Woraus besteht ein Programm bzw. eine Webseite?

Informationen

Was ist eine Information?

- ✓ Zahl
- ✓ Boolscher Wert (Ja/Nein)
- ✓ Text
- ✓ Bild
- ✓ Musik / Ton
- ✓ Video / Kamera
- ✓ Tabelle
- ✓ Verfügbarkeit (zb. Internetverbindung, Drucker, Scanner, WebCam, Sensor, etc.)

Kurz alles im Computer, jedes Programm und jede Webseite besteht aus Informationen.

Welche Arten gibt es?

Primitive Informationen:
Das sind die kleinsten Informationen, die es gibt: Zahlen, Boolscher Werte, Text

Datensätze (Arrays):
Ein Array kann leer sein, eine oder mehrere Informationen enthalten. Sie ist eine Art Liste.

Objekte:
Das sind komplexe Strukturen, die verschiedene Arten von primitiven Informationen, Arrays und ebenfalls Objekte enthalten können. Ein Objekt kann auch Funktionen enthalten.

Anweisungen

Was ist eine Anweisung?

Eine Anweisung ist ein Befehl an den Computer. Durch Anweisungen werden

- ✓ Aktionen ausgelöst
- ✓ Berechnungen durchgeführt
- ✓ Auf Eingaben (mit Tastatur bzw. Maus oder Touchscreen) reagiert
- ✓ Und vieles mehr

Merksatz: Daten alleine haben keinen Nutzen. Anweisungen können nicht ohne Daten erstellt werden.

Alles hat ein **Bezeichner** !!!!!!!!!!

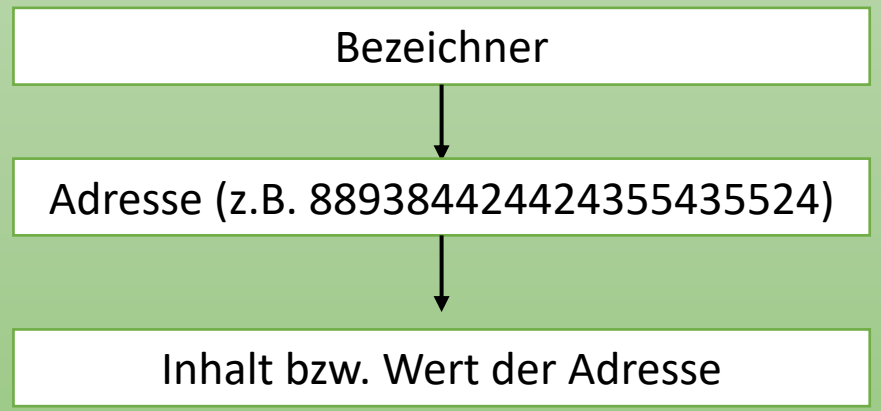
Was sind ein Bezeichner?

Jedes Element, die man im Quelltext definiert, bekommt einen Bezeichner.
Alle Variablen haben einen Bezeichner:

```
let a = 6; // a ist der Bezeichner, 6 ist der Wert zu Zuweisung

const b = 7; // b ist der Bezeichner, 7 ist der Wert der Zuweisung

let liste = [ 1, 2, 3 ]; // liste ist der Bezeichner, [1, 2,3] der Wert der Zuweisung

function machWas ()  // machWas ist der Bezeichner der Funktion
{
}

const fahrzeug = {};  // fahrzeug ist der Bezeichner des Objekts

class aparat {    // aparat ist der Bezeichner der Klasse
};
```

Was sind Bezeichner?

Da im Computer alles in Wirklichkeit aus Adressen besteht und dieser erst beim Laden es Programms oder Webseite bekannt werden, benutzt man beim Programmieren nicht die Adressen, sondern Platzhalter für die Adressen. Das sind die Bezeichner. Ein Bezeichner zeigt auf eine Adresse im Arbeitsspeicher, auf dem wir unseren Wert ausrichten.



Was ist eine Variable, ein Array oder ein Objekt?

Variable

Eine Variable ist ein Platzhalter für nur einen einzigen Wert.

Auch NULL oder 0 ist eine Information. Auch NICHTS oder LEER ist eine Information.

Eine Variable kann eine Ganzzahl (zb 5), eine Kommazahl (zb 1,7), einen boolischen Wert (zb. true / false) oder einen Text (zb. ‘Hallo Welt‘) aufnehmen.

Array

Ein Array ist im Prinzip eine Liste. Sie kann

- ✓ Leer sein
- ✓ Mindestens eine
- ✓ Bis mehrere Informationen enthalten

Ein Array macht es einfach, die Informationen die sie enthält mit Hilfe von Schleifen

- ✓ Auszuwerten
- ✓ Zu manipulieren

Objekt

Ein Objekt ist ein Konzept wie im realen Leben. Sie enthält Eigenschaften und Methoden. Ein Beispiel wäre das Auto. Ein Auto hat zum Beispiel die Eigenschaften:

- ✓ Rote Farbe
- ✓ Vier Türen
- ✓ Ein Dieselmotor
- ✓ 160 PS

Die Methoden eines Autos sind zum Beispiel:

- ✓ Fahren
- ✓ Gang wechseln
- ✓ Bremsen

Objekte können andere Objekte, primitive Variablen und Arrays gleichermaßen enthalten.

Arten von Variablen, Arrays und Objekte

Variable

- ✓ null (NICHTS)
- ✓ undefined (Keine Referenz)
- ✓ Integer (Ganzzahlen)
- ✓ Float (Kommazahlen)
- ✓ Boolean (wahr / flasch)
- ✓ String (Text)
- ✓ Anonyme Funktionen

Array

Ein Array kann jede Art von Information in einer Liste zusammentragen.

Objekt

JavaScript bietet eine Reihe Standartobjekte an, die vom Browser bereitgestellt werden.
Beispiele:

- ✓ Ereignisobjekt (Event)
- ✓ DOM (Element Manipulation)
- ✓ document (Dokument Manipulation, DOM ist ein Teil davon)
- ✓ window (Manipulation des Browser-Tabs bzw. Fensters)
- ✓ Date (Datum)
- ✓ Math (Mathematische Mittel)

Gemeinsamkeiten von Variablen, Arrays, Objekte und aller Daten

Editierbarkeit

Alle Variablen, Arrays und Objekte können auf drei Arten definiert werden:

var
Die deklarierte Adresse ist entweder im globalen Ebene oder in einer lokalen Ebene verfügbar. Sie kann gelesen und geschrieben werden.

let
eine deklarierte Adresse ist immer nur in der unmittelbar definierten Ebene verfügbar. Sie kann gelesen und geschrieben werden.

const
wie let. Die Adresse ist nur
Lesbar, kann nur einmal geschrieben werden.

Verfügbarkeit

Jede Information ist eine Adresse. Daher kann man dessen Verfügbarkeit prüfen, in de man eingibt:

If (element)
    Sie ist verfügbar bzw. enthält eine Information ungleich 0.
else
    Sie ist nicht verfügbar bzw. ist gleich 0.

Wir können auch den Typ einer Adresse ermitteln:

console.log(**typeof** element);

Wann braucht man eine Variable, wann ein Array, wann ein Objekt?

Variable

Eine Variable wird benötigt, wenn nur eine einzige Information benötigt wird. Typische Fälle sind:

- ✓ Einfache Rechenvorgänge wie c=a+b
- ✓ Darstellen einer einzigen Information wie console.log(x)
- ✓ Prüfen auf Verfügbarkeit wie if (x==y)
- ✓ Um wenig Speicher zu verbrauchen

Beispiele:

- ✓ Login Vorgang
- ✓ Kontaktformular
- ✓ Taschenrechner

Array

- ✓ Ein Array wird in der Regel dann benötigt, wenn man eine Listenlogik benötigt.
- ✓ Zudem wird ein Array verwendet, wenn man mit weniger Code, jedes Element in einem Array schnell erreichen und manipulieren bzw. auslesen muss. Stichwort: Schleife
- ✓ Ein Array kann Daten statisch oder dynamisch aufnehmen. Statische Daten werden im Quelltext der Array übergeben. Dynamische Daten werden zu Laufzeit in den Array übergeben.

Objekt

- ✓ Objekte werden bei größeren Projekten verwendet, um die gesamte Funktionalität des Programms in sinnvolle Abschnitte aufzuteilen.
- ✓ Das macht es später einfach, Erweiterungen zu erstellen, Fehler zu finden und zu beseitigen, Optimierungen durchzuführen und so weiter.

Beispielszenario:

Eine Webseite könnte aus folgenden Objekten bestehen:

- ✓ Kundendaten
- ✓ Produktdaten
- ✓ Aktuelle Angebote
- ✓ Bestellvorgang

Die Variable, das Array, das Objekt, die Funktion und dessen Inhalt

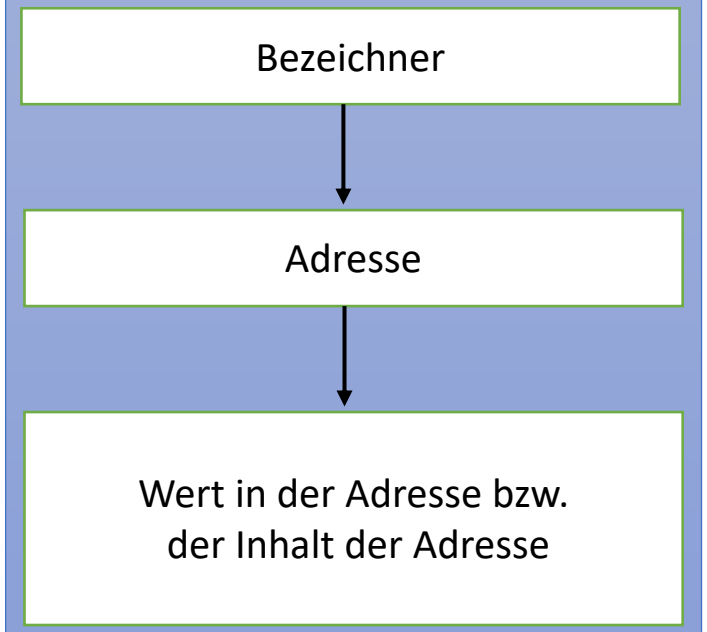
Bezeichner

Der Bezeichner ist der Name, dem man einer Variable, einem Array, einem Objekt oder einer Funktion vergibt:

```
const wert = 2;
let liste = [];
const objekt = {

};

const tuWas = () =>
{

}
Oder

function tuWas ()
{

}
```

Inhalt

Der Inhalt bzw. Wert eines Bezeichners ist eine Stelle im Arbeitsspeicher. Der Bezeichner hat nicht wirklich den Wert, sondern nur die Adresse, die auf den Wert zeigt:



Fehlerquellen erkennen

- ✓ Man darf niemals die Quelle mit dem Ziel gleichsetzen
- ✓ Ziel und Quelle sind immer von einander unabhängige Dinge

Beispiele:

```
const zeile1 = tabelle.insertRow(1)
```

Die beiden 1ner haben nichts mit einander zutun. „zeile1“ ist die Variable. insertRow ist eine Methode des Objekts “tabelle“ und die 1 ist ein Parameter der Methode.

```
const name = ´Das ist sein name´;
```

Die beiden „name“ Ausdrücke haben nichts gemeinsam. „const name“ erstellt eine Variable die den Bezeichner „name“ hat. Der andere „name“ ist nur Text in einem String.

Arbeiten mit einer Variable

Erstellen

var x;
var x = wert;

let x;
let x = wert;

const x = Wert; -> muss einen Wert haben!!!

Werte:

x = **true** oder x = **false** -> Boolischer Wert

x = 5 oder x = -5 -> Ganzzahl (Integer)
x = 0.5 oder x = -4.62 -> Kommazahl (Float)

x = ‘Hallo‘ -> Text

x = [] -> Leerer Array

x = [52, 31, 60, -94] -> Ein Array mit 4

Schreiben

Wie bei Werte von neben an, aber es geht weiter:

x = a + b

x = (a * b) / c

x = “Mein Name ist“ + name

x = ((a – 0.5) * b) + 4

x = produkt[0] + “ ist ausverkauft“

x[5] = x[2] - (a * y[z])

Lesen

Die Grundlage beim Lesen einer Variable, ist mit dem Schreiben gleich:

SchreibenIn = LesenAus

Beispiel:

x = a + b

Die Werte von a und b werden gelesen, addiert und in die x geschrieben. Weitere Methoden zum auslesen:

- ✓ console.log(x)
- ✓ alert(x)
- ✓ Objekt.value = x
- ✓ Objekt.innerText = x
- ✓ Objekt.innerHTML = x

Was ist eine Anweisung und wie erstellt man sie und was machen sie?

Eine Anweisung

Eine Anweisung ist ein Stück Code, der etwas macht. Die Anweisung braucht Informationen, um überhaupt definiert werden zu können.

Einfache Beispiele für eine Anweisung sind:

- ✓ x = a + b
- ✓ FunktionsAufruf()
- ✓ z = FunktionsAufruf(x,y)
- ✓ if (…) …

Allein, in Block

Anweisungen sind immer im Kontext einzubauen. Wenn der Kontext nur eine Anweisung benötigt, so gibt man nur die eine Anweisungen an. Wenn der Kontext mehrere Anweisungen benötigt, dann werden die Anweisungen in {} Klammern geschrieben. Der Kontext kann sein:

- ✓ If, else if, else
- ✓ for, while

Funktionen müssen die {} haben.

Funktion, globale Ebene

In JavaScript ist die globale Ebene bereits eine Art Funktion. Man kann darin Anweisungen schreiben:

```
<script>
  let x = 5;
  let y = 3;
  let r = 0;
  r = x + 5;
</script>
```

Das ist aber keine gute Lösung. Am besten schreibt man Anweisungen immer in eine Funktion:

```
function meineFunktion ()
{
 ….
}
```

Geltungsbereiche erkennen und Fehler vermeiden

Was ist ein Geltungsbereich ?

Der oberste Geltungsbereich ist eine leere JavaScript-Datei oder der unmittelbare Inhalt einer Scirpt-Tag. Dieser ist die sogenannte globale Ebene. Jeder Block ist wiederum ein weiterer Geltungsbereich. Blöcke sind immer {} Klammerninhalte.

```
Oberste globale Ebene
{                    Unterebene 1
  {                  Unterebene 2
    {                Unterebene 3
      …
    }
  }
}
```

Gültigkeit von Bezeichnern

Jeder Bezeichner ist immer in der Ebene und in dessen Unterebenen gültig, in dem der Bezeichner definiert wurde:

```
let a = 5;    // Gültig in allen Ebenen
{
let b = 2;  // Gültig hier & in unterebene
    {
        let c = 0;   // Gültig nur hier
    }
}
```

Arten von Anweisungen

Ziel – Quelle

Bei einer Ziel – Quelle – Anweisung gibt es zwei Operanden und einen Operator.

- ✓ Ein Operand ist entweder ein Ziel oder eine Quelle.
- ✓ Der Operator gibt die Art der Verarbeitung an.

Beispiel: x = y

x und y sind Operanden
= ist der Operator

Das Ziel erhält da Ergebnis aus der Quelle. Die Quelle kann ein Wert sein, oder eine komplexe Operation sein:

x = 5 -> x erhält den Wert 5 zugewiesen
x = 8 – f -> x erhält den Wert aus 8 – f
x = funktion() -> x erhält das Ergebnis der Funktion
x = (a + foo(b – 1, 3)) * 0.5 -> x erhält das Ergebnis

Aufruf

Eine Anweisung kann einfach das Aufrufen einer Funktion sein, wenn die Funktion nichts zurückgibt:

// Ohne Parameter
x()

// Mit einem Parameter
x(a)

// Mit mehreren Parametern
x(a, b, c)

Befehlsanweisung

Spezielle Sprachregeln wie if, else if, for, switch, case, while sind solche Befehlsanweisungen. Beispiele:

If (x == 5) -> Wenn x ist gleich 5

for (x of n) -> Geh jedes Element x von n durch

while (r != 0) -> Solange r nicht 0 ist…

Switch (x) -> Betrachte den Wert von x
case 2: -> Wenn der Wert von x = 2 ist …

Anweisungen : Ziel und Quelle

Ziel – Quelle - Verhältnis

Das Ziel einer solchen Anweisung ist immer eine Variable, Array oder Objekt. Das Ziel ist immer nur ein einziges Element. Die Quelle kann entweder nur eine Zahl, ein boolischer Ausdruck (true / false), eine null, ein String (“Hallo“), eine andere Variable, Array, Objekt oder eine Funktion sein. Solange die Quelle primitiv sein soll, kann sie auch eine Kombination aus primitiven Variablen, Zahlen, boolischen Ausdrücken und Strings sein.

Primitiv:

```
x = a + b + 5 – 0.5 + ´ Oh mein Gott ☺´
x = (a + b)
x = ´Hallo, mein Name ´ + ´ ist Anton´
x = 5
```

Array:

```
x = []
x = [ 2, 3, 4, ´Fisch´, ´Birne´, true, false ]
```

Objekt:

```
x = objekt
x  = {
        …
     }
```

Funktionen

Model 1:

```
x = function ()
{
   …
}
```

Model 2:

```
x = function (a, b, c)
{
   …
}
```

Model 3:

```
x = () =>
{
   …
}
```

Model 4:

```
x = (a, b, c) =>
{
   …
}
```

Operatoren:

`=` Überschreibt die Varaible

`x = 2 + 5 -> 7`

`+=` Addiert auf den bestehenden Wert:

```
x = 2
x += 7    -> 9
```

`-=` Zieht vom bestehenden Wert ab

```
x = 6
x -= 2   -> 4
```

`*=` Multipliziert drauf

`/=` Teilt durch den Wert

Wozu benötigt man Funktionen?

Konzept

Eine Funktion fasst einen ganzen Block von Anweisungen unter einer gemeinsamen Adresse, dem Funktionsnamen zusammen. Sie hat den Vorteil, einmal erstellt, kann sie im gesamten Programm aufgerufen werden.

Ein weiterer Vorteil einer Funktion ist, dass man ihr Parameter übergeben kann. Dank der Parameter bleibt die Funktion zwar immer dieselbe, kann aber unterschiedliche Resultate produzieren.

Beispiel Funktion

```
function rechner ( z1, z2, weg)
{
    if ( weg == ´+´ )
      return z1 + z2;
    else if ( weg == ´-´ )
      return z1 - z2;
    else if ( weg == ´*´ )
      return z1 * z2;
    else if ( weg == ´/´ )
      return z1 / z2;
    else
      return 0;
}
```

Funktion testen

```
// 5 + 3 = 8
console.log(    rechner( 5, 3, ´+´ )    );

// -2 + 7 = 5
console.log(    rechner( -2, 7, ´+´ )    );

// 9 - 5 = 4
console.log(    rechner( 9, 5, ´-´ )    );

// 6 * 2 = 12
console.log(    rechner( 6, 2, ´*´ )    );

// 21 / 3 = 7
console.log(    rechner( 21, 3, ´/´ )    );

// 5 % 3 = 0, weil % als Rechenweg nicht vorgesehen ist
console.log(    rechner( 5, 3, ´%´ )    );
```

Kontrollstruktur - If

Was ist das?

Eine Kontrollstruktur erlaubt es, den Programmablauf gezielt zu steuern. Das bedeutet, dass man an Hand von Daten entscheiden kann, was man als nächstes tun wird.

Wenn das Wetter sonnig ist, dann geh ich raus zum Fußball spielen. Andernfalls bleibe ich lieber zuhause und schaue fern. Die Wetterlange ist also ein Faktor, woraus ich eine Entscheidung treffe, was ich machen werden.

Schlüsselelemente

Begriffserklärung:

if = Wenn
else if = oder wenn
else = ansonsten

Logische Operatoren:

&& und
|| oder
! Das Gegenteil

Eine Anweisung kann direkt nach der Regel stehen, wenn mehrere Anweisungen benötigt werden, müssen diese in {} Klammern angegeben werden.

Aufbau

```
if ( ausdruck )
{
  …
}
```

```
if ( ausdruck && ausdruck || ausdruck )
{
   …
 }
```

Wenn nach einer IF-Kontrollstruktur, ein weitere „oder wenn“ erfolgen soll, oder ein „ansonsten“, so geht es:

```
if (…)
else if (…)
```

```
if (…)
else
```

```
if (…)
else if (…)
else if (…)
else if (…)
```

```
if (…)
else if (…)
else if (…)
else if (…)
else
```

Ein Ausdruck besteht aus zwei Quellen und einem Vergleichsoperator:

```
if( X == Y)   oder   if(x === y)   -> Wenn x und y gleich sind
if( X != Y )                       -> Wenn x und y unterschiedlich sind
if( X > Y   )                      -> Wenn x größer ist als y
if( X >= Y )                       -> Wenn x größer oder gleich y ist
if( X < Y   )                      -> Wenn x kleiner ist als y
if( X <= Y )                       -> Wenn x kleiner oder gleich y ist
if(x)                              -> x ist ein gültiger Ausdruck
```

```
// Wenn x
// gleich 5
// nicht ist
if ( !x == 5)
```

```
// Wenn das
// element
// nicht
// existiert
if ( ! element )
```

```
// Wenn a
// und b
// sich nicht
// unterschei-
// den
if ( ! a != b )
```

Kontrollstruktur – If - Beispiele

Login Vorgang

```
if ( user === dbuser &&
     pass === dbpass )
{
    login = true;
    userid = dbuserid;
    username = dbuser;
    console.log(`Hallo ${username}`);
}
else
{
     console.log(`Benutzername oder
Passwort falsch`);
}
```

Was wenn…

```
if ( money > 0 )
{
    console.log(`Du bist mit ${money}
im Plus! Herzlichen Glückwunsch.`);
}
else if ( money == 0 )
{
    console.log(`Kein Geld auf Konto!`);
}
else if ( money < 0 )
{
    console.log(`Du bist mit ${money}
im Minus! Verdiene Geld!!!`);
}
```

Verfügbarkeit prüfen

```
If ( document.getElementById(`element`)  )
{
    console.log(`Das Objekt existiert`);
}
else
{
    console.log(`Das Objekt existiert NICHT!`);
}
```

Kontrollstruktur - switch

Was ist das?

Auch switch ist eine Kontrollstruktur. Sie prüft aber nur auf “ist gleich“ und „sonst“.

Wenn das WETTER…

- ✓ Sonnig ist, gehe ich Fußball spielen
- ✓ Bewölkt ist, gehe ich einkaufen
- ✓ Regnerisch ist, gehe ich zum Rathaus
- ✓ Sonst bleibe ich zuhause und schaue fern

Schlüsselelemente

Begriffserklärung:

switch = Prüfe das
case = Im Fall von
default = ansonsten
break = bis hierher

Die switch-Anweisung hat immer einen {} Klammer. Darunter liegen alle Fälle (cases) und der Sonst-Fall (default).

Aufbau

```
switch ( ausdruck )
{
    case WERT1:
        Anweisung(en);
        break;
    case WERT2:
        Anweisung(en);
        break;
    default:
 Anweisung(en);
        break;
}
```

Der Ausdruck einer „switch“ ist in der Regel eine einzige Variable, aber es kann auch alles andere sein:

```
switch( x )
switch( array.length )
switch( array[element] )
switch( funktion() )
switch ( funktion(a,b,c) )
```