Sprint X Development

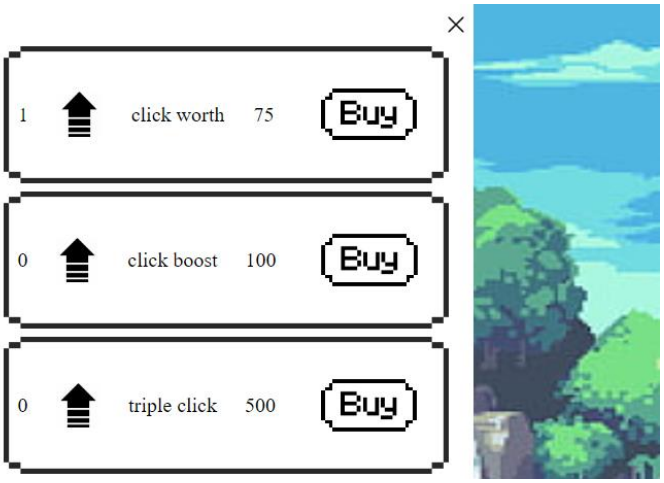
Opdracht:

Voor mijn sprint x heb ik gekozen een idle klikker game te coderen. Omdat ik graag verder wil gaan in development en dit mijn een goede manier leek om meer development doelstellingen aan te tonen. Via een idle clicker game kan ik namelijk simpel beginnen en gebaseerd op de hoeveelheid tijd de game steeds verder uitbreiden.

Proof of concepts

Shop:

```
{
    "id": "0",
    "name": "click worth",
    "upgradeType": "click",
    "upgrade": 1.8,
    "icon": "./images/level up.png",
    "level": 1,
    "cost": 75,
    "costMultiplier": 2
},
{
    "id": "1",
    "name": "click boost",
    "upgradeType": "chance",
    "chance": 0,
    "chanceUpgrade": 25,
    "upgrade": 1.8,
    "icon": "./images/level up.png",
    "level": 0,
    "cost": 100,
    "costMultiplier": 2
},
```




```
for (var i=0;i<upgradefile.length;i++){
  const shop = document.querySelector(".sidenav")
  const shopitem = shop.appendChild(document.createElement('div'));
  shopitem.classList.add('shopitem')
  const kader = shopitem.appendChild(document.createElement('img'));
  kader.classList.add('kader');
  kader.src = "./images/shopkader.png"
  const level = shopitem.appendChild(document.createElement('p'));
  level.classList.add("upgrade-level");
  level.innerText = upgradefile[i].level
  level.id = "L" +[i]
  const icon = shopitem.appendChild(document.createElement('img'));
  icon.classList.add("icon-size");
  icon.src = upgradefile[i].icon
  const name = shopitem.appendChild(document.createElement('p'));
  name.classList.add('upgrade-name')
  name.innerText = upgradefile[i].name
  const cost = shopitem.appendChild(document.createElement('p'));
  cost.classList.add('cost');
  cost.innerText = upgradefile[i].cost
  cost.id = "C" +[i]
  const buy = shopitem.appendChild(document.createElement('img'));
  buy.classList.add('buy');
  buy.src = "./images/buy.png"
  buy.id = upgradefile[i].id

}
```

De shop is gevuld via json objecten die via een for loop in de shop worden geplaatst. Deze vaardigheid heb ik geleerd uit de to-do list opdracht. Om te zorgen dat de upgrades ook koopbaar zijn heb ik een buy knop toegevoegd. Deze buy knop werkt door middel van een forEach method.

```
const shop = document.querySelectorAll('.buy')

shop.forEach( div =>
    div.addEventListener("click", (e) => {
        var purchase = e.target.id
        if(purchase <= 2){
          buy(purchase)
```

In de buy function word gekeken of de speler genoeg geld heeft om de upgrade te kopen. Als dit is dan worden de kosten van de hoeveelheid geld van de speler afgetrokken. Nadat er voor de upgrade betaald is word er gekeken wat voor type upgrade het is. Dit is belangrijk omdat er voor verschillende upgrades verschillende acties genomen moeten worden.

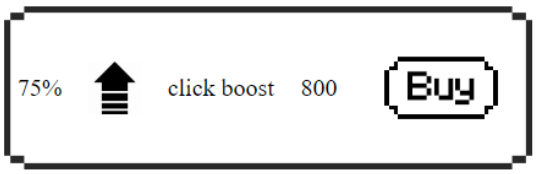
```
function buy(x){
    if(money >= upgradefile[x].cost){
      money -= upgradefile[x].cost
      updatescore()
                if (upgradefile[x].type == "cps"){
                  cps += upgradefile[x].cps;
                  levelup(x)
                }
                if (upgradefile[x].upgradeType == "chance"){

                  chancePercentage(x)
                  chanceup(x)
                  costup(x)
                }
                else if (upgradefile[x].upgradeType == "click"){
                  levelup(x)
                  clickupgrade(x)
                  costup(x)
                }
```

Bij de functie chancePercentage word de waarde van de kansberekening omhoog gedaan in het toepassende json item.

```
function chancePercentage(x){
  var chance = upgradefile[x].chance
  chance += upgradefile[x].chanceUpgrade
  upgradefile[x].chance = chance
}
```

Bij de functie chanceup word voor de speler visueel de kans verhoogt. Dit word weergegeven bij de upgrade in de shop

```
function chanceup(x){
  chancetext = document.getElementById("L"+[x])
  level = upgradefile[x].level
  chancetext.innerText = upgradefile[x].chance + "%"
}
```



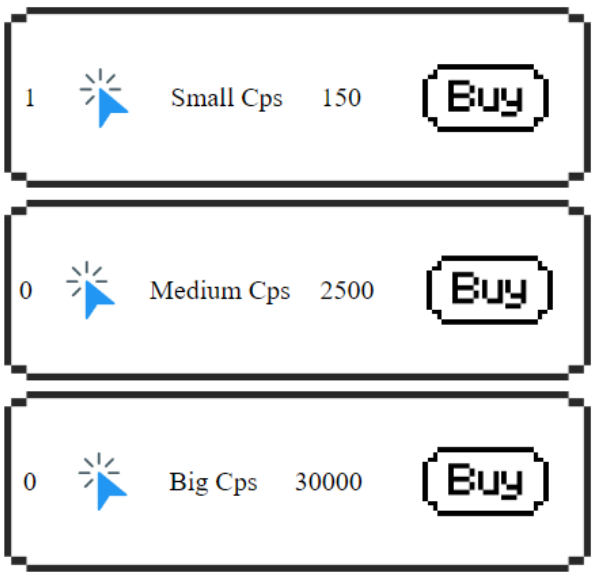
De functie costup verhoogt de kosten van de gekozen upgrade met de daarbij passende costmultiplier uit de json file. En update de prijs in de shop zelf.

```
function costup(x){
  costtext = document.getElementById("C"+[x])
  cost = upgradefile[x].cost
  cost = cost * upgradefile[x].costMultiplier
  costtext.innerText = cost
  upgradefile[x].cost = cost

}
```

De levelup functie verhoogt het level in de json file en het weergegeven level in de shop.

```
function levelup(x){
  leveltext = document.getElementById("L"+[x])
  level = upgradefile[x].level
    level += 1
    leveltext.innerText = level
    upgradefile[x].level = level
}
```

In de shop kunnen ook cps (clicks per second) upgrades gekocht worden. Deze upgrades geven de speler een een vaste hoeveelheid geld erbij zonder dat de speler hier iets voor hoeft te doen.



Elke cps upgrade geeft een vaste hoeveelheid extra cps. Deze hoeveelheid staat vastgesteld in de json file. De cps word aan de spelers geld toegevoegd door middel van een setinterval van 1 seconde.

```
function clicksPerSecond() {
  money += cps;
  updatescore()
}
setInterval(clicksPerSecond, 1000)
```

Wat heb ik geleerd:

- Hoe je data update in json
- Werken met json data
- Werken met eventlisteners
- Hoe je effectief de forEach method kan gebruiken
- Beter gebruik maken van addEventListners

Clicks:

```
function clickCalculation(mpc){
  var random = Math.random()* 100;
  random = Math.floor(random)
  if (random < upgradefile[1].chance) {
    mpc = mpc *2
  }
  if(random < upgradefile[2].chance){
    mpc = mpc * 3
  }
  Math.floor(mpc)
  return mpc
}
```

De waarde per click word berekend door middel van de functie clickCalculation. In deze functie word gekeken of er gebruik word gemaakt van de ‘chance upgrades’. Deze upgrades hebben de kans om de waarde per click te verhogen als de kans groter is als het random gekozen nummer.

```
document.querySelector(".click-button").addEventListener("click", () =>{

  var clickworth = clickCalculation(mpc)
  money += clickworth;
  updatescore();

})
```

Nadat dit gedaan is word de clickworth het geld toegevoegd en de score geüpdatet.

Wat heb ik geleerd:

- Waardeberekeningen uitvoeren
- Gebruik maken van math.Random en Math.floor
- Gebruik maken van de return functie

Random shop:

De random shop werkt op een interval timer van 2 minuten.

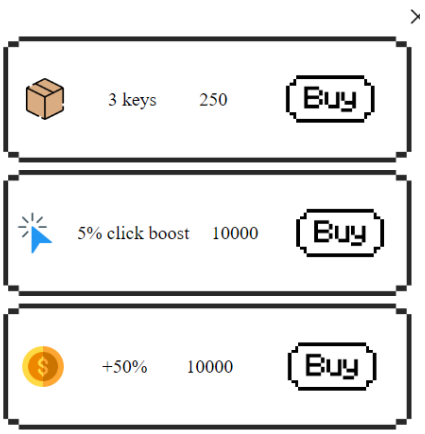
```
let delay= 120000
setInterval(shopRotatie, delay)

function shopRotatie(){
  const itemShop = document.querySelector(".sidenav-right")
    itemShop.innerHTML = ""

let shoparray = [];
shoparray = shopitems

let items = []
items = itemGenerator(items)
```

Om de 2 minuten word de shop leeggemaakt en worden er 3 nieuwe items toegevoegd. Dit word gedaan door de functie itemGenerator.

```
function itemGenerator(items){
    for(var i = 0; i <=2; i++ ){
    items.push((Math.floor(Math.random() * shoparray.length)))
    }
    return items
}
```

Via de for loop worden er 3 items in de shoparray toegevoegd.

```
items.forEach(element => {
const shopitem = itemShop.appendChild(document.createElement('div'));
shopitem.classList.add('shopitem')
const kader = shopitem.appendChild(document.createElement('img'));
kader.classList.add('kader');
kader.src = "./images/shopkader.png"
const icon = shopitem.appendChild(document.createElement('img'));
icon.classList.add("icon-size");
icon.src = shopitems[element].img
const name = shopitem.appendChild(document.createElement('p'));
name.classList.add('upgrade-name')
name.innerText = shopitems[element].itemName
const cost = shopitem.appendChild(document.createElement('p'));
cost.classList.add('cost');
cost.innerText = shopitems[element].cost
cost.id = "i" +[element]
const buy = shopitem.appendChild(document.createElement('img'));
buy.classList.add('buyRandom');
buy.src = "./images/buy.png"
buy.id = shopitems[element].id
```



Bij de random shop heb ik gekozen om de items to te voegen door middel van een forEach method. Ook heb ik bij de random shop ervoor gezorg dat na het kopen van 1 item, dat item uit de shop verdwijnt. Dit heb ik ook geleerd uit de to-do list opdracht.

```
buy.addEventListener("click", ()=> {
  itemShop.removeChild(shopitem)
})
```

In de random shop zitten specialere upgrades als in de normale shop zoals: keys, een 15 minuten warp, een 5% permanente clickboost en een 50% increase in geld.

```
function warp(){
  money += cps * 900000
  updatescore()
}
```

De warp geeft 15 minuten aan cps in een keer.

```
function plus(){
  money = money += money *0.5
  updatescore()
}
```

De 50% increase in geld verhoogd de hoeveelheid money die je hebt met 50%

```
function boost(target){
  mpc += mpc * shopitems[target].boostAmount
}
```

De boost verhoogt het aantal geld per click met 5 %.

```
function key(target){
  keyAmount += shopitems[target].uses
  keycounterText()

}
```

En het kopen van keys vehoogt het aantal keys dat je hebt met het aantal gekochte keys.

Wat heb ik geleerd:

- Werken met arrays
- De .push method gebruiken om dingen toe te voegen aan een array
- Werken met .removeChild

Lootbox systeem:

Met de keys die je uit de random shop kan kopen kun je lootboxen openen. Deze lootboxen kosten 1 key per box.

Bij het openen van een lootbox heb je de kans om tot maximaal 1/4de van je hoeveelheid money te winnen.

```
document.querySelector(".tiny-box").addEventListener('click', () => {
  if(keyAmount > 0){
      keyAmount -= 1
      keycounterText()
     let win=  Math.floor(Math.random() * money/4)
     document.querySelector(".recent-win").innerText = "Recent Win: " + win
  money += win
  }
```

Om feedback terug te geven aan de speler heb ik het zo gemaakt er na het openen van de lootbox te zien is hoeveel geld ze er uit hebben gekregen.

Wat heb ik geleerd:

Als ik verder zou gaan met dit project wat zou ik dan graag nog willen toevoegen?

Ik zou graag meer upgrades toevoegen en meer variatie in de upgrades. Een level up systeem toevoegen aan het klikken van de button om meer reden voor de speler te geven om te blijven klikken. En ik zou het lootbox systeem graag willen verbeteren om een hogere risk en reward factor toe te voegen. Bijvoorbeeld door hogere rarities van lootboxes toe te voegen die moeilijker betaalbaar zijn.