

Arquitectura y Diseño de Sistemas Web y C/S

Práctica 3: HTML 5 y CSS3. JavaScript y jQuery.

Sergio Orejón Pérez

Álvaro Navarro Marín

Santiago Hernández Delgado

Contenido

1. Ejercicios con el Lenguaje HTML 5	3
1.1. Aplicación de CSS en una web.....	3
1.2. Comprobación de funcionalidades HTML 5 en navegadores.....	6
1.3. Creación de página con elementos multimedia.....	7
1.4. Creación de página con formulario para su autovalidación	8
1.5. Ejercicio práctico de creación de páginas con otras funcionalidades	10
2. Ejemplos básicos de JavaScript	14
2.1. Abrir una nueva ventana	14
2.2. Mostrar en una ventana un valor del formulario	15
3. Ejercicio JavaScript	16

1. Ejercicios con el Lenguaje HTML 5

1.1. Aplicación de CSS en una web

Para esto hemos tenido que cambiar el formato de algunas de nuestras páginas (no hemos cogido la practica entera ya que la hicimos en html 4.01 y hemos tenido que cambiar al 5).

```
# style.css > ...
1  body{ background-image: url("../imagenes/bandera.jpg");
2  font: 150% sans-serif;
3  background-repeat: no-repeat;
4  background-size: cover;
5  background-position: center center;}
6  table{
7  |   width: 100%; text-align: left; border: 0; }
8  img{height: 600px; width: 700px; align-items: center;}
9  td{vertical-align: top; width: 50;}
10 |
```

Hemos aplicado eso al archivo CSS el cual lo que hace es cambiar la fuente del “body” de la página a Sans-serif y luego pone el fondo de España en todas las páginas.

*NOTA: las páginas las hemos validado todas en html5.

También con esto hemos adaptado la tabla, las imágenes y las columnas a como queremos y estos son los resultados en las páginas.



Captura 1. Entrada.html con CSS

PORTERO



BIOGRAFIA
 Iker Casillas Fernández es un futbolista español que juega como portero. Fue internacional absoluto con la selección española desde 2000 hasta 2016, de la cual fue su capitán desde 2006 hasta 2016. Es el segundo internacional español con más partidos de la historia y el decimoquinto a nivel mundial con 167.

Estadísticas en el Mundial

7 Partidos jugados
2 Goles Encajados
13 Pases
1 Penalti Parado

LATERAL DERECHO

CENTRAL DERECHO

CENTRAL IZQUIERDA

LATERAL IZQUIERDA

Captura 2. Portero.html

PORTERO



BIOGRAFIA
 Sergio Ramos García es un futbolista español que juega como defensa en el Sevilla F. C. de la Primera División de España. Juguó 15 años en el Real Madrid siendo el mejor defensor de su historia.

Estadísticas en el Mundial

7 Partidos jugados
3 Asistencias
58 Recuperaciones
30 Faltas Cometidas

PORTERO


CENTRAL DERECHO

CENTRAL IZQUIERDA

LATERAL IZQUIERDA

Captura 3. Lateral_derecho.html

PORTERO



BIOGRAFIA
 Gerard Piqué Bernabeu es un futbolista y empresario español que jugó como defensa, entre otros equipos, en el Fútbol Club Barcelona, con el cual ganó treinta títulos.

Estadísticas en el Mundial

7 Partidos jugados
0 Gol
1 Amarilla
67 Recuperaciones
9 Faltas Cometidas

PORTERO

LATERAL DERECHO

CENTRAL DERECHO

CENTRAL IZQUIERDA

LATERAL IZQUIERDA

Captura 4. Central_derecho.html

CARRILLO



BIOGRAFIA

Carlos Puyol Saforcada es un exfutbolista español. Jugaba en la posición de defensa y su único equipo durante su trayectoria fue el F. C. Barcelona, de la Primera División de España, del que fue capitán desde la temporada 2004-05.

Trae su retiro al trabajo como parte del Área de Dirección Deportiva de Fútbol del F. C Barcelona

Estatísticas en el Mundial

7 Partidos jugados
1 Gol
2 Asistencias
46 Recuperaciones
6 Fallos Cometidas

PORTERO

LATERAL DERECHO

CARRILLO

Captura 5. Central_izquierdo.html

BIOGRAFÍA

Juan Capdevita Méndez es un exfutbolista español que se desempeñaba como lateral izquierdo. Fue campeón del mundo con la selección española en el año 2010.

Estadísticas en el Mundial

7 Partidos jugados
6 Asistencias
26 Recuperaciones
5 Faltas Cometidas

PORTERO

LATERAL DERECHO

DEFENSA CENTRAL

DEFENSA LATERAL

ATAQUE CENTRAL

ATAQUE LATERAL

Captura 6. Lateral_izquierdo.html

1.2. Comprobación de funcionalidades HTML 5 en navegadores

Las principales diferencias en cuanto a funcionalidades HTML5 en los navegadores más utilizados son:

	Chrome 68	Edge 18	Firefox 60	Opera 45
Elements	27	23	26	25
Forms	64	64	52	64
Web components	10	2	2	10
Video	29	33	29	29
Audio	29	27	27	29
3d and VR	20	18	23	20
Animation	8	5	8	8
Communication	40	35	40	40
Streams	4	4	0	4
Peer to peer	40	33	40	40
User interaction	19	19	18	19
Performance	12	10	12	12
Security	29	29	27	29
Payments	5	5	0	0
Web applications	16	15	17	16
Scripting	30	28	30	27
Total general de puntos	528	496	497	518

Tras la comparativa de estos navegadores, hemos podido comprobar que el buscador Google Chrome es el más completo puesto es el que mayor puntuación ha obtenido mientras que el Microsoft Edge es el menos completo de estos cuatro.

1.3. Creación de página con elementos multimedia



Para este apartado hemos elegido la película “300” y hemos usado un trailer (.mp4), su BSO (.mp3) y el mejor fotograma(.jpg), para poder ver como interactúa html5 con distintos elementos multimedia. Hemos usado la funcionalidad de esta pagina https://www.w3schools.com/html/html_media.asp

```
P3 > 1.3 > <> peliculaFavorita.html > html > head > link
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4    <meta charset="UTF-8">
5    <title>Película Favorita</title>
6    <link rel="stylesheet" type="text/css" href="styles.css">
7  </head>
8  <body>
9
10   <h1>Película Favorita</h1>
11   <h2>Trailer de la Película</h2>
12   <video width="480" height="360" controls>
13     <source src="trailer.mp4" type="video/mp4">
14   </video>
15
16   <h2>Banda Sonora</h2>
17   <audio controls>
18     <source src="bso.mp3" type="audio/mpeg">
19   </audio>
20
21   <h2>Mejor Fotograma</h2>
22   
23
24 </body>
25 </html>
26
```


1.4. Creación de página con formulario para su autovalidación

Compra de Entrada de Cine

Nombre:

Apellidos:

Correo Electrónico:

Fecha de Nacimiento: 


Dirección:

Cantidad de Entradas:

Pago Requerido

Tarjeta de Credito:

Codigo de Seguridad (CVC):

Fecha de Caducidad: 

Pago Completado

¡Gracias por tu compra! El pago se ha completado con éxito.

Recibirás un correo de confirmación en la dirección proporcionada.

Que disfrutes.

Para este apartado, hemos diseñado 3 paginas html, la primera con autovalidación, fechas y correo, al igual que la segunda con números y fechas tmb. La tercera página es meramente decorativa.


```

<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Compra de Entrada de Cine</title>
</head>
<body>
  <h1>Compra de Entrada de Cine</h1>
  <form action="pago.html" method="post">
    <label for="nombre">Nombre:</label>
    <input type="text" id="nombre" name="nombre" required><br>

    <label for="nombre">Apellidos:</label>
    <input type="text" id="apellido" name="apellido" required><br>

    <label for="email">Correo Electrónico:</label>
    <input type="email" id="email" name="email" required><br>

    <label for="fecha_nacimiento">Fecha de Nacimiento:</label>
    <input type="date" id="fecha_nacimiento" name="fecha_nacimiento" required><br>

    <label for="direccion">Dirección:</label>
    <input type="text" id="direccion" name="direccion" required><br>

    <label for="cantidad_entradas">Cantidad de Entradas:</label>
    <input type="number" id="cantidad_entradas" name="cantidad_entradas" min="1" max="99" required><br>

    <input type="submit" value="Comprar Entrada">
  </form>
</body>
</html>

```

```

<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Pago de Entrada de Cine</title>
</head>
<body>
  <h1>Pago Requerido</h1>
  <form action="pagoCompletado.html" method="post">

    <label for="tarjeta_credito">Tarjeta de Credito:</label>
    <input type="text" id="tarjeta_credito" name="tarjeta_credito" pattern="[0-9]{13,16}" title="Por favor, ingrese una tarjeta de crédito válida" required><br>

    <label for="cvc">Codigo de Seguridad (CVC):</label>
    <input type="number" id="cvc" name="cvc" min="0" max="999" required><br>

    <label for="fecha_caducidad">Fecha de Caducidad:</label>
    <input type="date" id="fecha_caducidad" name="fecha_caducidad" required><br>

    <input type="submit" value="Comprar Entrada">
  </form>
</body>
</html>

```

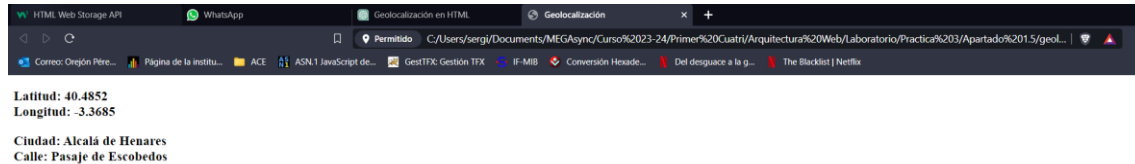
```

<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Pago Completado</title>
</head>
<body>
  <h1>Pago Completado</h1>
  <p>¡Gracias por tu compra! El pago se ha completado con éxito.</p>
  <p>Recibirás un correo de confirmación en la dirección proporcionada.</p>
  <p>Que disfrutes.</p>
</body>
</html>

```

1.5. Ejercicio práctico de creación de páginas con otras funcionalidades

-GEOLOCALIZACIÓN: Para la geolocalización nos hemos basado en lo que ponía en la página de w3schools https://www.w3schools.com/html/html5_geolocation.asp y a partir de ese código le hemos querido añadir una funcionalidad para sacar la ciudad así que nos hemos apoyado en el api nominatim y tenemos de resultado esta página web.



```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Geolocalización</title>
</head>
<body>
  <h3 id="demo"></h3>
  <h3 id="city"></h3>

  <script>
    const x = document.getElementById("demo");
    const cityElement = document.getElementById("city");

    function getlocation() {
      if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(showPosition);
      } else {
        x.innerHTML = "Geolocation is not supported by this browser.";
      }
    }

    function showPosition(position) {
      const latitude = position.coords.latitude;
      const longitude = position.coords.longitude;
      x.innerHTML = "Latitud: " + latitude + "<br>Longitud: " + longitude;

      // USAMOS LA API nominatim para sacar la ciudad a partir de las coordenadas
      const nominatimurl = `https://nominatim.openstreetmap.org/reverse?lat=${latitude}&lon=${longitude}&format=json`;

      fetch(nominatimurl)
        .then(response => response.json())
        .then(data => {
          const city = data.address.city || data.address.town || data.address.village || data.address.suburb;
          const street = data.address.road;
          cityElement.innerHTML = "Ciudad: " + city + "<br>Calle: " + street;
        })
        .catch(error => {
          cityElement.innerHTML = "La información de la ciudad no está disponible";
        });
    }

    getlocation();
  </script>
</body>
</html>
```

Captura X. geolocalización.html

- **LOCALSTORAGE:** Para demostrar el guardado de datos hemos usado la función `localStorage` la cual esta explicada en la página de: https://www.w3schools.com/html/html5_webstorage.asp
`localStorage` guarda los datos hasta después de cerrar el navegador así que lo tenemos todo el tiempo los datos.

Página de Guardado de Datos

INGRESE NOMBRE Y APELLIDO

EL HISTORIAL DE DATOS ES:

hola

XABIER ALONSO

que tal

que tal

Cristiano Ronaldo

que tal

Messi

que tal

que tal

Sergio Orejon

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Guardado de Datos</title>
</head>
<body>
  <h1>Página de Guardado de Datos</h1>
  <form id="myForm">
    <label for="inputData">INGRESE NOMBRE Y APELLIDO</label>
    <input type="text" id="inputData">
    <button type="submit">Guardar</button>
  </form>
  <p id="result"></p>
  <p id="historial"></p>
  <script>
    // Escuchar el evento 'submit' del formulario
    document.getElementById("myForm").addEventListener("submit", function (event) {
      event.preventDefault(); // Evitar que el formulario se envíe normalmente

      // Obtener el valor del input
      const inputData = document.getElementById("inputData").value;

      // Generar una clave única basada en la marca de tiempo
      const timestamp = new Date().getTime();
      const key = `userInput_${timestamp}`;

      // Guardar el valor en localStorage con la clave única
      localStorage.setItem(key, inputData);

      // Actualizar el contenido de un elemento en la página
      document.getElementById("result").innerHTML = "Dato guardado en localStorage: " + inputData;
      displayHistory();
    });

    function displayHistory() {
      const historyList = document.getElementById("historial");
      historyList.innerHTML = "EL HISTORIAL DE DATOS ES: ";

      for (let i = 0; i < localStorage.length; i++) {
        const key = localStorage.key(i);
        const value = localStorage.getItem(key);
        historyList.innerHTML += `<p>${value}</p>`;
      }
    }

    // Mostrar el historial al cargar la página
    displayHistory();
  </script>
</body>
</html>
```

Captura X. guardadoWeb.html

- **FuncionesPopUp:** Esta funcionalidad me ha costado un poco encontrarla porque hay realmente pocas funcionalidades para páginas simples, por lo que es una simple pregunta con un onClick que luego la respuesta de esta, aparece en otro onClick más abajo.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Presentación</title>

    <script>
      let nombre;

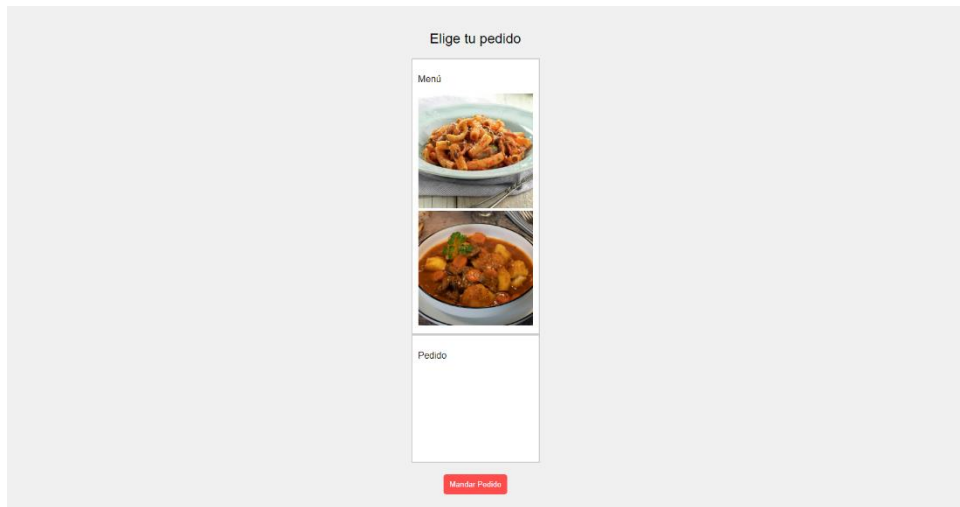
      function askName() {
        nombre = prompt('Dime tu nombre');
      }

      function greet() {
        let message = 'Hola ' + nombre + ', un placer conocerte!';
        alert(message);
      }
    </script>
  </head>
  <body>
    <h1>ENCANTADO!</h1>
    <p onclick="askName();">
      Pulsa aquí para escribir tu nombre.
    </p>

    <p onclick="greet();">
      Pulsa aquí para recibir un saludo.
    </p>
  </body>
</html>
```

Captura x. funciónPopUp.html

- **DRAG AND DROP:** Siguiendo las practicas de WEB hemos hecho un ejemplo de drag and drop basado en un sitio para pedir comida. Te hace elegir entre dos platos, los cuales arrastras, y mandas el pedido.



```
<script>
    function allowDrop(event) {
        event.preventDefault();
    }

    function drag(event) {
        event.dataTransfer.setData("text", event.target.id);
    }

    function drop(event) {
        event.preventDefault();
        var data = event.dataTransfer.getData("text");
        var draggedElement = document.getElementById(data);
        var copy = draggedElement.cloneNode(true);
        copy.removeAttribute("draggable");
        event.target.appendChild(copy);
    }

    function vaciarPedido() {
        var pedidoContainer = document.getElementById("pedido");
        pedidoContainer.innerHTML = "<p>Pedido</p>";
    }
</script>
</head>
<body>
    <div id="titulo">Elige tu pedido</div>

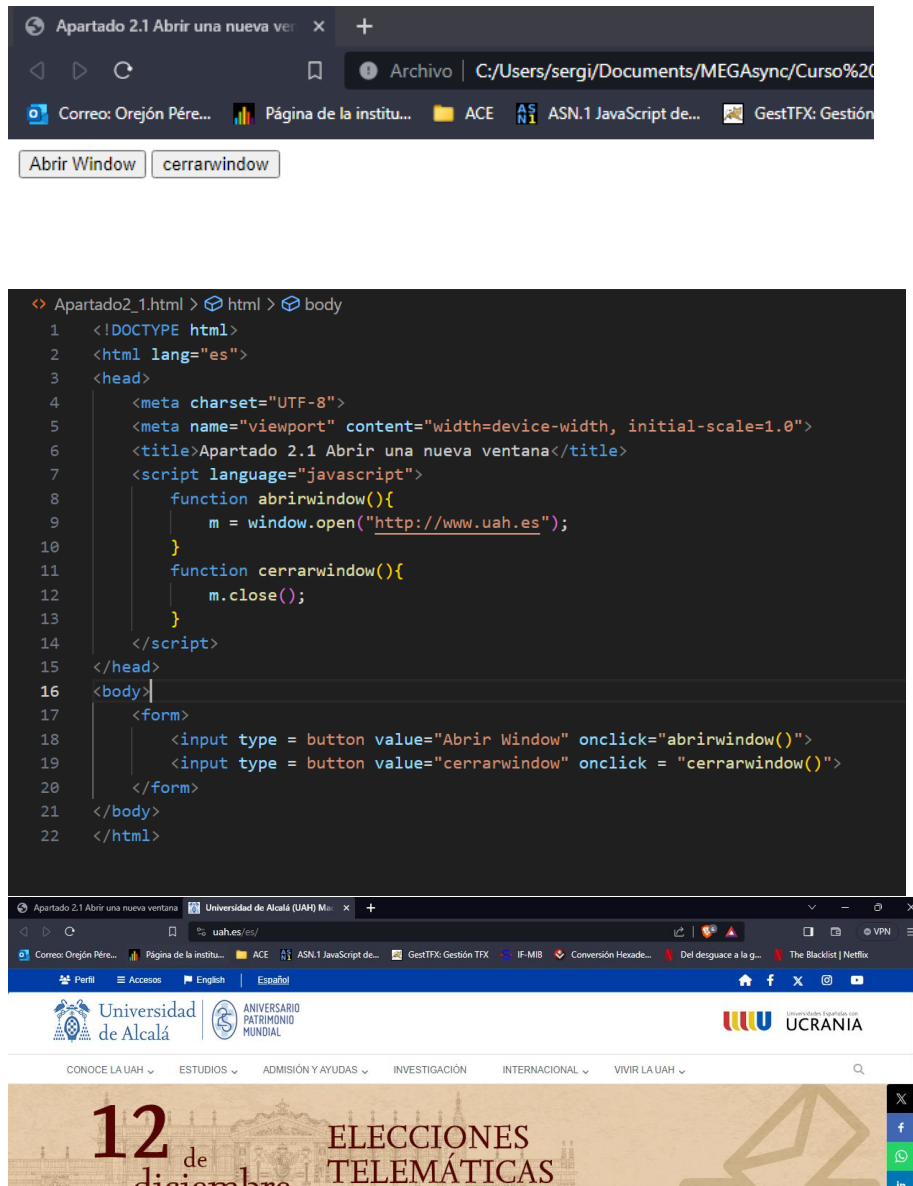
    <div id="menu" ondrop="drop(event)" ondragover="allowDrop(event)">
        <p>Menú</p>
        
        
    </div>

    <div id="pedido" ondrop="drop(event)" ondragover="allowDrop(event)">
        <p>Pedido</p>
    </div>

    <button id="vaciarPedido" onclick="vaciarPedido()">Mandar Pedido</button>
</body>
</html>
```

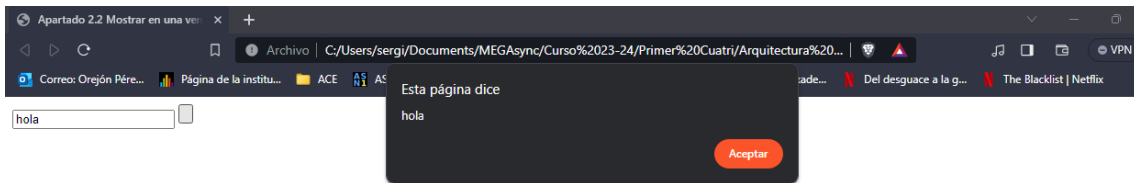
2. Ejemplos básicos de JavaScript

2.1. Abrir una nueva ventana



Captura X. Apartado2_1.html

2.2. Mostrar en una ventana un valor del formulario



```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Apartado 2.2 Mostrar en una ventana un valor del formulario</title>
7   <script language = "javascript">
8     function ventana(txt){
9       alert(txt);
10    }
11  </script>
12 </head>
13 <body>
14   <form name = "form1">
15     <input name = "entrada">
16     <input type = "button" onclick="ventana(form1.entrada.value)">
17   </form>
18
19 </body>
20 </html>
```

Captura X. Apartado2_2.html

3. Ejercicio JavaScript

Recomendamos probar para asegurarse de que es la funcionalidad requerida en el guión.

Primero de todo adjuntamos una captura del inicio del archivo html que hemos creado para el restaurante:

☒ Primeros platos
☐ Segundos platos
☐ Postres

Mesa 1 ▾

Puré de patata
Macarrones con queso
Garbanzos chorizo
Judías Verdes

☐ Café
☐ Copa
TOTAL:

Captura X. Apartado3.html

Explicación de las funcionalidades implementadas:

-Lista de las comidas ofrecidas por el restaurante:

- ☒ Primeros platos
☐ Segundos platos
☐ Postres

```
<div id="platos" star>
  <input type="radio" name="tipoPlato" value="primeros" checked> Primeros platos<br>
  <input type="radio" name="tipoPlato" value="segundos"> Segundos platos<br>
  <input type="radio" name="tipoPlato" value="postres"> Postres
</div>
```

Estos botones tipo radio los hemos implementado así y hemos dejado marcado por defecto el de los primeros platos. Y para programar que salga el tipo de plato que queremos por pantalla según el tipo seleccionado hemos implementado estas funciones:

```
// Escuchar el cambio en los radios de tipo de plato
$('input[name=tipoPlato]').on('change', function() {
  tipoPlatoSeleccionado = $('input[name=tipoPlato]:checked').val();
  actualizarCajaIzquierda();
});

// Función de ejemplo para obtener elementos según el tipo de plato
function obtenerElementosSegunTipoPlato(tipoPlato) {
  // Puedes personalizar esta lógica según tus necesidades
  if (tipoPlato === 'primeros') {
    return ["Puré de patata", "Macarrones con queso", "Garbanzos chorizo", "Judías Verdes", "Sopa de pescado"];
  } else if (tipoPlato === 'segundos') {
    return ["Merluza a la plancha", "Filete de ternera", "Bistec de Vaca", "Entrecot"];
  } else if (tipoPlato === 'postres') {
    return postres = ["Natillas", "Trufas", "Flanes", "Tarta de queso"];
  } else {
    return [];
  }
}
```



```
function actualizarCajaIzquierda() {
    // Lógica para obtener los elementos según el tipo de plato seleccionado
    var nuevosElementos = obtenerElementosSegunTipoPlato(tipoPlatoSeleccionado);
    // Limpiar la caja de la izquierda
    $('#caja').val('');
    // Agregar los nuevos elementos a la caja de la izquierda
    for (var i = 0; i < nuevosElementos.length; i++) {
        $('#caja').val($('#caja').val() + nuevosElementos[i] + '\n');
    }
}
```

Con la función de `$('#input[name=tipoPlato]').on('change',function(){})` hemos hecho que cambien los platos según si cambia el radio button a uno de los otros dos valores. Finalmente con `actualizarCajalIzquierda()`; actualizamos la textarea con los platos que ofrece el restaurante.

-Pasar platos elegidos con doble click a la caja de la derecha:

```
$('#caja').on('dblclick', function() {
    // Obtener todo el contenido de la línea en lugar de la palabra seleccionada
    var selectedLine = getSelectedLine(this);
    var clonedElement = $('#<textarea class="caja">' + selectedLine + '</textarea>');
    // Mover el elemento a la caja paralela
    $('#cajaparela').append(clonedElement.val() + '\n');
    actualizarCuenta(selectedLine);
});

function getSelectedLine(textarea) {
    // Obtener el contenido de la línea completa
    var lines = textarea.value.split('\n');
    var cursorPos = textarea.selectionStart;
    var lineStart = 0;
    var lineEnd = 0;
    // Encontrar el inicio y el final de la línea actual
    for (var i = 0; i < lines.length; i++) {
        lineEnd = lineStart + lines[i].length;
        if (lineEnd >= cursorPos) {
            break;
        }
        lineStart = lineEnd + 1; // +1 para saltar el carácter de nueva línea
    }
    // Devolver la línea completa
    return lines[i].trim();
}
```

Hemos utilizado la función `getSelectedLine(textarea){}` para que nos seleccione una línea entera cada vez que hacemos dobleclick en la izquierda y luego con la función `$('#cajaparela').append(clonedElement.val() + '\n');` los vamos añadiendo con un enter en el textarea de la derecha. Finalmente la función `actualizarCuenta(selectedLine)` lo que hace es extraer el ítem que hemos añadido y añadirle el precio a la cuenta de la mesa.

-Separación de las mesas:

```
$('#mesaSelect').on('change', function () {
    // Almacenar el contenido de la caja paralela en la mesa anterior
    cajasParalelas[mesaSeleccionada] = $('#cajaparela').val();
    cafeEstado = $('#cafe').prop('checked');
    copaEstado = $('#copa').prop('checked');

    // Cambiar la mesa seleccionada
    mesaSeleccionada = $('#mesaSelect').val();
    // Verificar si ya existe la caja paralela para esta mesa
    if (!cajasParalelas[mesaSeleccionada]) {
        $('#cafe').prop('checked', false);
        $('#copa').prop('checked', false);
        // Si no existe, crear una nueva cajaparela
        cajasParalelas[mesaSeleccionada] = $('#<textarea id="cajaparela"></textarea>');
        // Reemplazar la caja paralela actual con la nueva
        $('#cajaparela').replaceWith(cajasParalelas[mesaSeleccionada]);
    } else {
        // Si ya existe, simplemente actualizar la caja paralela con el contenido almacenado
        $('#cajaparela').val(cajasParalelas[mesaSeleccionada] || '');
    }
    restaurarEstadoMesa();
    $('#total').val(cuentas[mesaSeleccionada]);
});
```

Lo que hemos hecho para separar las mesas unas de otras y separar las cuentas es crear dos arrays, uno de cajas para mostrar el texto dependiendo de la mesa con índice la mesa y otro array con las cuentas con el índice la mesa.

También hemos añadido el código explicado abajo para conservar el estado del checkbox de café y de copa.

-Calculo de la cuenta:

```
const precios = {
    "Puré de patata":5,
    "Macarrones con queso":6,
    "Garbanzos chorizo":7,
    "Judías Verdes":5,
    "Sopa de pescado":5,
    "Merluza a la plancha":7,
    "Filete de ternera":9,
    "Bistec de Vaca":10,
    "Entrecot":15,
    "Natillas":4,
    "Trufas":5,
    "Flanes":5,
    "Tarta de queso":6,
    "cafe":4,
    "copa":6,
    "nocafe":-4,
    "nocopa":-6
}

//ESTO FUNCIONA
function actualizarCuenta(plato){
    if(!cuentas[mesaSeleccionada]){
        cuentas[mesaSeleccionada]=0;
        $('#total').val(cuentas[mesaSeleccionada]);
    }
    if(precios.hasOwnProperty(plato)){
        cuentas[mesaSeleccionada] += precios[plato];
        $('#total').val(cuentas[mesaSeleccionada]);
    }
}
```

Hemos creado un diccionario con los precios y lo que hemos ido haciendo es ir comprobando el item que íbamos metiendo en la caja de la derecha e irlo añadiendo a la cuenta y con esto podemos sacar cálculos exactos de las cuentas de las mesas

-Café y Copa:

```
// Restaurar el estado de café y copa según la mesa
function restaurarEstadoMesa() {
    cafeEstado = cafeEstados[mesaSeleccionada];
    copaEstado = copaEstados[mesaSeleccionada];
    console.info(cafeEstado + "CAFE ESTADO" + copaEstado + "COPA ESTADO");
    // Restaurar el estado de café y copa de la mesa actual
    $('#cafe').prop('checked', cafeEstado);
    $('#copa').prop('checked', copaEstado);
}

$('#cafe').on('change', function(){
    // Guardar el estado de café de la mesa actual
    cafeEstados[mesaSeleccionada] = $(this).prop('checked');
    actualizarCuenta(cafeEstados[mesaSeleccionada] ? cafe : noCafe);
})

$('#copa').on('change', function(){
    copaEstados[mesaSeleccionada] = $(this).prop('checked');
    actualizarCuenta(copaEstados[mesaSeleccionada] ? copa : noCopa);
})
```

Al ser dos checkbox comprobamos si están marcados o no y si lo están añadimos el valor de café/copa y si por algún motivo lo desmarcamos después de marcarlo quitamos el valor de lo que habíamos añadido.

Hemos creado un array con CafeEstados y CopaEstados para guardar el estado anterior del café y que se queden marcados o desmarcados los checkbox según estuvieran en la mesa antes de cambiar de mesa.

-Pagado:

```
function pagar() {
    // Establecer el valor de cuentas[mesaSeleccionada] a 0
    cuentas[mesaSeleccionada] = 0;
    // Limpiar la caja paralela de productos de esa mesaSeleccionada
    cajasParalelas[mesaSeleccionada]=null;
    $('#cajaparela').val('');
    $('#total').val('0');
    reiniciarEstadosCheckboxes();
    // Verificar si ya existe la caja paralela para esta mesa
    if (!cajasParalelas[mesaSeleccionada]) {
        // Si no existe, crear una nueva cajaparela
        cajasParalelas[mesaSeleccionada] = $('<textarea id="cajaparela"></textarea>');
        // Reemplazar la caja paralela actual con la nueva
        $('#cajaparela').replaceWith(cajasParalelas[mesaSeleccionada]);
    } else {
        // Si ya existe, simplemente actualizar la caja paralela con el contenido almacenado
        $('#cajaparela').val(cajasParalelas[mesaSeleccionada] || '');
    }
}

$('#pagado').on('click',function(){
    pagar();
})
```

```
function reiniciarEstadosCheckboxes() {  
  cafeEstados[mesaSeleccionada] = false;  
  copaEstados[mesaSeleccionada] = false;  
  $('#cafe').prop('checked', false);  
  $('#copa').prop('checked', false);  
}
```

Con pagar lo que hacemos es establecer la posición del array de esa mesa a 0 y vaciamos la caja y desmarcamos ambas checkbox si las habíamos marcado.