Universitat de Barcelona

Computació Orientada al Web

Práctica 4

Arthur Font Gouveia - 20222613

Barcelona

2022

Índice

1. Introducción

El objetivo de esta práctica fue añadir botones, funcionalidades de eventos y sus handlers e implementar interacciones con los elementos del DOM usando jQuery y JavaScript, implementar la comunicación cliente-servidor utilizando Ajax y jQuery y por fin, añadir efectos visuales usando jQueryUI, basándose en la página web desarrollada en la práctica anterior. Las nuevas funcionalidades fueron codificadas manualmente usando un editor de texto.

2. Apartado 1. Añadir las funcionalidades al cliente utilizando Java Script, jQuery y jQwidgets

En este apartado, he implementado las funcionalidades al cliente, simulando diversas formas para realizar una búsqueda de hoteles. Las implementaciones están en el fichero **page_load.js**.

El fichero page_load.js espera a que el DOM se cargue para ejecutarse mediante la función **ready()** y relaciona el evento del envío del formulario mediante la función **on(**"click"**)** y la función handler formSubmit(), creada para hacer el control de los datos introducidos por parte del cliente. En caso de que algún dato no pase en el control, el envío del formulario se para mediante la llamada a la función **event.preventDefault()**. En caso de que el formulario sea completado correctamente, es realizada una requisición AJAX para el envío del formulario. Para obtener el valor de de un elemento del formulario se usa el símbolo **$** y la función **val()**. A continuación están ilustradas las funcionalidades del código más relevantes:

```
jQuery(document).ready(function($){
   $("#search_button").on("click", formSubmit);
   …
});
```

El fichero page_load.js también es responsable por realizar interacciones y efectos visuales con elementos del DOM mediante las funciones **show()** y **effect()**, que definen el efecto en el primer parámetro.

```
// jQuery effects
$("#effect_phrase").show("bounce", 1500);
$("#center_header").delay(1000).show("scale", 500);
```

```
$("#env_header").delay(500).effect("shake", 1000); // effect after DOM object had already showed
$("#banner_img").delay(1000).show("bounce", 1000);
```

Además, fue implementada la funcionalidad de sugestiones para autocompletar durante la búsqueda de la ciudad de destino mediante la función **autocomplete()** y los widgets **datepicker()** y **spinner()** de jQWidget, que contribuyen para la usabilidad de la página web. Las sugestiones són añadidas después del elemento “destination” mediante la función **after()**. A continuación están ilustradas las funcionalidades del código más relevantes:

```
// Initiate jQuery date pickers and spinner
   $("#checkin").datepicker({
         showAnim: "slideDown", // default: show
   });
   $("#checkout").datepicker({
         showAnim: "slideDown", // default: show
   });
   $("#guests").spinner({
      min: 1,
      max: 20
   });

// Search destinations jQuery autocomplete
   $("#destination").autocomplete({
      source: function(request, response) {
         $.get('gethint.php', { term: request.term }, function(data) {
            $("#destination").after(data);
         });
      },
      minLength: 1
   });
```

3. Apartado 2. Implementar una transferencia de datos desde el servidor hacia el cliente usando JSON

En este apartado se ha modificado la funcionalidad de la búsqueda para que se utilice la comunicación AJAX usando JSON como el protocolo de datos. Para eso, se ha modificado la función formSubmit() en el parámetro de la requisición de ajax utilizando la función **JSON.stringify()**, que convierte un objeto JavaScript a un formato de cadena de texto JSON. En la función que controla la respuesta con suceso del servidor, los datos recibidos en formato de cadena de texto JSON son convertidos a

objeto JavaScript mediante la función **JSON.parse()**. Una vez recibida la respuesta del servidor, se crea una tabla que enseña los resultados de manera dinámica en el DOM.

```
$.ajax ({
      url: 'search_results.php',
      method: 'POST',
      data: { json: JSON.stringify({
         "destination": destination, "checkin": checkin,
         "checkout": checkout, "guests": guests
      })},
      success: function(data, textStatus) {
         //console.log(data);
         if (data.length > 0) {
            // Create a JSON object using JSON.parse
            var result = JSON.parse(data);
            …
      },
      error: function(error) {
         html += error + '</div>';
      }
});
```

Por parte del servidor, cabe destacar las funciones json_decode y json_encode, responsables por convertir una cadena de texto JSON en un objeto PHP y por convertir un objeto PHP en una cadena de texto JSON, respectivamente.

4. Apartado 3. Implementar una transferencia de datos desde el servidor hacia el cliente usando XML

En este apartado se ha modificado la funcionalidad del autocomplete para que se utilice la comunicación AJAX usando XML como protocolo de datos. Para eso, se ha modificado la función autocomplete(), que ahora recibe el documento xml como respuesta de la requisición GET. Para enseñar las destinaciones durante la búsqueda se usan los método **find()** y **after()**, para seleccionar un vector de destinaciones y añadir las sugestiones al DOM de forma dinámica, respectivamente.

```
$("#destination").autocomplete({
      source: function(request, response) {
         $.get('gethint.php', { term: request.term }, function(xml) {
            var names = $(xml).find("name");
            if (names.length > 0) {
               var html = '<datalist id="destlist">';
```

```
                    for (var i = 0; i < names.length; i++) {
                       html += `<option value="${$(names[i]).text()}"></option>`;
                    }
                    $("#destination").after(html);
                }
            });
        },
        minLength: 1
    });
```

Por parte del servidor, cabe destacar las funciones **new DOMDocument()**, **createElement()**, **appendChild()** y **saveXML()**, responsables por crear un nuevo documento DOM, crear un nuevo elemento, adicionar um elemento hijo a un elemento y producir una cadena de texto que contiene el XML, respectivamente.