TUDAI
Recuperatorio - WEB 1

30 JUNIO 2022

Alumno	DNI	Firma	#hojas (escritas)
Comisión Teórica: Javier / Matias		Comisión Práctica (1-15):	

Nota:

- *Es necesario **al menos un avance parcial en JS para aprobar***
- *El código debe estar prolijo e indentado.*

1- Verdadero y Falso

Marque verdadero o falso y justifique todas sus respuestas brevemente. Puede emplear un ejemplo para la justificación. Sin justificación la respuesta no tiene validez

a. En un proyecto de un sitio web es recomendable hacer uso de rutas absolutas para el vínculo o uso de archivos.

   F - Se deben hacer uso de rutas relativas para que el proyecto/rutas/recursos funcionen correctamente en cualquier computadora/servidor

b. Para realizar una modificación en un Servicio REST se utiliza el método POST

   F - Se debe usar PUT y ademas en la url se debe especificar el id a modificar

c. En CSS, herencia y cascada se usan para aplicar estilos de la misma manera.

   F - Son conceptos distintos, herencia se emplea para dar estilos por jerarquia y cascada por especificidad

d. Diseño Responsive (responsivo) y Diseño Adaptive (adaptado) no son equivalentes.

   V - Responsive se refiera a diseñar sobre el mismo sitio distintas vistas que responden dinamicamente de acuerdo al dispositivo mientras que adaptativos son diseños independientes que incluso pueden tener distintas URL´s

e. <head> y <header> no se usan con el mismo propósito.

   V - head es la etiqueta que esta al mismo nivel que body y define lo no visible del sitio (se incluyen metas, link css, favicons, titulo del sitio/pestaña, etc), mientras que header es una etiqueta semántica dentro del body para definir el encabezado del sitio

2- Seleccione la respuesta correcta. No es necesario justificar. Marcar sobre la misma hoja.

Respuesta mal contestada resta -0.25, sin contestar 0.00 y bien contestada +1.00.

A. Si a una página Web le sacamos el archivo CSS:

1. No se ve nada.
2. **Lo que se ve, dependerá de lo definido en html.**
3. No se puede sacar el css.
4. La página no se puede abrir en el navegador.

B. ¿ Qúe etiqueta es HTML5?

1. <head>
2.
3. **<canva>**
4. <script>

C. Usando el modelo de cajas defina el tamaño del bloque en px que corresponde a:

`div.card {`
`  width: 500px;`
`  height: 200px;`
`  font-size: 20px;`
`  padding: 20px;`
`  margin-left: 50px;`
`  border: 4px solid black;`
`}`
	Ancho: 598 px. Total = width: 500px + 2 x padding: 20px + margin-left: 50px + 2 x border: 4px
	Alto: 248 px. Total = height 200px + 2 x padding: 20px + 2x border: 4px

D. El DOM sirve para:

1. Incluir el Javascript desde HTML.
2. agregar comportamiento según el tipo de dispositivo.
3. **acceder y manipular HTML desde nuestro código Javascript.**
4. vincular html con CSS.

E. Al utilizar AJAX Partial Render

1. Puedo incluir contenido dinámico de igual modo que usando un Iframe
2. Puedo agregar fragmentos de contenido de modo sincrónico, transmitiendo menos información.
3. **Se puede agregar contenido html, mejorando la experiencia de usuario**

3. HTML + CSS

A. Analice el siguiente **código HTML + CSS** y complete la tercera columna con el **color resultante** de cada texto.

`<header>`
`  <p>Texto 1</p>`
`<header>`
`<p class="especial">Texto 2</p>`
`<p>`
`  Texto 3`
`  <span class="suave">Texto 4</span>`
`  <span class="resaltado">Texto 5</span>`
`</p>`
	`p, p.especial {`
	`  color: blue;`
	`}`
	
	`header p {`
	`  color: green;`
	`}`
	`p .resaltado {`
	`  color: pink;`
	`}`
	
	`.resaltado {`
	`  color: yellow;`
	`}`
	
	`p.suave {`
	`  color: gray;`
	`}`
	
	`.suave {`
	`  color: black;`
	`}`
		TEXTO 1: GREEN
		TEXTO 2: BLUE
		TEXTO 3: BLUE
		TEXTO 4: BLACK
		TEXTO 5: PINK

B. Escriba solo el **código CSS** para crear algunos estilos generales a un sitio web. **Utilice tags semánticos.**

i. La tipografía de las listas desordenadas debe ser “CoolSerif”. El menu debe tener la tipografia “CoolSerifThin”. Al pasar con el mouse sobre los items del menú el fondo debe ser azul.

```
ul {
    font-family: 'CoolSerif';
}

nav {
    font-family: 'CoolSerifThin';
}

nav ul {
    font-family: 'CoolSerifThin';
}

nav ul li:hover {
    background-color: blue;
}
```

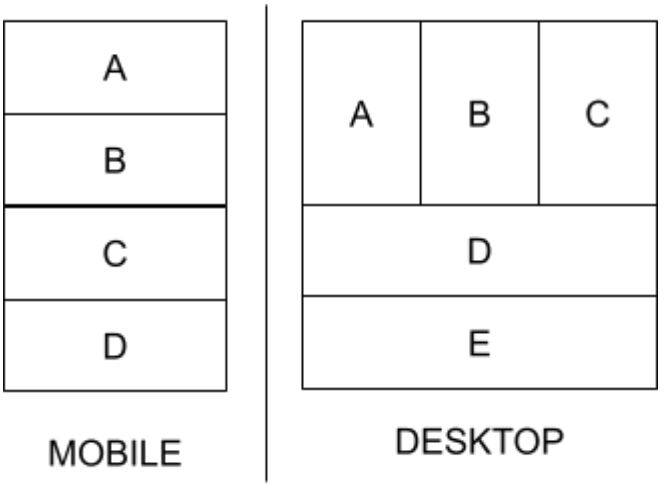
ii. Todas las imágenes deben tener un borde de 2px, de color azul, salvo las que se encuentren en los articulos con clase *“selectas”*, las cuales deben tener el borde en rojo y del mismo grosor.

```
img {
    border: solid 2px blue;
}

article img.selectas {
    border: solid 2px red;
}
```

4. Diseño responsive

Escriba el código CSS necesario para crear el siguiente layout responsive.

- Se debe realizar utilizando **Mobile First.**
- No importan las dimensiones, ni bordes de los elementos.
- No se puede utilizar ningún framework CSS



SOLUCION

ALTERNATIVA FLEX

```
<div class="container">
    <div class="top">
        <div>A</div>
        <div>B</div>
        <div>C</div>
    </div>
    <div>D</div>
    <div class="e">E</div>
</div>
```

```
.e {
    display: none;
}

@media only screen and (min-width: 720px) {
    .top {
        display: flex;
        flex-direction: row; /* es igual no poner la dir
    }

    .e {
        display: block; /* flex esta bien tambien */
    }
}
```

```
// PUEDEN NO USAR EL PADRE CONTENEDOR
```

ESTA ES LA VERSION MAS SENCILLA DONDE NO SE LE PONE NINGUN DISPLAY FLEX A LOS CONTENEDORES EN MOBILE

ALTERNATIVA GRID

```
<div class="container">
    <div class="a">A</div>
    <div class="b">B</div>
    <div class="c">C</div>
    <div class="d">D</div>
    <div class="e">E</div>
</div>
```

```
.container {
    display: grid;
    grid-template-areas:"A"
                        "B"
                        "C"
                        "D";
}

.a { grid-area: A; }
.b { grid-area: B; }
.c { grid-area: C; }
.d { grid-area: D; }
.e {
    grid-area: E;
    display: none;
}
```

	`@media only screen and (min-width: 720px) { .container { display: grid; grid-template-areas:"A B C" "D D D" "E E E"; } .e { display: block; /* flex esta bien tambien */ } }`

ALTERNATIVA GRID 2

OTRA ALTERNATIVA ES DEFINIR EN DESKTOP UNA GRILLA DE 3 X 3 CON ALGO COMO:

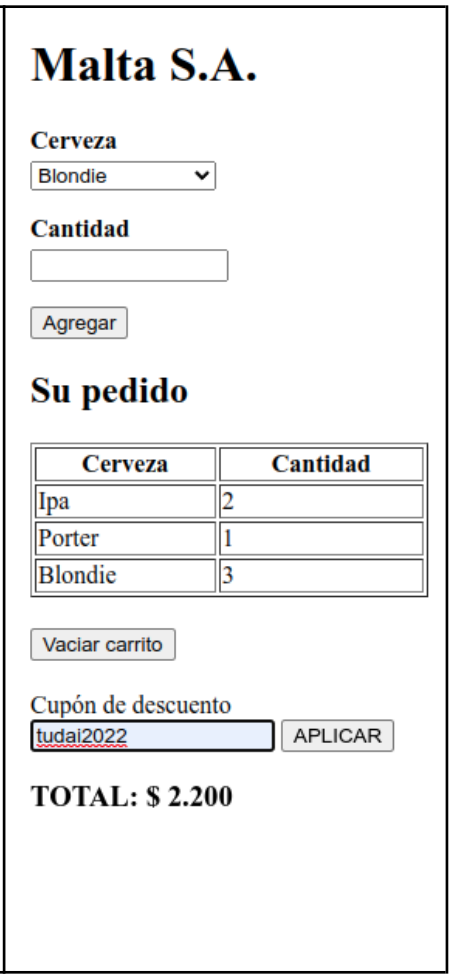
```
grid-template-columns: repeat(3, auto);
grid-template-rows: repeat(3, auto);
```

Y USAR grid-column y grid-row en cada item para posicionarlo donde quieran

5- Javascript

La cervecería Malta S.A. quiere agregar un carrito de compras en su página web. En este carrito los usuarios podrán ir cargando los artículos uno por uno y además podrán aplicar un cupón de descuento.

```
<h1>Malta S.A.</h1>
   <form id="form_cerveza">
       <label for="cerveza">Cerveza</label>
       <select id="cerveza" name="cerveza">
           <option value="blondie">Blondie</option>
           <option value="ipa">IPA</option>
           <option value="porter">Porter</option>
       </select>

      <label for="cantidad">Cantidad</label>
      <input id="cantidad" name="cantidad" type="number">

      <button id="btn_agregar" type="submit">Agregar</button>
   </form>

   <h2>Su pedido</h2>
   <table class="lista_pedido">
       <thead>
           <tr>
               <th>Cerveza</th>
               <th>Cantidad</th>
           </tr>
       </thead>
       <tbody>
           <!-- escribir el listado del carrito -->
       </tbody>
   </table>

   <button id="btn_vaciar">Vaciar carrito</button>


   <form id="form_cupon">
       <label for="cupon">Cupón de descuento</label>
       <input id="cupon" name="cupon" type="text">

       <button id="btn_aplicar" type="submit">APLICAR</button>
   </form>

   <h3>TOTAL: $ <span id="precio_total"> </span></h3>
```



- **El valor de cada cerveza es de $400**
- **El cupón de descuento es de $200 en total si ingresa la palabra “tudai2022”**

Escribir el código para cumplir los siguientes requerimientos:

- Se debe mostrar el listado del carrito de compras como lo indica la imagen a medida que se van agregando los items.
- El campo “cantidad” de cada cerveza es obligatorio y debe ser un valor entre 1 y 10.
- Se debe poder vaciar el carrito desde el botón “Vaciar carrito”
- Un usuario no puede comprar más de 20 cervezas en total. En caso de excederse cuando ingresa un ítem nuevo el sistema le debe informar por pantalla (alert) que no puede agregarlo.
- Solo se podrá aplicar el cupón de descuento una sola vez.

NOTA:

- si utiliza validaciones HTML5 agregue las propiedades al html en esta misma hoja

AYUDA: El valor de un select se lee de la misma manera que un input

RESOLUCION
Se aceptan versiones sin arreglos y solo variables globales que vayan llevando las cantidades.

```
"use strict";

let pedidos = [];
let descuentoAplicado = false;

let form = document.querySelector("#form_cerveza");
let vaciar = document.querySelector("#btn_vaciar");
let formCupon = document.querySelector("#form_cupon");
let total = document.querySelector("#precio_total");

form.addEventListener("submit",agregar);
vaciar.addEventListener("click",vaciarCarrito);
formCupon.addEventListener("submit",aplicar);

//funcion agregar para agregado de formulario
function agregar(e) {
   e.preventDefault(); //previene evento tipo submit
   let formData = new FormData(form);
   let cerveza = formData.get("cerveza");
   let cantidad = formData.get("cantidad");

   let totalPedidas = calcularCantidadCervezas();
   // lo importante es la validacion de la cantidad


   if (cantidad > 1 && cantidad < 10 && (totalPedidas + cantidad) <= 20) {
       let pedido = {
           'cerveza' : cerveza,
           'cantidad': cantidad,
       }
       pedidos.push(pedido);
       imprimir();
   } else {
        alert("no se puede cargar su pedido");
       //esto esta bien porque no esta especificada la notificacion al usuario
```

```
    }
}

function vaciarCarrito(e) {
    // aqui no es necesario el preventDefault
    let tBody = tablaPedidos.querySelector('#lista_pedido tbody');
    tBody.innerHTML = '';
    total.innerHTML = 0;
    pedidos = [];

}

function aplicar(e) {
    e.preventDefault(); // aqui si por el evento submit
    let formData = new FormData(formCupon);
    if (formData.get('cupon') == 'tudai2022') { // && !aplicadoDescuento
        aplicadoDescuento = true;
    }

    actualizar(); // pueden borrar todo e imprimir de vuelta

}

function calcularCantidadCervezas() {
    let sum = 0;
    for (let index = 0; index < pedidos.length; index++) {
        sum += pedidos[index].cantidad;
    }
    return sum;
}

function actualizar() {
 // imprimen la tabla de alguna manera, primero la vacian
  // CUALQUIER CODIGO PARA IMPRIMR LA TABLA

  let cantidad = calcularCantidadCervezas();
  let total = cantidad * 400;
  if (descuentoAplicado && total >= 200)
      total -= 200;
 total.innerHTML = total;
}
```

6. AJAX

Un servicio API REST tiene almacenado los alumnos censados en la universidad, con una url **http://unicenso/api/censados**

Al consultar la url de dicha API se obtiene una respuesta como la siguiente: [{ “id”: 1, "nombre": "Juan Garcia", "dni": 34671234 }, {	1. Escriba el código para agregar al servicio REST un dato fijo: "nombre": "Martina Lopez", " dni": 39512314 2. Escriba el código para consultar el servicio REST y mostrar por consola el dni de las personas censadas y la cantidad de personas que fueron censadas.

```
    “id”: 2,
     "nombre": María Gomez",
     "dni": 45123432
  },
  ...
]
```

Ayuda Fetch:

```
fetch( url, {
       "method": …,
       "headers": {"Content-Type": "application/json"},
       "body": …
     })
```

A.

```
let miUrl = "http://unicenso/api/censados";

//opcion 1
async function crearDatoA(){

    let persona = {
        "nombre": "Martina Lopez",
        "dni": 39512314
    };

    try {
        let res = await fetch(miUrl, {
            "method": "POST",
            "headers": { "Content-type": "application/json" },
            "body": JSON.stringify(persona)
        });
        //opcional
        let json = await res.json();
        console.log(json);
        }
    catch (error) {
    console.log(error);
    }
}

// opcion 2
function crearDatoB(){

    let persona = {
        "nombre": "Martina Lopez",
        "dni": 39512314
    };
```

```
    fetch(miUrl, {
        "method": "POST",
        "headers": { "Content-type": "application/json" },
        "body": JSON.stringify(persona)
    });

}
```

B.

```
//opcion 1
async function mostrarDatosA(){
    try {
        let res = await fetch(miUrl);
        let json = await res.json();
        console.log(json);
        console.log("Cantidad Censados: ", json.length);
        for(item of json){
            console.log(item.dni);
            };
        }
    catch (error) {
    console.log(error);
    }
}

//opcion 2
function mostrarDatosB() {
    fetch(miUrl).then( function(response){
        if (response.ok) {
          console.log("ok");
          console.log(response);
          response.json().then( function(r){
            console.log("Cantidad Censados:" , r.length);
            for(i of r){
                console.log(i.dni);
            }
          });
        }
        else
          console.log("Error en url");
      })
      .catch(function(response){
        console.log(response);
        console.log("Conexion Fallida");
      });
  }
```