```
●
add EventListener ("click", sort(cards);
add EventListener ("keydown", drawIfSpace);
add EventListener ("click", draw Card)
        ↓
┌─────────────────────────────────┐
│ hand [J; ablage Stapel [J        │
│ Interface Card { value: string;  │
│              color: string; }    │
└─────────────────────────────────┘
        ↓
┌───────────────────────┐
│ let deck: Card [] = [] │
└───────────────────────┘
        ↓
(  Prompt: Eingabe gewünschter  )
(       Hand karten             )
        ↓
┌─────────────────┐
│ random Card ++  │
└─────────────────┘
        ↓
    →◇
     ↓
( Schleife + random Card      )
(  ↳ Anzahl aus Prompt        )
(     wird aus deck           )
(     gezogen                 )
        ↓
( ++ place Card gibt Anzahl   )
( der karten in HTML aus      )
        ↓
       ◇
        ↓
       ●

i++

random Card ++    ●
        ↓
( Math.Floor n * deck.length )
        ↓
( weiter gabe an Schleife )
        ↓
       ☺

Place Card     ┌──────────────────────┐
               │ _value: string       │
               │ _color: string       │
               │ _handIndex: number   │
               └──────────────────────┘
                        ↓
               ┌──────────────────────┐
               │ div Element erstellen │
               └──────────────────────┘
                        ↓
               ( append Child zu      )
               ( div = id - hand      )
                        ↓
               ( erstellen von EventListener )
               ( auf jede karte             )
                        ↓
               ( Namerierung im Hand Index )
               ( für jede karte           )
                        ↓
                       ☺
```
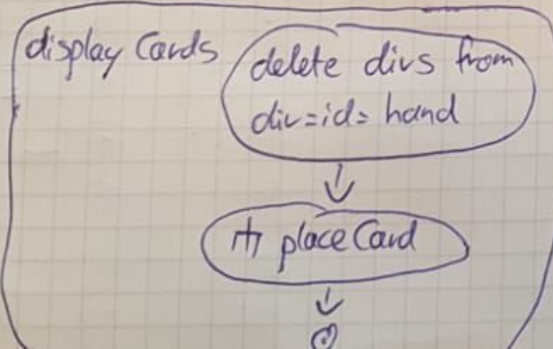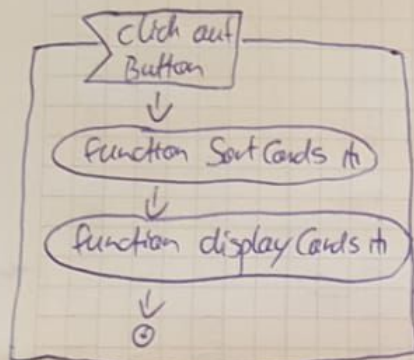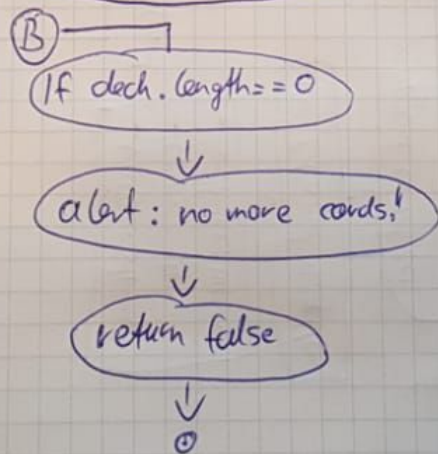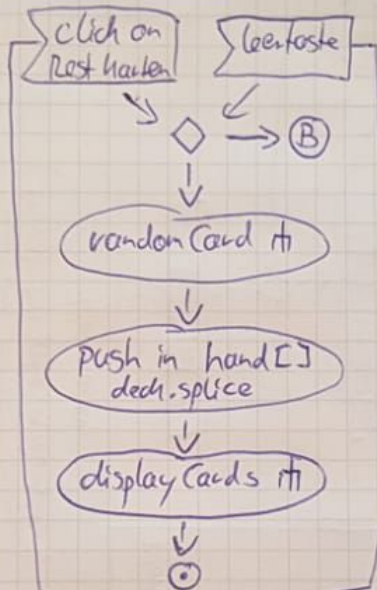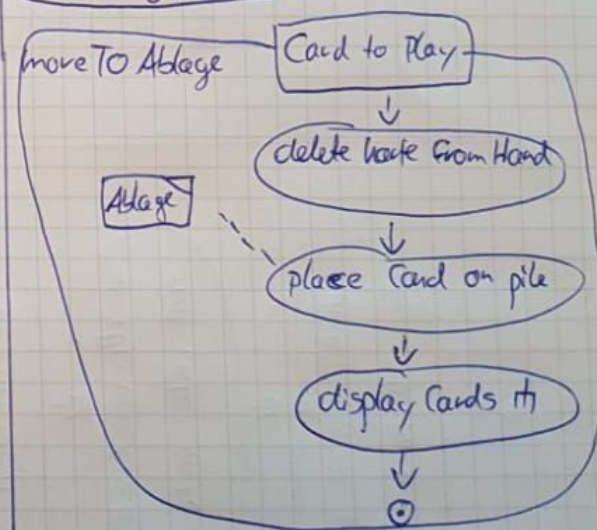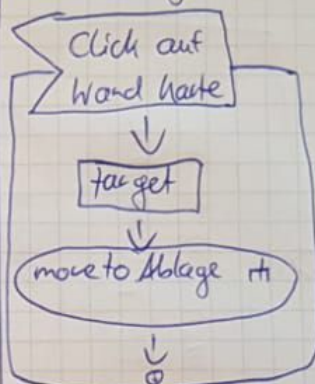
Farbsortieren:

```
┌─ click auf ──────────────────┐
│  Button                       │
│      ↓                        │
│  ( function Sort Cards ++ )   │
│      ↓                        │
│  ( function display Cards ++ )│
│      ↓                        │
│      ⊙                        │
└───────────────────────────────┘

┌ Sort Cards ( hand . sort ( Vergleich ) )──┐
│              ↓                             │
│        ( display Cards ++ )                │
│              ↓                             │
│              ⊙                             │
└────────────────────────────────────────────┘

┌ display Cards ( delete divs from ──┐
│                  div = id = hand )  │
│                    ↓                │
│             ( ++ place Card )       │
│                    ↓                │
│                    ⊙                │
```

Karten ziehen:

```
┌─ click on ──┐   ┌ leertaste ─┐
│  Rest karten │   │            │
│      ↘   ◇ → (B)              │
│           ↓                   │
│     ( random Card ++ )        │
│           ↓                   │
│     ( push in hand [ ]        │
│        dech . splice )        │
│           ↓                   │
│     ( display Cards ++ )      │
│           ↓                   │
│           ⊙                   │
└───────────────────────────────┘

(B)─┐
    │
( If dech . length == 0 )
        ↓
( alert : no more cards! )
        ↓
( return false )
        ↓
        ⊙
```

Karte ablegen:

```
┌─ Click auf ──────────┐
│  Hand karte           │
│      ↓                │
│   [ target ]          │
│      ↓                │
│ ( move to Ablage ++ ) │
│      ↓                │
│      ⊙                │
└───────────────────────┘

┌ move TO Ablage   [ Card to Play ]──┐
│                        ↓            │
│              ( delete karte from Hand )│
│   [ Ablage ]           ↓            │
│        ┆      ( place Card on pile )│
│        ┆               ↓            │
│               ( display Cards ++ )  │
│                        ↓            │
│                        ⊙            │
```

Aufgabe 3

UNO

USER

- Karten ablegen
- Karten nachziehen
- Karten sortieren

Sicht auf Spiel:

Rest Karten

DIV = ID = Restkarten

button ID = ColorSort

Button

Ablage

DIV = ID = Ablage

DIV = ID = MeinHand

+4   2   3   8

DIV = Class = Cards