**### 5. óra: Reguláris kifejezések**

**- Reguláris kifejezések használata JavaScript-ben**

**- Gyakorlati példák: reguláris kifejezések alapjai, egyszerűbb validációs feladatok.**

**Óraterv: Reguláris kifejezések JavaScript-ben**

**5. óra célja:** A diákok megértik a reguláris kifejezések alapjait és hogy hogyan használhatóak egyszerű szövegkeresésre, szűrésre és formátumellenőrzésre JavaScript-ben. Megtanulják a reguláris kifejezések szintaxisát és megismerkednek néhány gyakori mintával, amelyeket később alkalmazni tudnak a programozásban.

**Időtartam:** 45 perc

**Óra Script**

**0-5 perc: Bevezetés a reguláris kifejezésekbe**

* **Tanár mondja:** „Hallottatok már a reguláris kifejezésekről? Ezek egyfajta eszközt kínálnak arra, hogy mintákat írjunk le, és így azokat felhasználva gyorsan megtaláljunk vagy ellenőrizzünk bizonyos karakterláncokat, mint például egy e-mail címet, telefonszámot, vagy irányítószámot.”
* **Példa:** „Gondoljatok egy könyvtárra, ahol minden könyv azonosítója egy adott szabály szerint van felépítve. A reguláris kifejezésekkel megadhatjuk azt a szabályt, és így könnyen megtalálhatjuk a könyveket.”

**5-10 perc: Reguláris kifejezések alapjai és egyszerű szintaxis**

1. **Szövegkeresés:** „A reguláris kifejezések segítségével egyszerű mintákat is kereshetünk. Ha például a szövegben minden ’a’ betűt szeretnénk megtalálni, akkor az a mintát használnánk.”
2. **Minta szintaxis:** „A reguláris kifejezésekben a speciális karaktereknek különleges jelentése van. Például a . karakter bármilyen egyedi karaktert jelent, míg a \* azt jelenti, hogy az előtte lévő karakter bármennyiszer előfordulhat.”
   * **Példa kód:**

javascript

Copy code

let minta = /a/;

console.log(minta.test("alma")); // true

console.log(minta.test("körte")); // false

**10-20 perc: Alapvető reguláris kifejezések és szintaxis elemek**

1. **Speciális karakterek:**
   * **Pont (.):** „A . bármilyen egy karaktert jelöl. Ha egy olyan szót keresünk, amelyben az ’a’ bármilyen karakterrel követi, akkor az a. mintát használhatjuk.”
     + **Példa:** let minta = /a./;
     + „Ez a kifejezés minden olyan szóra illeszkedik, amely az ’a’ után egy karakterrel folytatódik, például az 'alma'.”
   * **Csillag (\*):** „A \* bármely előző karakter ismétlését jelzi. Például, ha azt akarjuk, hogy az a akárhányszor előfordulhat, akkor az a\* mintát használjuk.”
     + **Példa:** let minta = /ba\*na/;
     + **Magyarázat:** „Ez minden ’b’, ’na’ kombinációra illeszkedik, ahol az ’a’ akárhányszor megjelenhet. Például: ’banana’, ’bna’.”
2. **Metakarakterek és szekvenciák:**
   * **[abc]:** „A szögletes zárójel azt jelenti, hogy bármelyik karakter illeszkedhet a megadottak közül.”
     + **Példa:** let minta = /[aeiou]/;
     + „Ez minden magánhangzót keres a szövegben.”
   * **^ és $:** „A ^ a minta elejére, míg a $ a végére illeszkedést jelöli. Így ha azt akarjuk, hogy egy szó pontosan egy adott mintával kezdődjön vagy végződjön, ezeket használjuk.”
     + **Példa:** let minta = /^alma$/;
     + „Ez csak akkor ad vissza igazat, ha a teljes szó ’alma’.”
3. **Feladat:** „Keressetek egy mintát, amely illeszkedik minden olyan szóra, amely ’b’-vel kezdődik, és bármennyi karakter követheti.”
   * **Megoldás:** let minta = /^b.\*$/;

**20-30 perc: Gyakorlati példák reguláris kifejezések használatára**

1. **Feladat:** „Írjatok egy reguláris kifejezést, amely minden olyan szót felismer, ami ’e’-re végződik.”
   * **Megoldás:** let minta = /e$/;
   * **Tanár magyarázata:** „Ez a minta minden olyan szót felismer, ahol a végén ’e’ van, mert a $ a szöveg végét jelzi.”
2. **Gyakori minták:** „Vegyük például az e-mail címeket. Ezek rendszerint a következő formátumot követik: név@domain.com. Ezt egy reguláris kifejezéssel könnyen megadhatjuk.”
   * **Példa e-mail mintára:**

javascript

Copy code

let emailMinta = /^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;

console.log(emailMinta.test("pelda@domain.com")); // true

* + **Tanár magyarázata:** „Ez a minta azt biztosítja, hogy az e-mail cím formátuma helyes, de nem garantálja, hogy az e-mail valóban létezik.”

1. **Gyakorlati feladat:** „Írjatok egy mintát, amely az összes magyar irányítószámot ellenőrzi (négy számjegy, az első számjegy nem lehet 0).”
   * **Megoldás:** let iranyitoszamMinta = /^[1-9][0-9]{3}$/;
   * **Magyarázat:** „Itt az első számjegy 1 és 9 közé kell essen, majd három további számjegy követi.”

**30-40 perc: Különböző reguláris műveletek JavaScript-ben**

1. **test és match metódusok:**
   * **test használata:** „A test metódussal ellenőrizhetjük, hogy egy minta illeszkedik-e egy szövegre. Ha van találat, true-t ad vissza, ha nincs, akkor false-t.”
     + **Példa:**

javascript

Copy code

let minta = /alma/;

console.log(minta.test("Ez egy alma.")); // true

* + **match használata:** „A match metódussal az összes találatot lekérhetjük, nem csak azt, hogy van-e találat.”
    - **Példa:**

javascript

Copy code

let szoveg = "Ez egy alma, egy másik alma.";

let minta = /alma/g;

console.log(szoveg.match(minta)); // ["alma", "alma"]

* + **Feladat:** „Használjátok a test és match metódusokat egy szövegre, amelyben több szó van, és próbáljatok meg keresni egy konkrét szót, például 'fa'.”

**40-45 perc: Összefoglalás és kérdések**

* **Tanár mondja:** „Ma átvettük a reguláris kifejezések alapjait. Megnéztük, hogyan írhatunk különböző mintákat és hogyan használhatjuk őket JavaScript-ben egyszerű ellenőrzésre és keresésre. Van kérdésetek az eddig elhangzottakkal kapcsolatban?”
* **Házi feladat gyakorlásra:** „Gondolkodjatok el azon, milyen minták lehetnek hasznosak a mindennapi életben: például telefonszámok, dátumok, rendszámok, vagy akár jelszóformátumok. Próbáljátok megírni a reguláris kifejezéseket ezekre, és teszteljétek őket különböző szövegeken!”

**Óra összefoglalás**

Az óra során a diákok megtanulták, hogyan használhatják a reguláris kifejezéseket különböző minták felismerésére és ellenőrzésére. Kipróbálták a test és match metódusokat, és megértették, hogyan alkalmazhatják a reguláris kifejezéseket a gyakorlatban, például szövegszűrésre és formátumellenőrzésre.

### Óraterv: Reguláris kifejezések JavaScript-ben

\*\*5. óra célja:\*\* A diákok megértik a reguláris kifejezések alapjait és hogy hogyan használhatóak egyszerű szövegkeresésre, szűrésre és formátumellenőrzésre JavaScript-ben. Megtanulják a reguláris kifejezések szintaxisát és megismerkednek néhány gyakori mintával, amelyeket később alkalmazni tudnak a programozásban.

\*\*Időtartam:\*\* 45 perc

---

#### Óra Script

\*\*0-5 perc: Bevezetés a reguláris kifejezésekbe\*\*

- \*\*Tanár mondja:\*\* „Hallottatok már a reguláris kifejezésekről? Ezek egyfajta eszközt kínálnak arra, hogy mintákat írjunk le, és így azokat felhasználva gyorsan megtaláljunk vagy ellenőrizzünk bizonyos karakterláncokat, mint például egy e-mail címet, telefonszámot, vagy irányítószámot.”

- \*\*Példa:\*\* „Gondoljatok egy könyvtárra, ahol minden könyv azonosítója egy adott szabály szerint van felépítve. A reguláris kifejezésekkel megadhatjuk azt a szabályt, és így könnyen megtalálhatjuk a könyveket.”

\*\*5-10 perc: Reguláris kifejezések alapjai és egyszerű szintaxis\*\*

1. \*\*Szövegkeresés:\*\* „A reguláris kifejezések segítségével egyszerű mintákat is kereshetünk. Ha például a szövegben minden ’a’ betűt szeretnénk megtalálni, akkor az `a` mintát használnánk.”

2. \*\*Minta szintaxis:\*\* „A reguláris kifejezésekben a speciális karaktereknek különleges jelentése van. Például a `.` karakter bármilyen egyedi karaktert jelent, míg a `\*` azt jelenti, hogy az előtte lévő karakter bármennyiszer előfordulhat.”

- \*\*Példa kód:\*\*

```javascript

let minta = /a/;

console.log(minta.test("alma")); // true

console.log(minta.test("körte")); // false

```

\*\*10-20 perc: Alapvető reguláris kifejezések és szintaxis elemek\*\*

1. \*\*Speciális karakterek:\*\*

- \*\*Pont (`.`):\*\* „A `.` bármilyen egy karaktert jelöl. Ha egy olyan szót keresünk, amelyben az ’a’ bármilyen karakterrel követi, akkor az `a.` mintát használhatjuk.”

- \*\*Példa:\*\* `let minta = /a./;`

- „Ez a kifejezés minden olyan szóra illeszkedik, amely az ’a’ után egy karakterrel folytatódik, például az 'alma'.”

- \*\*Csillag (`\*`):\*\* „A `\*` bármely előző karakter ismétlését jelzi. Például, ha azt akarjuk, hogy az `a` akárhányszor előfordulhat, akkor az `a\*` mintát használjuk.”

- \*\*Példa:\*\* `let minta = /ba\*na/;`

- \*\*Magyarázat:\*\* „Ez minden ’b’, ’na’ kombinációra illeszkedik, ahol az ’a’ akárhányszor megjelenhet. Például: ’banana’, ’bna’.”

2. \*\*Metakarakterek és szekvenciák:\*\*

- \*\*[abc]:\*\* „A szögletes zárójel azt jelenti, hogy bármelyik karakter illeszkedhet a megadottak közül.”

- \*\*Példa:\*\* `let minta = /[aeiou]/;`

- „Ez minden magánhangzót keres a szövegben.”

- \*\*^ és $:\*\* „A `^` a minta elejére, míg a `$` a végére illeszkedést jelöli. Így ha azt akarjuk, hogy egy szó pontosan egy adott mintával kezdődjön vagy végződjön, ezeket használjuk.”

- \*\*Példa:\*\* `let minta = /^alma$/;`

- „Ez csak akkor ad vissza igazat, ha a teljes szó ’alma’.”

3. \*\*Feladat:\*\* „Keressetek egy mintát, amely illeszkedik minden olyan szóra, amely ’b’-vel kezdődik, és bármennyi karakter követheti.”

- \*\*Megoldás:\*\* `let minta = /^b.\*$/;`

\*\*20-30 perc: Gyakorlati példák reguláris kifejezések használatára\*\*

1. \*\*Feladat:\*\* „Írjatok egy reguláris kifejezést, amely minden olyan szót felismer, ami ’e’-re végződik.”

- \*\*Megoldás:\*\* `let minta = /e$/;`

- \*\*Tanár magyarázata:\*\* „Ez a minta minden olyan szót felismer, ahol a végén ’e’ van, mert a `$` a szöveg végét jelzi.”

2. \*\*Gyakori minták:\*\* „Vegyük például az e-mail címeket. Ezek rendszerint a következő formátumot követik: név@domain.com. Ezt egy reguláris kifejezéssel könnyen megadhatjuk.”

- \*\*Példa e-mail mintára:\*\*

```javascript

let emailMinta = /^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;

console.log(emailMinta.test("pelda@domain.com")); // true

```

- \*\*Tanár magyarázata:\*\* „Ez a minta azt biztosítja, hogy az e-mail cím formátuma helyes, de nem garantálja, hogy az e-mail valóban létezik.”

3. \*\*Gyakorlati feladat:\*\* „Írjatok egy mintát, amely az összes magyar irányítószámot ellenőrzi (négy számjegy, az első számjegy nem lehet 0).”

- \*\*Megoldás:\*\* `let iranyitoszamMinta = /^[1-9][0-9]{3}$/;`

- \*\*Magyarázat:\*\* „Itt az első számjegy 1 és 9 közé kell essen, majd három további számjegy követi.”

\*\*30-40 perc: Különböző reguláris műveletek JavaScript-ben\*\*

1. \*\*`test` és `match` metódusok:\*\*

- \*\*`test` használata:\*\* „A `test` metódussal ellenőrizhetjük, hogy egy minta illeszkedik-e egy szövegre. Ha van találat, `true`-t ad vissza, ha nincs, akkor `false`-t.”

- \*\*Példa:\*\*

```javascript

let minta = /alma/;

console.log(minta.test("Ez egy alma.")); // true

```

- \*\*`match` használata:\*\* „A `match` metódussal az összes találatot lekérhetjük, nem csak azt, hogy van-e találat.”

- \*\*Példa:\*\*

```javascript

let szoveg = "Ez egy alma, egy másik alma.";

let minta = /alma/g;

console.log(szoveg.match(minta)); // ["alma", "alma"]

```

- \*\*Feladat:\*\* „Használjátok a `test` és `match` metódusokat egy szövegre, amelyben több szó van, és próbáljatok meg keresni egy konkrét szót, például 'fa'.”

\*\*40-45 perc: Összefoglalás és kérdések\*\*

- \*\*Tanár mondja:\*\* „Ma átvettük a reguláris kifejezések alapjait. Megnéztük, hogyan írhatunk különböző mintákat és hogyan használhatjuk őket JavaScript-ben egyszerű ellenőrzésre és keresésre. Van kérdésetek az eddig elhangzottakkal kapcsolatban?”

- \*\*Házi feladat gyakorlásra:\*\* „Gondolkodjatok el azon, milyen minták lehetnek hasznosak a mindennapi életben: például telefonszámok, dátumok, rendszámok, vagy akár jelszóformátumok. Próbáljátok megírni a reguláris kifejezéseket ezekre, és teszteljétek őket különböző szövegeken!”

---

### Óra összefoglalás

Az óra során a diákok megtanulták, hogyan használhatják a reguláris kifejezéseket különböző minták felismerésére és ellenőrzésére. Kipróbálták a `test` és `match`

metódusokat, és megértették, hogyan alkalmazhatják a reguláris kifejezéseket a gyakorlatban, például szövegszűrésre és formátumellenőrzésre.