hier?

```
let loggedIn = true;
let hasMembership = false;

loggedIn && hasMembership
  ? console.log("show the video")
  : console.log("dont show  the video");
```

b) ältere Schreibweise: Verzweigungen mit »if ... else«

Verzweigungen werden mit Hilfe der Anweisung if ... else erzeugt. Nach dem if steht in runden Klammern eine Bedingung, die entweder erfüllt oder nicht erfüllt ist.

- Falls sie erfüllt ist, wird die folgende Anweisung oder der folgende Block von Anweisungen ausgeführt. Einen Block von Anweisungen erkennt man an den geschweiften Klammern { ... }.
- Falls die Bedingung nicht erfüllt ist, wird die Anweisung oder der Block von Anweisungen nach dem else ausgeführt, falls vorhanden.

```
if (Bedingung) {
      ...Anweisungen A
      } else {
      ...alternative Anweisungen
      }
```

Bedingungen werden mit Hilfe von Wahrheitswerten gebildet. Vergleichsoperatoren haben Wahrheitswerte als Ergebnis.

- Der Operator > steht für größer als;
- der Operator < steht für kleiner als;
- >= bedeutet größer als oder gleich;
- <= bedeutet kleiner als oder gleich.
- == prüft auf Gleichheit
- != auf Ungleichheit

Es gibt die if-Bedingung mit oder ohne nachgestellten else-Zweig. Wenn er da ist, kann darin wie oben eine Folge von alternativen Anweisungen stehen, die immer dann ausgeführt werden, wenn die Bedingung den Wert false liefert. Wenn bereits der if-Zweig ausgeführt wurde, wird der else-Zweig nicht ausgeführt.
Wenn der else-Zweig fehlt, bewirkt die gesamte Struktur nur etwas, wenn die Bedingung den Wert true liefert. Im Falle des Werts false passiert gar nichts.

Achtung:
Achte darauf, nach der Bedingung hinter einem if oder nach einem else **kein Semikolon** zu schreiben. Es sollte also

- nicht aussehen wie folgt: if(a > b);
- oder else;.

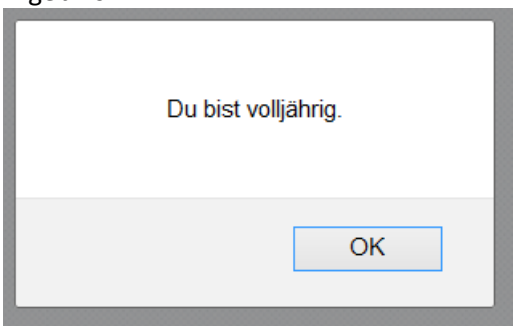
Dies würde dazu führen, dass die Verzweigung unmittelbar endet und die folgenden Anweisungen immer ausgeführt würden. Es könnte auch passieren, dass gar keine Ausgabe erfolgt, weil das else ohne sein zugehöriges if erzeugt wird. Diese typischen Einsteigerfehler sind schwer zu finden.

Beispiel:
Bei Volljährigkeit soll eine entsprechende Meldung ausgegeben werden.

```
<body>
    <script>
    var alter = 20;

    if (alter >= 18) {
        alert("Du bist volljährig.");
    } else {
        alert("Du bist zu jung und darfst nicht hinein.");
    }
    </script>
</body>
```

Speicher unter „3_js_alter.html“

Ergebnis:

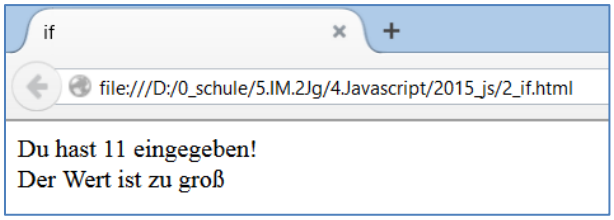


Beispiel:
In dem Beispiel wird mit der prompt()-Methode vom Anwender eine Zahl zwischen 0 und 9 entgegengenommen und dieser Wert der Variablen „erg“ zugewiesen.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <script>
        var erg = prompt("Gib eine Zahl zw. 0 und 9 ein");
        document.write("Du hast " + erg + " eingegeben! <br>");
        if (erg > 9){
            document.write("Der Wert ist zu groß");
        } else {
            document.write("Alles klar");
        }
    </script>
```

Speichern unter „3_js_erg.html“.

Ergebnis:



Übung Body-Mass-Index:

errechne den Body-Mass-Index mit folgender Berechnung:

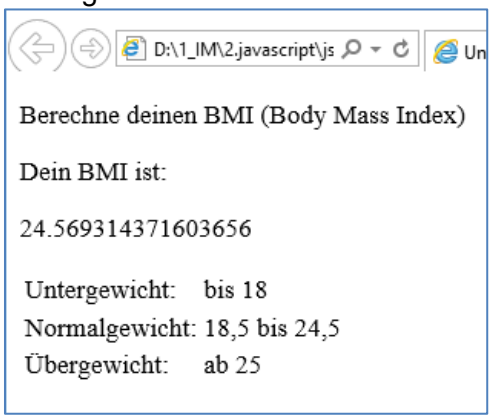
bmi = gewicht / (groesse * groesse) * 10000

- Die Größe und das Gewicht sollen jeweils vom Kunden mit Hilfe einer „prompt()“ – Anweisung eingegeben werden können.
- Darunter soll mit einer <table> eine kurze Erklärung erfolgen. Mit <tr> und <td>.

Es soll eine „if“-Anweisung regeln, ob der Kunde eine Größe eingibt:

```
if(groesse == 0) {
    alert("Körpergröße angeben!")
  } else {
    document.write(bmi);
  }
```

Lösung:



Übung: Was passiert hier?

```
// Zeichenketten
var land = "Spanien";
if(land == "Spanien"){
document.write("<p>Land ist Spanien<\/p>");}
else {
document.write("<p>Land ist nicht Spanien<\/p>");}
```

2)Bestätigung anfordern mit confirm()

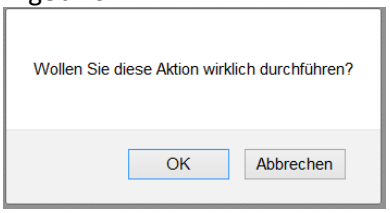
Die Funktion confirm() stellt im Zusammenhang mit einer Verzweigung eine weitere Möglichkeit zur Kommunikation mit dem Benutzer dar. Nach dem Aufruf der Methode erscheint ein Dialogfeld mit einer Frage sowie zwei Buttons mit der Aufschrift **OK bzw. Abbrechen**. Damit kann man eine einfache Ja – oder Nein Frage stellen.

Beispiel:

```
<script>
        var antwort = confirm("Wollen Sie diese Aktion wirklich durchführen?");
        if(antwort == true) {
        alert("Diese Aktion wird durchgeführt");
} else {
        alert("Diese Aktion wird nicht durchgeführt");}
</script>
```

- Falls der Benutzer den Button OK betätigt, wird true zurückgeliefert.
- Nach Betätigung des Buttons Abbrechen wird false geliefert. Dann wird der else-Zweig durchlaufen.

Ergebnis:



3)Beispiele mit if

3a)Radio Buttons auslesen (Bsp. Hak Anmeldung) – hier kann nur eine Auswahl getroffen werden

1)Kommunikation mit dem Benutzer – mit Kontrollkästchen
Das vorhandene Formular in HTML soll im <head> mit eine JavaScript-Funktion ergänzt werden. Diese soll alle vorhandenen Kontrollkästchen auswertet, ob sie ausgewählt wurden oder nicht.
Von einem beliebigen Benutzer kann mit dem Radio-Button genau ein Kästchen ausgewählt werden und dann mit dem Absenden-Button ausgegeben werden. So soll die gewählte Form unterhalb des Formulars ausgegeben werden. Somit weiß man, was der Kunde gewählt hat.
Info: Das passiert hier mit der „id=“output“ genau dort, wo es ausgegeben wird.

```
44  </form>
45  <h3>Sie haben folgende Ausbildung gewählt:</h3>
46      <div id="output"></div>
47      <h3>Danke für die Anmeldung.</h3>
```

- Erstelle dazu in einer Funktion „anmeldung()“
- Verwende entsprechende „document.getElementById()“ – Methoden und nutze die Verzweigung mit „if“ bzw. „elseif“

Info: https://www.w3schools.com/jsref/met_document_getelementbyid.asp

Info: else if https://www.w3schools.com/js/js_if_else.asp

Die else if-Anweisung

Verwenden Sie die `else if` Anweisung eine neue Bedingung angeben , wenn die erste Bedingung falsch ist.

Syntax

```
if (condition1) {
  //  block of code to be executed if condition1 is true
} else if (condition2) {
  //  block of code to be executed if the condition1 is false and condition2 is true
} else {
  //  block of code to be executed if the condition1 is false and condition2 is false
}
```

Begriffe:

- **id:** wird hier extra anders gewählt als der value, damit man es besser unterscheiden kann.

```
30 <form>
31 <fieldset>
32      <h3><input type="radio" name="schule" id="klassik" value="Hak Klassisch">
33      <label for="klassik">Hak Klassisch</label></h3>
```

- **value**: diese Eigenschaft kann dazu verwendet werden, um zum Kontrollkästchen gehörende Werte aufzunehmen
- **checked**: bei dieser Eigenschaft handelt es sich um einen Boole´schen Wert, der angibt, ob das Kontrollkästchen aktiviert worden ist oder nicht.

```
 5 <script>
 6     function anmeldung(){
 7         var klassik = document.getElementById("klassik"); //holt sich value von der ID
 8         var digbiz = document.getElementById("digbiz");
 9         var has = document.getElementById("has");
10
11         var output = document.getElementById("output");
12
13
14         if (klassik.checked)
15         {
16         var result =  klassik.value;
17         }
18             else if(digbiz.checked)
19             {
20             var result =  digbiz.value;
21             }
22             else
23             {
24             var result = has.value;
25             }
26
27         output.innerHTML = result;
28     }
29 </script>
30 </head>
```

- mit if und elseif und else: „ist etwas gechecked“?
- Die Eigenschaft „checked“ ermittelt, ob ein Haken gesetzt wurde oder nicht.

- Wenn es aktiviert wurde, gib den Inhalt der Eigenschaft „value“ zurück, der zu dem entsprechenden Kontrollkästchen gehört.

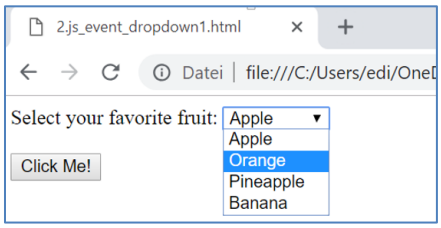
Gegeben ist das Formular schon vorher:

```
<form>
<fieldset>
    <h3><input type="radio" name="schule" id="klassik" value="Hak Klassisch">
    <label for="klassik">Hak Klassisch</label></h3>

    <h3><input type="radio" name="schule" id="digbiz" value="Hak DigBiz">
    <label for="digbiz">Hak DigBiz</label></h3>

    <h3><input type="radio" name="schule" id="has" value="Handelsschule">
    <label for="has">Handelsschule</label></h3>

    <br><br>
    <input type="button" onclick="anmeldung()" value="Anmeldung in der Hak/Has">
</fieldset>
</form>
<h3>Sie haben folgende Ausbildung gewählt:</h3>
    <div id="output"> </div>
    <h3>Danke für die Anmeldung.</h3>

</body>
</html>
```

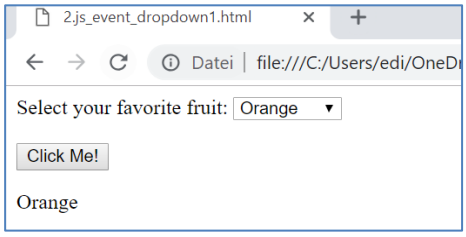
Info: die Radio-Buttons sind voneinander abhängig, da sie alle denselben „name“ haben. Dadurch wird gewährleistet, dass man nur immer genau einen anklicken kann.

3b)Beispiel: Anzeige einer gewählten Option in einer Dropdownliste

- Ereignis mit „onclick“ und „getElementById“ und „Objektzugriff“.
- Hier hat die Option keinen „value“. Hier wird das Ergebnis mit „innerHTML“ ermittelt.



Ergebnis:



- Ein Select-Objekt, auf das im Code verwiesen wird, sollte ein id-Feld haben.
- Das andere Element in diesem Formular ist eine Schaltfläche. Wenn man darauf geklickt hat, wird die Funktion „getOption()“ gestartet.
- Zeile 7 wird in den Bereich geschrieben, der mit dem Wert (demo) verknüpft ist und das ist die Zeile 25 mit id=“demo“.

Erstelle das File „2.js_event_dropdown1.html“:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <script>
5  function getOption() {
6      var obj = document.getElementById("mySelect");
7      document.getElementById("demo").innerHTML =
8      obj.options[obj.selectedIndex].text;
9  }
10 </script>
11 </head>
12 <body>
13
14 <form>
15 Select your favorite fruit:
16 <select id="mySelect">
17   <option>Apple</option>
18   <option>Orange</option>
19   <option>Pineapple</option>
20   <option>Banana</option>
21 </select>
22 <br><br>
23 <input type="button" onclick="getOption()" value="Click Me!">
24 </form>
25 <p id="demo"></p>
26 </body>
27 </html>
```

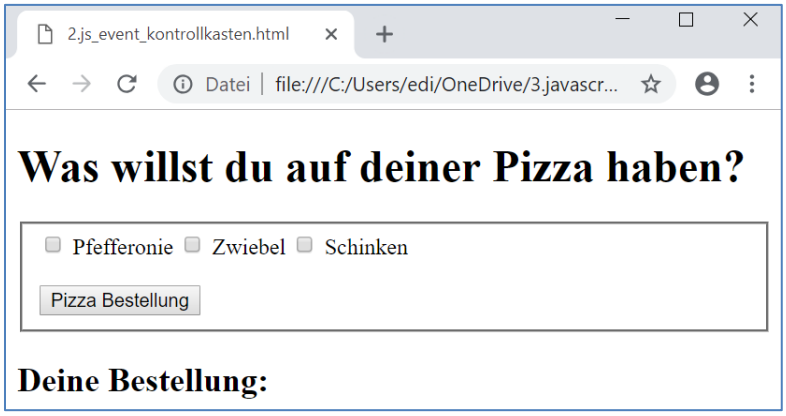
Code:

```
<!DOCTYPE html>
<html>
<head>
<script>
function getOption() {
   var obj = document.getElementById("mySelect");
   document.getElementById("demo").innerHTML =
   obj.options[obj.selectedIndex].text;
}
</script>
</head>
<body>

<form>
Select your favorite fruit:
<select id="mySelect">
  <option>Apple</option>
  <option>Orange</option>
  <option>Pineapple</option>
  <option>Banana</option>
</select>
<br><br>
<input type="button" onclick="getOption()" value="Click Me!">
</form>
<p id="demo"></p>
</body>
</html>
```

3c)Kontrollkästchen auslesen (Bsp. Pizzabestellung) - Hier können mehrere Auswahlen getroffen werden

Kontrollkästchen erfüllen eine nützliche Dateneingabe-Funktion: sie machen immer dann Sinn, wenn es um Boole´sche Daten geht. Wenn einige Werte wahr oder falsch sein können, sind Kontrollkästchen praktisch.
Es können beliebig viele Kontrollkästchen aktiviert werden.

- Kontrollkästchen sind voneinander unabhängig obwohl sie häufig gruppiert sind. Auf die Nachbarkästchen hat das keinen Einfluss.



Erstelle die Datei „2.js_event_kontrollkasten.html“.

```
<body>
  <h1>Was willst du auf deiner Pizza haben?</h1>
<form>
<fieldset>
      <p><input type="checkbox" id="pfeff" value="Pfefferoni">
      <label for="pfeff">Pfefferonie</label></p>

      <p><input type="checkbox" id="zwieb" value="Zwiebel">
      <label for="zwieb">Zwiebel</label></p>

      <p><input type="checkbox" id="schink" value="Schinken">
      <label for="schink">Schinken</label></p>
      <br><br>
      <input type="button" onclick="bestellung()" value="Pizza Bestellung">
  </fieldset>
  </form>
  <h2>Deine Bestellung:</h2>
      <div id="output"> </div>
  </body>
  </html>
```

Hier ist nur zuerst das Formular erstellt worden.
Zur Beschriftung der Checkboxen ist hier ein <label> verwendet worden.
Zusätzlich ist darin das „for-Attribut“ verwendet worden. Es ist nicht notwendig, aber es bindet das Label an ein Kontrollkästchen und unterstützt den Nutzer: er kann nämlich auch auf die Bezeichnung klicken, nicht nur in das Kontrollkästchen, um das Kästchen zu aktivieren.

Info zu Kontrollkästchen:

- **id:** wird hier extra anders gewählt als der value, damit man es besser unterscheiden kann.
- **value**: diese Eigenschaft kann dazu verwendet werden, um zum Kontrollkästchen gehörende Werte aufzunehmen
- **checked**: bei dieser Eigenschaft handelt es sich um einen Boole´schen Wert, der angibt, ob das Kontrollkästchen aktiviert worden ist oder nicht.

Nun folgt der Code für die Funktion „bestellung()“

- Die Eigenschaft „checked“ ermittelt, ob ein Haken gesetzt wurde oder nicht.
- Wenn es aktiviert wurde, gib den Inhalt der Eigenschaft „value“ zurück, der zu dem entsprechenden Kontrollkästchen gehört.
- Aber es benötigt auch ein „else“ mit leeren Ergebnis, da sonst, wenn es nicht angehakt ist, eine Anzeige kommt „undefined“.

a)Lösung ohne Aufzählungspunkte und mit einer neuen Variablen pro „if“-Anweisung:

```
<script>
    function bestellung(){
        var pfefferoni = document.getElementById("pfeff"); //value
        var zwiebel = document.getElementById("zwieb");
        var schinken = document.getElementById("schink");

        var output = document.getElementById("output");

        if (pfefferoni.checked){
            var result1 =  pfefferoni.value + "<br>";
        } else{
                var result1 = "";
                }//ende if
        if (zwiebel.checked){
            var result2 = zwiebel.value + "<br>";
        } else{
                var result2 = "";
                }//ende if
        if (schinken.checked){
            var result3 = schinken.value + "<br>";
        } else{
                var result3 = "";
                }//ende if


        output.innerHTML = result1 + result2 + result3;
    }
</script>
</head>
```

b)Lösung mit Additionszuweisungsoperator

Die **Additionszuweisungsoperator** (`+=`) addiert einen Wert einer Variablen zu .

Das heißt: es dient zur Verlängerung einer Zeichenkette.
Siehe auch: **https://www.w3schools.com/js/js_operators.asp**

Der `+=` Zuweisungsoperator kann auch (verketten) Strings hinzuzufügen verwendet werden:

Beispiel

```
var txt1 = "What a very ";
txt1 += "nice day";
```

Das Ergebnis txt1 wird:

```
What a very nice day
```

Code eines Teils, damit man nicht so viel selbst schreiben muss:

```
<script>
   function bestellung(){
      var pfefferoni = document.getElementById("pfeff"); //value
</script>
```

```
if (pfefferoni.checked){
        result += "<li>" + pfefferoni.value + "</li> \n";
      } //ende if
```

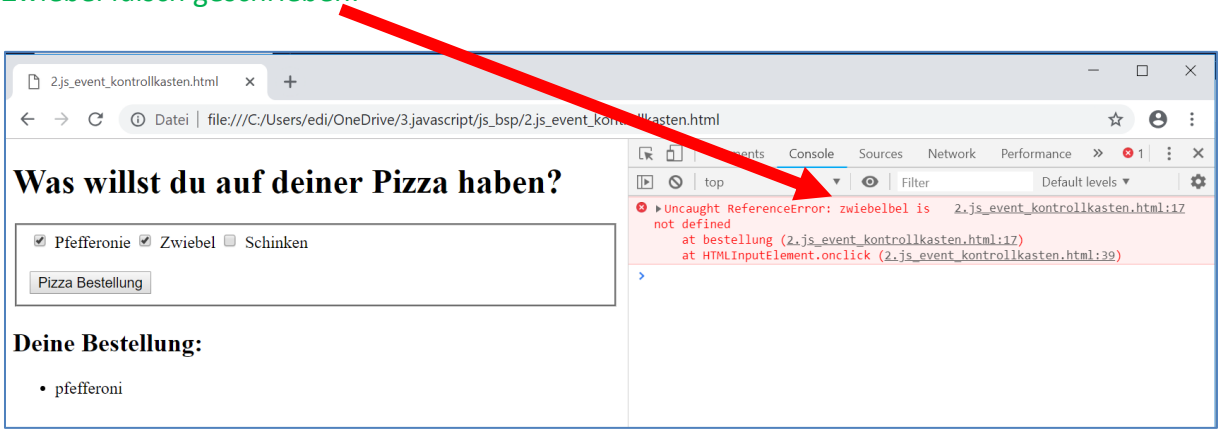
```
<head>
<script>
    function bestellung(){
        var pfefferoni = document.getElementById("pfeff"); //value
        var zwiebel = document.getElementById("zwieb");
        var schinken = document.getElementById("schink");

        var output = document.getElementById("output");
        var result = "<ul> \n"

        if (pfefferoni.checked){
            result += "<li>" + pfefferoni.value + "</li> \n";
        } //ende if
        if (zwiebel.checked){
            result += "<li>" + zwiebel.value + "</li> \n";
        } //ende if
        if (schinken.checked){
            result += "<li>" + schinken.value + "</li> \n";
        } //ende if

        result += "</ul> \n"
        output.innerHTML = result;
    }
</script>
</head>
<body>
```

Ergebnis:

Andy Harris, in: JavaScript für dummies, 2017, Verlag Wiley-VCH, Weinheim, S.176-185

Info: Verwendung von Entwicklertools: CONSOLE

Problem, dass nur Pfefferoni angezeigt werden. Lösung mittels „console“ zeigt warum: Zwiebel falsch geschrieben:



Quellen:

Andy Harris, in: JavaScript für dummies, 2017, Verlag Wiley-VCH, Weinheim, S. 161-173

Andy Harris, in: JavaScript für dummies, 2017, Verlag Wiley-VCH, Weinheim, S.176-185

Thomas Theis, in: Einstieg in JavaScript, 2015 Galileo Press, Bonn, S. 95-116

Florian Franke, Johannes Ippen; in: Apps mit HTML5, CSS3 und JavaScript: Rheinwerk Verlag, Bonn, 2015; S. 110-121

https://www.w3schools.com/js/js_objects.asp

offen, aber gut: https://www.youtube.com/watch?v=cAiSbkWvBQU Mario ab 1:15:45 Stunden:Minuten:Sekunden