Kathegorie			
Eisorten	preis name		
Toppings	preis id name		
Behälter	preis name		
Lieferoption	sofor express auf Uhrzeit		
Adresse	Name Vorname Straße ...		

▷ bleibt immer gleich

<input value:□ name:□ id:□
class:□

interface:

Kathegorie
Bezeichnung ≙ Name ≙ Schockolade
Preis

Kathegorien ≙ Array → alle Kath sind 1 Array: Fieldsets / Angebot

Element Array 1

Eissorten
- Schokolade , Preis
- Vanille , Preis
- Erdbeer , Preis

Behälter
- Waffel , Preis
- Becher , Preis

Toppings
- Streusel , Preis
- Soße , Preis

Lieferoptionen
- sofort , Preis
- abholen , Preis

Elemente
Array 2 = Angebot[]

teil des Interfaces der Elemente

Ts1

Funktionen

- write HTML
- check Angaben
- write In bestellung
- impot from Ts2

Ts2

Daten zum Pflegen/Tabellen
interface

create <fieldset>
↳ legend

create <input>

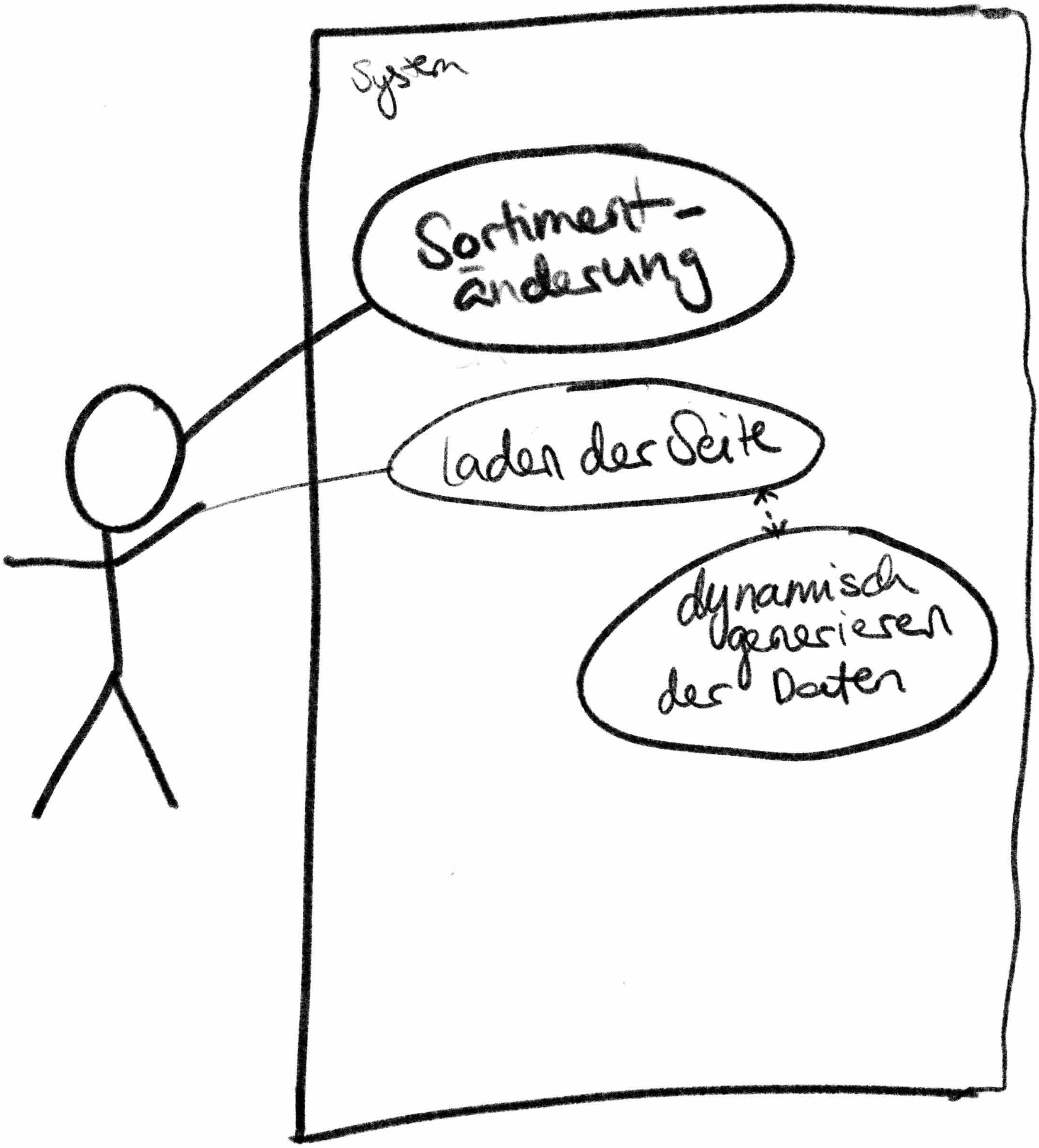
add Eventlistener

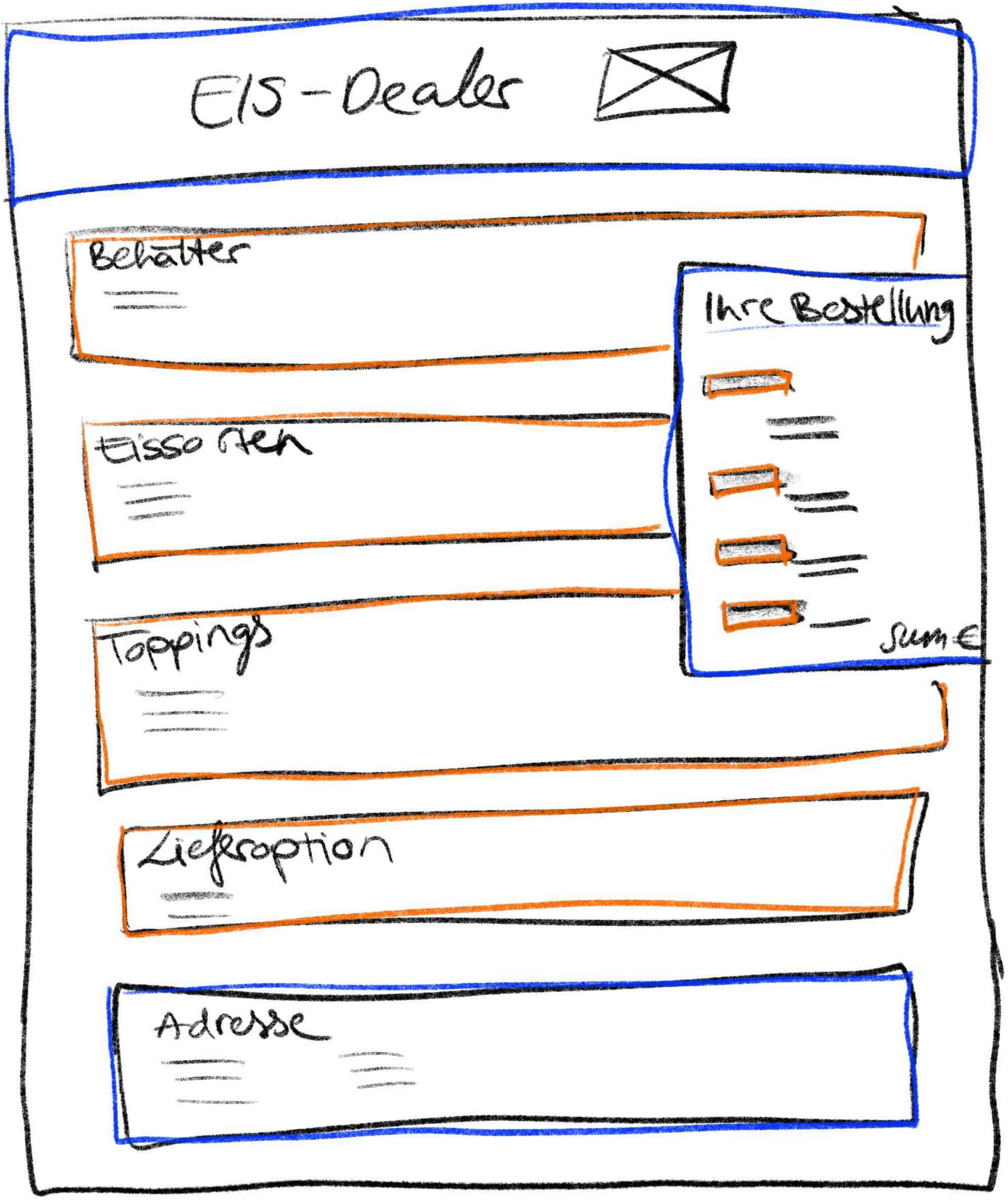
HTML

```
<DOC>
 <head> ... </head>
 <header> ... </header>
  <body>
   <div id=1>        </aside
   <aside> ...
   <div id=2> (für die fieldsets)

 </body>
 <footer> ... </footer>
```

Anwendungsfalldiagramm





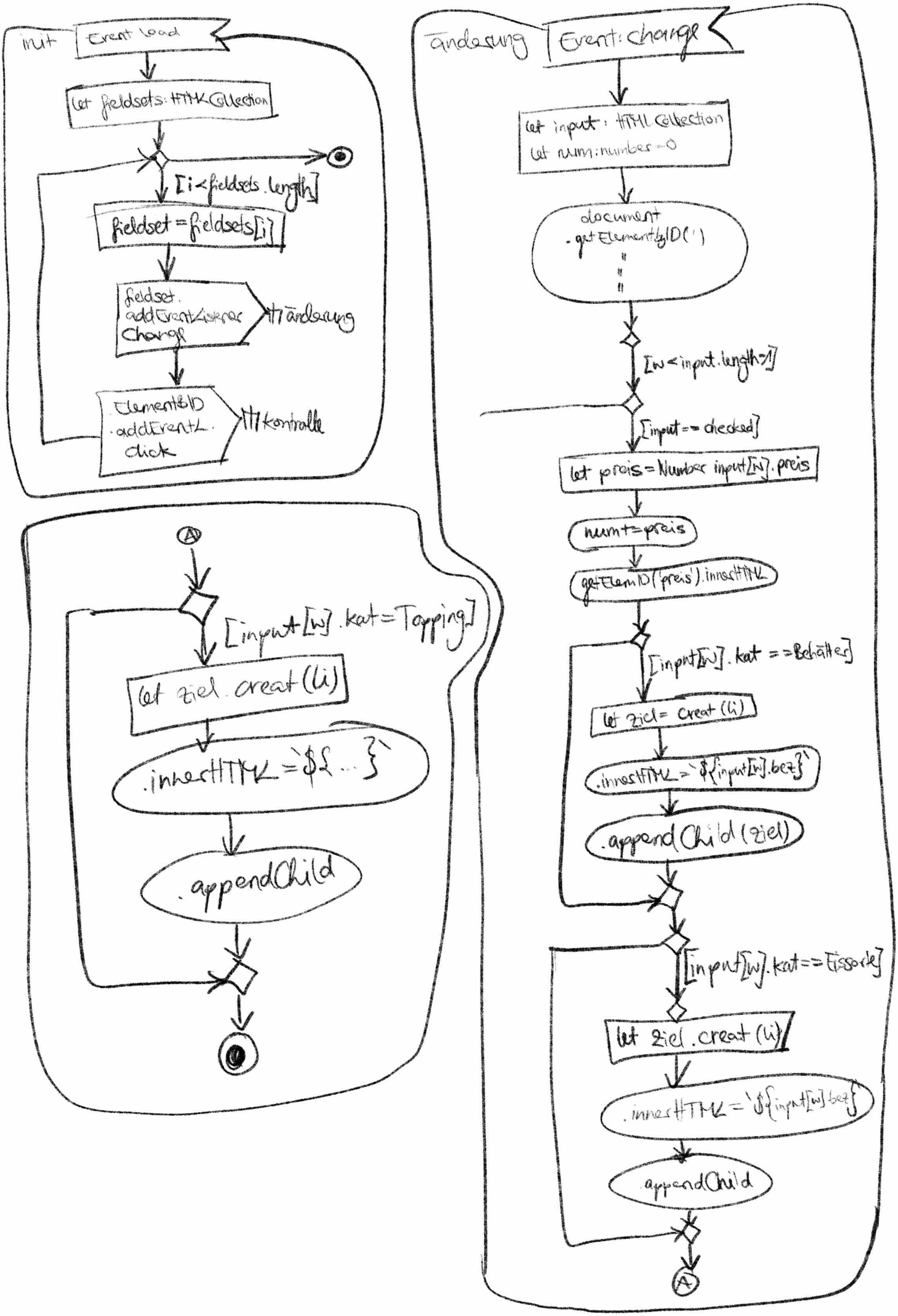
```
<DOC>
<head>
<header>
<aside> <p> ... </aside>
<main>
  <div>
    <fieldset> legend
      <input> ...
    <fieldset> legende
      <input> ...
      :
<fieldset> Adresse
  <input> ...
<footer>
```

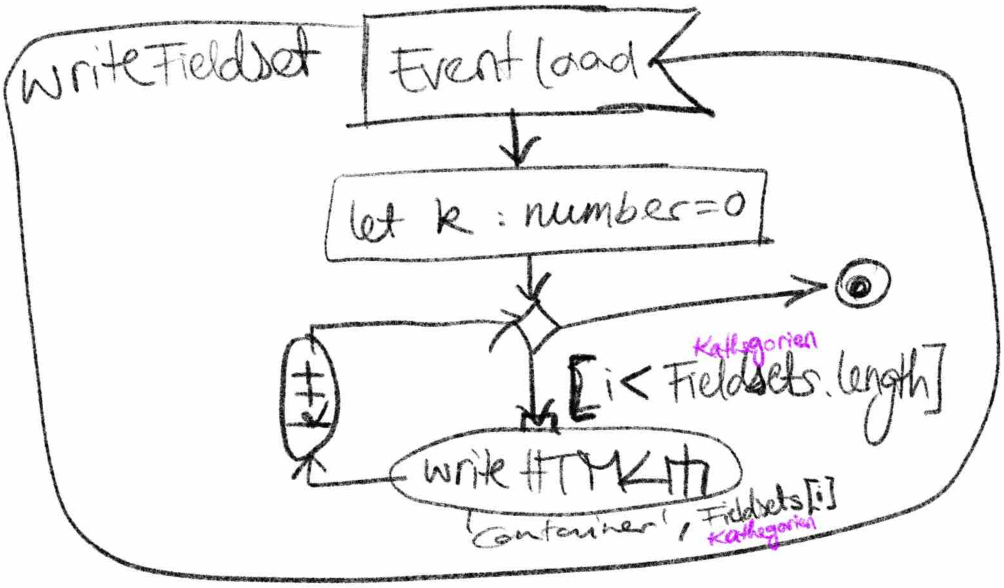
🟦 bleibt 🟧 wird dynamisch gebaut

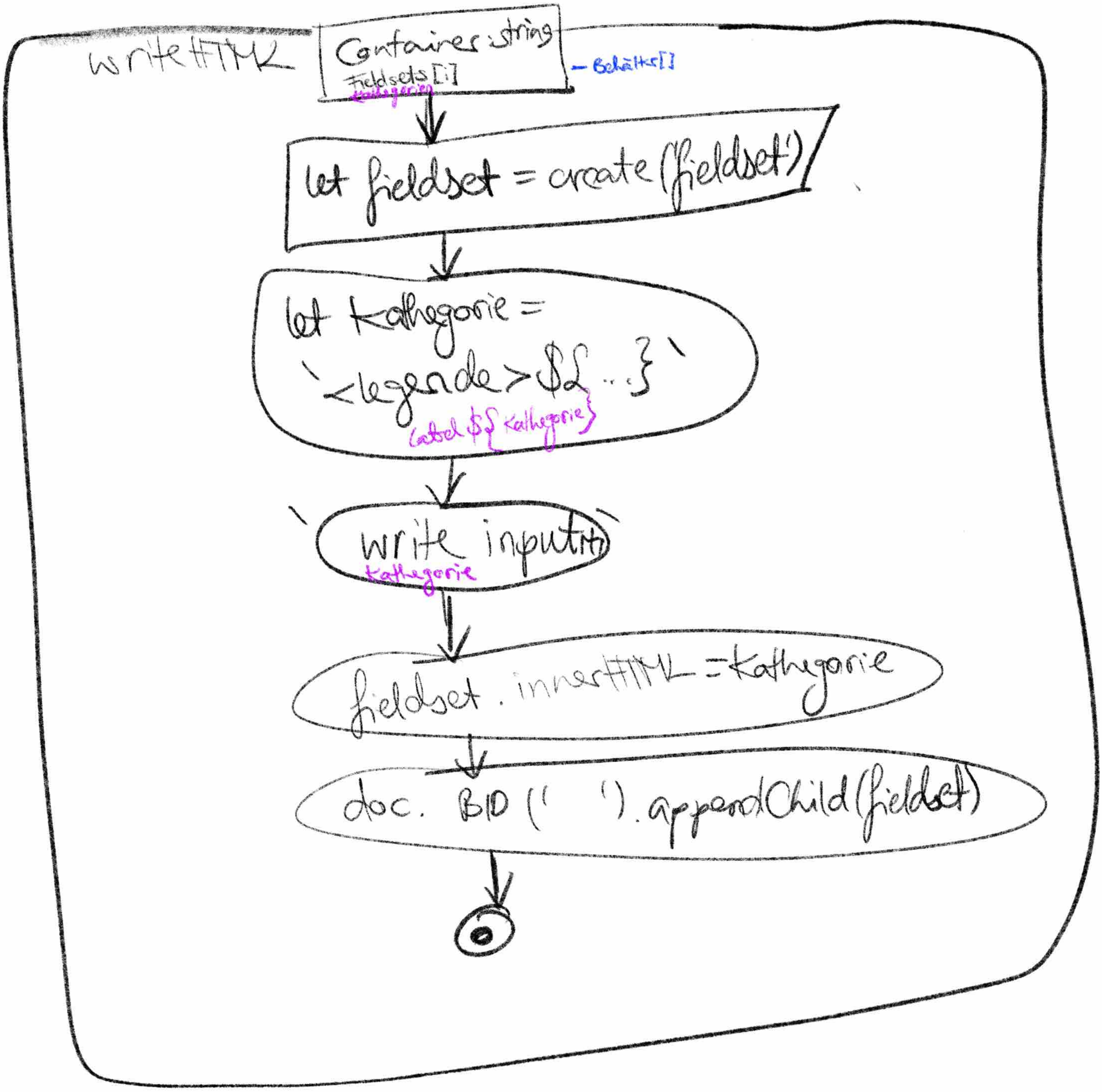
as Foo → fix this simply by a type assertion
you can add numbers and strings
for consistency

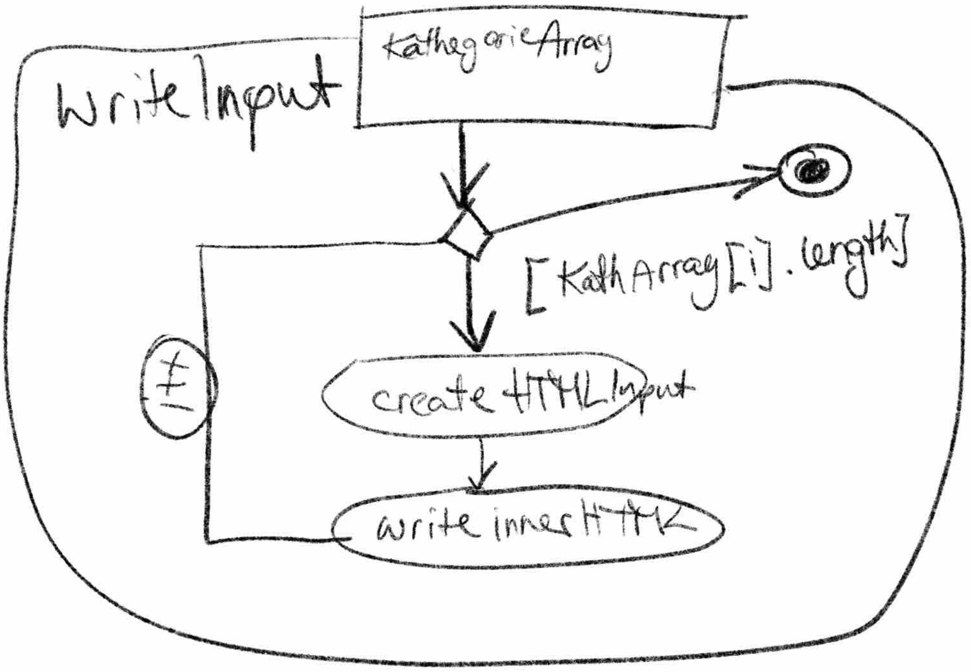
Casting: implies some sort of runtime support

assertions: compile time construct & way for you to provide
hints to the compiler on how you want your code
to be analyzed


init
Event load
let fieldsets: HTMLCollection
[i<fieldsets.length]
fieldset = fieldsets[i]
fieldset.addEventListener change
änderung
ElementByID.addEventL. click
kontrolle
änderung
Event: change
let input: HTML Collection
let num: number = 0
document.getElementByID('')
[w<input.length-1]
[input == checked]
let preis = Number input[w].preis
num += preis
getElemID('preis').innerHTML
[input[w].kat == Behälter]
let ziel = creat (li)
.innerHTML = `${input[w].bez}`
.append Child (ziel)
[input[w].kat == Eissorte]
let ziel.creat (li)
.innerHTML = `${input[w].bez}
appendChild
A
A
[input[w].kat = Topping]
let ziel.creat (li)
.innerHTML = `${...}`
.appendChild


writeFieldset
Event load
let k : number=0
i++
Kathegorien
[i< Fieldsets.length]
writeHTML()
'Container', Fieldsets[i]
Kathegorien


writeHTML
Container:string
Fieldsets[i]
Kathegorien
— Behälter[]
let fieldset = create('fieldset')
let kathegorie =
`<legende>${...}`
label ${kathegorie}
write input()
kathegorie
fieldset.innerHTML =kathegorie
doc. BID(' ').appendChild(fieldset)


writeInput
KathegorieArray
[KathArray[i].length]
i++
createHTMLInput
write innerHTML