

Creamos la clase Persona y ordenamos los constructores en cada clase. Después eliminamos los datos que pasábamos fijos a los constructores:

```
class Persona {
    #_nombre;
    #_apellidos;
    #_edad;

    constructor(nombre, apellidos, edad) {
        this.#_nombre = nombre;
        this.#_apellidos = apellidos;
        this.#_edad = edad;
    }

    //Genero Getters y Setters
    get nombreJugador() {
        return `${this.#_nombre} ${this.#_apellidos}`;
    }

    set nombreJugador(nombre) {
        this.#_nombre = nombre;
        return 'Nombre cambiado!!!';
    }

    set apellidoJugador(apellidos) {
        this.#_apellidos = apellidos;
        return 'Apellidos cambiados!!!';
    }

    get edadJugador() {
        return this.#_edad;
    }

    set edadJugador(edad) {
        this.#_edad = edad;
        return 'Edad cambiada!!!';
    }
}
```

```
class Jugador extends Persona {
    #_entrenamiento;

    constructor(nombre, apellidos, edad) {
        super(nombre, apellidos, edad);
        this.#_entrenamiento = 0;
    }

    get nivelEntrenamiento() {
        return this.#_entrenamiento;
    }

    set nivelEntrenamiento(nivel) {
        this.#_entrenamiento = nivel;
        return 'Nivel cambiado!!!';
    }
}
```

Preparamos los dummies para que funcionen con los constructores:

```
//Jugadores dummy para practicar
let jugador1 = new Jugador('Adrián', 'Escribano', 20);
let jugador2 = new Jugador('Alberto', 'Martínez', 33);
```

Refactorizamos el nombre del botón cambiándolo por un nombre que nos dé más información:

```
//Creamos variables de arranque
let botonNuevoJugador = document.getElementById('nuevoJugador');
botonNuevoJugador.addEventListener('click', introducirJugador);
```

Cambiamos en la función de introducir jugadores quitamos el evento que le pasamos a la función.

También metemos una función que deshabilita los botones de “inicio” y de “nuevo Jugador”

```
//Función raíz añadir jugador
function introducirJugador() {
  disableButtons();
  muestraForm();
  vaciarInputs();

  return;
}
```

```
function disableButtons() {
  const botones = document.getElementsByTagName('button');

  for (let intCont = 0; intCont < botones.length - 1; intCont++) {
    botones[intCont].disabled = true;
  }

  return;
}
```

Función deshabilitar botones

También hemos decidido sacar las funciones muestraForm() y vaciarInputs() como funciones independientes para mejorar la modularización del código:

```
function muestraForm() {
  const nodoAddJugador = document.getElementsByClassName('addJugador')[0];

  if (nodoAddJugador.children.length === 0) {
    let camposJugadores = document.getElementsByClassName('addJugador')[0];
    let campos = crearCampos();
    camposJugadores.appendChild(campos);

    return;
  }

  nodoAddJugador.classList.remove('hidden');
  return;
}

function vaciarInputs(): void {
  const nodos: HTMLCollection = document.getElementsByClassName('addJugador')[0].children;

  if (nodos) {
    for (const nodo: Element of nodos) {
      nodo.value = '';
    }
    return;
  }
  return;
}
```

En la función crearCampos() de muestra form hemos decidido que sería una mejor idea separar cada campo creándolo en una función independiente, de forma que se disminuya la repetición de código:

```
function crearCampos() : DocumentFragment {
  const campoNombre = generaElemento(
    tipoElemento: 'input',
    idElemento: 'nombreJugador',
    tipoDato: 'text',
    placeholder: 'Nombre'
  );
  const campoApellidos = generaElemento(
    tipoElemento: 'input',
    idElemento: 'apellidosJugador',
    tipoDato: 'text',
    placeholder: 'Apellidos'
  );
  const campoEdad = generaElemento(
    tipoElemento: 'input',
    idElemento: 'edadJugador',
    tipoDato: 'number',
    placeholder: 'Edad'
  );
  const botonInsertar = generaElemento(
    tipoElemento: 'button',
    idElemento: 'botonInsertar',
    tipoDato: 'button',
    placeholder: 'Insertar jugador'
  );
  botonInsertar.addEventListener('click', insertarJugador);

  return generaFragmento(campoNombre, campoApellidos, campoEdad, botonInsertar);
}
```

La función generaElemento() va a recibir una serie de parámetros y en base a ello va a generar un tipo de elemento u otro con sus atributos propios.

```
function generaElemento(tipoElemento, idElemento, tipoDato, placeholder) : any {
  const elementoReturn = document.createElement(tipoElemento);
  elementoReturn.id = idElemento;
  elementoReturn.type = tipoDato;

  if (tipoElemento === 'button') {
    elementoReturn.innerText = placeholder;
    return elementoReturn;
  }

  elementoReturn.placeholder = placeholder;
  return elementoReturn;
}
```

Por último creamos una función generaFragmento() que recibe un número indeterminado de atributos los cuales se estructuran en un array que se anidan en el fragmento, como resultado de la función se devuelve el fragmento con todos los elementos anidados:

```
function generaFragmento(... elementos) : DocumentFragment {
  const fragmento : DocumentFragment = document.createDocumentFragment();

  for (const elemento of elementos) {
    fragmento.appendChild(elemento);
  }

  return fragmento;
}
```

Al botón de insertar jugador se le asocia una función que genera un jugador y en el caso de no ser nulo, lo asocia al array de jugadores posibles:

```
function insertarJugador() : void {
    const nuevoJugador : Jugador = creaJugador();

    if (nuevoJugador !== null) {
        misJugadores.push(nuevoJugador);
        ocultarForm();
        document.getElementById( elementId: 'nuevoJugador').disabled = false;
    }
}
```

Esta función de generación de jugadores contiene un control de errores básico de forma que se impide la generación de jugadores con valores vacíos o con una edad que sea un String.

```
function creaJugador() : Jugador | null {
    const campoNombre = document.getElementById( elementId: 'nombreJugador').value;
    const campoApellidos = document.getElementById( elementId: 'apellidosJugador').value;
    const campoEdad = document.getElementById( elementId: 'edadJugador').value;

    let nuevoJugador : null = null;

    if (campoNombre === '' || campoApellidos === '' || !isNaN(parseInt(campoEdad))) {
        nuevoJugador = new Jugador(campoNombre, campoApellidos, campoEdad);
        return nuevoJugador;
    }

    return nuevoJugador;
}
```

Por último, modularizamos la función para ocultar el formulario una vez que se pulsa el botón insertar.

```
function ocultarForm() : void {
    let camposJugadores : Element = document.getElementsByClassName( className: 'addJugador')[0];
    camposJugadores.classList.add('hidden');
}
```

En cuanto a la funcionalidad de inicio del juego también hemos dividido la función general en mini-funciones:

```
function inicioJuego() : void {
    // Deshabilito adición de jugadores
    disableButtons();

    //Recibo dos jugadores nuevos
    dosJugadores = dameJugadores();
    mostrarJugadores(dosJugadores);

    //Arranco los listeners
    preparadosYa();
}
```

Hemos decidido cambiar el nombre de una de las funciones a “mostrarJugadores()” porque creemos que es más representativo de lo que ocurre al ejecutarla.

En la función dameJugadores() hemos cambiado algunas cosas:

```
function dameJugadores() : [any, any] {  
    let numeroJugador1 : number = getRandom(misJugadores);  
    let numeroJugador2;  
  
    do {  
        numeroJugador2 = getRandom(misJugadores);  
    } while (numeroJugador1 === numeroJugador2);  
  
    return [misJugadores[numeroJugador1], misJugadores[numeroJugador2]];  
}
```

Por ejemplo hemos creado una función externa getRandom() que devuelve un número aleatorio para que pueda ser utilizada también en la fase de selección de letras, eliminando de esta manera la repetición de código:

```
function getRandom(array) : number {  
    const lenght = array.length;  
    return Math.floor(Math.random() * lenght);  
}
```

También hemos cambiado el while por un do-while ya que como la primera opción siempre se va a ejecutar creemos que se mejora un poco la funcionalidad del código.

En la función de mostrarJugadores lo que hacemos es pasar por el array de jugadores para imprimirlos en las secciones correspondientes con la función infoJugador()

```
function mostrarJugadores(jugadoresPasados) : void {  
    for (let contador : number = 0; contador < jugadoresPasados.length; contador++) {  
        const jugador : string = `jugador${ contador + 1 }`;   
        infoJugador(jugadoresPasados[contador], document.getElementById(jugador));  
    }  
}
```

```
function infoJugador(jugador, nodo) : void {  
    let misP = nodo.getElementsByTagName('p');  
  
    misP[0].innerHTML = `<strong>Nombre:</strong> ${jugador.nombreJugador}`;  
    misP[1].innerHTML = `<strong>Edad:</strong> ${jugador.edadJugador}`;  
    misP[2].innerHTML = `<strong>Entrenamiento:</strong> ${jugador.nivelEntrenamiento}`;  
}
```

En cuanto a la funcionalidad del juego hemos decidido que es un mejor feedback que el usuario vea que aumenta el contador al pulsar la tecla y no al soltarla de forma que creamos las siguientes funciones para activar y desactivar eventos de escucha:

```
function activoTeclas() : void {  
    document.addEventListener('keydown', detectarTecla);  
    document.addEventListener('keyup', sueltaTecla);  
}
```

```
function desactivoTeclas() : void {  
    document.removeEventListener('keydown', detectarTecla);  
    document.removeEventListener('keyup', sueltaTecla);  
}
```

En la función de detectarTecla() ahora necesitaremos una bandera que nos diga si la tecla está pulsada o no:

```
function detectarTecla(evento) : void {
  if (evento.key === letras[0] && banderaTeclas[0]) {
    let contador : number = Number.parseInt(contador1.innerText);
    banderaTeclas[0] = false;
    contador++;
    contador1.innerText = contador.toString();
    return;
  }

  if (evento.key === letras[1] && banderaTeclas[1]) {
    let contador : number = Number.parseInt(contador2.innerText);
    banderaTeclas[1] = false;
    contador++;
    contador2.innerText = contador.toString();
  }
}
```

Esto se consigue gracias a la función sueltaTecla() asociada al evento 'keyup':

```
function sueltaTecla(evento) : void {
  if (evento.key === letras[0]) {
    banderaTeclas[0] = true;
  }


  if (evento.key === letras[1]) {
    banderaTeclas[1] = true;
  }
}
```

Para que esto sea posible hemos creado una variable global que inicializamos en true:

```
let banderaTeclas : any[] = [true, true];
```

Esta variable se inicializa a true al iniciar una partida:

```
function inicioJuego() : void {
  // Deshabilito adición de jugadores
  disableButtons();

  // Recibo dos jugadores nuevos
  dosJugadores = dameJugadores();
  mostrarJugadores(dosJugadores);
  
  banderaTeclas = [true, true];

  // Arranco los listeners
  preparadosYa();
}
```