

Projecte UF1: Joc de cartes de color

Desenvolupament del projecte

Elecció d'entorn de desenvolupament

Per a desenvolupar el projecte, he decidit emprar Jigsaw, un framework derivat de Laravel que aporta totes les eines necessàries per al desenvolupament de pàgines estàtiques. Permet desenvolupar en ES6 i transpilar-lo fàcilment a ES5, per a la seva execució a la majoria de navegadors.

He fet aquesta elecció per a poder realitzar el desenvolupament amb ES6 i ES2016+ sense haver-me de preocupar de si l'entorn d'execució suporta els elements del llenguatge més modern que he emprat.

Com a IDE, he usat Atom, el editor de text desenvolupat per Github. És de codi obert i és molt similar a Sublime.

La resta de tecnologies que he fet servir són HTML5 i CSS3.

Diseny del programa

He dividit el codi en múltiples funcions. Evito repetir codi, i per tant he intentat crear totes les funcions possibles.

El utilitzar codi ES6 m'ha facilitat molt el desenvolupament del programa, ja que m'ha estalviat d'escriure moltes línies de codi, sobretot amb l'ús del spread operator.

He decidit crear les cartes com a objectes per tal de tindre propietats nombrades, i les he emmagatzemat en un array. Així és fàcil afegir cartes al joc, i és una de les millores que he realitzat.

Altres millores que he realitzat és la possibilitat d'augmentar o reduir el nombre de cartes màximes amb les que es pot jugar només variant la constant maxCards, ja que els elements de les cartes es generen dinàmicament.

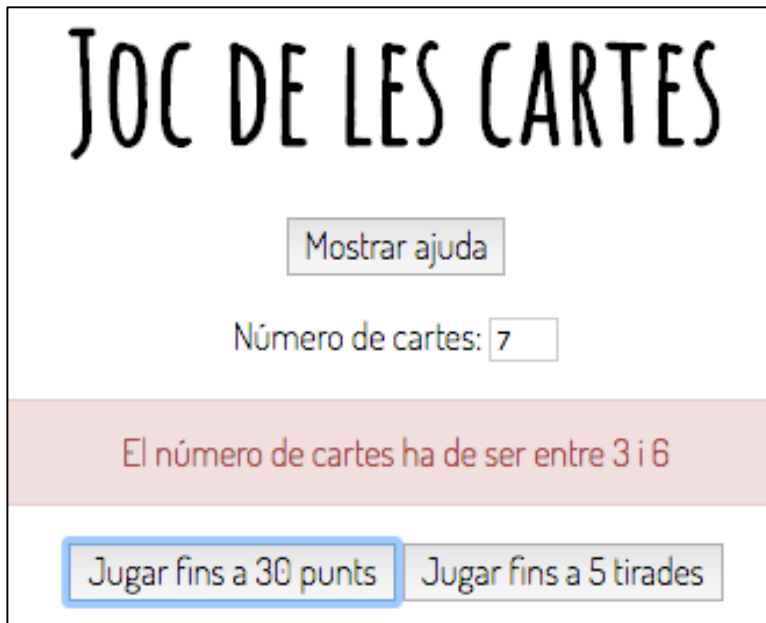
També he inclòs una petita ajuda en el joc, que es pot veure al prémer el botó "Mostrar ajuda".

Escollir cartes

Al començament del joc es demana a l'usuari el número de cartes que vol tirar per tirada. Ho he realitzat amb un input numèric amb valors màxims i mínims, tal i com es veu a la il·lustració 1. També realitza la validació en el codi, comprovant que el nombre introduït sigui entre 3 i 5, i mostrant una alerta a l'usuari en cas de que no es compleixi aquesta condició, tal i com es veu a la il·lustració 2.

Número de cartes:

Il·lustració 1



JOC DE LES CARTES

Mostrar ajuda

Número de cartas:

El número de cartas ha de ser entre 3 i 6

Jugar fins a 30 punts Jugar fins a 5 tirades

Il·lustració 2

Codi de validació

```
let numCards = $(selectorNumCards).value;
if(numCards >= minCards && numCards <= maxCards) {
  hideNumCardError();
  hideOtherButtons(e);
  throwCards(numCards);
} else {
  showNumCardError();
}
```

Disseny cartes

Les cartes son divs que es creen amb un estil comú en començar la partida. Es creen tantes cartes com s'hagin triat en les opcions de la partida. A l'estil comú per a totes les cartes hi ha propietats com l'ample, l'alçada, la vora, i les fa invisibles. Al donar-li un altre classe a més de la comuna de carta, per exemple "red" per a la carta vermella, es fa visible, a més d'estilar-se com una carta vermella.

Codi HTML generat

```
<div class="playCards">
  <div class="card"></div>
  <div class="card"></div>
  <div class="card"></div>
  <div class="card"></div>
  <div class="card"></div>
  <div class="card"></div>
</div>
```

Codi CSS

```
.card {
  display: inline-block;
  margin-left: 1rem;
  margin-top: 1rem;
  width: 10rem;
  height: 15rem;
  visibility: hidden;
  border-radius: 1rem;
  border-color: #000000;
  border-width: 0.1rem;
  border-style: solid;
}

.card.red {
  background: linear-gradient(135deg, #FF0000 22px, #ffffff 22px, #ffffff 24px, transparent 24px,
transparent 67px, #ffffff 67px, #ffffff 69px, transparent 69px), linear-gradient(225deg, #FF0000 22px,
#ffffff 22px, #ffffff 24px, transparent 24px, transparent 67px, #ffffff 67px, #ffffff 69px, transparent
69px)0 64px;
  background-color: #FF0000;
  background-size: 64px 128px;
  visibility: visible;
}
```

Mètode o tipus de joc

El joc es pot jugar en **dos modes**:

- **Màxim de punts:** El jugador realitza tirades fins a arribar als 30 punts.
- **Màxim de tirades:** El jugador realitza un màxim de 5 tirades.

Aquestes condicions de joc estan definides en les constants globals `defaultPoints` i `defaultThrows`. També es defineixen les constants `defaultCards`, `minCards` i `maxCards`, que defineixen el nombre de cartes a emprar per defecte, el mínim de cartes necessàries per a jugar, i el màxim.

```
const defaultPoints = 30;
const defaultThrows = 5;
const defaultCards = 3;
const minCards = 3;
const maxCards = 5;
```

Al prémer un dels botons de mode de joc, s'inicia el joc en aquell mode, a més de realitzar-se la primera tirada. El text del botó premut canvia per "Tornar a tirar". A tots els elements de configuració del joc, excepte al botó premut, i que estan marcats amb la classe "hideable", se'ls hi assigna la classe "invisible", que els amaga.

Codi JS

```
function hideOtherButtons(e) {
  $(selectorHideable, true).forEach(el => {
    if (e.currentTarget !== el) el.classList.add(classHidden);
    else el.innerHTML = 'Tornar a tirar';
  });
}
```

Colors i puntuació

Les cartes, com ja he mencionat anteriorment, s'emmagatzemen en un Array en forma d'objectes, per tal de tindre les seves propietats nombrades. Això permet que les cartes es puguin crear, modificar i esborrar fàcilment.

Codi JS

```
const cards = [{  
  style: "yellow",  
  points: 1  
},  
{  
  style: "blue",  
  points: 2  
},  
{  
  style: "green",  
  points: 3  
},  
{  
  style: "red",  
  points: -2  
},  
{  
  style: "purple",  
  points: 5  
}];
```

Com veiem, cada carta té dos propietats:

- **Style:** Estil que s'aplicarà quan s'hagi de mostrar aquest tipus de carta.
- **Points:** Puntuació d'aquest tipus de carta.

En aquest exemple es veu un tipus de carta nou, "purple", que suposarà sumar 5 punts cada vegada que aparegui.

Per a afegir una carta al joc, només hem de crear-la dins de l'array i crear l'estil al CSS.

Exemple de creació d'una carta

En aquest exemple, afegirem una carta negra que valdrà 1000 punts. Només cal realitzar unes petites modificacions al codi del joc per a que funcioni.

Afegir a l'array cards el següent objecte:

Codi JS

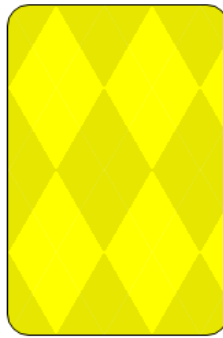
```
{  
  style: "black",  
  points: 1000  
}
```

Afegir a la fulla d'estils la següent classe:

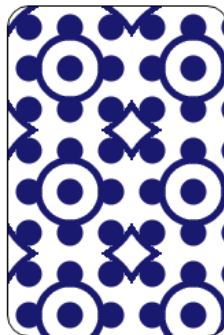
Codi CSS

```
.card.black {  
  background-color: #000000;  
  visibility: visible;  
}
```

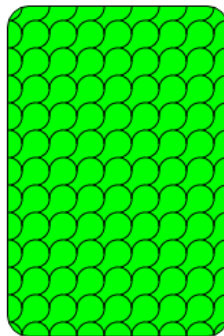
Forma de les cartes



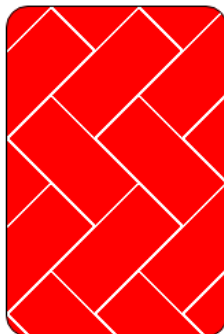
Aquesta carta val 1 punt



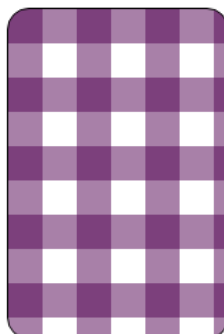
Aquesta carta val 2 punts



Aquesta carta val 3 punts



Aquesta carta val -2 punts



Aquesta carta val 5 punts

Generar cartes

La funció `throwCards` genera un número aleatori **entre 0 i 999** per cada carta que s'ha de generar. Després, per tal de reduir-lo a un valor usable, mitjançant l'operador `mod` obtenim el residu de la divisió del número aleatori entre la quantitat de possibles cartes que hi ha, obtenint aquest valor de la longitud de l'array `cards`. Després, fem aquest valor usable com a índex de l'array.

Codi JS

```
function throwCards(num) {  
  var thrownCards = [];  
  var randomNumbers = [];  
  $(selectorCard, true).forEach((el, i) => {  
    let random = Math.trunc(Math.random()*1000);  
    let card = convertNumberToCard(card);  
    setColor(el, card);  
    addPoints(card);  
    randomNumbers.push(random);  
    thrownCards.push(card);  
  });  
  currthrows++;  
  printThrowInfo(thrownCards, randomNumbers);  
  checkEndGame(thrownCards);  
}
```

En el codi que realitza la tirada no gestionem directament les cartes, sinó que ens limitem a cridar les funcions necessàries per a configurar les cartes, afegir els punts, imprimir els resultats i comprovar si s'ha d'acabar el joc.

`setColor(el, card);`

S'encarrega d'obtenir l'estil de l'objecte `card` i assignar-lo a l'element `el`.

`addPoints(card);`

Afegeix els punts que corresponen a la carta.

`printThrowInfo(thrownCards, randomNumbers);`

Imprimeix informació de la tirada.

`checkEndGame(thrownCards);`

Comprova si s'ha arribat al final de la partida.

Impressió de informació de la tirada

S'imprimeix informació dels números aleatoris generats, els punts generats, el número de tirades realitzades, els punts que sumen aquesta tirada, i els punts acumulats en la partida actual.

Utilitza la funció `map` dels Arrays per a generar un nou array només amb els valors de les puntuacions. Després, utilitza la funció `reduce` dels Arrays per a reduir tot l'array a un sol valor, en aquest cas sumant tots els valors.

Codi JS

```
function printThrowInfo(throwPoints, randomNumbers) {  
  let el = $(selectorOutput);  
  let throwPoints = thrownCards.map(thrCard => thrCard.points);  
  el.innerHTML = `

Els aleatoris generats en aquesta jugada són: ${randomNumbers.join(', ')}</p>  
    <p>Els punts generats en aquesta jugada són: ${throwPoints.join(', ')}</p>  
    <p>Número de tirades: ${currthrows}</p>  
    <p class="textPoints">PUNTUACIÓ</p>  
    <p>Punts tirada: ${throwPoints.reduce((a,b) => a+b)}</p>  
    <p>Punts totals: ${currpoints}</p>`;  
}


```

Exemple de sortida

Els aleatoris generats en aquesta jugada són: 122, 126, 664, 994, 432, 895

Els punts generats en aquesta jugada són: 3, 2, 5, 5, 3, 1

Número de tirades: 3

PUNTUACIÓ

Punts tirada: 19

Punts totals: 33

PARTIDA FINALITZADA AMB 33 PUNTS I 3 TIRADES

Fi del joc

El joc acaba en quan es compleix la condició del mode de joc, **o quan totes les cartes són iguals**. Aquesta última condició la he afegit per a afegir una altra forma de finalitzar la partida.

La funció que comprova si s'ha acabat el joc comprova quin botó és visible per a determinar el mode de joc. Després, determina si es compleixen les condicions d'aquest botó. Per últim, comprova si els objectes de totes les cartes son iguals i, si ho son, acaba la partida també.

Donat que no s'instancia, aquests objectes seran iguals. Si fossin instàncies, s'hauria de comparar-los d'un altre forma, per exemple comparant-ne la propietat d'estil.

Codi JS

```
function checkEndGame(thrownCards) {  
  if (!$(selectorButton + selectorPlayPoints).classList.contains(classHidden) && currpoints >=  
    defaultPoints) endGame();  
  else if (!$(selectorButton + selectorPlayThrows).classList.contains(classHidden) && currthrows >=  
    defaultThrows) endGame();  
  else if (thrownCards.every((val, i, arr) => val == arr[0])) endGame();  
}
```

La funció comprova quin dels dos botons està visible amb l'invers de si conté la classe que defineix que està amagat. En cas d'estar visible, comprova si es compleix la condició per a finalitzar la partida. Si és visible i compleix la condició, executem la funció per a finalitzar la partida.

Un últim cas comprova si totes les cartes tirades son iguals. Ho comprova comparant tots els valors d'un array on hem emmagatzemat l'objecte de cada carta. En cas afirmatiu, cridem a la funció endGame.

```
function endGame() {  
  let el = $(selectorOutput);  
  el.innerHTML = el.innerHTML + `<h3>PARTIDA FINALITZADA AMB ${currpoints} PUNTS I ${currthrows}  
  TIRADES</h3>`;   
  restartGame();  
}
```

La funció endGame() imprimeix en l'element de sortida un text per indicar que s'ha acabat la partida. Per últim, reinicia la partida.

Joc de proves

Partida a 5 tirades amb 5 cartes

S'introdueixen les dades: 5 cartes amb màxim 5 tirades.

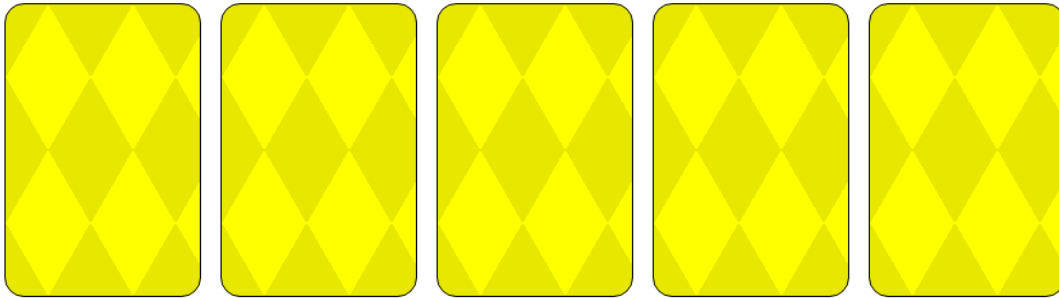
Joc de les cartes

Mostrar ajuda

Número de cartas: 5

Jugar fins a 30 punts

Jugar fins a 5 tirades



Els aleatoris generats en aquesta jugada són: 800, 610, 870, 600, 265

Els punts generats en aquesta jugada són: 1, 1, 1, 1, 1

Número de tirades: 4

PUNTUACIÓ

Punts tirada: 5

Punts totals: 40

PARTIDA FINALITZADA AMB 40 PUNTS I 4 TIRADES

La partida ha acabat prematurament al sortir totes les cartes iguals.

Partida a 30 punts amb 4 cartes

S'introdueixen les dades: 5 cartes amb màxim 30 punts.

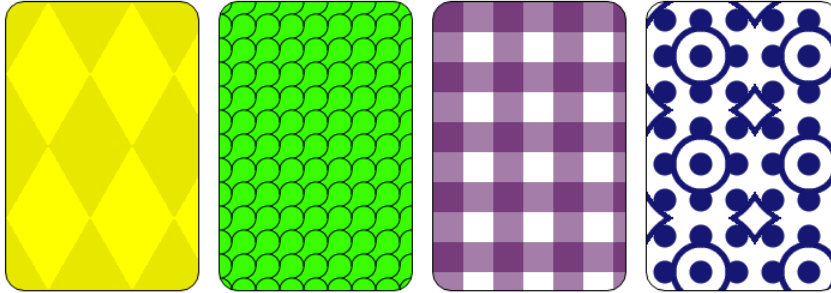
Joc de les cartes

Mostrar ajuda

Número de cartas: 4

Jugar fins a 30 punts

Jugar fins a 5 tirades



Els aleatoris generats en aquesta jugada són: 160, 372, 174, 856

Els punts generats en aquesta jugada són: 1, 3, 5, 2

Número de tirades: 3

PUNTUACIÓ

Punts tirada: 11

Punts totals: 34

PARTIDA FINALITZADA AMB 34 PUNTS I 3 TIRADES

La partida ha funcionat bé, l'únic que falla és que al no haver connexió a internet no ha carregat la font.

Partida a 10 tirades amb 6 cartes

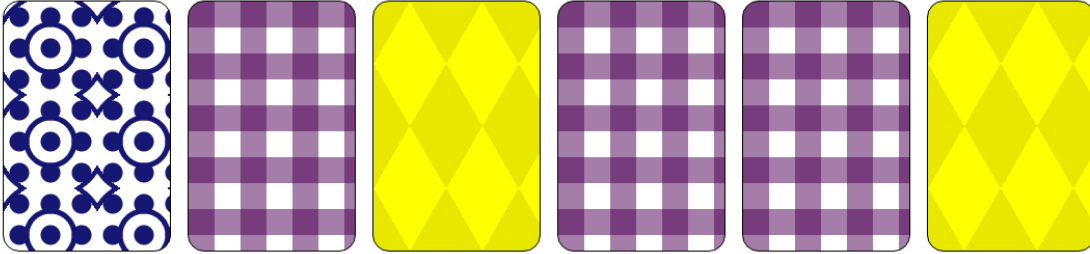
Per a fer aquest exemple, es varia el codi font. Es modifiquen les constants `defaultThrows` a valor 10, i `maxCards` a 6. No es modifica res més. Aquest cas valora la facilitat de modificar el funcionament del programa.

Joc de les cartes

Mostrar ajuda

Número de cartas: 6

Jugar fins a 30 punts Jugar fins a 10 tirades



Els aleatoris generats en aquesta jugada són: 336, 219, 590, 384, 924, 210

Els punts generats en aquesta jugada són: 2, 5, 1, 5, 5, 1

Número de tirades: 10

PUNTUACIÓ

Punts tirada: 19

Punts totals: 130

PARTIDA FINALITZADA AMB 130 PUNTS I 10 TIRADES

Veiem que el botó ha canviat correctament, s'han realitzat 10 tirades i han aparegut 6 cartes en comptes de 5. Veiem que s'ha imprès tot bé.

Partida a 5 tirades amb 6 cartes

Retornem els valors canviats a la prova anterior als per defecte que teníem al començar l'exercici. Provem a introduir valors no vàlids al número de cartes.

JOC DE LES CARTES

Mostrar ajuda

Número de cartes:

El número de cartes ha de ser entre 3 i 5

Jugar fins a 30 punts

Jugar fins a 5 tirades

Veiem que dona error la validació i no ens deixa jugar.

Innovació extra

A més de les innovacions citades al llarg d'aquesta documentació, n'he realitzat un altre que m'ha permès prescindir d'incloure una llibreria tant pesada com és jQuery.

Per a evitar l'ús de jQuery però conservar la facilitat d'escriure `$('selectorCSS')` que ens dona aquesta llibreria, he redactat una funció que, amb 7 línies, ens permet tant seleccionar elements del DOM d'aquesta manera, com afegir funcions a executar al finalitzar la càrrega del DOM amb la sintaxi `$(function())` que també ens proporciona jQuery. A més, ens permet forçar la selecció de múltiples elements passant un segon argument després del selector CSS, que haurà de ser true per a forçar aquesta execució.

Està clar que amb 7 línies no ens aporta tota la funcionalitat, però si que ens permet escriure codi d'una forma molt més còmoda. A més, JS ha anat incorporant funcionalitats que només es trobaven a jQuery, com `.toggleClass()` que ens permetia afegir o eliminar una classe en un element, que ara podem fer amb JS pur amb `.classList.toggle()`

Codi JS

```
function $(sel, multi = false) {  
  if (typeof sel == 'function') return window.addEventListener('load', sel);  
  else if (typeof sel == 'string') {  
    if (multi || document.querySelectorAll(sel).length > 1) return document.querySelectorAll(sel);  
    else return document.querySelector(sel);  
  }  
}
```