JEGYZŐKÖNYV

Web technológia alapjai

Peugeot bemutató oldal

Készítette: Borbély Ádám

Neptunkód: R41N5H

Dátum: 2024. május

Tartalomjegyzék

Feladat leírás:

Peugeot bemutatóoldalt készítettem, az alábbi szempontoknak megfelelően:

HTML

 Legalább öt HTML fájl legyen.

 HTML5 szerkezet megtervezése (kép - oldal tetején, menü stb).

 HTML elemek: div, span, p, címsorok, képek, táblázat, linkek, html5 elemei stb.

 Űrlap elemek: szöveges beviteli mező: egy, több soros, adatlista, jelölőnégyzet, rádiógomb, színválasztó, dátumválasztás, gombok, számozott lista.

 Video: az adott témakörhöz tartozó rövid video elkészítése, egyes gombok vezérlése JavaScript kóddal.

CSS

 style attribútumban néhány elem formázása,

 azonosító alapján formázást,

 osztály alapján formázást,

 táblázat formázása,

 menü kialakítása,

 háttérszín,

 linkek formázása,

 űrlap elemek, gombok formázása,

 egy vagy több .CSS fájlban is legyen formázás, és a html-ben a <style> tag-ben is.

JavaScript, jQuery, JSON, AJAX

 form ellenőrzés, hiba esetén az egyes űrlap elemeknél kiírja, hogy mi a hiba, css-t állít pl. piros keret, ha hibás az adott űrlap elem,

 jQuery animáció,

 elemek kiválasztása: html tag név alapján, osztály alapján, azonosító (id) alapján stb.

 új html elem készítése, meglévő html elem módosítása.

 egy fájl elkészítése JSON formátumba (beágyazott).

 JavaScript, jQuery – AJAX metódus használatával JSON fájl megjelenítése a weboldalon.

stílus megvalósítása

css-ben

Először is egy alapot adtam a style.css fájlban, főként a headerek és footerek alapjai kerültek megvalósításra, ideértve a headerben megjelenő Peugeot logót is. Ezen kívül a hátteret is formáztam meg, valamint a body szerkezetet a szövegnek. Található itt section, article formázás is, és a navigációk, és a navigált felsorolásoknál, gomboknál is. pl:

```
nav ul li a {
    color: #fff;
    text-decoration: none;
    padding: 8px 12px; /* Kisebb belső tér a gombok között */
    border-radius: 5px; /* Kerekített szélek */
    transition: background-color 0.3s; /* Animáció a hover effekthez */
}
```

html kódokban

Az alapvető stílusokon túl a html kódokban is találhatóak formázások, mind gombokra, mind szövegre, mind táblázatra, videóra, valamint a meglévők felülírása is esetenként.

Ezen kívül külső forrást is használtam, például a modellek előtti ikonokhoz:

```
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css"> <!-- Font Awesome ikonok stílusainak
importálása -->
```

Ezeket pedig így tudtam importálni, a fas fa class-al választottam ki a számomra megfelelőt.

```
<i class="fas fa-motorcycle">
```

Táblázat

A táblázatom a lehető legegyszerűbb módon oldottam meg, a feladatomban nem kapott sok helyet a táblázat, inkább másra koncentráltam, nem tartottam fontosnak, hogy több táblázattal dolgozzak:

```
<div class="container">
    <table>
        <thead>
            <tr>
                <th>Szolgáltatás</th>
```

```
                <th>Leírás</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>Autó értékesítés</td>
                <td>Új és használt autók értékesítése, széles
választékban.</td>
            </tr>
            <tr>
                <td>Autó kölcsönzés</td>
                <td>Rövid és hosszú távú autókölcsönzés, kedvező árakon.</td>
            </tr>
            <tr>
                <td>Autó szerviz</td>
                <td>Teljes körű autószerviz szolgáltatások, gyors és
megbízható munkával.</td>
            </tr>
        </tbody>
    </table>
</div>
```

Json használata

Json fájl letöltése jQuery-vel

Létrehoztam a models.json fájlt, ahova manuálisan felvittem az autómodellek adatait.

Ennek a fájlnak a letöltésére lehetőséget adtam a modellek oldalon egy gombnyomással, amit egy függvény segítségével oldottam meg:

```
$(document).ready(function() {
    // Az exportálás gombra kattintáskor futtatott függvény
    $("#exportButton").click(function() {
        // Gyűjtsük össze az összes modell adatait
        var models = [];
        $("section").each(function() {
            var model = {};
            model.name = $(this).find("h3").text();
            model.details = [];
            $(this).find(".model-info p").each(function() {
                model.details.push($(this).text());
            });
            models.push(model);
        });

        // Konvertáljuk a modell adatokat JSON formátumba
        var jsonData = JSON.stringify(models, null, 2);
```

```
        // Exportáljuk a JSON adatokat fájlba
        var blob = new Blob([jsonData], {type: "application/json"});
        var url = URL.createObjectURL(blob);
        var a = document.createElement('a');
        a.href = url;
        a.download = 'models.json';
        document.body.appendChild(a);
        a.click();
        setTimeout(function() {
            document.body.removeChild(a);
            window.URL.revokeObjectURL(url);
        }, 0);
    });
});
```

JavaScript-nek a szkriptjét használtam, mely használja a jQuery-t. Először is csak akkor tudja letölteni, ha már az egész HTML betöltődött, erre szolgál a ready függvény. az exportbutton click funkciójával kezdeményezem a letöltését, az alapján amit az each függvényem összegyűjt a megadott paraméterek alapján. A 'JSON.stringify'-vel konvertálja át a formátumot. Letöltéskor egy URL-t hozok létre az 'URL.createObjectURL' segítségével, átnevezi a megadott nevemre, majd a ' window.URL.revokeObjectURL' metódussal törlöm az URL, az erőforrások feleseges használatát mellőzve.

Json betöltése AJAX segítségével

```
$(document).ready(function() {
    // Megjelenítjük a felugró ablakot, ha a felhasználó rákattint a gombra
    $('#popupButton').click(function() {
        $('#myModal').css('display', 'block');
        // AJAX kérés az adatok betöltésére
        $.ajax({
            url: 'models.json', // Az AJAX kérés URL-je, itt állítsd be a JSON
fájl elérési útvonalát
            dataType: 'json', // Várt adattípus: JSON
            success: function(data) { // Sikeres válasz esetén
                // Feltöltjük a JSON tartalmát a megfelelő HTML elembe
                var content = '<ul>';
                $.each(data, function(index, model) {
                    content += '<li>' + model.name + ': ';
                    if (model.details.length > 0) {
                        content += '<ul>';
                        $.each(model.details, function(i, detail) {
                            content += '<li>' + detail + '</li>';
                        });
                        content += '</ul>';
```

```
                    } else {
                        content += 'Nincsenek részletek';
                    }
                    content += '</li>';
                });
                content += '</ul>';
                $('#popupContent').html(content);
            }
            ,
            error: function(xhr, status, error) { // Hiba esetén
                console.error(status + ': ' + error); // Hibaüzenet kiírása a
konzolra
                $('#popupContent').html('<p>Hiba történt a JSON
betöltésekor.</p>'); // Hibajelzés megjelenítése
            }
        });
    });
```

Szintén a jQuery használatával egy függvényt hoztam létre az előzőhöz hasonlóan, mely szintén a HTML teljes betöltése után kezd dolgozni. A popup buttonnal hajtódik végre a kód belseje. Mymodal-val jelenítek meg egy modális ablakot, ahol megjelenítem az előre megírt json fájl tartalmát. Az AJAX-val adjuk meg az url-t amivel betölti a models.json fájlt json formátumban. Itt ha lefut a függvény egyszerűen kiírja a kívánt szöveget, más esetben hibaüzenetet dob fel. Alapvetően hibaüzenetet dob fel, LiveServer-vel kell megnyitni, hogy ez a funkció működni tudjon, más esetben csak a hibaüzenetet láthatjuk.

Űrlap

Az űrlapot az autórendelés oldalon valósítottam meg. Az adatokat listáztam, valamint inputokat adtam meg választási lehetőségnek. Egyszerű választási lehetőségek állnak rendelkezésre, melyeknél van lehetőség a törlésre, egy alaphelyzetbe állításra.

```
document.getElementById('carRequestForm').addEventListener('submit',
function(event) {
        event.preventDefault(); // Az alapértelmezett űrlap elküldésének
megakadályozása

        // Űrlapadatok lekérése
        var formData = new FormData(this);

        // Autó kérés elküldése (itt lehetne az AJAX kérés)
        console.log("Autó kérés elküldve:");
        for (var pair of formData.entries()) {
            console.log(pair[0] + ': ' + pair[1]);
        }
        alert("Autó rendelés elküldve:");
    });
```

Ezzel megakadályozza, hogy üresen küldjék be az űrlapot, valamint az elküldött adatokat kiírja console-ra, ezt a megoldást találtam a legjobb megoldásnak, mivel nem használatos oldal, így elég ide küldenie.

Videó

A videót csak letöltöttem az internetről, beillesztettem és 3 gombot hoztam létre, amivel el lehet indítani/megállítani, valamint előre és hátra tekerni 10 másodperccel.

```
<div class="video-container">
    <video id="video-player" width="1024" height="768" controls autoplay=1>
        <source src="Peugeot.mp4" type="video/mp4">
        Your browser does not support the video tag.
    </video>
    <div id="video-controls">
        <button class="video-controls-button"
onclick="playPause()">Lejátszás/Megállítás</button>
        <button class="video-controls-button" onclick="skip(10)">Előre 10
másodperc</button>
        <button class="video-controls-button" onclick="skip(-10)">Vissza 10
másodperc</button>
    </div>
</div>
```

Valamint ennek az egyszerű megoldására function-t írtam az alábbi módon:

```
 function playPause() {
        if (video.paused) {
            video.play();
        } else {
            video.pause();
        }
    }

    function skip(time) {
        video.currentTime += time;
    }

    $(document).ready(function() {
        // Elem animálása osztály alapján
        $(".box").click(function() {
            $(this).animate({opacity: 0.5}, 1000); // 1000 ms (1 másodperc)
alatt animálja az átlátszóságot
        });
    });
```