**### 10. óra: Arrow functions**

**- Arrow functions (nyílfüggvények) használata**

**- Gyakorlati feladatok: egyszerű és összetettebb arrow functions létrehozása.**

**10. Óra: Arrow Functions (Nyílfüggvények)**

**1. Bevezetés (10 perc)**

* **Tanár mondja:** "Ma az arrow functions, azaz nyílfüggvények világába fogunk betekinteni. Ezek a függvények egy egyszerűbb és tömörebb szintaxisú módját kínálják a függvények definiálásának. Sokszor találkozhatunk velük modern JavaScript kódokban, és segítenek a kód egyszerűsítésében."
* **Miért fontosak?**
  + Kompaktabbá teszik a kódot, így az olvashatóság is javul.
    - **Példa:** Egy hagyományos, hosszan megírt függvényt helyettesíthetünk egyetlen sorral, ha arrow function-t használunk.
  + Gyakori, egyszerű műveletekhez ideálisak.
    - **Példa:** Egy tömb elemeinek összegét kiszámolhatjuk egy arrow function segítségével, amit könnyedén beilleszthetünk más kódrészekbe.
  + Egyedi tulajdonságaik (pl. nincs saját this-környezetük) megoldanak olyan problémákat, amelyeket a hagyományos függvényeknél manuálisan kellene kezelni.
    - **Magyárat:** Eseménykezelésnél, ahol a this tévesen mutat másik objektumra, az arrow functions az aktuális kontextust örökíti, elkerülve ezzel a hibákat.
* **Példa alkalmazási területekre:**
  + **Egyszerű számítási műveletek:** "Egy tömb elemeinek összegét kiszámolhatjuk egyszerűsített szintaxissal, pl. const osszeg = arr.reduce((a, b) => a + b, 0);."
  + **Eseménykezelés a DOM-ban:** "Például egy gombra kattintáskor az arrow function biztosítja, hogy a this az aktuális objektumra mutasson, nem pedig az eseménykezelőre."
  + **Callback függvények:** "Amikor egy async műveletben vagy iterációs ciklusban van szükség egyszerű függvényekre, az arrow function gyorsabb és átláthatóbb."

**2. Arrow Functions Alapjai (15 perc)**

* **Tanár mondja:** "Az arrow function a => szimbólum alapján kapta a nevét. Az alapgondolat az, hogy egy adott feladatot egyszerűen és gyorsan definiáljunk függvényként."
* **Példa bemutatása:**

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Automatikusan generált leírás



* **Tanár magyázza:** "A hagyományos függvényekhez képest az arrow function szintaxisa sokkal egyszerűbb. Nincs szükség function kulcsszóra, és ha a függvény egyetlen kifejezést tartalmaz, a kapcsos zárójelek és a return kulcsszó is elhagyható."
* **Fontos megjegyzések:**
  + Ha több soros a függvény, akkor kapcsos zárójelek és return szükségesek:

A képen szöveg, Betűtípus, képernyőkép látható

Automatikusan generált leírás



Ha nincs paraméter, akkor üres zárójeleket kell használni:

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

****

**Gyakorlati feladat:** "Alakítsatok át egy hagyományos függvényt arrow functionné:

A képen szöveg, képernyőkép, Betűtípus, Multimédiás szoftver látható

Automatikusan generált leírás



**3. Paraméterek Kezelése Arrow Functions-ban (15 perc)**

* **Tanár mondja:** "Az arrow functions ugyanúgy kezelik a paramétereket, mint a hagyományos függvények. Egy paraméter esetén a zárójelek elhagyhatók, de ha több paraméter van, akkor a zárójelek kötelezők."
* **Egyszerű példa:**

A képen szöveg, képernyőkép, Betűtípus látható

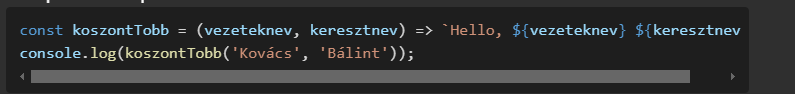
Automatikusan generált leírás

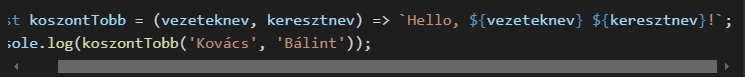
****

**Több paraméteres példa:**

const koszontTobb = (vezeteknev, keresztnev) => `Hello, ${vezeteknev} ${keresztnev}!`;

console.log(koszontTobb('Kovács', 'Bálint'));





const koszontTobb = (vezeteknev, keresztnev) => `Hello, ${vezeteknev} ${keresztnev}!`;

console.log(koszontTobb('Kovács', 'Bálint'));

* **Gyakorlati feladat:** "Készítsetek egy arrow function-t, amely egy számot fogad, és visszaadja annak köbét!"

**4. Arrow Functions Speciális Jellemzői (15 perc)**

* **Tanár mondja:** "Az arrow functions egyik legfontosabb jellemzője, hogy nincs saját this-környezetük. Ez azt jelenti, hogy az örökölt this-t használják az őket tartalmazó környezetből."
* **Példa:** A képen szöveg, képernyőkép, képernyő, szoftver látható

  Automatikusan generált leírás

function Hagyomanyos() {

this.nev = 'Hagyományos';

setTimeout(function() {

console.log(this.nev); // undefined

}, 1000);

}

function NyilFuggveny() {

this.nev = 'Nyílfüggvény';

setTimeout(() => {

console.log(this.nev); // Nyílfüggvény

}, 1000);

}

new Hagyomanyos();

new NyilFuggveny();

* **Magyázat:** "A hagyományos függvényeknél a this a hívó objektumra mutat, ami problémákat okozhat. Az arrow functions viszont nem definiál saját this-t, ezért az őket tartalmazó környezet this-t használják. Ez különösen fontos, amikor callbackeket vagy aszinkron műveleteket kezelünk."

**5. Gyakorlás és Kérdések (10 perc)**

* **Gyakorlati feladatok:**
  1. "Készítsetek egy arrow function-t, amely egy tömböt fogad, és visszaadja az elemei összegét."
  2. "Készítsetek egy arrow function-t, amely egy szöveges üzenetet fogad, és minden szó első betűjét nagybetűsíti."
* **Tanár mondja:** "Az arrow functions egyszerűséget és tömörséget hoznak a kódunkba, különösen akkor, ha kis műveletekhez használjuk őket. Ha valakinek bármilyen kérdése van, most tegye fel!"
* **Otthoni feladat:** "Otthon írjatok egy arrow function-t, amely egy tömb minden elemét megszorozza 2-vel, és visszaadja az új tömböt."