1. Feladat - HTML+CSS

Leírás

Készítsen el az alábbi leírásnak megfelelően, HTML és CSS segítségével egy egyszerű weboldalt, ügyeljen arra, hogy a feladatokhoz tartozó tesztesetek lefussanak, valamint a tiszta kód elveinek használatára is törekedjen a kód készítése során. A formázásokat, csak és kizárólag INLINE CSS segítségével hajtsa végre, amennyiben a feladat másképp nem kéri!

A weboldal alap HTML-jét kell folytatni a megadott feladat szerint.

Feladatok

1. Az oldalon hozzon létre egy H1-es node-ot, aminek a tartalma legyen az, hogy „Főoldal”.
    a. Állítsa be inline CSS-ben, hogy a betűköz ezen a node-on 3 pixeles legyen.
    b. Állítsa be inline CSS-ben, hogy a szöveg legyen aláhúzott.
2. Hozzon létre egy szekció konténer elemet a H1-es node alá (az azonosítója legyen „container”)
    a. A szélessége ennek a konténer elemnek legyen 1200 pixel
    b. A minimum magassága legyen 200 pixel
    c. A háttérszíne legyen EFEFEF színkódú
3. A konténer elemen belül hozzon létre két paragrafust, melynek tartalma legyen 1-1 bekezdésnyi Lorem ipsum szöveg.
    a. Az első bekezdés kezdődjön „Lorem ipsum” szöveggel!
    b. A második bekezdésben az egyik szó legyen egy link a Google-re (https://www.google.com)
4. Helyezzen el az első bekezdésben egy képet (*„html.png”*)
    a. A kép címe és helyettesítő szövege is legyen „HTML”
    b. A kép legyen 300 pixel széles

2. Feladat - Bootstrap

Leírás

Készítsen el egy rendelési űrlapot Bootstrap segítségével, ügyeljen arra, hogy a feladatokhoz tartozó tesztesetek lefussanak, valamint a tiszta kód elveinek használatára is törekedjen a kód készítése során. A design formázásokat, csak és kizárólag bootstrap osztályok segítségével hajtsa végre!

A weboldal alap HTML-jét kell folytatni a megadott feladat szerint.

Feladatok

1. Hozzon létre egy konténer elemet section node-dal.
    a. Ez legyen úgy pozícionálva, hogy 6 oszlop szélességet foglaljon el az oldal közepén nagy (large) kijelzőre igazítva.
        i. Ehhez biztosítsa a szerkezetben a szükséges egyéb elemeket is! (Működjön a grid system és középen legyen a 6 oszlop széles konténer közrefogva a megfelelő méretű side elemekkel)
2. Hozzon létre a konténerben egy form elemet aminek a átküldési metodikája legyen POST, a form végpontja pedig legyen egy *„feldolgoz.php”* állomány relatív eléréssel. *(A feldolgoz.php nem fog létezni, de ez a feladat szempontjából nem érdekes!)*
3. A form elemen belül hozzon létre 3 input **szekciót**, amik a következő beviteli mezőknek fogják biztosítani a helyet (ezek még nem a beviteli mezők!):
    a. email
    b. név
    c. rendelés szövege
4. Az előző elemekre egységesen használja a bootstrap form csoportosító osztályát
    a. Ezekben a szekciókban hozza létre az input mezőket a következő azonosítókkal:
        i. email → email
        ii. név → name
        iii. rendelés szövege → text_message
    b. Minden inputhoz tartozzon egy címke is, ami megadja, hogy mit kell ott kitölteni.
    c. Biztosítsa, hogy minden input ugyanezekkel a név attribútumokkal is rendelkezzen.
    d. Biztosítsa, hogy a text_message mező egy többsoros beviteli mező legyen.
5. Legyen a form alján önállóan egy submit gomb, aminek a név attribútuma legyen *„Ok”*.
    a. Ez legyen egy elsődleges gomb Bootstrap szerint
6. Biztosítsa, hogy minden input (beviteli vezérlő) a Bootstrap megfelelő osztályát implementálja (vezérlő, gomb stb.)

3. Feladat – JavaScript és programozási paradigmák

Leírás

Készítse el az alábbi JS függvényeket a leírásnak megfelelően a forrás állományba elhelyezve. A megoldáshoz natív JavaScript-et használjon, nem szükséges egyéb keretrendszer alkalmazása ehhez a feladathoz. Ügyeljen arra, hogy a feladatokhoz tartozó tesztesetek lefussanak, valamint a tiszta kód elveinek használatára is törekedjen a kód készítése során.

A weboldal alap HTML-jét kell folytatni a megadott feladat szerint.

Feladatok

1. Készítsen egy új függvényt, mely a paraméterként megkapott tömb elemei közül kiválogatja azokat, amik párosak és 50-nél nagyobbak és ezeket az elemek tömb szerkezetben ugyancsak visszaadja.
    a. Függvény neve: kivalogatParosOtven
    b. Bemeneti paraméterek:
        i. array → amit ellenőrizni kell (számokkal lesz feltöltve, ezt nem kell ellenőrizni)
    c. Visszatérés:
        i. array → amiben benne vannak azok a számok, amik teljesítették a feltételt
    d. Példa:
        i. kivalogatParosOtven([10, 20, 3, 5, 100, 50, 60, 53]) → [100, 60]
2. Készítsen függvényt, mely beolvassa az első div node-ból ami a body-ban található annak belső szöveges tartalmát és megadja, hogy hány „a” betű található abban a szövegben. Biztos lehet benne, hogy a div node létezik, ezt nem kell külön tesztelni.
    a. Függvény neve: aBetukSzamaDiv
    b. Bemeneti paraméterek:
        i. nincs bemeneti paraméter
    c. Visszatérés:
        i. darabszám → az „a” betűk száma az első DIV-ben
    d. Példa:
        i. **a**gi21bvg8ibvg**a**vgvt3zv8gfz**a**vgzuv9gzu**a**vgzu55vgz
            1. aBetukSzamaDiv() → 4
3. Készítsen függvényt, mely egy bemeneti paraméterként megadott osztály alapján megtalálja az első elemet a DOM fában és ebben létrehoz egy új paragrafust, melynek tartalma az átadott másik két szám összege és a matematikai felírásuk (x + y = osszeg). Az első megtalálható node elembe hozza létre a paragrafust. Figyeljen rá oda, hogy a bemeneten megjelenő két szám paraméter nem biztos, hogy szám típusként is érkezik meg!
    a. Függvény neve: osszeadasFeliras
    b. Bemeneti paraméterek:
        i. osztály (class) megnevezés → ebbe kell majd rakni az eredményt
        ii. első operandus → az első szám (biztos, hogy szám érték, de nem biztos, hogy Number type)
        iii. második operandus → a második szám (biztos, hogy szám érték, de nem biztos, hogy Number type)
    c. Visszatérés
        i. nincs visszatérés
    d. Példa:
        i. osszeadasFeliras(”eredmeny”, 2, 3)
        ii. Eredmény:
            1.
               2 + 3 = 5

4. Feladat – TypeScript

Leírás

Készítse el az alábbi TS függvényeket a leírásnak megfelelően. A megoldáshoz TypeScript-et, vagy arra épülő keretrendszert használjon (nem szükséges, de megengedett). Ügyeljen arra, hogy a feladatokhoz tartozó tesztesetek lefussanak, valamint a tiszta kód elveinek használatára is törekedjen a kód készítése során.

A teszteléshez készült HTML állományba importálja be a lefordított JS kódot.

Feladatok

1. Készítsen egy függvényt, ami bemeneten egy számot vár, a kimeneten pedig megadja annak a számnak az összes egész osztóját emelkedő sorrendben. A kimeneten jelenjen meg az 1 és maga a szám is mint osztó! Csak pozitív egész számokra kell készülni – ezt ellenőrizni sem kell a függvényen belül.
    a. Függvény neve: osszesOszto
    b. Bemeneti paraméterek:
        i. egy szám típusú szám bemenet
    c. Visszatérés
        i. szám tömb
2. Készítsen egy függvényt, ami a bemenetén egy szám típusú tömböt vár és megadja, hogy hány páros szám van abban. Csak pozitív egész számokra kell készülni a tömbben – ezt ellenőrizni sem kell a függvényen belül.
    a. Függvény neve: parosDarab
    b. Bemeneti paraméterek:
        i. egy szám tömb
    c. Visszatérés
        i. egy szám → hány páros volt a tömbben
3. Készítsen egy függvényt, ami bemeneten egy függvényt vár, melyet futtatnia kell. A függvény visszatérése egy szöveg lesz és ezt a szöveget kell ellenőrizni, hogy palindrom-e. (fordítva is ugyan azt jelenti, mint rendesen pl.: indul a görög aludni). A feladat megoldása során a szóközöket nem kell figyelembe venni a teszteléskor (mint ahogy a példában sem jönne ki azzal együtt), viszont más karakterekre nem kell felkészülni, tehát tab, enter stb. nem lehet a szövegben.
    a. Függvény neve: fuggvenyhivasPalindrom
    b. Bemeneti paraméterek:
        i. egy függvény, melynek nincs bemeneti paramétere, a kimenete pedig szöveg
    c. Visszatérés
        i. logikai → megadja, hogy a függ

Megjegyzés

A feladathoz tartozó teszt JS-ben lett implementálva, így a TS kódot először fordítani kell, majd úgy lehet az ellenőrzést végrehajtani, hogy a fordított JS állományt behúzzuk az alap HTML szerkezetbe. Beadni azonban a TS állományt szükséges, a fordított JS állományt opcionális.

5. Feladat – Angular

Leírás

Töltse ki az alábbi Angular projektet végig vezető tesztsort. A tesztsor egy egyszerű listát megjelenítő Angular projekt létrehozásáról szól, ahol magának a projektnek nem kell létrejönnie, azonban a parancsokat ki kell tölteni a kiadott HTML dokumentumban. Ehhez a feladathoz tartozó tesztek szöveges kiértékelésűek, így a parancsokat a rendszer automatikusan értékeli és megadja, hogy helyes, vagy helytelen.

A kiadott HTML állomány forrásába kell elhelyezni megoldásokat, melyeket egy teszt JS rendszer automatikusan ki is fog értékelni.

A megoldás során természetesen használhatja az Angular CLI-t, megírhatja a projekt kérdéseinek megfelelő kódokat, de beadni csak a

Feladat

1. Angular CLI-ben, hogyan hozza létre a „listaFeladat” nevezetű Angular projektet?
    a. A választ a „newProj” id-jú div tartalmába írja bele!
2. Angular CLI-ben, hogyan hoz létre egy új „listacomp” nevű komponenst?
    a. A választ a „newComp” id-jú div tartalmába írja bele!
3. TypeScript-ben, hogyan definiál egy „elemek” nevű tömböt, melyben szövegek lehetnek és a következő 3 statikus szöveges elemet tartalmazza: „Elem1”, „Elem2” és „Elem3”?
    a. A választ a „elemekArray” id-jú div tartalmába írja bele!
4. HTML-ben, Angular segítségével, hogyan listázza ki az előző tömb tartalmát felsorolás elemekként? (Csak a felsorlás elem a lényeges és egy elem megnevezése legyen „elem”)
    a. A választ a „angularFelsorolas” id-jú div tartalmába írja bele!

Megjegyzés

A tesztek futtatása során előfordulhat, hogy a feladat megoldása jó, de a teszt nem tud minden inputra felkészülni – egy plusz szóköz, esetleg fordítható paraméterek stb. – ebből adódóan, ha egy parancs sikeresen lefutott, de a teszt azt állítja, hogy nem jó sem kell kicserélni, a vizsga úgy is manuális ellenőrzésen fog átesni.