Dokumentáció

**Projektfeladat 1**

**Készítette: Dávid Roland, Szabó Noé Bendegúz, Rák Máté Zsombor**

**Feladatok leosztása csapattagok között**

HTML, CSS, JavaScriptA weboldalt Rák Máté Zsombor készítette el. A közös ötleteket összegyűjtve, a csapattal folyamatosan egyeztetve lett elkészítve a tartalom, a dizájn és a funkciók is.

Cypress tesztek  
A Cypress teszteket Dávid Roland készítette el. A Cypress egy modern és kifinomult tesztelési keretrendszer böngésző alapú alkalmazásokhoz. A Cypress könnyen használható, és számos beépített funkcióval rendelkezik, amelyek lehetővé teszik az aszinkron és szinkron tesztek írását. Emellett a Cypress nagy hangsúlyt fektet a stabilitásra és a gyors visszajelzésre, amely segít az automatizált tesztek hatékony készítésében és karbantartásában.

Szabadon választott tesztelési keretrendszerrel elkészített JS tesztek  
Szabó Noé Bendegúz készítette el a JavaScript működését ellenőrző teszteket. Ehhez a feladathoz a Mocha tesztelési keretrendszert választotta, amely kiválóan alkalmas arra, hogy strukturált és könnyen olvasható teszteseteket írjon a szoftvertesztelő. Emellett támogatja a különféle assertion library-ket, például a Chai-t, ami lehetővé teszi a különböző tesztelési stílusok alkalmazását az egyedi igényeknek megfelelően.

Selenium Webdriver automatizált tesztek  
A Selenium Webdriver segítségével elkészített automatizált teszteket Szabó Noé Bendegúz készítette el. A Selenium Webdriver tesztelési környezet lehetővé teszi a böngésző alapú automatizált tesztek írását és végrehajtását. A Selenium WebDriver rugalmas és erőteljes eszköztár a böngészőalkalmazások interaktív elemekkel és felhasználói műveletekkel történő tesztelésére.

Dokumentáció  
A projekt dokumentálása során egyenlően osztottuk le a munkát, a feladat szerint mindenki részletesen leírta a saját részét a projektben. Emellett elosztottuk a kiegészítő részeit a dokumentációnak egymás között, így mindenki teljesítette a terjedelmi követelményt.

**Weboldal**A csatolt config.js fájlt kérem helyezze a /js almappába a weboldal megfelelő működéséhez!

Téma  
A csoportunk az európai kontinenstől északra található, Jeges-tengeren fekvő szigetcsoportról, a Spitzbergákról készített weboldalt. Azért választottuk ennek a helyszínnek a bemutatását, mert a kevesek által ismert változatos, gazdag állatvilág és az elbűvölő havas tájak varázslatos látványt nyújtanak.

Felépítés  
A weboldal egy főoldalból áll, nincsenek hozzá tartozó aloldalak. Az oldalt megnyitva egy, a szemnek kellemes színpalettával ellátott ismeretterjesztő weboldalon találhatja magát a felhasználó. Legfelül található egy navigációs sáv, melynek segítségével könnyedén kiigazodhatunk az alcímekre való egyszerű ,,ugrálással”. A navigációs sáv mindig az oldal tetejéhez ,,ragadva” segíti a felhasználót, tehát hogyha lejjebb görget, akkor is egyszerű kattintásokkal el tud navigálni az egész weblapon. A főcím alatt látható három kép, melyek random generálódnak le, tehát a weboldal megnyitáskor mindig más-más fotókkal találkozhat a felhasználó. Ez a funkció az Unsplash Image API-t felhasználva működik. Ezután következnek az ismertető bekezdések, külön alcímekkel, fő szempontokra elkülönítve: történet, földrajz, turizmus és élővilág. Az oldal alján egy képgaléria látható.

Felhasználói interakciók  
A weboldal egy ismeretterjesztő oldal, szóval kevés felhasználói interakció található rajta. Az előző bekezdésben már részletezett navigációs sávban található hivatkozásokra kattintva, a felhasználó gyorsan kiigazodhat a weblapon. Ezen kívül az élővilág alcímű bekezdésben található egy legördülő menü, amelyből, ha a felhasználó kiválaszt egy állatot, akkor megjelenik a kiválasztott állatról egy ismertető szöveg. Továbbiakban egy galéria is található az oldalon. Itt, hogyha egy képre kattint a felhasználó, akkor az megjelenik egy új lapon nagyobb méretben.

**Cypress tesztek**Használat:  
Kérem telepítse a Cypress-t a használathoz!

A tesztelés során a következőket ellenőrizzük:

* Oldal betöltése: Ellenőrizzük, hogy a weboldal betöltődik-e megfelelően

it('Oldal betöltése', () => {

      cy.visit('http://127.0.0.1:5500/index.html')

    })

**HTML elemek ellenőrzése:**

* Navigációs linkek: Ellenőrizzük, hogy a navigációs linkek a megfelelő helyekre mutatnak-e

it('Navigációs linkek ellenőrzése', () => {

      cy.visit('http://127.0.0.1:5500/index.html')

      cy.get('#navbar a').each((link) => {

        cy.wrap(link).click()

        cy.url().should('include', link.attr('href'))

      })

    })

* Címkép: Ellenőrizzük, hogy a headerben lévő kép megjelenik-e

it('Címkép megjelenítésének ellenőrzése', () => {

      cy.visit('http://127.0.0.1:5500/index.html')

      cy.get('header img').should('be.visible')

    })

* Történet rész: Ellenőrizzük, hogy a történet rész megjelenik-e

it('Történet rész megjelenítésének ellenőrzése', () => {

      cy.visit('http://127.0.0.1:5500/index.html')

      cy.get('#tortenet').should('be.visible')

    })

* Galéria rész: Ellenőrizzük, hogy a galéria rész megjelenik-e

    it('Galéria rész megjelenítésének ellenőrzése', () => {

      cy.visit('http://127.0.0.1:5500/index.html')

      cy.get('#galeria').should('be.visible')

    })

* Footer: Ellenőrizzük, hogy a footer megjelenik-e

it('Footer megjelenítésének ellenőrzése', () => {

      cy.visit('http://127.0.0.1:5500/index.html')

      cy.get('#footer').should('be.visible')

    })

**CSS stílusok ellenőrzése:**

* Címek stílusa: Ellenőrizzük a címek középre igazítását

it('Címek stílusának ellenőrzése', () => {

      cy.visit('http://127.0.0.1:5500/index.html')

      cy.get('h1').should('have.css', 'text-align', 'center')

      cy.get('h2').should('have.css', 'text-align', 'center')

    })

* Szöveg bekezdéseinek stílusa: Ellenőrizzük a szövegek igazítását

it('Szöveg bekezdéseinek stílusának ellenőrzése', () => {

      cy.visit('http://127.0.0.1:5500/index.html')

      cy.get('p').should('have.css', 'text-align', 'justify')

    })

* Header kép stílusa: Ellenőrizzük a header kép igazítását

it('Header kép megjelenítésének ellenőrzése', () => {

      cy.visit('http://127.0.0.1:5500/index.html')

      cy.get('header img').should('have.css', 'object-fit', 'cover')

    })

* Navbar formázása: Ellenőrizzük a navbar stílusát és linkek színét

it('Navbar formázásának ellenőrzése', () => {

      cy.visit('http://127.0.0.1:5500/index.html')

      cy.get('#navbar').should('have.css', 'overflow', 'hidden')

      cy.get('#navbar a').should('have.css', 'color', 'rgb(242, 242, 242)')

    })

* Háttérszín: Ellenőrizzük a háttérszín beállítását

it('Háttérszín ellenőrzése', () => {

      cy.visit('http://127.0.0.1:5500/index.html')

      cy.get('body').should('have.css', 'background-color', 'rgb(239, 239, 239)')

    })

**Szabadon választott tesztelési keretrendszerrel elkészített JS tesztek – Mocha**Használat:  
Telepítse a Mocha-t a használathoz!

A tesztelés során a következőket ellenőrizzük:

* Az API kulcs meglétének ellenőrzése

it("config van?", function () {

        assert.notEqual(config.config.UNSPLASH\_ACCESS\_KEY, "");

    });

* Megfelelő számú kép lekérésének ellenőrzése

it("3 képet kér le?", async function () {

        let t = await kep\_leker(3);

        assert.equal(t.length, 3);

    });

* A lekért képek formátumának(jpg) ellenőrzése

it("jpg formátumú képek jönnek?", async function () {

        let t = await kep\_leker(3);

        t.forEach((kep) => {

            assert.equal(kep.includes("fm=jpg"), true);

        });

    });

* Egy, a legördülő menüben nem szereplő állatnak kiválasztása során nem jelenik meg szöveg

it("rossz állatra nem ad vissza szöveget?", function () {

        assert.equal(allat\_szoveg("gorilla"), "");

    });

**Selenium Webdriver automatizált teszt**Használat:  
Telepítse a Seleniumot! A szkriptet futtatva a böngészőben a megadott URL címre navigál, majd végrehajtja a definiált műveleteket. A teszt eredményeit a konzolra naplózza.

Lépések sorban:

* Böngészőablak betöltése és inicializálása

let driver = await new Builder().forBrowser(Browser.FIREFOX).build();

* Oldalcím lekérdezése és hosszának naplozása

 let oldalcim = await driver.getTitle();

        let cim\_hossz = oldalcim.length;

        console.log("Oldal címe " + oldalcim + ", hossza: " + cim\_hossz);

* Böngészőablak méretének lekérdezése és logolása

 let size = await driver.manage().window().getSize();

        let w = size.width;

        let h = size.height;

        console.log("Böngészőablak szélessége: " + w + ", hosszúsága: " + h);

* Navbar elem osztályattribútumának tartalmának lekérdezése és kiírása

 let navbar\_osztaly = await driver.findElement(By.css("#navbar")).getAttribute("class");

        console.log("A navbar class attribútumának tartalma: " + navbar\_osztaly)

* Véletlenszerű képek URL-jének lekérdezése és naplózása

 let random\_kepek = await driver.findElements(By.css("#kepek img"));

        console.log("A három random kép url-je: ");

        random\_kepek.forEach(async (kep) => {

            let url = await kep.getAttribute("src");

            console.log(url);

        });

* Navigáció a menüpontok között

let tortenet\_link = await driver.findElement(By.css("#navbar a[href=\"#tortenet\"]"));

        await actions.move({origin: tortenet\_link}).click().perform();

(És a további 3 eset)

* "Elővilág" menüponton belül egy select („sarkiroka”) elem kiválasztása

let elovilag\_select = new Select(await driver.findElement(By.css("#elovilag select")));

        await elovilag\_select.selectByValue("sarkiroka");

* Galéria megnyitása és véletlenszerű kép kiválasztása

 let galeria\_link = await driver.findElement(By.css("#navbar a[href=\"#galeria\"]"));

        await actions.move({origin: galeria\_link}).click().perform();

        var idx = Math.round(Math.random() \* 6);

        let galeria\_kepek = await driver.findElements(By.css("#galeria img"));

        await sleep(1000);

        await galeria\_kepek[idx].click();

* Képernyőkép készítése és mentése

driver.takeScreenshot().then(async (image, err) => {

            await writeFile("screenshot-" + Date.now() + ".png", image, "base64");