Scratch-től JavaScript-ig

Németh Tamás[1], Tornai Henrietta[2]

[1] tnemeth@inf.u-szeged.hu

Szegedi Tudományegyetem

[2] tornai.henrietta@stud.u-szeged.hu

SZTE TTIK

Absztrakt: Az elmúlt években egyre több olyan tananyag íródott és képzés indult, ami a Scratch használatára épül. Az eredetileg gyerekeknek szánt környezetben bármely korosztály könnyen és szórakozva szerezheti meg a programozáshoz szükséges alapokat. Azt tapasztaltuk, hogy Scratch-ben jól boldogulnak és szívesen programoznak a gyerekek, ám amikor áttérnek valamely programozási nyelv tanulására, elveszik a motiváció és sokan lemorzsolódnak. A kutatásunk fő célja feltárni, hogy pontosan melyik tananyag-egységnél és konkrétan mi okozza ezt a lemorzsolódást, és hogyan lehet ezt csökkenteni.

Kulcsszavak: gondolkodás, algoritmusok, programozás, informatika, Scratch, JavaScript, lemorzsolódás, oktatás, közoktatás

1. Bevezető

A következőkben szeretnénk ismertetni a Scratch és a Java programozási nyelv működését. A Java egy széleskörűen elterjedt objektumorientált nyelv, a Scratch pedig szintén objektumorientált alapokra épít.

A két programozási nyelv oktatásának bemutatása után a köztük lévő szakadékot elemezzük röviden. A szakadék csökkentésére egy lehetséges megoldást is mutatunk.

2. A Scratch és oktatása

Napjainkban széleskörűen elterjedt, az eredetileg gyerekeknek (8-18 éves korosztály) készült programozási környezet. Izgalmas, kreatív, azonnal látható eredménnyel, ennek köszönhetően szórakozva tanul vele gyerek és felnőtt egyaránt. Már több tanterv is íródott, ami ezen a felületen keresztül oktatja a programozás alapjait.

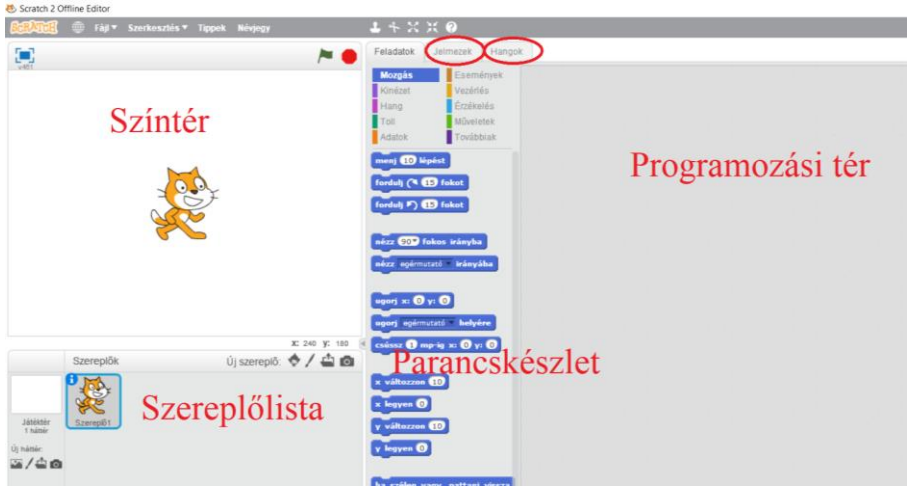
2.1. Felépítése

A tervezőknek sikerült egy jól átlátható, könnyen kiismerhető platformot kialakítani. Összesen négy fő részt különíthetünk el az indításkor megjelenő ablakban.

- *Színtér:* Itt láthatjuk a programunk eredményét. A felső sávban elhelyezett *zöld zászló* és *piros kör* a program futtatására és megállítására szolgál, az integrált fejlesztői környezetekben használt gombokhoz hasonlóan.
- *Szereplőlista:* A programunkban használt szereplők listája található itt, valamint a háttereket is itt tudjuk kezelni. A program által kínált lehetőségek közül is válogathatunk, de saját képeket is betölthetünk szereplőnek, vagy háttérnek, sőt rajzolhatunk is saját mintát a programon belül.
- *Parancskészlet:* A *Feladat* fül alatt kategorizálva találhatók a blokkok, melyekkel a kiválasztott szereplőnek (vagy a háttérnek) utasításokat adhatunk.

- *Programozási tér:* Erre a területre kell behúzni a használni kívánt blokkokat.

A kiválasztott szereplő jelmezét szerkeszthetjük, és ugyanazon szereplőnek lehet több jelmeze is. Ezeket a változtatásokat a *Jelmezek* fülre kattintás után tehetjük meg, ami a *Feladatok* fül mellett található. Az előzőek mellett lévő *Hangok* fül alatt pedig a szereplő által kiadott hangokat szerkeszthetjük.



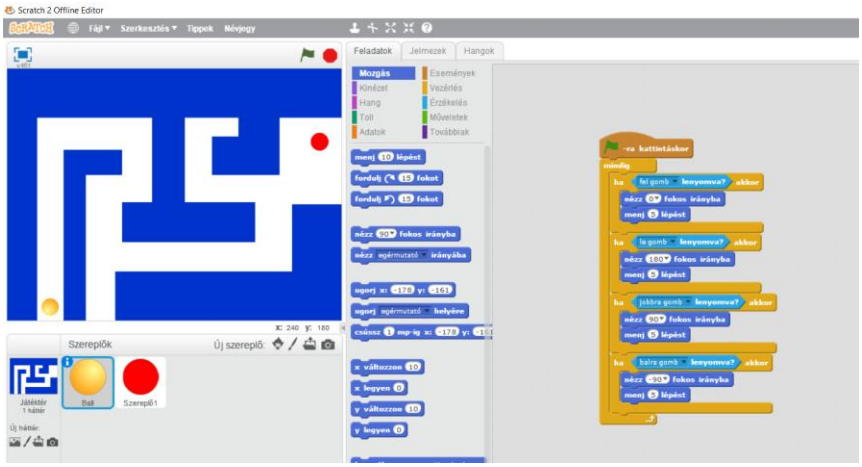
1. ábra: Scratch kezdőképernyő

2.2. Oktatása

Az első lépés a környezet megismertetése a gyerekekkel. Magyar nyelven is elérhető, könnyen átlátható, így gyorsan felfedeztethető a diákokkal. Igazából az előző pontban ismertetett főbb egységeket elég megmutatni nekik, a későbbiek során ezen információk alapján ők is megkereshetik az egyéb funkciókat, vagy a feladatok megoldásánál rávezethetők.

A tantervek többségében az első lépés a környezet megismerése után, hogy valamit mozgásra bírjunk a színtéren. A legegyszerűbben ezt az indításkor alapértelmezetten megjelenő macska szereplő megfelelő utasításával elő is idézhetjük. A Scratch megnyitásakor a *Feladatok* fül alatt található *Parancskészlet* a *Mozgás* kategória blokkjait mutatja, ahol az ehhez szükséges utasítások vannak. Így sokáig keresgélni sem kell, máris látványos eredményt érhetünk el. Szinte az első percek után sikerélményt szerezhetnek vele. Az kezdeti sikerek után érdemes hagyni, hogy egy kicsit önállóan fedezzék fel a lehetőségeket, próbálkozzanak. Hirtelen nagyon sok kérdésük lesz azzal kapcsolatban, hogy mit és hogyan lehet megvalósítani, ötletek repkednek a fejekben, és ezeket nem tartják magukban.

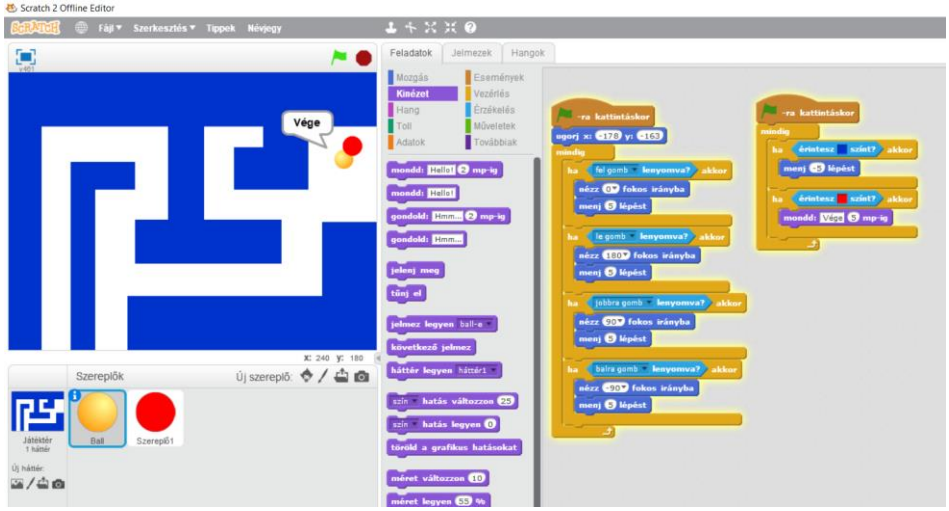
Ezután egy olyan játékot érdemes közösen megvalósítani, amit a csoport ismer, és egyszerűen kivitelezhető Scratch-ben. Ilyen például egy labirintus játék. Az alapjátékhoz megfelelő hátteret, cél mezőt és szereplőt kell alkotni. Könnyen be lehet állítani, hogy a szereplőt billentyűkkel irányíthassuk.



2. ábra: Labirintus játék - első rész

Ha ezzel elkészültünk, szinte mondani sem kell, már rögtön ki is próbálják a diákok, majd meglepődnek azon, hogy a szereplő átmegy a falon. Ez egy remek alkalom felhívni a figyelmüket arra, hogy a program csak azt tudja végrehajtani, amire utasították korábban. Rögtön látják az eredményen, hogy a program nem gondolatolvasó, és hiába szeretnék, ha a labda megütközne a falnál, az mégis átmegy rajta. Sajnos gyakori hiba a programozás alapozó kurzusokon, hogy a hallgatók elvárják a programtól azt, amit egyáltalán nem, vagy nem úgy programoztak bele, ahogy szeretnék, hogy működjön.

Egy néhány perces felvezetés, majd közös ötletelés után a gyerekekkel együtt már orvosolható a falnak ütközés problémája. Hasonló gondolkodással az is megoldható, hogy célba érés esetén történjen valami.



3. ábra: Labirintus játék – második rész

Az alapjátékot érdemes közösen elkészíteni. Fejleszteni sokféleképpen lehet, ahogy egyre több funkciót megismernek a Scratch-ben. Akár a következő órákon együtt is megvalósíthatóak, de ajánlottabb egy új játékot készíteni, és azáltal fedezni fel újabb lehetőségeket, így változatos marad a képzés, és a kreativitásukat sem gátoljuk. Mivel minden

diák más, így nem szerethetik ugyanazt a játékot, és nem élvezik egyformán bármelyik elkészítését. A számukra legkedvesebb játékot úgy is tökélyre fejlesztik otthon a saját kedvük szerint.

A Scratch-nek van online és offline változata is. Az offline verzió ingyenesen letölthető, és percek alatt telepíthető. Az online változatot regisztráció nélkül is ki lehet próbálni, regisztrációt követően pedig online bárhonnan elérhetjük előző munkáinkat, és mások nyilvános projektjei között is nézelődhetünk.

A Scratch-ben közvetlen vagy közvetett módon megvalósíthatóak a következő programozási problémák:

- elágazás (egy vagy kétágú)
- ciklus (számlálós, végtelen, feltételes)
- szekvencia
- változó
- lista
- eseményvezérlés
- többszálúság
- rekurzió
- eljárás
- függvény

3. A Java nyelv oktatása

A Java egy széleskörűen elterjedt, tisztán objektumorientált programozási nyelv. A fejlesztéshez, a fordításhoz, valamint a futtatáshoz szükséges program és környezet ingyenesen elérhető az interneten. A programkód hordozható, és a számítógépes alkalmazások mellett, telefonon és szervereken futó alkalmazások készítésére is alkalmas.

A tanmenetek az első órát a fejlesztői környezet megismertetésére szánják, illetve a nyelv alapinformációinak ismertetésére. Az első program, amit rögtön ezen az órán, a motiváció felkeltése érdekében ajánlott megmutatni, esetleg megíratni a diákokkal, a szokásos “Hello Világ!” kiíratása.

Ha a “Hello Világ!” típusú feladaton túl vagyunk, akkor a továbbiakban folytathatjuk a képernyőre írással kapcsolatos ismeretek átadásával. Ez azért is hasznos, mert rögtön látható a munkájuk eredménye.

A további egységek tanításának egy lehetséges sorrendje:

- azonosítók, adattípusok, változók
- konstansok, kifejezések, operátorok
- utasítás, blokk, vezérlési szerkezetek (elágazások, ciklusok)
- adatszerkezetek (egy- és többdimenziós tömbök, karakterlánc)
- osztályok, objektumok, alaposztályok
- *private* és *public* hozzáférés
- öröklődés, polimorfizmus, absztrakt osztályok
- kivételkezelés
- fájlkezelés
- dátum és időkezelés
- *Collection* interfész (halmaz, rendezett halmaz, lista)
- hashtábla, dinamikus vektor, verem

Ezen ismeretek elsajátítása és gyakorlása során a legtöbbször felöltöztetett feladatokkal találkozik a diák, amelyek nem táplálják a motivációt, és nem látják, hogy miért is lenne ez

számukra hasznos. Ha nem felöltöztetett, akkor valamilyen fizikai jelenség modellezése, vagy matematikai probléma megoldása kerül feladatként kitűzésre, azonban az ezen tárgyakkal szembeni ellenszenvük miatt, a motiváció csökkenése jellemző.

4. Távolság a kettő között

Az ismertetések alapján már érezhető miben adnak más élményt, de néhány gondolatban szeretnénk kiemelni a diákok által szerintünk fontosnak gondolt különbségeket.

Míg a Scratch-ben folyamatosan látják a munkájuk eredményét, ami színes, mozgó, ingergazdag, addig Java programozás közben kevésbé szembetűnő az eredmény. A másik nagy különbség, hogy Scratch-ben elég csak a már meglévő blokkokat a megfelelő helyre húzni, Java-ban pedig már a diáknak kell begépelnie a kódot, ami sokkal több hibázási lehetőséget rejt a pontos gépelés miatt. Egy elfelejtett írásjel, vagy egy elgépelt szó következtében már vagy nem úgy, vagy egyáltalán nem fog működni a kódunk. A fejlesztői környezet azonban sokszor a hozzájárul ezen hibák minimalizálásához.

A fejlesztői környezet segítsége ellenére is könnyű tabulálatlan, vagy rosszul tabulált kódot gépelni. A Scratch blokkjainak kapcsolódásai megoldják a tabulálási problémát.

```
binaris.java
1  import java.util.*;
2  public class fa {
3      int elemszam ;
4      class fapont{
5          int key, value ;
6          fapont bal, jobb, apa ;
7      }
8      fapont nil, gyoker ;
9      void fa() {
10         fapont q = new fapont() ;
11         nil=q ;
12         gyoker=q ;
13         elemszam=0 ;
14     }
15     boolean beszur(int x, int v) {
16         fapont p,papa ;
17         p=gyoker ;
18         papa=nil ;
19         while (p!=nil && p.key!=x) {
20             papa=p ;
21             if (x<p.key) p=p.bal;
22             else p=p.jobb;
23         }
24         if (p==nil) {
25             fapont ujpont = new fapont();
26             if (gyoker==p) gyoker=ujpont ;
27             ujpont.key=x ;
28             ujpont.value=v ;
29             ujpont.bal=nil;
30             ujpont.jobb=nil;
31             ujpont.apa=papa;
32             if (papa!=nil) {
33                 if (x>papa.key) papa.jobb=ujpont;
34                 else  papa.bal=ujpont;
```

4. ábra: Java kódrészlet

5. ábra: Scratch kódrészlet

Egy Java kód megírásához ismerniük kell a nyelv szintaktikáját is, nem elég ha tudja az algoritmust, amivel meg lehet oldani a problémát. Scratch-ben csak azt kell tudnia, hogy hol találja a megoldáshoz szükséges blokkot. Látja az összes lehetőséget, és kikeresheti a megfelelőt, így az algoritmikus gondolkodásra nagyobb hangsúlyt lehet fordítani.

A Scratch-ről Java-ra való áttérésben a legnagyobb buktatót a programkód írása jelenti. Kezdve azzal, hogy nem elég csak a megfelelő blokkot, a megfelelő helyre húzni, és helyette gépelni kell a kódot, de még arra is figyelniük kell, hogy megfelelő írásjelet használjanak, ott ahol kell.

Ha még ezekkel a nehézségekkel sikeresen meg is küzd a diák, sokkal ingerszegényebb eredményt lát viszont a képernyőn a kezdetekben.

5. Ötletek a távolság csökkentésére

Olyan megoldást szeretnénk találni, amivel könnyebbé tehetjük ezt a váltást, motiváltak maradhatnak a gyerekek. Egy lehetséges megoldást látunk a következőkben ismertetett rendszerekben.

5.1. JavaScript

A JavaScript egy scriptnyelv, weboldalak dinamikussá tételére használják elsősorban. A felhasználó mindamellett, hogy információt szerez a weboldalról, ő is küldhet információt kattintással, szöveg bevitelével. Ezeket az eseményeket kezelhetjük JavaScript-tel. Legfőbb alkalmazási területe az űrlapok kezelése.

A JavaScript kódhoz nincs szükség speciális környezetre, csak egy szövegszerkesztőre és egy böngészőre. A böngésző sorról sorra haladva hajtja végre a parancsokat. A JavaScript kódot HTML állományhoz kell kapcsolni, erre több lehetőség is van. Beágyazhatjuk a HMTL kódunkba, vagy külön fájlban is írhatjuk (.js kiterjesztés), ilyenkor csatolni kell a HTML kódba.

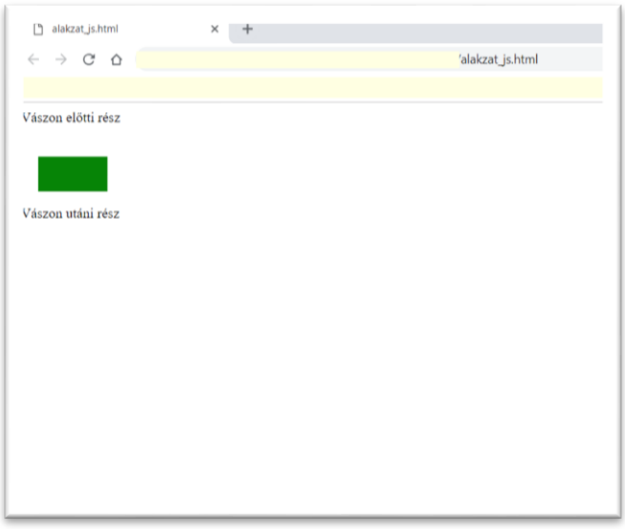
Egy egyszerű weblapot már pár sor kóddal létre lehet hozni, az eredményt mentés után a böngészőben megtekinthetjük. Ha ehhez JavaScript kódot is társítunk, interaktívvá tehetjük az oldalt,ami egy frissítés után a megváltozott weblapot mutatja, így azonnali sikerélményt garantál.

Nem csak űrlapok kezelésére képes, szép színes minták rajzolására is, aminek köszönhetően a Scratch-hez hasonlóan a munkánknak látványos kimenetele lesz. Lehetőség van minden apró változtatás esetén megfigyelni miben különbözik a frissített weblap. A grafikus elemek megjelenítéséhez a *canvas* HTML elemre van szükségünk.

alakzat_js.html

```
1  <html>
2  <head>
3  </head>
4  <body>
5   <p>Vászon előtti rész</p>
6   <canvas width="120" height="60"></canvas>
7   <p>Vászon utáni rész</p>
8  <script>
9     let canvas = document.querySelector("canvas");
10    let context = canvas.getContext("2d");
11    context.fillStyle = "green";
12    context.fillRect(20, 20, 80, 80);
13 </script>
14 </body>
15 </html>
```

6. ábra: Grafikus elem megjelenítése - kód



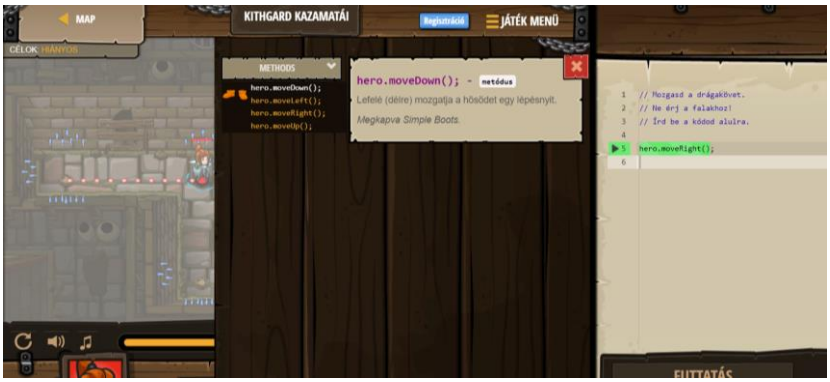
7. ábra: Grafikus elem megjelenítése - weblap

Mindamellett, hogy a látványvilágában majdnem olyan szintű élményeket nyújt, mint a Scratch, a JavaScript a C++ és a Java szintaxisán alapszik, így átvezetésként jól működhet a két programozás között.

A JavaScript oktatásának kezdésére kiváló lehetőség a CodeCombat. A későbbiekben a látványos eredmény vonalon maradva remek környezetek és kisegítő lehetőségek várnak ránk. Ezek közül is ismertetnénk néhányat.

5.1.1. CodeCombat

A CodeCombat egy olyan platform, ami játékkal motivál programozásra. Python mellett JavaScript nyelv is elérhető. A feladat pedig az, hogy a választott programozási nyelven adott utasításokkal segítsük hősünket abszolválni a pályákat. A sikeres teljesítés pontokat ér, szinteket léphetünk. Van fejlődési lehetőség, a diákok között versenyhelyzet alakulhat ki. Mint mindenhol, itt is több jó megoldása lehet egy problémának, ilyenkor a rövidebb kód több pontot ér, ezzel is ösztönzi a felhasználót arra, hogy megtalálja a legjobb algoritmust.



8. ábra: CodeCombat felület játék közben

Ez ideális felület, a JavaScript játékos bevezetésére, még jobban csökkentve a szakadékot a Scratch blokk-alapú kódolása, és a Java között.

5.1.2. HTML SVG

Az SVG egy XML alapú leírónyelv. Segítségével felbontástól függetlenül készíthetünk álló és mozgó képeket egyaránt. A *canvas* elem ezzel ellentétben csak előre meghatározott felbontással rendelkező grafikákat készíthetünk.

5.1.3. Node.js

A Node.js újabb lehetőséget ad a JavaScript programunk futtatására, nem webszerver. Ingyenes letölthető telepítők installálása után vagy shell-ben futtatjuk a kódunk, vagy pedig a JavaScript fájlt adjuk át neki.

5.1.4. Vue-CLI

A Vue.js egy reaktív keretrendszer. Egyszerű, könnyen elsajátítható lépésről lépésre, és akár komolyabb alkalmazásokat is készíthetünk vele. A Vue-CLI pedig ezt a rendszert támogató eszköz. Teljes grafikus környezetet biztosít Vue.js alkalmazásunk készítéséhez.

9. ábra: Vue-CLI logó

6. Hogyan mérjük?

Terveink között szerepel a módszer kipróbálása, és tesztelése. Jelenleg még a megfelelő tesztcsoport keresése zajlik. Szeretnénk egy előzetes ismeret és motiváció-felmérést végezni, a folyamat közben és végén pedig megismételni.

Jelenleg is vannak olyan iskolák, ahol a bevezető képzés Scratch-ben zajlik, majd JavaScript-re térnek át. Velük együttműködve, a tapasztalatok megosztással tökélesíthető a módszer. Minél szélesebb körben próbálhatjuk ki, annál több tényező által javítható, pontosítható a tanmenet.

7. Összegzés

A mai világban már más igényei vannak a diákoknak. Jobban vágynak az erős és állandó külső impulzusokra, így ha első programozással kapcsolatos élményük a többsoros kódgépelés, nem lesz számukra elég érdekes ahhoz, hogy továbbiakban is azzal foglalkozzanak. Szerencsére egyre több lehetőség van a kreatív programozás oktatásra. Már csak kísérletező szellemű oktatókra, diákokra, vezetőkre és megfelelő környezetre van szükség ezek közoktatásba történő beillesztéséhez.

Irodalom

1. Takács Valéria: *Tananyagkészítés a Sratch programozási környezethez*, ELTE IK, Budapest (2009) http://grafit.netpositive.hu/download/scratch/szakdolgozat.pdf (utoljára megtekintve: 2018.11.03.)
2. https://pcworld.hu/szoftver/scratch-suli-kezdjunk-el-programozni-175636.html (utoljára megtekintve: 2018.11.03.)
3. https://scratch.mit.edu/about (utoljára megtekintve: 2018.11.03.)
4. Kovács Zsuzsanna: *JAVA programozási nyelv NetBeans fejlesztőkörnyezetben*, Budapest (2009) http://www.petrik.hu/files/tamop/SZINFO13/SZINFO13_TK.pdf (utoljára megtekintve: 2018.11.04.)
5. Tömösközi Péter: *Programozás Javában*, Eger (2013) http://mek.oszk.hu/14200/14282/pdf/14282.pdf (utoljára megtekintve: 2018.11.04.)
6. Horváth László András: *A Java nyelv tanítása középiskolában*, DE IK, Debrecen (2009) https://dea.lib.unideb.hu/dea/bitstream/handle/2437/88910/Szakdolgoat.pdf?sequence=1 (utoljára megtekintve: 2018.11.04.)
7. http://www.inf.u-szeged.hu/~tnemeth/source/faclassjava.txt (utoljára megtekintve: 2018.11.04.)
8. Horváth Győző , Menyhárt László Gábor: *Oktatási környezetek vizsgálata a programozás tanításához*
   In: Szlávi Péter, Zsakó László (szerk.)
   INFODIDACT 2014: Informatika Szakmódszertani Konferencia. Konferencia helye, ideje: Zamárdi, Magyarország, 2014.11.20-2014.11.21. Budapest: Webdidaktika Alapítvány, 2014. (ISBN:9789631206272)
9. Németh Tamás, Széll Réka, Tornai Henrietta: *Az algoritmikus gondolkodás fejlesztésének fontossága a közoktatásban*
   In: Szlávi Péter, Zsakó László (szerk.)
   INFODIDACT 2017: Informatika Szakmódszertani Konferencia. Konferencia helye, ideje: Zamárdi, Magyarország, 2017.11.23-2017.11.25. Budapest: Webdidaktika Alapítvány, 2017. (ISBN: 978-615-80608-1-3)
10. http://www.inf.u-szeged.hu/~tnemeth/source/faclassjava.txt (utoljára megtekintve: 2018.11.04.)
11. https://codecombat.com/ (utoljára megtekintve: 2018.11.04.)
12. https://html5.ugyesen.com/2013/06/a-html5-inline-svg-hasznalata/ (utoljára megtekintve: 2018.11.04.)
13. https://eloquentjavascript.net/17_canvas.html (utoljára megtekintve: 2018.11.04.)
14. https://schonherzbazis.hu/hirek/reszletek/Node.js-teljesen-kezdoknek (utoljára megtekintve: 2018.11.04.)
15. https://ithub.hu/blog/post/Bevezetes-a-Vuejs-keretrendszerbe/(utoljára megtekintve: 2018.11.04.)
16. https://cli.vuejs.org/ (utoljára megtekintve: 2018.11.04.)