Documentación de los ejercicios realizados.

Hecho por Luis Stive Silva Prudencio

Solicitud R2 (HTML)

- En primer lugar, puse un h1 para poner el titulo
- Luego cree un div con la clase “table-container” para poder estilizarlo en css y que sirva de contenedor de toda la tabla. Dentro del div hay un table, el cual contiene un thead (cabecera de la tabla) y un tbody (cuerpo de la tabla).

```
<body>
    <h1 class="titulo">Tabla</h1>
    <div class="table-container">
        <table>
            <thead>···
            </thead>
            <tbody>···
            </tbody>
        </table>
    </div>
</body>
```

- Dentro del thead introduje un tr (fila) el cual contiene 6 th (celda de la cabecera). Estos 6 th sirven para identificar que la tabla tendrá 6 tablas.

```
<thead>
    <tr>
        <th class="fijo">Código</th>
        <th>Nombre</th>
        <th>Edad</th>
        <th>Fecha de ingreso</th>
        <th>Puesto</th>
        <th>Sueldo</th>
    </tr>
</thead>
```

- Dentro del tbody hay 10 tr, esto significa que la tabla tendrá 10 filas (sin contar el encabezado). Estas tr tienen 6 td (celda de tabla), los cuales tienen relación con las th de su columna. También, a cada elemento de la primer columna, le asigne una clase llamada “fijo”.

```
<tbody>
    <tr>
        <td class="fijo">cod100</td>
        <td>Jorge Fernández</td>
        <td>53</td>
        <td>28/10/1994</td>
        <td>Gerente</td>
        <td>2300</td>
    </tr>
    <tr>···
    </tr>
    <tr>···
    </tr>
    <tr>···
    </tr>
    <tr>···
    </tr>
    <tr>···
    </tr>
    <tr>···
    </tr>
    <tr>···
    </tr>
    <tr>···
    </tr>
    <tr>···
    </tr>
    <tr>···
    </tr>
    <tr>···
    </tr>
    <tr>···
    </tr>
</tbody>
```

Solicitud R2 (CSS)

- Primero puse los estilos bases y asigne variables en el root para poder usarlas en cualquier posición.

```css
:root {
  /* VARIABLES */
  /* Colores */
  --background-body: #efefef;
  --background-grey-table: #ccc;
  --background-white-table: #eee;
}

html {
  font-size: 100%;
  box-sizing: border-box;
  font-size: 62.5%;
  scroll-behavior: smooth;
}

*,
*::before,
*::after {
  box-sizing: inherit;
}

body {
  margin: 0;
  padding: 0;
  font-family: "Open Sans", sans-serif;
  font-size: 1.7rem;
  background-color: var(--background-body);
}

h1,
h2,
h3,
p {
  margin: 0;
  padding: 0;
}
```

- Al título lo posicioné al centro y le di un margen de 2rem arriba y abajo.

```css
.titulo {
  text-align: center;
  margin: 2rem auto;
}
```

- El table-container tiene un ancho máximo que ocupa todo el ancho del padre (body), el overflow-x con scroll nos permite navegar por la tabla sin mover toda la pantalla cuando usemos móvil, Tablet o algún dispositivo con poco ancho de pantalla. Por ultimo, le puse margen para que la tabla no esté pegada a la pantalla y también para que se separe un poco del título.

```css
.table-container {
  max-width: 100%;
  overflow-x: scroll;
  margin: 15px;
}
```

- A la tabla le asigné que todos sus textos estén alineados a la izquierda. Además le puse borde arriba y abajo de 1 pixel color negro. Como estos bordes generaban unos pequeños espacios, le puse 0 a border-spacing. Y el margin auto permite que la tabla está centrada.

```css
table {
  text-align: left;
  border-top: 1px solid black;
  border-bottom: 1px solid black;
  border-spacing: 0;
  margin: auto;
}
```

- Acá le asigne a todas las celdas de la tabla un padding en los laterales de 10 px para que no esté tan pegado al borde. Además, les puse un borde a la derecha, un min-width de 180px que probé para que todas las celdas se vean parejas y un height de 37px para que no estén pegadas con los bordes de arriba y abajo. Trate de ponerle padding arriba y abajo pero las palabras largas se hacían salto de línea y no me gustaba

```css
table th,
table td {
  padding: 0 10px;
  border-right: 1px solid black;
  min-width: 180px;
  height: 37px;
}
```

- A las celdas de la cabecera le puse un background color blanco. Después, a las filas impares del cuerpo de la tabla le puse un background gris y a las pares un background blanco.

```css
th {
  background-color: var(--background-white-table);
}

table tbody tr:nth-child(odd) {
  background-color: var(--background-grey-table);
}
table tbody tr:nth-child(even) {
  background-color: var(--background-white-table);
}
```

- Como mencioné anteriormente, a las primeras columnas de la tabla le puse una clase llamada “fijo”. Esto para que cuando naveguemos por la tabla con, por ejemplo, un celular, podamos seguir viendo la primer columna que, generalmente, es la más importante en una tabla. Para que esté fija le puse un position sticky y un left 0. Por último le puse un borde en la izquierda, antes ese borde lo tenia la tabla general, pero mientras probaba como quedaba, me di cuenta que al desplazarnos en celular, el borde desaparecía, por ello se la di a la columna fija y problema resuelto.

- Luego, como en la imagen anterior, le asigné background a la columna fija dependiendo si eran par o impar.

```css
.fijo {
  position: sticky;
  left: 0;
  border-left: 1px solid black;
}

table tbody tr:nth-child(odd) .fijo {
  background-color: var(--background-grey-table);
}
table tbody tr:nth-child(even) .fijo {
  background-color: var(--background-white-table);
}
```

- Para finalizar, les agregue un hover a las filas, cambiándoles el color del baground y el de la fuente.

```css
table tbody tr:hover,
table tbody tr:hover .fijo {
  background-color: royalblue;
  color: white;
}
```

Solicitud R2 (Resultados)

- Escritorio

Tabla

Código	Nombre	Edad	Fecha de ingreso	Puesto	Sueldo
cod100	Jorge Fernández	53	28/10/1994	Gerente	2300
cod101	Jessica Valencia	22	03/03/2019	Atención al cliente	300
cod102	Agustin Tobarez	26	19/06/2017	Ventas	400
cod103	Adrian Tolaba	19	01/03/2021	Atención al cliente	300
cod104	Brescia Muñoz	20	28/09/2019	Ventas	400
cod105	Federico Bravo	42	30/01/2018	Limpieza	500
cod106	Marcelo Uncal	31	15/02/2010	Administración	600
cod107	Florencia Lazarte	22	10/10/2020	Marketing	600
cod108	José Ferrer	24	16/05/2019	Ventas	400
cod109	Emma Medrano	46	19/05/2011	Administración	800

- Mobile

Tabla

Código	Nombre
cod100	Jorge Fernández
cod101	Jessica Valencia
cod102	Agustin Tobarez
cod103	Adrian Tolaba
cod104	Brescia Muñoz
cod105	Federico Bravo
cod106	Marcelo Uncal
cod107	Florencia Lazarte
cod108	José Ferrer
cod109	Emma Medrano

Tabla

Código	ueldo
cod100	2300
cod101	300
cod102	400
cod103	300
cod104	400
cod105	500
cod106	600
cod107	600
cod108	400
cod109	800

Solicitud R3 (HTML)

- En primer lugar guardé todo el contenido en un div llamado “page-container”. Dentro de este hay 3 divs que también sirven de contenedores. Uno es para la ventana modal, otro para la imagen y el último para el texto.
- Utilicé 2 iconos de Font awesome (un botón de play y una X).

```
<body>
    <div class="page-container">
        <div class="modal-container" id="modal-container">
            <div class="modal-ventana modal-cerrado" id="ventana">
                <i class="fas fa-times-circle boton-cerrar" id="cerrar"></i>
                <video class="video" id="video" src="lollapalooza2021.mp4" controls></video>
            </div>
        </div>
        <div class="imagen-container">
            <img class="imagen" src="lollapalooza-chicago-4.jpg" alt="Lollapalooza">
            <i class="far fa-play-circle icono" id="play"></i>
        </div>
        <div class="text-container">
            <h1 class="titulo">Lollapalooza ha regresado. Sé el primero en conseguir tu entrada.</h1>
            <p class="texto">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quam exercitatio
        </div>
    </div>
    <script src="https://kit.fontawesome.com/dc56f53266.js" crossorigin="anonymous"></script>
    <script src="scripts.js"></script>
</body>
```

Solicitud R3 (CSS)

- En el css primero me ocupe de posicionar todos los elementos de manera mobile first. Y la ventana modal del video lo deje para lo último.
- Page-container: Le di un alto mínimo que ocupe todo el viewport-alto de la pantalla. También un background gradient que quedó bien.
- Imagen: lo importante acá es el alto máximo que le asigne, cuando aumentaba el tamaño de la ventana con el inspector, veía como aumentaba la imagen y no me gustaba, por ello lo deje a que se quede en esa altura (en celular). El filter se lo coloqué porque el botón no se distinguía bien por la imagen.

```css
.page-container {
  min-height: 100vh;
  background-image: linear-gradient(120deg, #fccb90 0%, #d57eeb 100%);
}
.imagen {
  width: 100%;
  max-height: 300px;
  object-fit: cover;
  filter: grayscale(0.5);
}
```

- Todos los estilos que tiene el imagen-container sirve para que el icono de play esté centrado.
- El ícono tiene position absolute para que se pueda centrar con su padre que tiene relative y el transition es para un hover que le puse más adelante.

```css
.imagen-container {
  display: flex;
  justify-content: center;
  align-items: center;
  position: relative;
}
.icono {
  color: rgb(255, 0, 255);
  font-size: 8rem;
  position: absolute;
  cursor: pointer;
  transition: transform 300ms linear;
  z-index: 100;
}
```

- Estilos básicos para los textos

```css
.text-container {
  padding: 15px;
}

.titulo {
  line-height: 1.2;
  margin-bottom: 2rem;
}
```

- Teniendo todo maquetado en mobile, pase a estilizar su versión de escritorio. Lo que tiene el body sirve para que el page-container quede centrado en la página.
- Al page-container le asigne unos valores de ancho y alto para dimensionarlo. Luego, el display, justify y align-ítems fueron para centrar su contenido.
- El width de 50% que tiene el imagen-container y el text-container son para que el page-container tenga un 50% de imagen y un 50% de texto.

```css
@media screen and (min-width: 1024px) {
  body {
    display: flex;
    justify-content: center;
    align-items: center;
  }
  .page-container {
    max-width: 1024px;
    min-height: 500px;
    display: flex;
    justify-content: space-around;
    align-items: center;
    border: 2px solid black;
    overflow: hidden;
  }
  .imagen-container {
    width: 50%;
  }
  .imagen {
    min-height: 500px;
  }
  .text-container {
    width: 50%;
  }
}
```

- Esta parte es para que cuando se vea la página en Tablet, la imagen no sea tan pequeña.

```css
@media screen and (min-width: 600px) {
  .imagen {
    min-height: 500px;
  }
}
```

- Terminado todo, continúe con la ventana modal.
- El modal-container tiene una opacidad nula y su visibility hidden. El position fixed sirve para sacar al elemento del flujo y, en caso de que haya más contenido en la página, se siga viendo la venta modal al scrollear. El contenedor centra todos sus elementos y tiene un z-index superior al resto.
- El modal-ventana tiene un width de 80% y un transition para cuando aparezca al apretar play.

```css
.modal-container {
  opacity: 0;
  visibility: hidden;
  position: fixed;
  width: 100%;
  height: 100vh;
  top: 0;
  left: 0;
  display: flex;
  justify-content: center;
  align-items: center;
  background-color: rgba(0, 0, 0, 0.5);
  z-index: 1000;
}

.modal-ventana {
  width: 80%;
  transition: transform 500ms linear;
}
```

- Como 80% de width en el modal ventana en la vista de escritorio era mucho, lo reduje a 50% solo para escritorio.
- Al modal ventana le asigne una clase llamada “modal cerrado”, esta clase está por defecto y hace que la ventana del modal esté fuera de la pantalla.

```css
@media screen and (min-width: 1024px) {
  .modal-ventana {
    width: 50%;
  }
}
.modal-cerrado {
  transform: translateY(-200%);
}
```

- Por último, le di estilos al video, al icono de cerrar y al posicionar el mouse sobre el icono de play, hará que este crezca 1.2 de su escala.

```css
.video {
  width: 100%;
}

.boton-cerrar {
  font-size: 30px;
  color: #ccc;
  cursor: pointer;
}

.icono:hover {
  transform: scale(1.2);
}
```

Solicitud R3 (JavaScript)

- Primero, asigné los elementos que iba a manipular a variables.

```
let abrir = document.getElementById("play")
let cerrar = document.getElementById("cerrar")
let modal=document.getElementById("ventana")
let modalCont = document.getElementById("modal-container")
let video = document.getElementById("video")
```

- Le asigné un evento de escucha al botón abrir. Este evento se activa al clickear el icono de play. Cuando esto ocurre, cambiamos los estios del modal-container para que sea visible y le sacamos la clase “modal-cerrado” a la ventana modal. Por último, le asigno al elemento video el atributo src con la ruta del video. Esto lo explicare luego. Con todo esto, aparece la ventana modal desde arriba y se puede ver el video.

```
abrir.addEventListener("click", ()=>{
    modalCont.style.visibility="visible"
    modalCont.style.opacity="1"
    modal.classList.toggle("modal-cerrado")
    video.setAttribute("src", "lollapalooza2021.mp4")
})
```

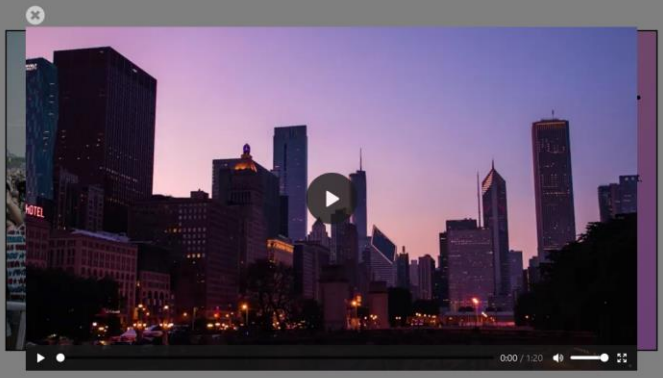
- Le asigné un evento de escucha al botón de cerrar que se activa al clickearlo. Este le vuelve a dar la clase “modal-cerrado” a la ventana modal. Luego le puse un setTimeout con 500ms para que coincida con la transición de la ventana-modal y para que el video se corte al cerrarlo, le asigno un atributo src vacio al elemento video.

```
cerrar.addEventListener("click", ()=>{
    modal.classList.toggle("modal-cerrado")
    setTimeout(() => {
        modalCont.style.visibility="hidden"
        modalCont.style.opacity="0"
        video.setAttribute("src", "")
    }, 500);
})
```

Solicitud R3 (Resultados)

- Escritorio (sin ventana modal)

- Escritorio (con ventana modal)



- Mobile (Sin ventana modal)

Lollapalooza ha regresado. Sé el primero en conseguir tu entrada.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quam exercitationem quasi ab magni rem impedit reiciendis, autem quis quas tempore dolore! Temporibus omnis officia et at laboriosam, animi blanditiis fugiat nisi similique hic vitae asperiores laborum perferendis totam in cum modi commodi cumque nihil. Accusamus, rerum corrupti? Rem, dolor expedita!

Mobile (Con ventana modal)

Lollapalooza ha

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quam exercitationem quasi ab magni rem impedit reiciendis, autem quis quas tempore dolore! Temporibus omnis officia et at laboriosam, animi blanditiis fugiat nisi similique hic vitae asperiores laborum perferendis totam in cum modi commodi cumque nihil. Accusamus, rerum corrupti? Rem, dolor expedita!

Solicitud R4 (HTML)

- La estructura base consiste en un titulo, un contenedor de la tabla y la tabla con una cabecera de 6 columnas y el cuerpo de la tabla está vacío (lo completaremos con js)
- Los estilos de la tabla son los mismos de la solicitud R2

```
<body>
    <h1 class="titulo">Clima en las capitales de sudamérica</h1>
    <div class="table-container">
       <table>
        <thead>
            <tr>
                <th class="fijo">País</th>
                <th>Capital</th>
                <th>Temperatura</th>
                <th>Humedad</th>
                <th>Viento</th>
                <th>Icono</th>
            </tr>
        </thead>
        <tbody id="table-body">

        </tbody>
       </table>
    </div>
    <script src="scripts.js"></script>
</body>
```

Solicitud R4 (JavaScript)

- Primero creamos la variable key, esta contiene la clave de la api.
- Luego cree un array con las capitales de los países que constituyen Sudamérica.
- Además, el elemento con id “table-body” lo guardé en una constante.

```
let key = "e545ec04f84cefd2ed0600237fcf9466"
let cityArray = ['buenos aires', 'la paz', 'brasilia', 'santiago
de chile', 'bogota', 'quito', 'georgetown', 'asuncion','lima',
'paramaribo','montevideo','caracas']
const tableBody = document.getElementById('table-body')
```

- La función getData necesita 2 parámetros, key y city. Estos son necesarios para el url que va a necesitar fetch para recuperar los datos. Luego esos datos son convertidos a json. Después escribimos código html en tableBody, en esté podemos ver que se crea una fila de la una tabla con 6 celdas con datos obtenidos por la consulta.

```
function getData(key, city) {
    fetch(`http://api.weatherstack.com/current?access_key=${key}&
    query=${city}`)
    .then(res => res.json())
    .then(res=>{
            tableBody.innerHTML+=`
                <tr>
                    <td class="fijo">${res.location.country}</td>
                    <td>${res.location.name}</td>
                    <td>${res.current.temperature} °C</td>
                    <td>${res.current.humidity} %</td>
                    <td>${res.current.wind_speed} km/h</td>
                    <td><img src="${res.current.weather_icons[0]}
                    "></td>
                </tr>
            `
    })
    .catch(err => {
        console.error(err);
    });
}
```

- Por último, la función renderizar tiene como parámetro un array. Dentro de la función hay un ciclo for que va a ejecutar la función getData dependiendo del tamaño del array dado como parámetro. Con esta función, al darle el cityArray creado anteriormente, podemos renderizar todas las capitales de los países de Sudamérica.

```
const renderizar = (array) =>{
    for(let i=0; i<array.length;i++){
        getData(key,array[i])
    }
}

renderizar(cityArray)
```

Solicitud R4 (Resultado)

- Escritorio

Clima en las capitales de sudamérica

País	Capital	Temperatura	Humedad	Viento	Icono
Colombia	Bogota	18 °C	49 %	20 km/h	
Cuba	Chile	26 °C	84 %	6 km/h	
Bolivia	La Paz	4 °C	81 %	7 km/h	
Argentina	Buenos Aires	13 °C	77 %	22 km/h	
Brazil	Brasilia	24 °C	65 %	6 km/h	
Ecuador	Quito	17 °C	67 %	5 km/h	
Guyana	Georgetown	26 °C	94 %	7 km/h	
Venezuela	Caracas	25 °C	74 %	9 km/h	
Suriname	Paramaribo	25 °C	89 %	7 km/h	
Peru	Lima	20 °C	68 %	10 km/h	
Uruguay	Montevideo	13 °C	82 %	35 km/h	
Paraguay	Asuncion	13 °C	72 %	7 km/h	

Fin

Gracias por leer