JavaScript kódolási konvenciók

Megkérünk benneteket, hogy a házi feladatok elkészítésénél figyeljetek az ECMAScript 6 szabványnak megfelelelő JavaScript kódolási konvenciók betartására. Az ide tartozó laborfeladatot csak abban az esetben fogadjuk el, ha megfelel ezen konvencióknak.

Elnevezések

- a változó- és függvényneveket `camelCase` írásmóddal írjuk. Ez azt jelenti, hogy kisbetűvel kezdjük és a szóösszetételeknél az első betűt nagybetűsítjük. Példa:
  `js const firstName = 'John'; const lastName = 'Doe'; function getAvarageOfNotes() { /* ... */ }`
- használjunk beszédes neveket: `js let t = obj.returnValue(); // mi a t? time, testValue? let time = obj.returnTime(); // helyes`
- a változókat `const` vagy `let` kulcsszóval deklaráljuk és nem `var`ral: `js var testVariable = 52; // Helytelen let variable = 25; // Ennek a változónak később új értéket lehet adni const constant = 'This is a constant'; // Nem lehet megváltoztatni az értékét const exampleObject = new ExampleObject(); // Nem lehet átirányítani, mindig erre az objektumra fog mutatni a referencia.`

Stringek és objektumok

- a (változókat nem tartalmazó) stringek létrehozásához használjunk sima idézőjeleket: `js const apple = "Apple"; // helytelen const apple = `Apple`; // helytelen const apple = 'Apple'; // helyes`
- ne `+` operátorral fűzzük össze a stringeket, hanem használjunk sablonliterálokat (template literals): „`js const apple =`Apple‘; // helytelen, nincs dinamikus érték const orange = ‘Orange’; // helyes
  // helytelen const fruits = apple + ‘,’ + orange; // helyes
  const fruits = `${apple}, ${orange}`; „‘

- az üres tömböket és objektumokat a `[]`/`{}` operátorokkal inicializáljuk, nem a `new` metódushívást használva: `js const items = new Array(); // helytelen const items = []; // helyes const object = new Object(); // helytelen const object = {}; // helyes`

Függvények

- használjunk konstruktorfüggvényeket a `this` implicitváltozóval, illetve a `new` operátort az egységbezáráshoz: `js function Jedi(name='no name') { this.name = name; this.toString = () => `Jedi - ${this.name}`; } let jedi = new Jedi();`
- inline függvényátadáskor (pl. amikor paraméterként adunk át egy függvényt, ld. `forEach`, `map`, `filter`), használjuk az arrow functioneket: „‘js // helytelen [1, 2, 3].map(function(x) { const y = x + 1; return x * y; });

  // helyes [1, 2, 3].map((x) => { const y = x + 1; return x * y; }); „‘
- egysoros arrow függvények esetén mellőzzük a kapcsos zárójelet és a `return` kulcsszót: „‘js // helytelen const sqr = (x) => { return x * x; };

  // helyes const sqr = (x) => x * x; „‘
- ahol lehet, adjunk alapértelmezett (default) értékeket a függvényparamétereinknek: `js function pow(x, a=2) { // ... }`

Kódstruktúra

- kerüljük a felesleges elágazásokat:
  „‘js if (feltétel) { // Nem csinál semmit } else { obj.függvényHívás(); } // helytelen!

  if (!feltétel) { obj.függvényHívás(); } // helyes „‘
- kerüljük a felesleges ellenőrzéseket: „‘js if (változó === érték) { return true; } else { return false; } // helytelen!

  return változó === érték; // helyes „‘

A forráskód formázására vonatkozó konvenciók

- az operátorokat egy-egy szóközzel válasszuk el: `js const x = y + z; const fruits = ['Apple', 'Orange', 'Banana'];`
- a forráskód indentálására használjunk mindig *két darab szóközt*. Modern fejlesztői felületek esetén be lehet állítani, hogy a tab karakterek helyett szóközöket tegyen: `js function toCelsius(fahrenheit) { return (5 / 9) * (fahrenheit - 32); }`
- minden kijelentés végére tegyünk pontosvesszőt: `js const values = ['Volvo', 'Saab', 'Fiat']; const person = { firstName: 'John', lastName: 'Doe', age: 50, eyeColor: 'blue' };`
- a `for` és `if` struktúrák esetén a nyitó zárójelet a sor végére helyezzük el és mindig egy szóközzel válasszuk el a paraméterlistától. A metódust záró zárójelet új sorba tegyük. Ne tegyünk elé vagy utána szóközt sem pontosvesszőt:
  `js for (i = 0; i < 5; i += 1) { x += i; }`
- az olvashatóság megőrzésének érdekében ne írjunk hosszabb sorokat, mint **100 karakter**. Ha egy sor nem fér ki 100 karakter hosszan, akkor igyekezzünk valamelyik operátornál elválasztani, például:
  „‘js // helytelen document.getElementsByClassName(demo’).getAttributeNames().forEach((attributeName) => console.log(attributeName));

// helyes document.getElementsByClassName(demo').getAttributeNames() .forEach((attributeName) => console.log(attributeName)); „‘

- egy metódusnak lehetőleg ne legyen több mint **8 paramétere**. Figyelem: 8 paraméter is már soknak számít. Ez olyankor szokott előfordulni, ha nem burkoljuk megfelelőképpen az adatainkat, például egy tömb elemeit egyenként adjuk át. Próbáljuk meg az adatainkat egy egységbe zárni és egy egységként átadni (használjunk objektumokat).
- **Ne hagyjunk „szemetet" a kódban**, pl. nem használt, megjegyzésbe tett részek, korábbi háziból vagy példaprogramból. A programkód egy szellemi értéket képvisel, próbáljuk meg minél tisztábban tartani.

Hivatkozások

1. AirBnB JavaScript konvenciók: `https://github.com/airbnb/javascript`
2. ECMAScript 6 specifikáció: `https://www.w3schools.com/js/js_es6.asp`