Objektumok Magyarázata JavaScriptben

Az objektumok az egyik legfontosabb és leggyakrabban használt adatstruktúrák JavaScriptben. Az objektumok lehetővé teszik, hogy összefüggő adatokat egyetlen egységben tároljunk. Ezek az adatok kulcs-érték párokként kerülnek tárolásra, ahol a kulcs egy egyedi azonosító, az érték pedig bármilyen típusú adat lehet.

Objektum Létrehozása

Objektumliterálok

Az objektumokat leggyakrabban objektumliterálok segítségével hozzuk létre. Az objektumliterálokat kapcsos zárójelek (`{}`) határolják, és a kulcs-érték párok vesszővel vannak elválasztva.

Példa:

```
let szemely = {
    nev: "János",
    kor: 30,
    foglalkozas: "programozó",
};
```

Ebben a példában a `szemely` objektum három tulajdonságot (kulcs-érték párt) tartalmaz: `nev`, `kor` és `foglalkozas`.

Objektum Tulajdonságainak Hozzáférése

Pont Notáció

A pont notációval hozzáférhetünk az objektum tulajdonságaihoz.

Példa:

```
console.log(szemely.nev); // Kiírja: "János"
console.log(szemely.kor); // Kiírja: 30
console.log(szemely.foglalkozas); // Kiírja: "programozó"
```

Kapcsos Zárójel Notáció

A kapcsos zárójel notációval is hozzáférhetünk az objektum tulajdonságaihoz. Ez különösen hasznos, ha a tulajdonság neve dinamikusan kerül meghatározásra.

Példa:

```
console.log(szemely["nev"]); // Kiírja: "János"
console.log(szemely["kor"]); // Kiírja: 30
console.log(szemely["foglalkozas"]); // Kiírja: "programozó"
```

Objektum Tulajdonságainak Módosítása

Az objektum tulajdonságainak értékét megváltoztathatjuk a pont vagy a kapcsos zárójel notáció használatával.

Példa:

```
szemely.kor = 31; // Módosítja a 'kor' tulajdonságot
szemely["foglalkozas"] = "vezető programozó"; // Módosítja a 'foglalkozas' tulajdonságot

console.log(szemely); // Kiírja: { nev: 'János', kor: 31, foglalkozas: 'vezető programozó' }
```

Új Tulajdonság Hozzáadása

Új tulajdonságot is hozzáadhatunk egy objektumhoz.

Példa:

```
szemely.lakcim = "Budapest";
console.log(szemely); // Kiírja: { nev: 'János', kor: 31, foglalkozas: 'vezető programozó', lakcim: 'Budapest' }
```

Tulajdonság Eltávolítása

Egy tulajdonságot eltávolíthatunk a `delete` operátor segítségével.

Példa:

```
delete szemely.lakcim;
console.log(szemely); // Kiírja: { nev: 'János', kor: 31, foglalkozas: 'vezető programozó' }
```

Tömbök Magyarázata JavaScriptben

A tömbök (arrays) alapvető adatstruktúrák JavaScriptben, amelyek lehetővé teszik, hogy több értéket tároljunk egyetlen változóban. A tömbök különösen hasznosak, amikor egy halom hasonló adatot szeretnénk kezelni és rendszerezni.

Tömbök Létrehozása

Tömbök létrehozására többféle mód van JavaScriptben. A leggyakoribb módszer az, amikor a tömböt szögletes zárójelek (`[]`) segítségével hozzuk létre, és az elemeket vesszővel választjuk el egymástól.

Példák:

```
let uresTomb = []; // Üres tömb
let szamok = [1, 2, 3, 4, 5]; // Számokat tartalmazó tömb
let szovegek = ["alma", "banán", "cseresznye"]; // Sztringeket tartalmazó tömb
let vegyes = [1, "alma", true, null]; // Vegyes típusú elemeket tartalmazó tömb
```

Tömbök Hozzáférése és Módosítása

A tömb elemeihez indexeléssel férhetünk hozzá. Az indexelés 0-tól kezdődik, tehát az első elem indexe 0, a másodiké 1, és így tovább.

Példák:

```
let gyumolcsok = ["alma", "banán", "cseresznye"];
console.log(gyumolcsok[0]); // Kiírja: "alma"
console.log(gyumolcsok[1]); // Kiírja: "banán"
console.log(gyumolcsok[2]); // Kiírja: "cseresznye"

gyumolcsok[1] = "körte"; // A második elem módosítása
console.log(gyumolcsok); // Kiírja: ["alma", "körte", "cseresznye"]
```

Tömbök Hossza

A tömb hosszát a `length` tulajdonsággal érhetjük el, amely megadja, hogy hány elem található a tömbben.

Példa:

```
let szamok = [1, 2, 3, 4, 5];
console.log(szamok.length); // Kiírja: 5
```

Tömb Műveletek

JavaScriptben számos beépített metódus áll rendelkezésre a tömbökkel való munkához. Néhány alapvető művelet:

- **Elem hozzáadása a végéhez**: `push`

  ```
  let szamok = [1, 2, 3];
  szamok.push(4); // Hozzáadja a 4-et a tömb végéhez
  console.log(szamok); // Kiírja: [1, 2, 3, 4]
  ```

- **Elem eltávolítása a végéről**: `pop`

  ```
  let szamok = [1, 2, 3];
  let utolsoElem = szamok.pop(); // Eltávolítja az utolsó elemet és visszaadja azt
  console.log(utolsoElem); // Kiírja: 3
  console.log(szamok); // Kiírja: [1, 2]
  ```

- **Elem hozzáadása az elejéhez**: `unshift`

  ```
  let szamok = [1, 2, 3];
  szamok.unshift(0); // Hozzáadja a 0-t a tömb elejéhez
  console.log(szamok); // Kiírja: [0, 1, 2, 3]
  ```

- **Elem eltávolítása az elejéről**: `shift`

  ```
  let szamok = [1, 2, 3];
  let elsoElem = szamok.shift(); // Eltávolítja az első elemet és visszaadja azt
  console.log(elsoElem); // Kiírja: 1
  console.log(szamok); // Kiírja: [2, 3]
  ```