DigitalHouse > Coding School

Desarrollo Web Full Stack Node

Ejercitación - M08C20

Eventos, timers y formularios

Ejercicio: abracadabra.

En esta práctica tu trabajo será trabajar en el archivo /assets/js/main.js. En algunas de las consignas puede que encuentres que también debas modificar el archivo HTML.

1. Primero lo primero. Una de las primeras cosas que deberíamos hacer en nuestro archivo JS para asegurar que no tengamos problemas a futuro es encerrar todo nuestro código dentro del evento **onload**. Tu trabajo entonces, es definir este evento.
2. Bien, vamos asegurar que esté todo bien vinculado agregando un **alert** en el archivo dentro del evento **onload**. Luego y con la tranquilidad de que todo funciona bien, eliminaremos la alerta.
3. Hora de definir un primer evento:
    a. Debes definir un evento al hacer click en el botón que dice **Abracadabra**
    b. Al hacer click deberás lanzar un **prompt** que le diga al usuario que ingrese su nombre.
    c. Tras ingresar su nombre, debes reemplazar el texto que dice “Hi, I’m **Jane Doe**” por “Hi, I’m NOMBRE” donde NOMBRE debe ser el nombre ingresado por el usuario.
4. Vamos con un segundo evento que suceda en el botón que dice **Get in touch with me** pero cuando se haga **doble click** sobre el mismo. En ese caso, modificar el color del párrafo con clase **“parrafo-color”** (se encuentra arriba del botón) al color rojo.

(Todo esto se encuentra en el **footer** de nuestro HTML)

5. Definir una función **colorAlAzar**. Esta función debe armar un array con 5 colores. Luego, sortear al azar un número entre 0 y 4 y retornar el color correspondiente del array.
6. Modificar el evento de doble click sobre el botón **Get in touch with me** para que el texto se modifique a un color al azar en vez de siempre pasar a rojo…
7. Definir 3 eventos sobre los 3 elementos que se presentan en la foto (Estamos refiriendonos a las etiquetas **section** aunque tendrás que modificar el HTML para poder ser específico). Al hacer click sobre estos elementos deberás:
    a. Obtener el **h3** dentro del elemento clickeado. Para esto recomendamos utilizar **this.querySelector**. Funciona muy similar al **document.querySelector** pero solo busca dentro del elemento clickeado.
    b. Imprimir en un alerta el mensaje “Clickeaste sobre **Consequat Lorem**”. Donde el texto debe ser reemplazado por el contenido del **h3**

1. Generar 2 eventos sobre el botón que dice “See some of my recent work”
    a. Cuando se pase el mouse por encima, el párrafo que se encuentra justo encima debe cambiar de color a un color al azar
    b. Al quitar el mouse del botón, el párrafo debe volver a su color original #888
2. Definiremos un evento al clickear en la foto de la lechuza:
    a. Primero debe disparar una alerta que diga “Preparate para el futuro…”
    b. Luego de 5 segundos disparar una segunda alerta que diga “Y el futuro ya llegó!”
3. Definir un evento que al presionar cualquier tecla diga “Ey, tocaste el teclado!”

4. Modificar el evento anterior para que el mensaje sea “Ey, tocaste la barra!” y solo se vea ese mensaje si el usuario presiona la barra espaciadora
5. Modificar el evento anterior para que el evento solo suceda si se presiona la barra espaciadora estando en el campo de **email**. Además, el evento debe evitar el comportamiento por defecto (no se debería poder escribir un espacio en un campo de email)
6. Vamos con un desafío bastante más complejo… vamos a crear una **máquina de estados**. Nuestro objetivo será detectar cuando el usuario tipee de corrido la palabra “secreto”. El problema es que solamente podemos definir un evento cuando el usuario presiona una tecla y no cuando escribe toda una palabra. Por eso es que para empezar el ejercicio vamos a definir una variable **estadoSecreto** que empiece con el número 0. A partir de ahí, vamos a implementar un código interno que solo nosotros sabemos:

   0 significa que todavía no escribió nada
   1 significa que escribió “s”
   2 significa que escribió “se”
   3 significa que escribió “sec”
   4 significa que escribió “secr”
   5 significa que escribió “secre”
   6 significa que escribió “secret”

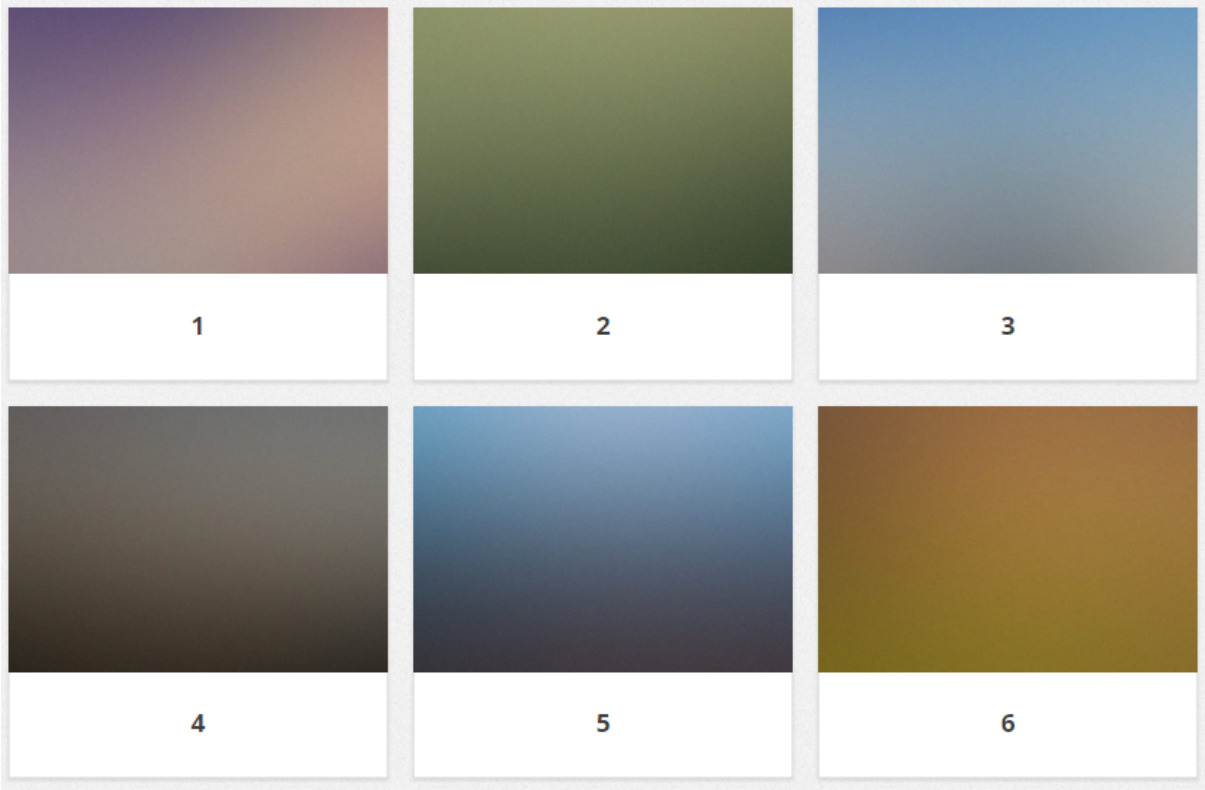
   ¿Qué debe hacer nuestro código?

   Definiremos un evento al presionar una tecla que implemente la siguiente lógica:

1) Si el estado es 0 y se presiona la tecla S, la variable **estadoSecreto** pasa a 1.
2) Si el estado es 1 y se presiona la tecla E, la variable **estadoSecreto** pasa a 2.
3) Si el estado es 2 y se presiona la tecla C, la variable **estadoSecreto** pasa a 3.
4) Si el estado es 3 y se presiona la tecla R, la variable **estadoSecreto** pasa a 4.
5) Si el estado es 4 y se presiona la tecla E, la variable **estadoSecreto** pasa a 5.
6) Si el estado es 5 y se presiona la tecla T, la variable **estadoSecreto** pasa a 6.
7) Si el estado es 6 y se presiona la tecla O, la variable **estadoSecreto** vuelve a 0 y se dispara una alerta que diga “SECRETO MAGICO”.
8) Si no se cumple ninguna de las condiciones el estado vuelve a 0.

Extra Extra

1. A continuación vamos a crear un juego, este consistirá en que el usuario deberá elegir uno de los siguientes bloques, donde el bloque ganador será generado previamente de manera aleatoria. Le daremos a saber cuando elija el ganador.



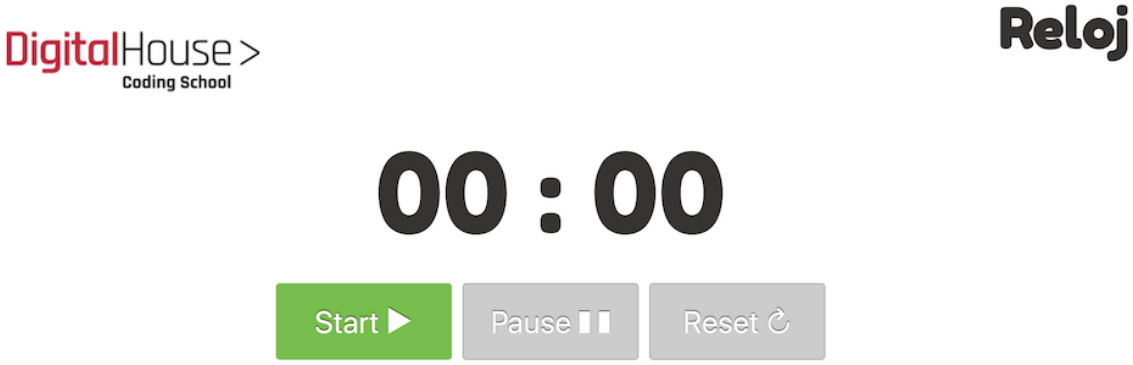
Para deberás hacer lo siguiente:

1. Crear una **variable** que guarde un número **aleatorio entre 1 y 6** (este será el número ganador) .
2. Luego crear una **variable** por cada bloque donde guardaremos la selección de todas las cajas. Deberían ser 6…. (Las cajas se encuentran como **articles** en el HTML**).**

Ejercicio: Cronómetro

¡Vamos a competir! **Tenés 30 minutos** para realizar el ejercicio desde el momento que el profesor lo indique. Descargá los archivos complementarios y leé bien la consigna.

En este ejercicio deberás escribir el código javascript que haga funcionar el reloj.



Contarás con los archivos de base html, css e imágenes dentro del campus. En la carpeta "**/js**" se encuentra el archivo **"functions.js"** dentro del cual deberás escribir el código que permita funcionar al reloj.

El reloj inicia haciendo click en el botón **"Start"** por lo tanto comenzarán a actualizarse los segundos y los minutos de acuerdo al tiempo transcurrido. Tené en cuenta que la vista de los segundos menores a 10 debe verse: 01, 02, 03, 04,….,09.

El botón **"Pause"** detendrá el reloj respetando los valores del tiempo transcurrido. Es decir, no debe volverlo a cero.

Si el usuario clickea en **"Start"** luego de **"Pause"** el reloj debe continuar desde donde frenó.

El botón **“Reset”** volverá el reloj a cero.

Cronómetro - punto bonus

Pensemos un momento en cómo funciona la botonera…. Debería trabajar de acuerdo a la siguiente descripción:

Mientras el reloj no inició únicamente podremos clickear en el botón **“Start”**. Si ya iniciamos el reloj, es decir que hicimos click en **“Start”,** el botón debe deshabilitarse para evitar que el usuario lo presione por accidente. Tampoco, mientras está andando el reloj, debe ser posible hacer click en el botón **“Reset”.**

Sólo si el usuario detiene el reloj con el botón **“Pause”** se habilitarán los botones **“Start”** y **“Reset”**; que por lógica son 2 acciones que solo pueden hacerse al estar el reloj detenido.

Si el código es correcto el css hará su magia para ayudar en la UX.

Si aceptás ir por el bonus tu trabajo será incorporar el comportamiento de la botonera dentro del código que ya escribiste en el punto anterior. ¡Éxitos!

Validaciones de formularios

Tomando como punto de partida el archivo **formulario.html** vamos a generar un archivo **js/validaciones.js** y vincularemos los dos archivos.

El objetivo de esta práctica es **validar completamente el formulario.** Dicha validación deberá seguir los siguientes requerimientos:

A. Todos los campos son obligatorios. Ninguno puede estar vacío.
B. Si se trata de enviar el formulario (presionando el botón ENVIAR) estando todos o algún campo vacío. No será posible enviar el formulario y a aquellos campos input con error se les deberá agregar la clase **“is-invalid”** (necesitamos esta clase para que css haga su magia).
   Las validaciones no solo deberán hacerse al enviar el formulario si no también al momento en el que el visitante interactúa con cada campo (validación on-time).
C. De igual manera, aquellos campos que poseen error deberán tener un texto que especifique el tipo de error. Dicho texto deberá estar presente en el elemento con clase “**invalid-feedback**” (necesitamos esta clase para que css haga su magia).
D. El formulario contará con validaciones especiales para los siguientes campos:
   a. Correo electrónico: deberá validar que el texto ingresado coincida con un formato de email valido.
   b. Teléfono de contacto: deberá validar que el texto ingresado no contenga caracteres alfabéticos, sólo números.
   c. Contraseña y repetir contraseña: deberá validarse que los dos campos contengan exactamente el mismo texto. Y adicionalmente la contraseña no podrá ser inferior a 4 caracteres.

Una vez el formulario esté validado y tras enviarse el mismo:

A. Deberá desaparecer/ ocultarse el formulario.
B. Deberá mostrar en pantalla un listado <ul> con los valores de cada uno de los campos a excepción de las contraseñas y un texto adicional que diga "Gracias por registrarte".