

Boilerplate

Los ejemplos que vamos a codificar los puedes probar en la sandbox de TypeScript, si no sabes cómo funciona, échale un vistazo al módulo de setup y si te quedan dudas contacta con tu mentor.

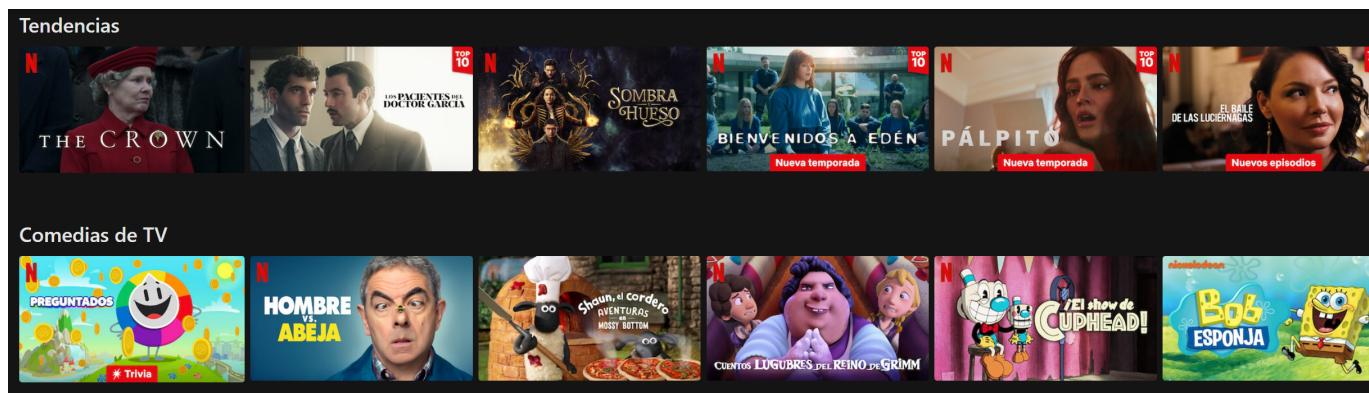
Intro

Te puede parecer que llevas muy poco aprendido en lo que llevas de Bootcamp, pero te vas a sorprender que con las bases que llevamos ya puedes hacer cosas interesantes.

En este caso vamos a simular la portada de un portal tipo *NextFlix*.

Enunciado

Cuando nos logamos en nuestro proveedor de streaming de pelis favoritos nos aparece una pantalla tal que así:



Lo normal para un usuario es que haga una sola petición a servidor y le devuelva un JSON (un fichero con datos en JavaScript) con todas las películas del usuario con diferentes campos (genero, favorita, premiada con Oscar, más vistas...).

```
[  
  {  
    titulo: "Babe el cerdito valiente",  
    resumen: "Un cerdito que quiere ser perro pastor",  
    genero: "Familiar",  
    masVisto: true,  
    calificacionImdb: 6.7,  
    premioGalardon: true,  
    fechaEstreno: new Date("1995-08-04"),  
    imagen:  
      "https://raw.githubusercontent.com/Lemoncode/fotos-ejemplos/main/mflix/babe.png",  
  },  
  {  
    titulo: "Intocable",  
  },
```

```
resumen: "Un hombre que se queda tetraplejico y contrata a un cuidador",
genero: "Familiar",
masVisto: true,
calificacionImdb: 8.5,
premioGalardon: true,
fechaEstreno: new Date("2011-11-02"),
imagen:
  "https://raw.githubusercontent.com/Lemoncode/fotos-ejemplos/main/mflix/intocable.png",
},
{
  titulo: "The Karate Kid",
  resumen: "Un niño que aprende karate",
  genero: "Familiar",
  masVisto: true,
  calificacionImdb: 7.2,
  premioGalardon: false,
  fechaEstreno: new Date("1984-06-22"),
  imagen:
    "https://raw.githubusercontent.com/Lemoncode/fotos-ejemplos/main/mflix/karate-kid.png",
},
{
  titulo: "Poli de Guardería",
  resumen: "Un policia que se hace pasar por profesor de guarderia",
  genero: "Familiar",
  masVisto: false,
  calificacionImdb: 5.6,
  premioGalardon: false,
  fechaEstreno: new Date("1990-12-21"),
  imagen:
    "https://raw.githubusercontent.com/Lemoncode/fotos-ejemplos/main/mflix/poli-guarderia.png",
},
{
  titulo: "El príncipe de Zamunda",
  resumen: "Un principe africano que se va a Nueva York a buscar esposa",
  genero: "Familiar",
  masVisto: false,
  calificacionImdb: 7,
  premioGalardon: false,
  fechaEstreno: new Date("1988-06-29"),
  imagen:
    "https://raw.githubusercontent.com/Lemoncode/fotos-ejemplos/main/mflix/principe-zamunda.png",
},
{
  titulo: "Angry Birds",
  resumen: "Una pelicula de los pajaros de angry birds",
  genero: "Animacion",
  masVisto: false,
  calificacionImdb: 6.3,
  premioGalardon: false,
  fechaEstreno: new Date("2016-05-11"),
```

```
imagen:  
    "https://raw.githubusercontent.com/Lemoncode/fotos-  
ejemplos/main/mflix/angry-birds.png",  
,  
{  
    titulo: "Los Croods",  
    resumen:  
        "Una familia de cavernicolas que se encuentra con un hombre moderno",  
    genero: "Animacion",  
    masVisto: false,  
    calificacionImdb: 7.2,  
    premioGalardon: false,  
    fechaEstreno: new Date("2013-03-15"),  
    imagen:  
        "https://raw.githubusercontent.com/Lemoncode/fotos-  
ejemplos/main/mflix/croods.png",  
,  
{  
    titulo: "José, el rey de los sueños",  
    resumen: "La historia de José de la biblia",  
    genero: "Animacion",  
    masVisto: false,  
    calificacionImdb: 6.5,  
    premioGalardon: false,  
    fechaEstreno: new Date("2000-10-27"),  
    imagen:  
        "https://raw.githubusercontent.com/Lemoncode/fotos-  
ejemplos/main/mflix/jose.png",  
,  
{  
    titulo: "La patrulla canina",  
    resumen: "Una pelicula de la patrulla canina",  
    genero: "Animacion",  
    masVisto: false,  
    calificacionImdb: 4.7,  
    premioGalardon: false,  
    fechaEstreno: new Date("2021-08-20"),  
    imagen:  
        "https://raw.githubusercontent.com/Lemoncode/fotos-  
ejemplos/main/mflix/patrulla-canina.png",  
,  
{  
    titulo: "Los pitufos",  
    resumen: "Una pelicula de los pitufos",  
    genero: "Animacion",  
    masVisto: false,  
    calificacionImdb: 5.4,  
    premioGalardon: false,  
    fechaEstreno: new Date("2011-07-29"),  
    imagen:  
        "https://raw.githubusercontent.com/Lemoncode/fotos-  
ejemplos/main/mflix/pitufos.png",  
,  
{
```

```
    titulo: "El cazador y la reina del hielo",
    resumen: "La historia de la reina del hielo",
    genero: "Aventuras",
    masVisto: false,
    calificacionImdb: 6.1,
    premioGalardon: false,
    fechaEstreno: new Date("2016-04-08"),
    imagen:
      "https://raw.githubusercontent.com/Lemoncode/fotos-
ejemplos/main/mflix/cazador.png",
  },
{
  titulo: "Ghost Rider",
  resumen: "Un hombre que se convierte en un esqueleto en llamas",
  genero: "Aventuras",
  masVisto: false,
  calificacionImdb: 5.2,
  premioGalardon: false,
  fechaEstreno: new Date("2007-02-16"),
  imagen:
    "https://raw.githubusercontent.com/Lemoncode/fotos-
ejemplos/main/mflix/ghost-rider.png",
},
{
  titulo: "La momia",
  resumen: "La historia de la momia",
  genero: "Aventuras",
  masVisto: false,
  calificacionImdb: 5.2,
  premioGalardon: false,
  fechaEstreno: new Date("2008-08-01"),
  imagen:
    "https://raw.githubusercontent.com/Lemoncode/fotos-
ejemplos/main/mflix/momia.png",
},
{
  titulo: "Vampire Academy",
  resumen: "Una academia de vampiros",
  genero: "Aventuras",
  masVisto: false,
  calificacionImdb: 5.5,
  premioGalardon: false,
  fechaEstreno: new Date("2014-02-14"),
  imagen:
    "https://raw.githubusercontent.com/Lemoncode/fotos-
ejemplos/main/mflix/vampire-academy.png",
},
{
  titulo: "Warcraft: El origen",
  resumen: "La historia de Warcraft",
  genero: "Aventuras",
  masVisto: false,
  calificacionImdb: 6.8,
  premioGalardon: false,
```

```
fechaEstreno: new Date("2016-06-03"),
imagen:
  "https://raw.githubusercontent.com/Lemoncode/fotos-
ejemplos/main/mflix/warcraft.png",
},
];
```

En las categorías queremos sacar lo siguiente:

- Una pantalla en la que vamos a tener categoría y listado de películas.
- Cada listado debe de venir filtrado por categoría.

Va a ver listados:

- Genero familiar
- Genero de animación.
- Genero de aventuras.
- Películas más vistas.
- Películas premiadas con Oscar.
- Ordenar películas de mayor a menor valoración.

Queremos poder hacer scroll en horizontal en cada categoría (como en NetFlix).

Queremos poder ver el listado de categorías y películas

El contrato de los datos que recibimos:

```
export interface Pelicula {
  titulo: string;
  resumen: string;
  genero: string;
  masVisto: boolean;
  calificacionImdb: number;
  premioGalardon: boolean;
  fechaEstreno: Date;
  imagen: string;
}
```

Un ejemplo de datos:

```
{
  titulo: "Babe el cerdito valiente",
  resumen: "Un cerdito que quiere ser perro pastor",
  genero: "Familiar",
  masVisto: true,
  calificacionImdb: 6.7,
  premioGalardon: true,
  fechaEstreno: new Date("1995-08-04"),
  imagen:
```

```
    "https://raw.githubusercontent.com/Lemoncode/fotos-
ejemplos/main/mflix/babe.png",
}
```

Eyyy... un inciso aquí, para crear una fecha estamos usando el objeto *Date* ¿Por qué no me explicaste esto con los tipos? Bueno técnicamente hablando *Date* es un objeto que representa una fecha y una hora:

- Cuando creamos el objeto (usamos *new* para ello), le pasamos la fecha que queremos representar, esto de objetos y clases lo veremos en un módulo más adelante.
- El propio *Date* expone una serie de métodos para poder por ejemplo darle formato a una fecha (año mes día, o mes día año, o...)
- Para ciertas operaciones puede que lo que trae *Date* se quede corto y tengamos que usar librerías externas como *dateFns*

Como resolver el problema

Cuando vamos a implementar algo más complejo, tenemos que plantearnos como resolverlo, si lo intentamos atacar de un tirón nos vamos a liar y no vamos a saber por dónde empezar.

Lo primero es aplicar *divide y vencerás*, vamos a dividir el problema en partes más pequeñas y vamos a ir resolviendo cada una de ellas.

¿Qué desafíos tenemos?

1. Aspecto

No sé cómo mostrar una lista de películas en el HTML y que se vea bien:

- Vamos a trabajar directamente con HTML (si sabes de diseño ponte con Figma 😊).
- Luego le vamos a dar estilos al HTML con CSS.
- Y por último vamos a añadir un poco de contenido a nuestro HTML y ver cómo se vería.

Vamos a ello

- Primero creamos un HTML de pruebas, que nos va a servir de referencia para luego construir el resto. En este ejemplo hemos añadido un *header* y un *nav* con un logo y un menú de navegación. Es solo una simulación, no tiene funcionalidad.
- Vamos a crear un contenedor para el listado de películas, que va a ser un contenedor flexbox, con un título y un listado de películas. Para hacernos una idea de cómo quedaría.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>MFlix</title>
```

```
</head>
<body>
  <div class="container">
    <header>
      <nav>
        <div class="container-header">
          
          <ul>
            <li>Inicio</li>
            <li>Series</li>
            <li>Películas</li>
            <li>Más recientes</li>
            <li>Mi lista</li>
          </ul>
        </div>
      </nav>
      <div class="hero">
        <h1>Películas y series ilimitadas.</h1>
        <p>Disfruta en tu TV, computadora, tablet o celular.</p>
        <button>Comenzar</button>
      </div>
    </header>
    <div id="principal" class="principal"></div>
  </div>

  <script type="module" src="/src/main.ts"></script>
</body>
</html>
```

En este HTML hemos añadido un *header* y un *nav* con un logo y un menú de navegación. Y por otra parte, hemos creado un título, un párrafo y un botón para el *hero*. Es solo una simulación, no tiene funcionalidad.

Y el CSS:

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: "Franklin Gothic Medium", "Arial Narrow", Arial, sans-serif;
  background-color: #141414;
}

.container-header {
  max-width: 1200px;
```

```
margin: 0 auto;
padding: 0 20px;
display: flex;
justify-content: space-between;
align-items: center;
}

nav {
background-color: #141414;
color: #ffffff;
height: 80px;
display: flex;
justify-content: space-between;
align-items: center;
}

nav img {
height: 40px;
}

nav ul {
display: flex;
}

nav li {
list-style-type: none;
margin-right: 20px;
}

.hero {
height: 300px;
display: flex;
flex-direction: column;
justify-content: center;
align-items: center;
color: #ffffff;
}

.hero h1 {
font-size: 48px;
margin-bottom: 20px;
}

.hero p {
font-size: 24px;
margin-bottom: 40px;
}

.hero button {
background-color: #e50914;
color: #ffffff;
font-size: 24px;
padding: 10px 20px;
border: none;
```

```
border-radius: 5px;
cursor: pointer;
}

.principal {
  display: flex;
  flex-direction: column;
  gap: 50px;
  color: #ffffff;
  margin-left: 10px;
  margin-bottom: 50px;
}

.lista-peliculas {
  position: relative;
}

.peliculas-contenedor {
  display: flex;
  overflow: hidden;
  scroll-behavior: smooth;
  justify-content: flex-start;
  gap: 10px;
}

.peliculas {
  margin-left: 5px;
  display: flex;
  flex-direction: column;
  gap: 10px;
}

.pelicula img {
  transition: all 0.3s ease;
  cursor: pointer;
}

.pelicula:hover img {
  transform: scale(1.1);
  transition: transform 0.2s ease-in-out;
}

.pelicula h3 {
  font-size: 18px;
  margin-top: 10px;
}

.flecha-izquierda,
.flecha-derecha {
  position: absolute;
  cursor: pointer;
  top: 80px;
  width: 60px;
  z-index: 1;
```

```
}

.scroll .flecha-izquierda,
.scroll .flecha-derecha {
  display: block;
}

.flecha-izquierda {
  left: 0;
}

.flecha-derecha {
  right: 0;
}
```

También hemos añadidos los estilos que usaremos para nuestra aplicación, el color de fondo, el tipo de letra que vamos a usar. Y le hemos añadido estilos al *header*, al *nav*, al *hero*, a las flechas y a la lista de películas.

Ahora, vamos a añadirle un poco de contenido a nuestro HTML, vamos a ver como quedaría el HTML con un par de películas:

```
<!DOCTYPE html>
<html lang="en">
( ... )
<div id="principal" class="principal">
+ <h2>Familiar</h2>
+ <div class="lista-peliculas" id="lista-peliculas">
+   <div class="flecha-izquierda">
+     
+   </div>
+   <div class="flecha-derecha">
+     
+   </div>
+   <div class="peliculas-contenedor">
+     <div class="pelicula" id="pelicula">
+       
+       <h3>Babe el cerdito valiente</h3>
+     </div>
+     <div class="pelicula" id="pelicula">
+       
+       <h3>Intocable</h3>
+     </div>
+   </div>
+ </div>
</div>
( ...)
```

Así se vería en el navegador:



2. Datos + UI

¿Y ahora como paso esa lista al HTML?

Vamos a arrancar implementando la solución para una lista simple, una vez que tengamos esto ya nos preocupamos de dar el siguiente paso.

Podemos pensar en desarrollar de esta manera, en como si estuviéramos escalando una montaña, subimos unos metros, aseguramos la cuerda con la chapa y luego vamos a por la siguiente chapa, de esta manera vamos teniendo puntos seguros, y si algo no sale la caída es menor.

Primero vamos a crear un archivo para poner nuestro modelo, lo llamamos *modelo.ts* y lo ponemos en la carpeta *src*:

src/modelo.ts

```

export interface Pelicula {
  titulo: string;
  resumen: string;
  genero: string;
  masVisto: boolean;
```

```
    calificacionImdb: number;
    premioGalardon: boolean;
    fechaEstreno: Date;
    imagen: string;
}
```

Ahora creamos un archivo para los datos, lo vamos a llamar *datos.ts*. En él vamos a crear una lista de películas, que vamos a exportar para poder usarla en el resto de archivos.

src/datos.ts

```
import { Pelicula } from "./modelo";

export const peliculas: Pelicula[] = [
  {
    titulo: "Babe el cerdito valiente",
    resumen: "Un cerdito que quiere ser perro pastor",
    genero: "Familiar",
    masVisto: true,
    calificacionImdb: 6.7,
    premioGalardon: true,
    fechaEstreno: new Date("1995-08-04"),
    imagen:
      "https://raw.githubusercontent.com/Lemoncode/fotos-ejemplos/main/mflix/babe.png",
  },
  {
    titulo: "Intocable",
    resumen: "Un hombre que se queda tetraplejico y contrata a un cuidador",
    genero: "Familiar",
    masVisto: true,
    calificacionImdb: 8.5,
    premioGalardon: true,
    fechaEstreno: new Date("2011-11-02"),
    imagen:
      "https://raw.githubusercontent.com/Lemoncode/fotos-ejemplos/main/mflix/intocable.png",
  },
  {
    titulo: "The Karate Kid",
    resumen: "Un niño que aprende karate",
    genero: "Familiar",
    masVisto: true,
    calificacionImdb: 7.2,
    premioGalardon: false,
    fechaEstreno: new Date("1984-06-22"),
    imagen:
      "https://raw.githubusercontent.com/Lemoncode/fotos-ejemplos/main/mflix/karate-kid.png",
  },
]
```

```
    titulo: "Poli de Guardería",
    resumen: "Un policia que se hace pasar por profesor de guarderia",
    genero: "Familiar",
    masVisto: false,
    calificacionImdb: 5.6,
    premioGalardon: false,
    fechaEstreno: new Date("1990-12-21"),
    imagen:
      "https://raw.githubusercontent.com/Lemoncode/fotos-ejemplos/main/mflix/poli-
guarderia.png",
  },
  {
    titulo: "El príncipe de Zamunda",
    resumen: "Un principe africano que se va a Nueva York a buscar esposa",
    genero: "Familiar",
    masVisto: false,
    calificacionImdb: 7,
    premioGalardon: false,
    fechaEstreno: new Date("1988-06-29"),
    imagen:
      "https://raw.githubusercontent.com/Lemoncode/fotos-
ejemplos/main/mflix/principe-zamunda.png",
  },
  {
    titulo: "Angry Birds",
    resumen: "Una pelicula de los pajaros de angry birds",
    genero: "Animacion",
    masVisto: false,
    calificacionImdb: 6.3,
    premioGalardon: false,
    fechaEstreno: new Date("2016-05-11"),
    imagen:
      "https://raw.githubusercontent.com/Lemoncode/fotos-
ejemplos/main/mflix/angry-birds.png",
  },
  {
    titulo: "Los Croods",
    resumen:
      "Una familia de cavernicolas que se encuentra con un hombre moderno",
    genero: "Animacion",
    masVisto: false,
    calificacionImdb: 7.2,
    premioGalardon: false,
    fechaEstreno: new Date("2013-03-15"),
    imagen:
      "https://raw.githubusercontent.com/Lemoncode/fotos-
ejemplos/main/mflix/croods.png",
  },
  {
    titulo: "José, el rey de los sueños",
    resumen: "La historia de José de la biblia",
    genero: "Animacion",
    masVisto: false,
    calificacionImdb: 6.5,
```

```
premioGalardon: false,
fechaEstreno: new Date("2000-10-27"),
imagen:
  "https://raw.githubusercontent.com/Lemoncode/fotos-
ejemplos/main/mflix/jose.png",
},
{
  titulo: "La patrulla canina",
  resumen: "Una pelicula de la patrulla canina",
  genero: "Animacion",
  masVisto: false,
  calificacionImdb: 4.7,
  premioGalardon: false,
  fechaEstreno: new Date("2021-08-20"),
  imagen:
    "https://raw.githubusercontent.com/Lemoncode/fotos-
ejemplos/main/mflix/patrulla-canina.png",
},
{
  titulo: "Los pitufos",
  resumen: "Una pelicula de los pitufos",
  genero: "Animacion",
  masVisto: false,
  calificacionImdb: 5.4,
  premioGalardon: false,
  fechaEstreno: new Date("2011-07-29"),
  imagen:
    "https://raw.githubusercontent.com/Lemoncode/fotos-
ejemplos/main/mflix/pitufos.png",
},
{
  titulo: "El cazador y la reina del hielo",
  resumen: "La historia de la reina del hielo",
  genero: "Aventuras",
  masVisto: false,
  calificacionImdb: 6.1,
  premioGalardon: false,
  fechaEstreno: new Date("2016-04-08"),
  imagen:
    "https://raw.githubusercontent.com/Lemoncode/fotos-
ejemplos/main/mflix/cazador.png",
},
{
  titulo: "Ghost Rider",
  resumen: "Un hombre que se convierte en un esqueleto en llamas",
  genero: "Aventuras",
  masVisto: false,
  calificacionImdb: 5.2,
  premioGalardon: false,
  fechaEstreno: new Date("2007-02-16"),
  imagen:
    "https://raw.githubusercontent.com/Lemoncode/fotos-
ejemplos/main/mflix/ghost-rider.png",
},
```

```
{
  titulo: "La momia",
  resumen: "La historia de la momia",
  genero: "Aventuras",
  masVisto: false,
  calificacionImdb: 5.2,
  premioGalardon: false,
  fechaEstreno: new Date("2008-08-01"),
  imagen:
    "https://raw.githubusercontent.com/Lemoncode/fotos-
ejemplos/main/mflix/momia.png",
},
{
  titulo: "Vampire Academy",
  resumen: "Una academia de vampiros",
  genero: "Aventuras",
  masVisto: false,
  calificacionImdb: 5.5,
  premioGalardon: false,
  fechaEstreno: new Date("2014-02-14"),
  imagen:
    "https://raw.githubusercontent.com/Lemoncode/fotos-
ejemplos/main/mflix/vampire-academy.png",
},
{
  titulo: "Warcraft: El origen",
  resumen: "La historia de Warcraft",
  genero: "Aventuras",
  masVisto: false,
  calificacionImdb: 6.8,
  premioGalardon: false,
  fechaEstreno: new Date("2016-06-03"),
  imagen:
    "https://raw.githubusercontent.com/Lemoncode/fotos-
ejemplos/main/mflix/warcraft.png",
},
];
}
```

Ahora vamos a ir por la parte de la interfaz, para ello vamos a crear un fichero `ui.ts` donde vamos a ir añadiendo las funciones que nos permitan pintar la interfaz.

- Añadimos una función para crear los títulos de las secciones:

`src/ui.ts`

```
const crearTitulo = (tituloSeccion: string): HTMLHeadingElement => {
  const titulo = document.createElement("h2");
  titulo.textContent = tituloSeccion;
  return titulo;
};
```

- Añadimos una función para crear los contenedores de las secciones, le añadimos una clase para poder darle estilos con CSS y una *id* para poder identificarlo:

src/ui.ts

```
const crearContenedor = (nombreClase: string): HTMLDivElement => {
  const listaPeliculas = document.createElement("div");
  listaPeliculas.classList.add(nombreClase);
  listaPeliculas.id = nombreClase;
  return listaPeliculas;
};
```

- Añadimos una función para pintar la lista de las películas, le pasamos el contenedor donde queremos pintarlas y la lista de películas:

src/ui.ts

```
import { Pelicula } from "./modelo";

(...)

export const pintarListaPeliculas = (
  tituloSeccion: string,
  listaPeliculas: Pelicula[]
): void => {
  // obtener el div principal
  const appDiv = document.getElementById("principal");
  // comprobar que existe
  if (appDiv && appDiv instanceof HTMLDivElement) {
    // crear un div para las películas
    const creaDivPeliculas = crearContenedor("peliculas");
    // añadimos el div de películas al div principal
    appDiv.appendChild(creaDivPeliculas);

    // crear título
    const titulo = crearTitulo(tituloSeccion);
    // añadir el título al div de películas
    creaDivPeliculas.appendChild(titulo);

    // crear div lista de películas
    const divListaPeliculas = crearContenedor("lista-peliculas");
    // añadir div lista de películas al div de películas
    creaDivPeliculas.appendChild(divListaPeliculas);

    // crear div contenedor de películas
    const divPeliculasContenedor = crearContenedor("peliculas-contenedor");
    // añadir div contenedor de Películas al div lista de películas
    divListaPeliculas.appendChild(divPeliculasContenedor);

    // pintar películas
    listaPeliculas.forEach((pelicula) => {
```

```
// crear div película
const divPelicula = crearContenedor("pelicula");
// añadir datos a la pelicula
divPelicula.innerHTML =
  `![${pelicula.titulo}](${pelicula.imagen})

```

De momento hemos creado esto, pero si te fijas, da un poco de mal olor, está función volveremos a ella y la refactorizaremos (cuando haces software sigues un proceso iterativo, primero das con una solución y después la vas puliendo, como un escultor con un bloque de mármol).

Pero como vemos no queda bien tener *magic strings* por ahí como "lista-peliculas", "peliculas-contenedor", etc. Vamos a crear un objeto que nos guarde los nombres de las clases y de esta manera si tenemos que cambiar el nombre de alguna clase, solo tenemos que cambiarlo en un sitio.

src/modelo.ts

```
export const nombreClases = {
  peliculas: "peliculas",
  listaPeliculas: "lista-peliculas",
  peliculasContenedor: "peliculas-contenedor",
  pelicula: "pelicula",
};
```

Vamos a cambiar la función *pintarListaPelículas* para que use el objeto que acabamos de crear:

src/ui.ts

```
- import { Pelicula } from "./modelo";
+ import { Pelicula, nombreClases } from "./modelo";
(...)

export const pintarListaPelículas = (
  tituloSección: string,
  listaPeliculas: Pelicula[]
): void => {
  // obtener el div principal
  const appDiv = document.getElementById("principal");
  // comprobar que existe
  if (appDiv && appDiv instanceof HTMLDivElement) {
    // crear un div para las películas
```

```

-   const creaDivPeliculas = crearContenedor("peliculas");
+   const creaDivPeliculas = crearContenedor(nombreClases.peliculas);
    // añadir el div de películas al div principal
    appDiv.appendChild(creaDivPeliculas);

    // crear título
    const titulo = crearTitulo(tituloSeccion);
    // añadir el título al div de películas
    creaDivPeliculas.appendChild(titulo);

    // crear un div lista de películas
-   const divListaPeliculas = crearContenedor("lista-peliculas");
+   const divListaPeliculas = crearContenedor(nombreClases.listaPeliculas);
    // añadir div lista de películas al div de películas
    creaDivPeliculas.appendChild(divListaPeliculas);

    // crear div contenedor de películas
-   const divPeliculasContenedor = crearContenedor("peliculas-contenedor");
+   const divPeliculasContenedor =
crearContenedor(nombreClases.peliculasContenedor);
    // añadir div contenedor de películas al div lista de películas
    divListaPeliculas.appendChild(divPeliculasContenedor);

    // pintar películas
    listaPeliculas.forEach((pelicula) => {
        // crear div película
-       const divPelicula = crearContenedor("pelicula");
+       const divPelicula = crearContenedor(nombreClases.pelicula);
        // añadir datos a la película
        divPelicula.innerHTML =
            `![${pelicula.titulo}](${pelicula.imagen})${pelicula.titulo}`;
        // añadir div película al div contenedor de películas
        divPeliculasContenedor.appendChild(divPelicula);
    });
} else {
    console.error("No se encontró el elemento");
}
};

```

Creamos el archivo *shell.ts*.

```

import { peliculas } from "./datos";
import { pintarListaPeliculas } from "./ui";

document.addEventListener("DOMContentLoaded", () => {
    pintarListaPeliculas("Todas las películas", peliculas);
});

```

Y lo importamos en el *main.ts*.

```
import "./style.css";
+ import "./shell";

- console.log("Hello Typescript!");
```

Y borramos el html que introducimos anteriormente de prueba.

```
(...)
<div id="principal" class="principal">
-   <h2>Familiar</h2>
-   <div class="lista-peliculas" id="lista-peliculas">
-     <div class="flecha-izquierda">
-       
-     </div>
-     <div class="flecha-derecha">
-       
-     </div>
-     <div class="peliculas-contenedor">
-       <div class="pelicula" id="pelicula">
-         
-         <h3>Babe el cerdito valiente</h3>
-       </div>
-       <div class="pelicula" id="pelicula">
-         
-         <h3>Intocable</h3>
-       </div>
-     </div>
-   </div>
</div>
(...)
```

Así ya tenemos la lista de películas pintada en la interfaz.

The screenshot shows a dark-themed movie streaming interface. At the top, there's a navigation bar with the logo 'MFLIX' and links for 'Inicio', 'Series', 'Películas', 'Más recientes', and 'Mi lista'. Below the navigation is a large promotional banner with the text 'Películas y series ilimitadas.' and 'Disfruta en tu TV, computadora, tablet o celular.' A red button labeled 'Comenzar' is centered below the banner. Underneath, there's a section titled 'Todas las películas' with a grid of movie thumbnails. The visible thumbnails include 'Babe el cerdito valiente', 'Intocable', 'The Karate Kid', and 'Poli de Guardería'. Each thumbnail has a small arrow indicating it can be swiped.

- Ahora lo siguiente que podríamos hacer es agregar las típicas flechas que utiliza Netflix para navegar entre las películas que se muestran en cada sección. Para ello vamos a crearnos un nuevo archivo, que lo llamaremos *constantes*, y añadimos un objeto que nos guarde las imágenes de las flechas:

src/constantes.ts

```
+export const flechas = {
+  left: "https://raw.githubusercontent.com/Lemoncode/fotos-
ejemplos/7f4b7d5a25593dd2d2857f2aaa467c5fd87b5cbb/mflix/flecha-izquierda.svg",
+  right:"https://raw.githubusercontent.com/Lemoncode/fotos-
ejemplos/7f4b7d5a25593dd2d2857f2aaa467c5fd87b5cbb/mflix/flecha-derecha.svg",
+};
```

- Ahora vamos a crear una función que nos permita crear las flechas, le pasamos la dirección de la flecha:

Pero antes vamos a crear un *type* para el tipo de flecha que vamos a utilizar:

src/modelo.ts

```
export type TipoFlecha = "izquierda" | "derecha";
```

src/ui.ts

```
+ import { flechas } from "./constantes";
- import { Pelicula, nombreClases } from "./modelo";
+ import { Pelicula, nombreClases, TipoFlecha } from "./modelo";
(...)
+ const añadirFlecha = (
+  contenedor: HTMLDivElement,
+  tipo: TipoFlecha,
+ ): void => {
```

```
+ const divFlecha = document.createElement("div");
+ divFlecha.className = `flecha-${tipo}`;

+ const imgFlecha = document.createElement("img");
+ imgFlecha.src = tipo === "izquierda" ? flechas.left : flechas.right;
+ divFlecha.appendChild(imgFlecha);

+ divFlecha.addEventListener("click", () => {
+   if (tipo === "izquierda") {
+     contenedor.scrollBy({
+       left: -contenedor.clientWidth,
+       behavior: "smooth",
+     });
+   } else {
+     contenedor.scrollBy({
+       left: contenedor.clientWidth,
+       behavior: "smooth",
+     });
+   }
+ });

+ contenedor.appendChild(divFlecha);
+ };
(...)
```

Qué hace esta función:

- Creamos un contenedor para la flecha.
- Le añadimos una clase para poder darle estilos con CSS.
- Creamos la imagen de la flecha y le añadimos la dirección de la imagen.
- Le añadimos un evento *click* para que cuando pulsemos sobre la flecha se mueva el contenedor de películas.
- Añadimos la flecha al contenedor de las películas.

Ahora vamos a modificar la función que pinta la lista de películas para que nos añada las flechas:

```
export const pintarListaPeliculas = (
  tituloSeccion: string,
  listaPeliculas: Pelicula[]
): void => {
  // obtener el div principal
  const appDiv = document.getElementById("principal");
  // comprobar que existe
  if (appDiv && appDiv instanceof HTMLDivElement) {
    // crear un div para las películas
    const creaDivPeliculas = crearContenedor(nombreClases.peliculas);
    // añadir el div de peliculas al div principal
    appDiv.appendChild(creaDivPeliculas);

    // crear titulo
```

```
const titulo = crearTitulo(tituloSeccion);
// añadir el título al div de películas
creaDivPeliculas.appendChild(titulo);

// crear un div lista de películas
const divListaPeliculas = crearContenedor(nombreClases.listaPeliculas);
// añadir div lista de películas al div de películas
creaDivPeliculas.appendChild(divListaPeliculas);

// crear div contenedor de películas
const divPeliculasContenedor = crearContenedor(
    nombreClases.peliculasContenedor
);
// añadir div contenedor de películas al div lista de películas
divListaPeliculas.appendChild(divPeliculasContenedor);

+ // añadir flechas
+ añadirFlecha(divPeliculasContenedor, "izquierda");
+ añadirFlecha(divPeliculasContenedor, "derecha");

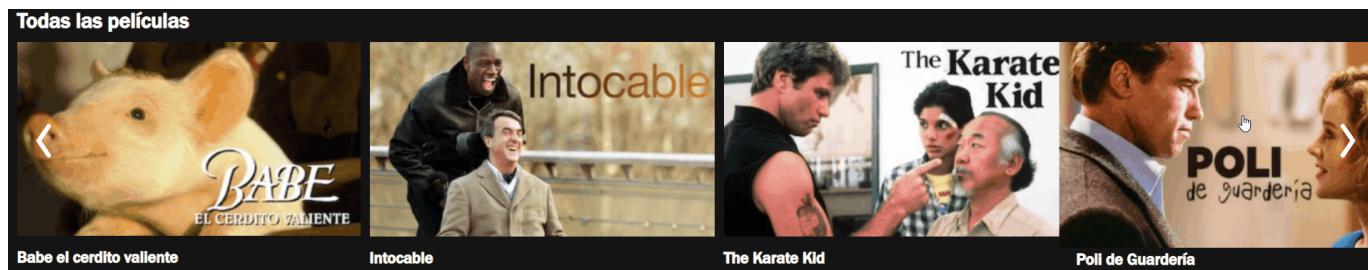
// pintar películas
listaPeliculas.forEach((pelicula) => {
    // crear div película
    const divPelicula = crearContenedor(nombreClases.pelicula);
    // añadir datos a la película
    divPelicula.innerHTML =
        `![${pelicula.titulo}](${pelicula.imagen})
```

De esta forma ya tenemos las flechas en la interfaz y funcionando. Cómo veis hemos tenido que añadir el listado de películas y las flechas a un contenedor, llamado *peliculas-contenedor*, para que funcione el scroll.

```
<div class="lista-peliculas" id="lista-peliculas">
    <div class="peliculas-contenedor" id="peliculas-contenedor" flex == $0>
        <div class="flecha-izquierda"><...></div>
        <div class="flecha-derecha"><...></div>
        <div class="pelicula" id="pelicula">...</div>
        <div class="pelicula" id="pelicula">...</div>
```

Si no lo hacemos así, se haría sobre el *div* que contiene el título de la sección, las flechas y el listado de películas y no funcionaría correctamente.

Ya tenemos el scroll funcionando:



Esta función ha quedado muy grande y hace demasiadas cosas.

**** NO PEGAR ESTE CODIGO ****

```
export const pintarListaPelículas = (
  tituloSección: string,
  listaPelículas: Película[]
): void => {
  // obtener el div principal
  const appDiv = document.getElementById("principal");
  // comprobar que existe
  if (appDiv && appDiv instanceof HTMLDivElement) {
    // crear un div para las películas
    const creaDivPelículas = crearContenedor(nombreClases.películas);
    // añadir el div de películas al div principal
    appDiv.appendChild(creaDivPelículas);

    // crear título
    const título = crearTítulo(tituloSección);
    // añadir el título al div de películas
    creaDivPelículas.appendChild(título);

    // crear un div lista de películas
    const divListaPelículas = crearContenedor(nombreClases.listaPelículas);
    // añadir div lista de películas al div de películas
    creaDivPelículas.appendChild(divListaPelículas);

    // crear div contenedor de películas
    const divPelículasContenedor = crearContenedor(
      nombreClases.películasContenedor
    );
    // añadir div contenedor de películas al div lista de películas
    divListaPelículas.appendChild(divPelículasContenedor);

    // añadir flechas
    añadirFlecha(divPelículasContenedor, "izquierda");
    añadirFlecha(divPelículasContenedor, "derecha");

    // pintar películas
    listaPelículas.forEach((película) => {
```

```

    // crear div película
    const divPelicula = crearContenedor(nombreClases.pelicula);
    // añadir datos a la película
    divPelicula.innerHTML =
      
      <h3>${pelicula.titulo}</h3>
    ;
    // añadir div pelicula al div conteneor de películas
    divPeliculasContenedor.appendChild(divPelicula);
  });
} else {
  console.error("No se encontró el elemento");
}
};


```

Vamos a romperla en varias y le aplicamos el principio de responsabilidad única.

Crearíamos las siguientes funciones:

- *agregarTitulo*: recibe el título de la sección y el contenedor donde queremos que se pinte y nos pinta el título.

```

const agregarTitulo = (
  tituloSeccion: string,
  contenedor: HTMLDivElement
): void => {
  const titulo = crearTitulo(tituloSeccion);
  contenedor.appendChild(titulo);
};


```

Y modificamos la función *pintarListaPelículas* para que use esta función:

```

export const pintarListaPelículas = (
  tituloSeccion: string,
  listaPelículas: Película[]
): void => {
  // obtener el div principal
  const appDiv = document.getElementById("principal");
  // comprobar que existe
  if (appDiv && appDiv instanceof HTMLDivElement) {
    // crear un div para las películas
    const creaDivPelículas = crearContenedor(nombreClases.películas);
    // añadir el div de películas al div principal
    appDiv.appendChild(creaDivPelículas);

    - // crear título
    - const titulo = crearTitulo(tituloSeccion);
    - // añadir el título al div de películas
  }
};


```

```
-     creaDivPelículas.appendChild(título);
+     agregarTítulo(títuloSección, creaDivPelículas);
```

- Vamos a refactorizar la función *crearContenedor*, y la hacemos más genérica, vamos a meter el *appendChild* dentro de la función, pero ¿sería esto buena idea?, pues depende. Si la función la vamos a utilizar en varios sitios, sí, pero si la vamos a utilizar solo en un sitio, no, porque estaríamos acoplando la función a un sitio y no sería reutilizable. En este caso, como la vamos a utilizar en varios sitios, sí que lo vamos a hacer.

```
const crearContenedor = (
  nombreClase: string,
+  contenedor: HTMLDivElement
): HTMLDivElement => {
-  const listaPelículas = document.createElement("div");
-  listaPelículas.classList.add(nombreClase);
-  listaPelículas.id = nombreClase;
+  const div = document.createElement("div");
+  div.classList.add(nombreClase);
+  div.id = nombreClase;
+  contenedor.appendChild(div);
-  return listaPelículas;
+  return div;
};
```

Y modificamos la función *pintarListaPelículas* para que use esta función:

```
export const pintarListaPelículas = (
  títuloSección: string,
  listaPelículas: Película[]
): void => {
  // obtener el div principal
  const appDiv = document.getElementById("principal");
  // comprobar que existe
  if (appDiv && appDiv instanceof HTMLDivElement) {
    // crear un div para las películas
-    const creaDivPelículas = crearContenedor(nombreClases.películas);
-    // añadir el div de películas al div principal
-    appDiv.appendChild(creaDivPelículas);
+    const creaDivPelículas = crearContenedor(nombreClases.películas, appDiv);

    agregarTítulo(títuloSección, creaDivPelículas);

    // crear un div lista de películas
-    const divListaPelículas = crearContenedor(nombreClases.listaPelículas);
-    // añadir div lista de películas al div de películas
-    creaDivPelículas.appendChild(divListaPelículas);
+    const divListaPelículas = crearContenedor(nombreClases.listaPelículas,
  creaDivPelículas);
```

```

    // crear div contenedor de películas
-  const divPeliculasContenedor = crearContenedor(
-      nombreClases.peliculasContenedor
-  );
-  // añadir div contenedor de películas al div lista de películas
-  divListaPeliculas.appendChild(divPeliculasContenedor);
+  const divPeliculasContenedor =
crearContenedor(nombreClases.peliculasContenedor, divListaPeliculas);
(...)

// pintar películas
listaPeliculas.forEach((pelicula) => {
    // crear div película
-  const divPelicula = crearContenedor(nombreClases.pelicula);
+  const divPelicula = crearContenedor(nombreClases.pelicula,
divPeliculasContenedor);
    // añadir datos a la película
    divPelicula.innerHTML =
        
        <h3>${pelicula.titulo}</h3>
    ;
-  // añadir div pelicula al div conteneor de películas
-  divPeliculasContenedor.appendChild(divPelicula);

```

- *pintarFlechas*: recibe el contenedor de las películas y nos pinta las flechas.

```

const pintarFlechas = (peliculaContenedor: HTMLDivElement): void => {
    añadirFlecha(peliculaContenedor, "izquierda");
    añadirFlecha(peliculaContenedor, "derecha");
};

```

Modificamos la función *pintarListaPelículas* para que use esta función:

```

// añadir flechas
-  añadirFlecha(divPeliculasContenedor, "izquierda");
-  añadirFlecha(divPeliculasContenedor, "derecha");
+  pintarFlechas(divPeliculasContenedor);

```

- *pintarPelícula*: recibe una película y el contenedor de las películas y nos pinta la película.

```

const pintarPelícula = (
    pelicula: Película,
    peliculaContenedor: HTMLDivElement
): void => {
    const divPelicula = crearContenedor(
        nombreClases.pelicula,
        peliculaContenedor
);

```

```
divPelicula.innerHTML =
  
  <h3>${pelicula.titulo}</h3>
  ;
};
```

- *pintarPelículas*: recibe una lista de películas y el contenedor de las películas y nos pinta todas las películas.

```
const pintarPelículas = (
  películas: Película[],
  películaContenedor: HTMLDivElement
): void =>
  películas.forEach((pelicula) => {
    pintarPelícula(pelicula, películaContenedor);
});
```

Y así nos quedaría ahora nuestro función *pintarListaPelículas*:

```
export const pintarListaPelículas = (
  títuloSección: string,
  listaPelículas: Película[]
): void => {
  // obtener el div principal
  const appDiv = document.getElementById("principal");
  // comprobar que existe
  if (appDiv && appDiv instanceof HTMLDivElement) {
    // crear un div para las películas
    const creaDivPelículas = crearContenedor(nombreClases.películas, appDiv);

    agregarTítulo(títuloSección, creaDivPelículas);

    // crear un div lista de películas
    const divListaPelículas = crearContenedor(
      nombreClases.listaPelículas,
      creaDivPelículas
    );

    // crear div contenedor de películas
    const divPelículasContenedor = crearContenedor(
      nombreClases.películasContenedor,
      divListaPelículas
    );

    // añadir flechas
    pintarFlechas(divPelículasContenedor);

    // pintar películas
    - listaPelículas.forEach((pelicula) => {
```

```

-     // crear div película
-     const divPelicula = crearContenedor(
-       nombreClases.pelicula,
-       divPeliculasContenedor
-     );
-     // añadir datos a la película
-     divPelicula.innerHTML =
-       `![${pelicula.titulo}](${pelicula.imagen})${pelicula.titulo}
-       `;
-   });
+   pintarPeliculas(listaPeliculas, divPeliculasContenedor);
} else {
  console.error("No se encontró el elemento");
}
};

```

3. Datos y categorías

Pero en Netflix no solo hay una lista de películas, hay varias. Por ejemplo, la lista de películas que son de Aventuras, la lista de películas que tienen premios, la lista de películas mejor valoradas, etc. ¿Cómo podemos hacer para que nuestra aplicación muestre todas estas listas?

Lo primero que vamos a hacer es crear una función que nos devuelva una lista de películas en función de una categoría. Para ello vamos a crear un archivo llamado *motor.ts* y vamos a crear la función:

src/motor.ts

```

import { Pelicula } from "./modelo";

export const filtrarPeliculasPorGenero = (
  peliculas: Pelicula[],
  genero?: TipoGenero
): Pelicula[] => peliculas.filter((pelicula) => pelicula.genero === genero);

```

Esta función recibe una lista de películas, un género y nos devuelve una lista de películas que tengan ese género.

Vamos a *modelo.ts* y creamos un tipo de lista que va a ser un string y de momento solo va a tener el valor de género.

src/modelo.ts

```

type TipoCaracteristica = "genero";

```

Creamos otro con los tipos de Géneros que tenemos en nuestra lista de películas:

```
export type TipoGenero = "Familiar" | "Aventuras" | "Animacion";
```

Y añadimos un nueva interfaz que se va a llamar FiltroPeliculas que va a tener dos propiedades, el tipo de característica y el género.

```
export interface FiltroPeliculas {
  genero?: TipoGenero;
  caracteristica: TipoCaracteristica;
}
```

Lo siguiente que vamos a hacer es crearnos una función, que va a ser un *switch* que nos va a devolver una lista de películas en función a lo que le digamos que haga y va a utilizar la interfaz que acabamos de crear.

Va a tener tres parámetros de entrada, la lista de películas, el tipo de lista que queremos, es decir, queremos que nos filtre por categoría, y le pasamos el género de la película que en este caso va a ser opcional porque no todas las listas van a tener género.

A esta función la vamos a llamar *filtrarPelículas* y va a ser así:

src/motor.ts

```
- import { Pelicula } from "./modelo";
+ import { Pelicula, FiltroPeliculas } from "./modelo";
( ... )

+ export const filtrarPeliculas = (
+   peliculas: Pelicula[],
+   filtro?: FiltroPeliculas
+ ): Pelicula[] => {
+   if(!filtro) return peliculas;
+   switch (filtro.caracteristica) {
+     case "genero":
+       return filtrarPeliculasPorGenero(peliculas, filtro.genero);
+     default:
+       return peliculas;
+   }
+};
```

Ahora vamos de nuevo al modelo y vamos a crearnos una nueva interfaz que se va a llamar ListaPeliculasConfiguracion que va a tener dos propiedades, el título de la sección y el filtro de películas.

src/modelo.ts

```
export interface ListaPeliculasConfiguracion {
  titulo: string;
```

```
    filtro: FiltroPeliculas;
}
```

Vamos al archivo donde estamos definiendo la *ui*. La función que pinta la lista de películas y vamos a modificarla para que nos pinte la lista de películas en función de lo que le digamos.

src/ui.ts

```
- import { Pelicula, nombreClases, TipoFlecha } from "./modelo";
+ import { Pelicula, nombreClases, ListaPeliculaConfiguracion } from "./modelo";
+ import { filtrarPeliculas } from "./motor";
( ... )
export const pintarListaPeliculas = (
-   tituloSeccion: string,
-   listaPeliculas: Pelicula[]
+   listaPeliculas: Pelicula[],
+   configuracion: ListaPeliculaConfiguracion
): void => {
( ... )

+   const peliculasFiltradas = filtrarPeliculas(listaPeliculas, filtro);

-   pintarPeliculas(listaPeliculas, peliculaContenedor);
+   pintarPeliculas(peliculasFiltradas, peliculaContenedor);
} else {
  console.error("No se encontró el elemento");
}
};
```

Ahora vamos a *shell.ts* y vamos a modificar la llamada a la función *pintarListaPelículas* para que nos pinte la lista de películas de Aventuras.

src/shell.ts

```
document.addEventListener("DOMContentLoaded", () => {
-   pintarListaPeliculas("Todas las películas", peliculas);
+   pintarListaPeliculas(peliculas, { titulo: "Todas las películas" });
+   pintarListaPeliculas(peliculas, {
+     titulo: "Películas de Aventuras",
+     filtro: { genero: "Aventuras", caracteristica: "genero" },
+   });
});
```

Si vamos a la aplicación, vemos que nos pinta la lista de películas de Aventuras.

Películas de Aventuras



Pues ya tenemos nuestra aplicación funcionando con listas de películas, vamos a añadir la lista de películas de Familiares y de Animación.

src/shell.ts

```
document.addEventListener("DOMContentLoaded", () => {
  pintarListaPeliculas(peliculas, { titulo: "Todas las películas" });
  pintarListaPeliculas(peliculas, {
    titulo: "Películas de Aventuras",
    filtro: { genero: "Aventuras", caracteristica: "genero" },
  });
  + pintarListaPeliculas(peliculas, {
  +   titulo: "Películas Familiares",
  +   filtro: { genero: "Familiar", caracteristica: "genero" },
  + });
  + pintarListaPeliculas(peliculas, {
  +   titulo: "Películas de Animación",
  +   filtro: { genero: "Animacion", caracteristica: "genero" },
  + });
});
```

Ahora vamos a crear nuevas listas de películas, por ejemplo, una lista de películas que tengan premios, filtrar por películas que sean más vistas y por último ordenarlas por calificación.

Te toca (¿a qué lo echabas de menos?), vamos a crear nuevas listas de películas, por ejemplo, una lista de películas que tengan premios, filtrar por películas que sean más vistas y por último ordenarlas por calificación.

Aquí van un par de pistas:

- Para filtrar por las películas que tengan premios, tenemos que crear una nueva función en el *motor*, que nos devuelva una lista de películas que tengan premios, es decir, que la propiedad *premioGalardon* sea *true*.
- Para filtrar por las películas más vistas, tenemos que crear una nueva función que nos devuelva una lista de películas que sean más vistas, es decir, que la propiedad *masVisto* sea *true*.
- Para ordenar las películas por calificación, tenemos que crear una nueva función que nos devuelva una lista de películas ordenadas por calificación, es decir, que la propiedad *calificacionImdb* sea mayor que la otra. Para eso podemos usar el método *sort*.
- Luego tendremos que modificar el *type tipoLista* y añadir *premios*, *masVistas* y *calificacion*.

- Tendremos que añadir nuevos casos al switch, de la función *filtrarPelículas*
- Por último, nos iremos al archivo *shell.ts* y añadiremos las nuevas listas de películas.

Poned pausa el video, intentad hacerlo, no preocuparos sino lo conseguís, que a continuación vemos la solución.

Para ello vamos a crear estas nuevas funciones en *motor.ts* para abarcar todos estos casos.

src/motor.ts

```
+ const filtrarPeliculasPorPremios = (peliculas: Pelicula[]): Pelicula[] =>
+   peliculas.filter((pelicula) => pelicula.premioGalardon);

+ const filtrarPeliculasMasVistas = (peliculas: Pelicula[]): Pelicula[] =>
+   peliculas.filter((pelicula) => pelicula.masVisto);

+ const ordenarPeliculasPorCalificacion = (peliculas: Pelicula[]): Pelicula[] =>
+   peliculas.sort((peliculaA, peliculaB) => peliculaB.calificacionImdb -
peliculaA.calificacionImdb);
```

Vamos a cambiar el *type* *Tipo* lista para que acepte estos nuevos tipos de listas.

src/modelo.ts

```
- type TipoCaracteristica = "genero";
+ type TipoCaracteristica = "genero" | "premios" | "masVistas" | "calificacion";
```

Modificamos la función *filtrarPelículas* para qué nos devuelva una lista de películas en función de lo que le digamos.

src/motor.ts

```
export const filtrarPeliculas = (
  peliculas: Pelicula[],
  filtro?: FiltroPeliculas
): Pelicula[] => {
  if(!filtro) return peliculas;

  switch (tipoLista) {
    case "genero":
      return filtrarPeliculasPorGenero(peliculas, genero);
+    case "premios":
+      return filtrarPeliculasPorPremios(peliculas);
+    case "masVistas":
+      return filtrarPeliculasMasVistas(peliculas);
+    case "calificacion":
+      return ordenarPeliculasPorCalificacion(peliculas);
    default:
```

```
        return peliculas;
    }
};
```

src/shell.ts

```
document.addEventListener("DOMContentLoaded", () => {
    pintarListaPeliculas(peliculas, { titulo: "Todas las películas" });
    pintarListaPeliculas(peliculas, {
        titulo: "Películas de Aventuras",
        filtro: { genero: "Aventuras", caracteristica: "genero" },
    });
    pintarListaPeliculas(peliculas, {
        titulo: "Películas Familiares",
        filtro: { genero: "Familiar", caracteristica: "genero" },
    });
    pintarListaPeliculas(peliculas, {
        titulo: "Películas de Animación",
        filtro: { genero: "Animacion", caracteristica: "genero" },
    });
+   pintarListaPeliculas(peliculas, {
+     titulo: "Películas más vistas",
+     filtro: { caracteristica: "masVistas" },
+   });
+   pintarListaPeliculas(peliculas, {
+     titulo: "Películas con mejor calificación",
+     filtro: { caracteristica: "calificacion" },
+   });
+   pintarListaPeliculas(peliculas, {
+     titulo: "Películas con premios",
+     filtro: { caracteristica: "premios" },
+   });
});
```

Este archivo *shell.ts* lo podríamos refactorizar para que nos quedara más limpio, pero lo dejamos así para que se vea más claro.

Resumen

En esta práctica guiado hemos aprendido:

- Como plantear un problema complejo.
- Como dividirlo en partes más pequeñas.
- Como resolver cada una de las partes.
- ¡Hemos simulado la portada de NetFlix ! 😊