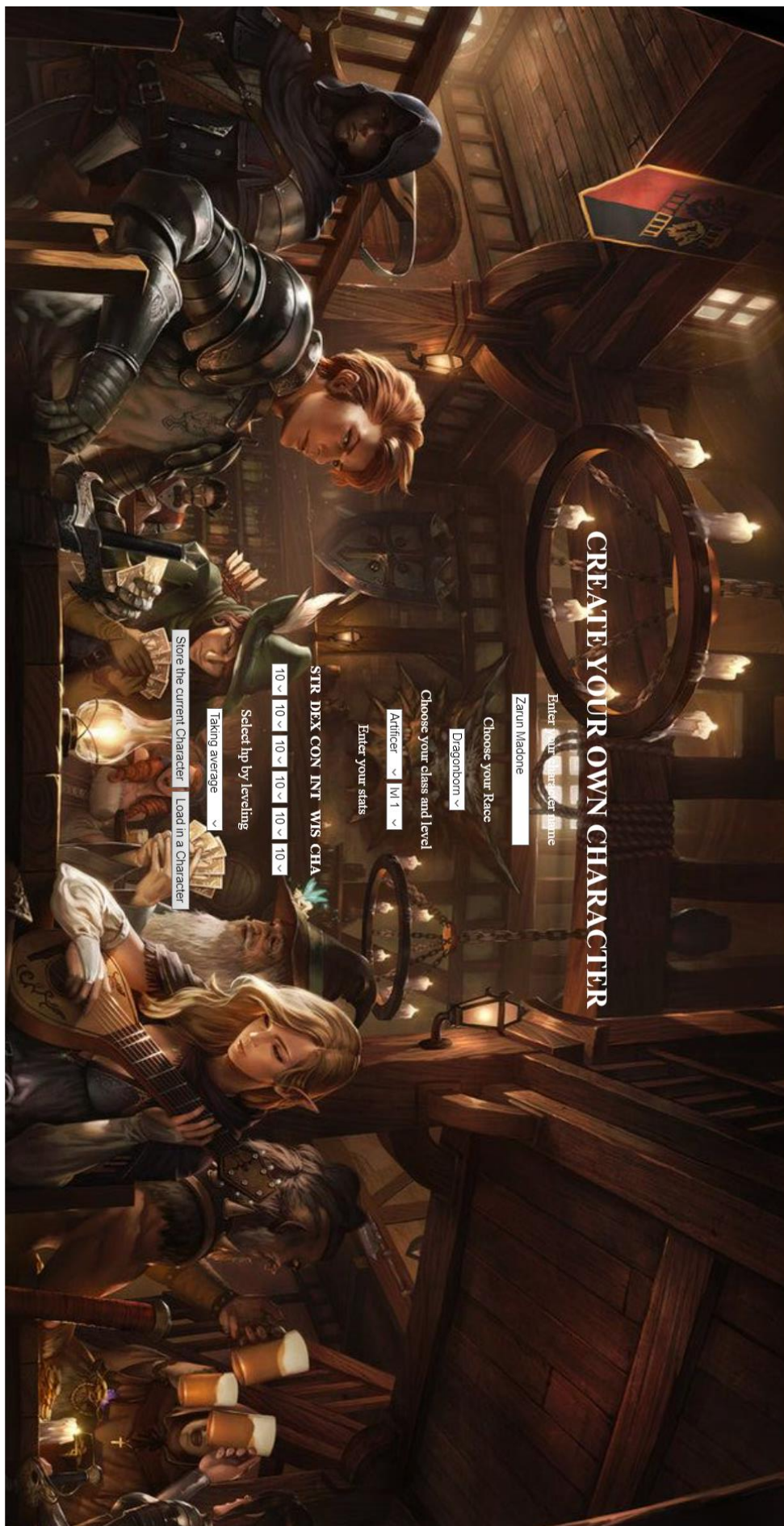


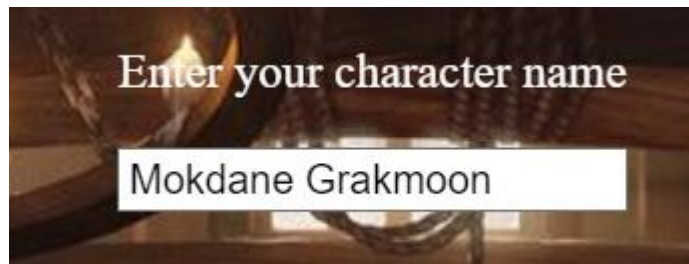
# Webtechnológiák 1 féléves beadandó

Bodgál Attila Zoltán GVFPFT



A program egy „Dungeons and Dragons” nevezetű szerepjáték karakter kreáló részét segíti egyszerűen megvalósítani internetes felületen.

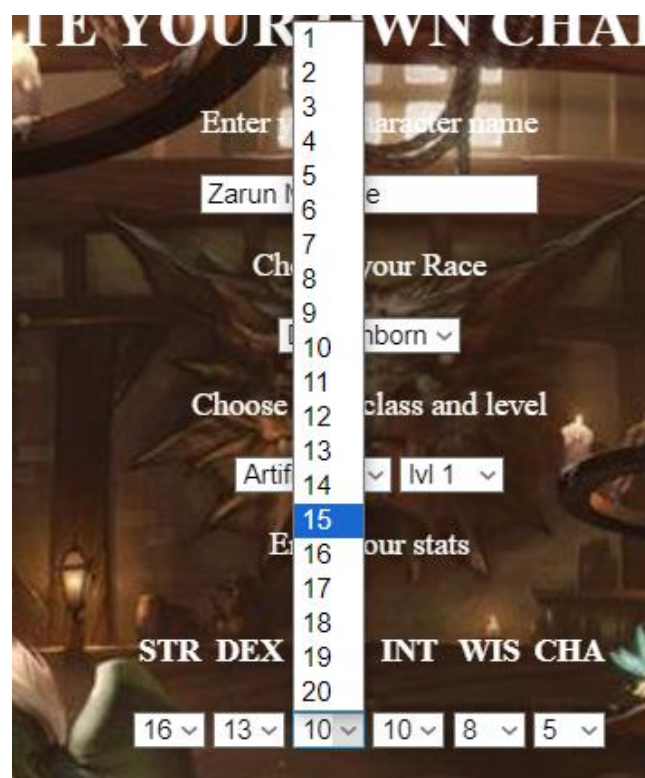
Az első dolog ami lehetséges az egy névadás a karakternek, ezt egy szövegdobozba írhatja be amit kiolvasunk mikor szükségünk van rá.



Ezek után a fajt lehet kiválasztani valamint az osztájt amivel akarunk játszani. Ezek után egy szintet választhatunk ki amin kreáljuk a karaktert. Ez később kell nekünk hogy megtudjuk állapítani hogy mennyi életereje legyen a karakternek.

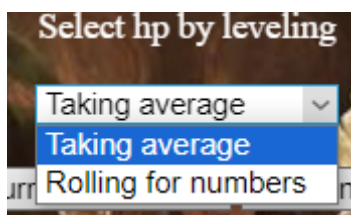


Mindezek után a karakter „statjainak” az értékeit állíthatjuk be. Ezek mindegyike 10-re van alapból beállítva.

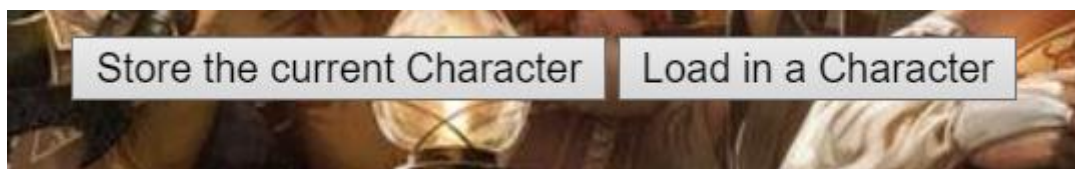




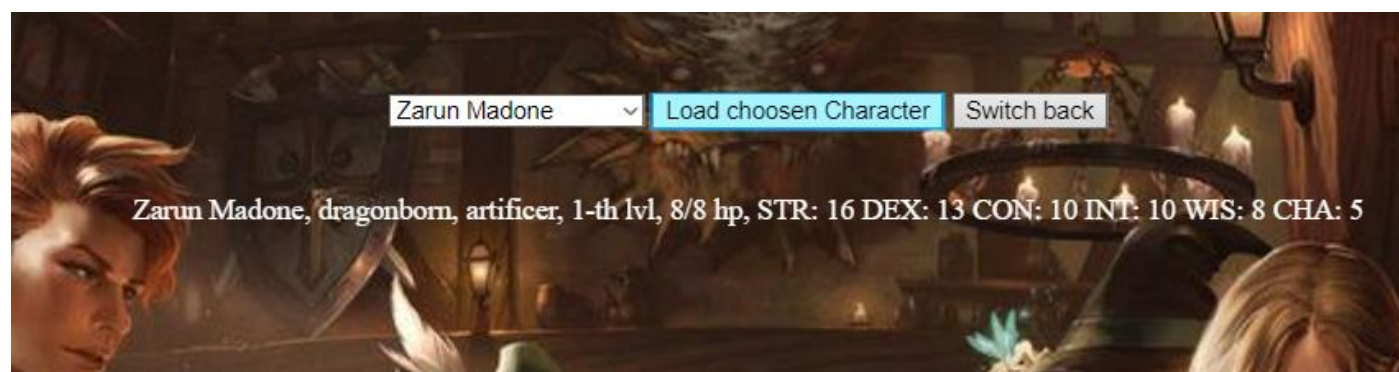
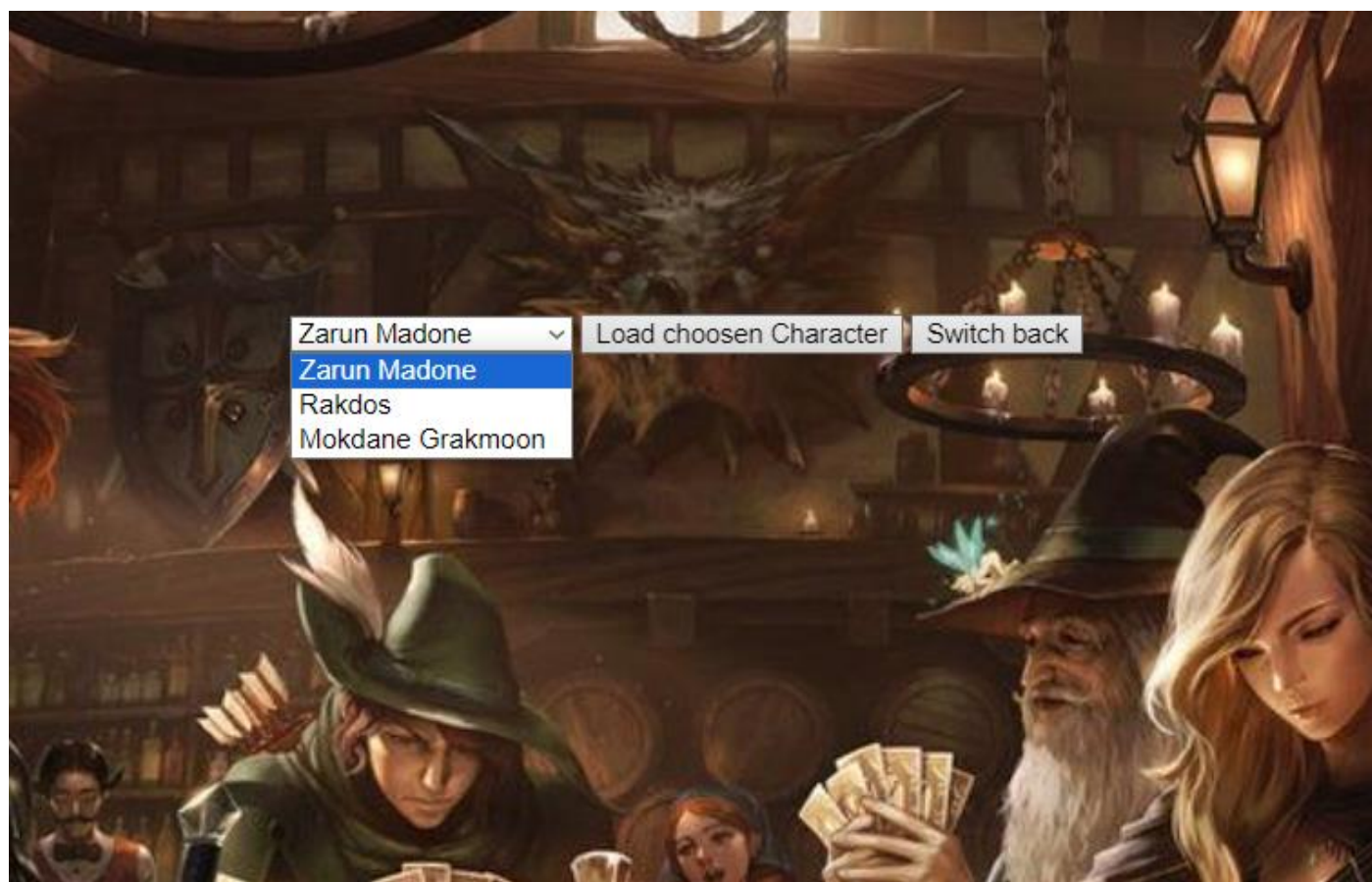
Legutoljára pedig azt választhatjuk ki hogy hogyan szeretnénk a karakter szintenkénti életerejét kiszámolni. Alapjáraton átlagot számítunk ami a megfelelő oldalú dobókockával számol, pl egy 8 oldalú kockának az átlag értéke 5 lesz, ez azt elenti hogy minden szinten az első kivételével 5 életerő pontot kap lapjáraton. A 2. lehetőség hogy minden szinten dobunk a kockával és azt az értéket adjuk hozzá a maximum elérhető életerőhöz.



Mindezek után a karaktert elmenthetjük hogy később megnézhessük mik is lettek beállítva a karakter(ek)nek.



Valamint átmehetünk megnézni ezeket a karaktereket.



A html és js kódokban a lap formázása elég egyszerűen van megoldva, néhány igazítás és háttér beállításon kívül más nem nagyon történik

```
<Style>
  h1{
    color: white;
    text-align: center;
  }

  h3{
    text-align: right;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  }

  table,th,tr{
    text-align: center;
    margin-left: auto;
    margin-right: auto;
  }

  p {
    color: white;
  }

  .centerAlignement {
    position: fixed;
    top: 50%;
    left: 50%;
    -webkit-transform: translate(-50%,-50%);
    transform: translate(-50%,-50%);
    text-align: center;
  }

  body {
    background-image: url("DnD_Tavern.jpg");
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-size: cover;
  }
</Style>
```

A két lap között egy egyszerű megjelenítés átállítás történik az egyik gomb megjelölésakor.

```
<button type="button" id="LoadIn" onclick="SwitchSelectToCreateOrViseversa()">Load in a Character</button>

function SwitchSelectToCreateOrViseversa(){ //switches between 2 dirs by changing their display
  if (characterSelector) {
    updateChSelector();
    document.getElementById("CharacterCreator").style.display = "none";
    document.getElementById("CharacterSelector").style.display = "block";
    characterSelector = false;
  }
  else{
    document.getElementById("CharacterSelector").style.display = "none";
    document.getElementById("CharacterCreator").style.display = "block";
    characterSelector = true;
  }
}
```

Az updateSelector() függvény frissíti hogy milyen karakterek vannak eltárolva éppen hogy azokat ki tudjuk választani és megakarjuk jeleníteni valamelyiket.

```
function updateChSelector(){ //updates the character selector element to the currently stored characters
    var selector = document.getElementById("CharacterSelection");
    while (selector.length > 0) {
        selector.remove(selector.length-1);
    }
    for (let index = 0; index < characters.length; index++) {
        var newOption = document.createElement("option");
        newOption.value = index;
        newOption.text = characters[index].getName();
        selector.add(newOption);
    }
}
```

A karakter egy js osztályban van definiálva aminek az egyik metódusa annak kiírása szöveges formátumba ami olvasható az emberi szemnek.

```
getDescription() { //a general description of the character in text format
    return this.getName() + ", " + this.getRace() + ", " + this.getClass() +
        ", " + this.getLvl() + "-th lvl, " + this.getCurrentHp() + "/" + this.getMaxHp() +
        " hp, STR: " + this.GetStat(0) + " DEX: " + this.GetStat(1) + " CON: " +
        this.GetStat(2) + " INT: " + this.GetStat(3) + " WIS: " + this.GetStat(4) + " CHA: " + this.GetStat(5);
}
```

Az eltárolás egy szimpla tömbben történik az alábbi módon: mikor rákattintunk a tárolás gombra létrehozunk egy új karaktert a felhasználó által megadott és kiválasztott értékek alapján.

```
function addCharacter(){ //add a character to the stored character list
    characters.push(new Character(GetValueOf("classes"),GetValueOf("Icname"),GetValueOf("races"),
    GetValueOf("level"),readInStats(),GetValueOf("average")))
}
```

A getValue() függvény csak egy egyszerűsített érték kiolvasó függvény ami a html elem ID-ja alapján megkeresi, kiolvassa és visszaadja annak értékét.

```
function GetValueOf(id) { //shortcut for simpler value reading
    return document.getElementById(id).value;
}
```

Van még 1-2 függvény ami lehetővé teszi a gyors kiolvasást például a „statok” kiolvasására való függvény ami rögtön egy használható tömbként adja vissza azok értékeit

```
function readInStats(){ //reads the stats in
    let stats = [];
    stats[0] = GetValueOf("Strenght");
    stats[1] = GetValueOf("Dexterity");
    stats[2] = GetValueOf("Constitution");
    stats[3] = GetValueOf("Intelligence");
    stats[4] = GetValueOf("Wisdom");
    stats[5] = GetValueOf("Charisma");
    return stats;
}
```

Vagy a GetHitDie() függvény, ami a kiválasztott osztály alapján visszaadja a megfelelő kockát ami annak az osztálynak az életerejének számolásához használunk.

```
function GetHitDie(cl){ //t
    switch (cl) {
        case "wizard":
        case "sorcerer":
            return 6;
        case "artificer":
        case "bard":
        case "cleric":
        case "druid":
        case "monk":
        case "rogue":
        case "warlock":
            return 8;
        case "fighter":
        case "paladin":
        case "ranger" :
            return 10;
        case "barbarian":
            return 12;
        default:
            break;
    }
}
```