

• Datenstrukturen

- ① Heterogenes assoziatives Array mit vordefinierten Schlüsseln:

```
export interface waren {  
  bezeichnung: string;  
  preis: number;  
}
```

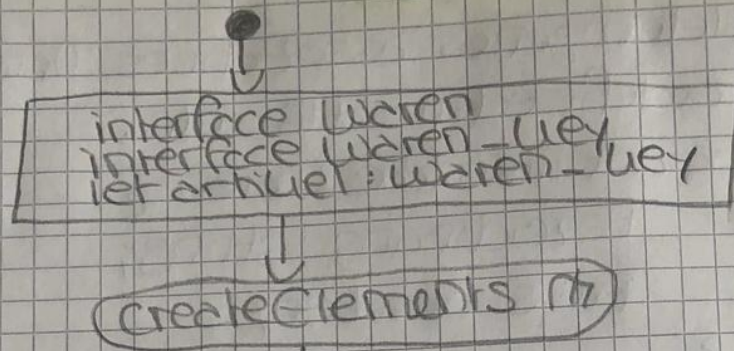
- ② Homogenes assoziatives Array mit variablen Schlüsseln, ein String wird abgebildet auf ein Array mit Objekten von obigen Typ:

```
export interface waren-key {  
  [key: string]: waren[];  
}
```

- ③ Neue Produkte und Kategorien hinzufügen:

```
export let artikel: waren-key = {  
  "Sorte": [ { bezeichnung: "...", preis: ... },  
             { bezeichnung: "...", preis: ... } ],  
  "Halter": [ { bezeichnung: "...", preis: ... },  
              { bezeichnung: "...", preis: ... } ],  
  :  
  :  
}
```


• Aktivitätsdiagramme



createElements
(Bsp. Baumart)

```
graph TD; Fork(( )) --> CreateElements[let sorte: HITULDivElement  
doc.getelementById("sorte")  
-> Auf Schlüssel zu "Baumarten"  
in extra Datei zugreifen  
mit displayWaren(artikel["sorte"]); CreateElements --> Join(( ))];
```

UML Activity Diagram showing the "createElements" activity (Bsp. Baumart):

- The fork node leads to a large oval node containing:
 - let sorte: HITULDivElement
 - doc.getelementById("sorte")
 - > Auf Schlüssel zu "Baumarten"
 - in extra Datei zugreifen
 - mit displayWaren(artikel["sorte"]);
- This node leads to a join node (circle with a solid black dot).

```
graph TD; Join(( )) --> Halterung[let halterung: HITULDivElement  
doc.getelementById("halterung")  
-> ebenfalls zugreifen mit  
displayWaren(artikel["halter"]); Halterung --> End(( ))];
```

UML Activity Diagram showing the "halterung" activity:

- The join node leads to a large oval node containing:
 - let halterung: HITULDivElement
 - doc.getelementById("halterung")
 - > ebenfalls zugreifen mit
 - displayWaren(artikel["halter"]);
- This node leads to an end node (solid black circle).

↓

- : Gleich für Schmuckartikel,
- : Beleuchtung und Lieferoptionen

End node (solid black circle).