- Introducción.

En este tutorial continuaremos trabajando con las funciones de cadena y también introduciremos los llamados arrays o matrices de datos. Aprenderemos a crear nuestras propias funciones para ver sus ventajas.

1.- Convertir a mayúsculas y a minúsculas.

Existe un código (función) que permite convertir en mayúsculas el contenido de una variable.
La función es esta**:** to**UpperCase();**
Aquí podéis ver un ejemplo con este interactivo. Se puede escribir la solución con letras mayúsculas y/o minúsculas indistintamente.

Comprobar

```
on (press) {
   if(_root.palabra.toUpperCase() == "DINOSAURIO") {

   _root.palabra="MUY BIEN"
}
}
```

La variable **_root.palabra** está vinculada a un campo de texto de forma que la palabra que escribe se convierte a mayúsculas.

Del mismo modo podemos utilizar la función que convierte las mayúsculas a minúsculas. La función es esta: **toLowerCase();**

2.- Recorrer cadenas con un bucle.

Vamos a crear un programa que permita detectar los errores cuando se escribe una palabra incorrectamente. El programa nos debe indicar con un signo de interrogación cada letra que esté mal escrita o que falte.

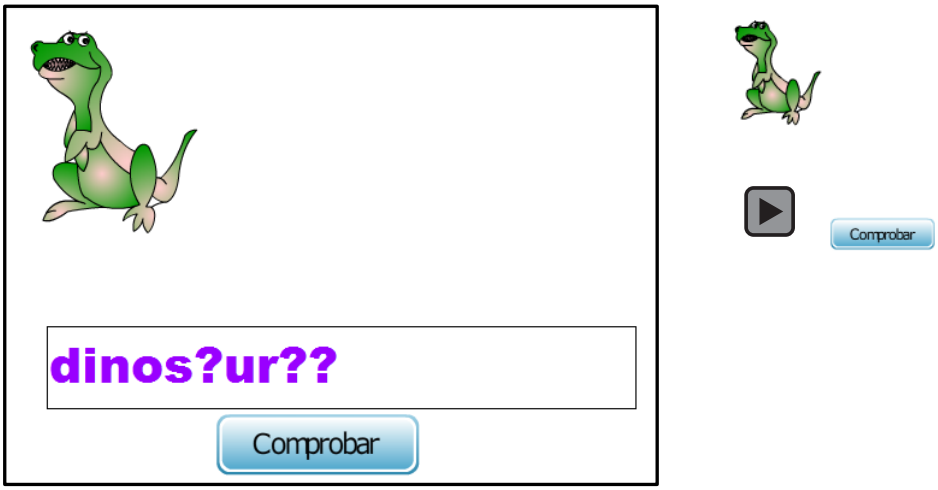


Imagen 1

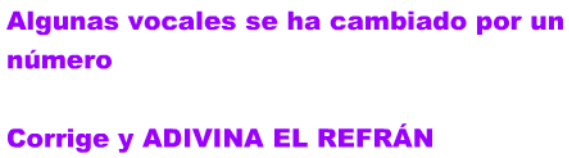
```
on (press) {
   _root.solucion = "dinosaurio";
   _root.pista = "";

   for (n=0; n<_root.solucion.length; n++) {

      if (_root.palabra.substr(n, 1) == _root.solucion.substr(n, 1)) {

         _root.pista += _root.solucion.substr(n, 1);

      } else {
         _root.pista += "?";
      }
   }

   if (_root.palabra == _root.solucion) {

      _root.palabra = "MUY BIEN";
   } else {
      _root.palabra = _root.pista;
   }
}
```

Como podemos observar en el código se utiliza un bucle donde recorre letra por letra de la palabra que ha escrito el usuario y la compara con la solución. Si coincide ***if (_root.palabra.substr(n, 1) == _root.solucion.substr(n, 1))***, entonces acumula en una variable llamada **pista** la letra correcta.

De lo contrario (**else**) lo que hace es acumular el signo de interrogación (?). El número de veces que se repite el bucle es igual a la longitud de la palabra escrita por el usuario.

3. Creación de arrays o matrices.

Ahora vamos a crear un programa pueda generar frases en las que unas vocales se cambian por números y el usuario ha de adivinarlas.



Vamos a introducir una nueva acción que nos permite crear listas ordenadas de datos. Se llaman **matrices o arrays**. Son como variables pero numeradas e indexadas.

Para crear un array es tan fácil como hacer esto:

```
frase = new Array();
```

Después les damos un valor. En nuestro ejemplo son frases.

```
frase[0] = "Abril aguas mil";
frase[1] = "Dime con quién andas y te diré quién eres";
frase[2] = "A mal tiempo buena cara";
frase[3] = "No por mucho madrugar amanece más temparano";
```

Ahora con la siguiente acción genero una frase al azar y la almaceno en otra variable:

```
_root.solucion = frase[random(4)];
```

Creo una variable donde voy a almacenar la nueva frase donde algunas vocales serán sustituídas por números:

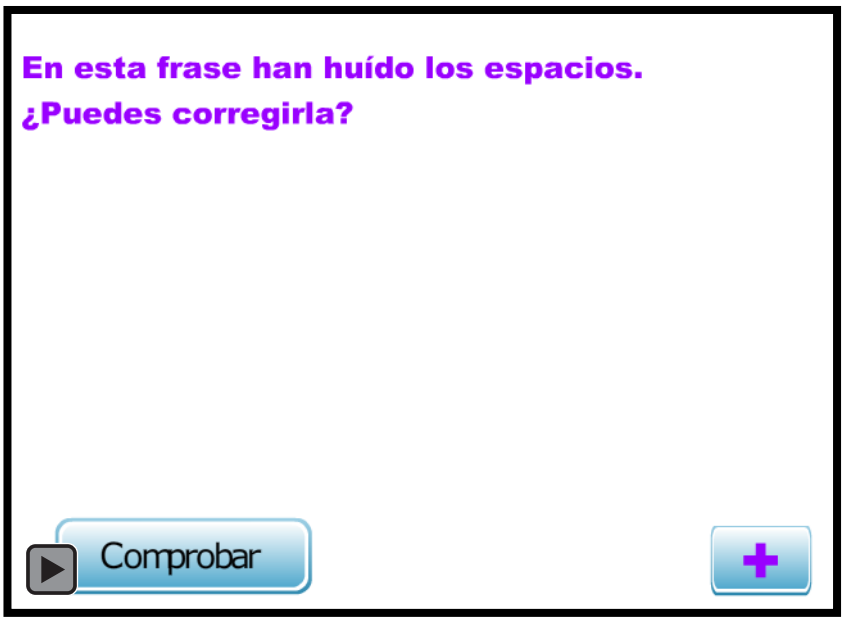
```
_root.palabra_nueva = "";
```

Y ahora con el código siguiente voy a recorrer letra por letra y voy a sustituir las vocales por los números. En caso de que no haya las vocales que quiero sustituir almaceno la letra que ya existía:

```
for (n=0; n<_root.solucion.length; n++) {//repite  num. veces = longitud de la palabra.

        if (_root.solucion.substr(n, 1) == "a") { // si es la letra “a”
        _root.palabra_nueva += "1"; }//registra “1”
        else if (_root.solucion.substr(n, 1) == "e") { //si es igual a “e”
        _root.palabra_nueva += "2"; } // registra “2”
        else if (_root.solucion.substr(n, 1) == "i") {// si es igual a “i”
        _root.palabra_nueva += "3";//registra “3”
        } else { _root.palabra_nueva += _root.solucion.substr(n, 1); }// de lo contrario registra la
        letra original.
}
```

4.- Creación de funciones personalizadas.

En este apartado vamos a crear una función que nos permita extraer los espacios en blanco de una frase. Y lo aprovecharemos para crear un juego donde se generan frases sin espacios y el usuario debe dividir las palabras para reconstruir las frases.



La primera parte es la misma que en los juegos anteriores donde ya se explicó cómo crear matrices o arrays:

```
frase = new Array();
frase[0] = "Abril aguas mil";
frase[1] = "Dime con quién andas y te diré quién eres";
frase[2] = "A mal tiempo buena cara";
frase[3] = "No por mucho madrugar amanece más temparano";
```

Ahora vamos a crear la función:

```
function quitar_espacios() { //se crea la función.
    _root.palabra_nueva = "";//creamos una variable  y le damos valor de vacío.
   _root.solucion = frase[random(4)];//se elige al azar una frase.
    for (n=0; n<_root.solucion.length; n++) {// se inicia el bucle para recorres la frase.

        if (_root.solucion.substr(n, 1)<>" ") {//compruba que no es un espacio

            _root.palabra_nueva += _root.solucion.substr(n, 1);//entonces registra la letra en la
variable.
        }
    }
} //finalizamos la creación de la función.
```

Aquí es donde ejecuto la función que hemos creado:

```
_root.quitar_espacios();
```

Como se puede observar es fácil crear funciones. La estructura es:

function nombre_de_la_función() {

Aquí se pone el código que queramos

}

Ahora nos podemos preguntar: ¿y para qué me sirven las funciones?

Pues en este caso para no tener que repetir el código cada vez que lo ejecutemos. En este programa se ejecuta la función cuando se inicia el programa y cuando pulsamos el **botón +** para generar nuevos ejercicios:

```
on (press) {//botón + que genera nuevas frases.
_root.quitar_espacios();// aquí ya no hace falta escribir otr vez todo el código
}
```

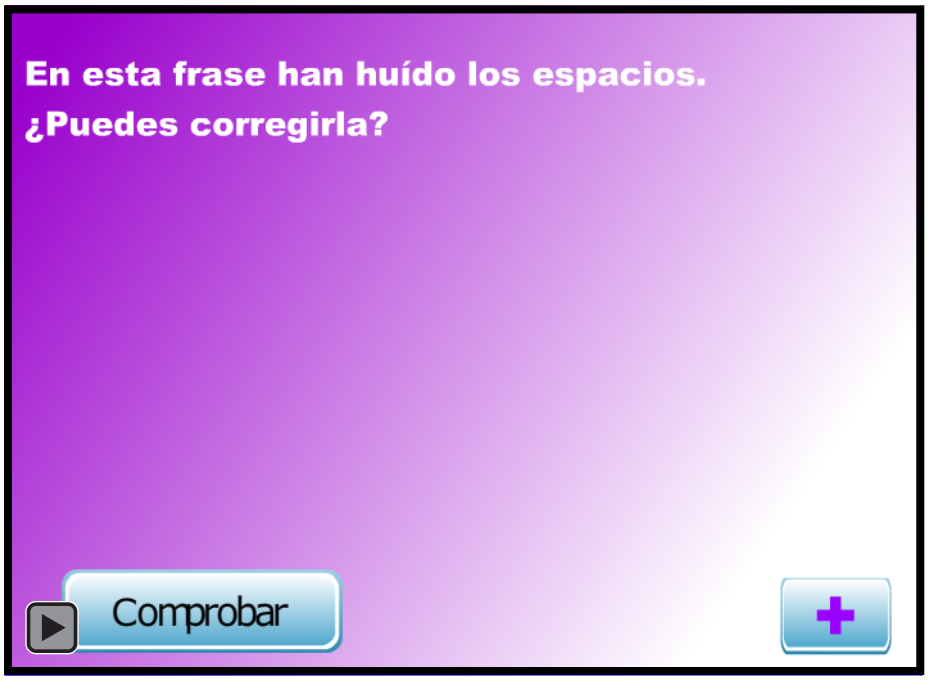
Cómo podemos observar gracias a las funciones con una línea ejecuto todo el código. Y si tengo que modificar el código no debo modificarlo en todos los lugares, solo en la función.

Finalmente con el **botón comprobar** verificamos la respuesta del usuario.

```
on (press) {
        if (_root.palabra_nueva == _root.solucion) {
                _root.palabra_nueva = "MUY BIEN";
        }
}
```

5.- Otro ejemplo de aplicación.

Para aprender a crear aplicaciones donde se procesa texto es importante practicar mucho. En este apartado vamos a crear una aplicación que genere frases que deben leerse al revés para que después el usuario las escriba correctamente.



Nos basamos en el mismo código para empezar:

```
frase = new Array();
frase[0] = "Abril aguas mil";
frase[1] = "Dime con quién andas y te diré quién eres";
frase[2] = "A mal tiempo buena cara";
frase[3] = "No por mucho madrugar amanece más temparano";
```

Creamos la función que permite elegir una frase aleatoria y dar la vuelta la frase de izquierda a derecha:

```
function frases_alreves() {
   _root.solucion = frase[random(4)];
   _root.palabra_nueva = "";
   for (n=_root.solucion.length; n>-1; n--) {// repetimos desde el final de la frase
      _root.palabra_nueva += _root.solucion.substr(n, 1);//vamos registrando letra por
letra desde el final.

   }
}

 _root.frases_alreves();//Ejecutamos la función.
```

Código del botón +

```
 on (press) {
_root.frases_alreves();
}
```

Código del botón comprobar

```
on (press) {
   if (_root.palabra_nueva == _root.solucion) {

      _root.palabra_nueva = "MUY BIEN";
   }
}
```

6.- Ejercicios propuestos.

- Una aplicación que permita pegar un texto en un campo de texto y mediante un botón quitar todos los acentos (abiertos y cerrados) de las vocales. El usuario deberá corregir el texto y el programa comprobar que lo ha hecho bien.

- Una aplicación de libre elección que implique el uso de los nuevos conceptos aprendidos.

- Actividad avanzada. Una aplicación que divida a una palabra en letras (clips de películas) que se puedan arrastrar por el escenario.