**Mladi za napredek Maribora 2014**

**31. srečanje**

# ORGANIZATOR

**RAČUNALNIŠTVO**

Inovacijski predlog

Avtor: ĢAN MAJERIĻ

Mentor: JOĢE ĠTRUCL

Ġola: SREDNJA ELEKTRO-RAĻ UNALNIĠKA ĠOLA MARIBOR

**2014, Maribor**

# Vsebina



# Kazalo slik

# 1. Povzetek

Mentor me je seznanil s programom ExeCute, ki ga je razvila mariborska fakulteta za naravoslovje in matematiko, na podlagi že obstoječe zamisli, več o programu bom predstavil tudi tekom te raziskovalne naloge, v glavnem je program namenjen e-gradivom. Ta aplikacija omogoča da lahko dodajaš svoje »module« oz. javascript applete. Poleg tega pa mi je mentor predstavil še zanimivo javascript knjižnico, ki izrisuje HTML5 slike (canvase) - KineticJS.

Ker sam nisem prišel do nobene zamisli, hkrati pa sem imel željo izvesti raziskovalno nalogo iz tega področja (računalništva), zaradi zanimivih vsebin, mi je mentor sam predlagal da bi mu pomagal pri kreiranju teh modulov in nekaterih applet-ov, kateri bi mu nasploh olajšali kreiranje le-teh.

## 2. Uvod

Čeprav je program že nekoliko brez podpore (iz slovenske strani) in nekaterimi nedokončanimi funkcionalnostmi, zna biti še zmeraj uporaben, saj pri učenju zmeraj pride prav kakšna stvar več.

Dani moduli, ki sem jih pregledal so precej nedodelani in občutek sem imel da se da stvar izboljšati. Da se stvar da izboljšati, pa ni le en od ciljev raziskovalne naloge, poenostavljanje je pomembna stvar, saj bi lahko spodbudilo k ustvarjanju novih ti. modulov.

Namen je da se v javascript jeziku, ki se nato zapakira v jsx file - ki zajema celoten projekt (resources in dodatnimi javascript knjižnicami), naredi/uredi novi/obstoječi modul ter se doda knjižnica ki poenostavlja delo z moduli. Čeprav je za javascript na voljo precej knjižnic (angl. Framework-ov, kot so: »AngularJS. KnockoutJS, YUI«, vsak od teh ima svoj namen in prednost, odvisno od problema ki se rešuje pa izberemo najbolj ustreznega), sem sam razvijal v KineticJS, dani primeri na internetu so izgledali dovolj zanimivi, da jo uporabim.

## 3. Metodologija dela

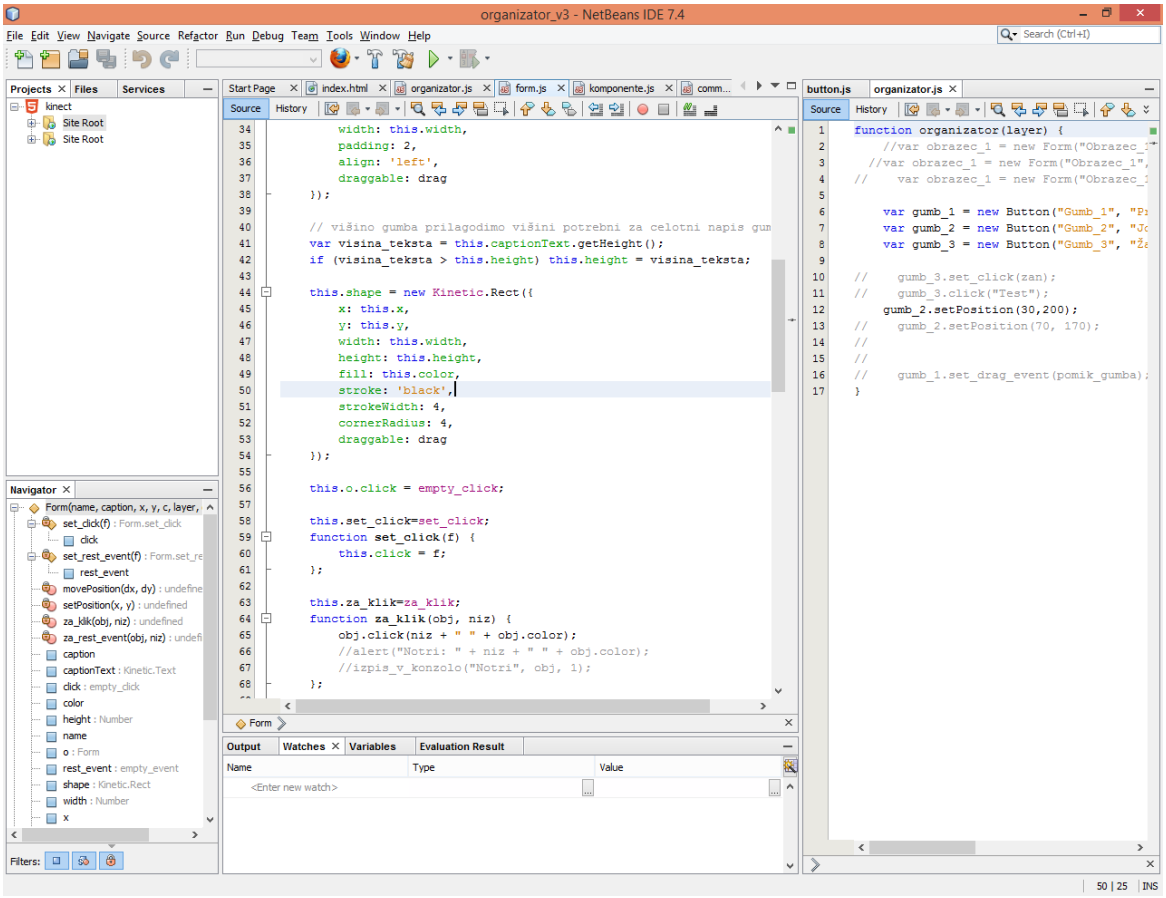
Ko je padla zamisel, sem sprva pregledal kako kaj v Javascriptu stvari potekajo, kakšen jezik je, sintaksa, logika… Sicer sem jezik poznal že od prejšnjih let na šoli (kjer smo sicer programirali na precej nižji ravni), poleg tega znam tudi nekaj drugih jezikov, zato ni bilo nič kaj novega glede samega jezika, in sem stvari lahko usmeril na bolj pomembna področja.

Za začetek je bilo potrebno namestiti delovno okolje (angl. IDE), spet je bila izbira kar precejšnja ampak sem se odločil za NetBeans, ker so dosegljivi tudi na šolskih računalnikih. Z okoljem sem se nato lahko lotil spreminjanja nekaj že obstoječih »aplikacij«, ustvarjenih z KineticJS. Nekaj časa sem porabil za pregled zgradbe samega frameworka, za vajo sem jih nekaj spremenil in že sem vedel na kakšni osnovi je sestavljen.

Šele na tej točki sem bil seznanjen s programom ExeCute in na sam način dodajanja ustvarjenih aplikacij v sam program.

## 4. Razvojno okolje (IDE) – NetBeans

NetBeans je razvojno okolje, osredotočeno predvsem na programiranje v Javanskem jeziku, lahko pa se uporablja tudi za druge kot so PHP, C/C++ in HTML. Okolje je spisano v Javi in je multi-platform, kar pomeni da teče na vseh bolj razširjenih operacijskih sistemih (Windows, Linux, MacOS in drugimi)




: Slika 1: Primer vmesnika v NetBeans 7.4 (Vir: domači računalnik)

## 5. Javascript

Javascript je dinamičen programski jezik, vgrajen v spletne strani, ki dovoljuje interakcijo skript na strani klienta, z uporabnikom, kontrolira brskalnik in komunicira asinhrono. Vse to naredi spletne strani dosti bolj interaktivne, preprostejše in bolj privlačne za uporabo. V zadnjem času se je začel tudi precej pogosto uporabljati v izdelavi iger in razvijanju namiznih aplikacij.

Jezik je sicer povsem neodvisen od jave, kljub temu da si z njo delo marsikatero lastnost in strukturo, pa sta jezika nepovezana in različna. Imenuje se tudi skriptni jezik, ki je po sintaksi precej podoben C-jevski, za programiranje pa se lahko uporablja objekte.

Javascript se večinoma uporablja za ustvarjanje dinamičnih spletnih strani. Program se vgradi ali pa vključi v HTML, da opravlja naloge, ki niso mogoče samo s statično stranjo. Na primer odpiranje novih oken, preverjanje pravilnost vnesenih podatkov, enostavni izračuni ipd. Veliko je odvisno tudi od brskalnikov saj vsak uporablja javascript na različne načine, in je potrebno za večjo podporo pisati tudi več različnih funkcij. JavaScript se zadnje čase uporablja tudi v različnih orodjih, ne samo v spletu. Adobe Reader ga podpira v datotekah PDF, lahko pa programiramo tudi za Windows operacijski sistem.


Slika 2: Vzorec javascript-a (Vir: google slike)

## 5.1. Nastanek

Leta 1995 so pri podjetju Sun razvili nov programski jezik, poimenovan Java. Namenjen je pisanju programov, ki jih lahko nespremenjene izvajamo na vsakem računalniku, ne glede na vrsto procesorja ali operacijski sistem. Zaradi te lastnosti so ti programi idealni za vključitev na spletne strani, saj tam ne vemo vnaprej, na kakšnih vrstah računalnikov bodo uporabniki prebirali naše strani. Da bi izboljšali podporo programčkom, napisanim v jeziku Java, so pri podjetju Netscape še istega leta razvili programski jezik, ki so ga sprva poimenovali »LiveScript«, iz tržnih razlogov pa so ga kmalu preimenovali v Javascript (zaradi popularnosti Jave). To ni bila ravno dobra poteza, saj je preveč uporabnikov mislilo, da je Java in Javascript isto. Kljub začetnim pomanjkljivostim (slaba varnost, nestandardiziranost, pomanjkanje

razvojnih orodij) je Javascript sčasoma postal eno od najbolj priljubljenih orodij za izdelavo dinamičnih spletnih strani.

### 5.2. Framework

Javascript knjižnice so kompleti, že vnaprej spisane kode, ki nam oz. programerju omogočajo lažje razvijanje poljubne aplikacije in so narejene za točno določeno zadevo (če potrebujemo funkcije za delo z slikami, potem izberemo kakšno knjižnico za slike). Za razliko od tega Frameworki ponujajo nekoliko razširjene zmogljivosti in lastnosti, ki jih v klasičnih knjižnicah ne najdemo. Večina frameworkov je izdanih pod določeno licenco.

Nekaj primerov knjižnic: jQuery, MooTools, Prototype

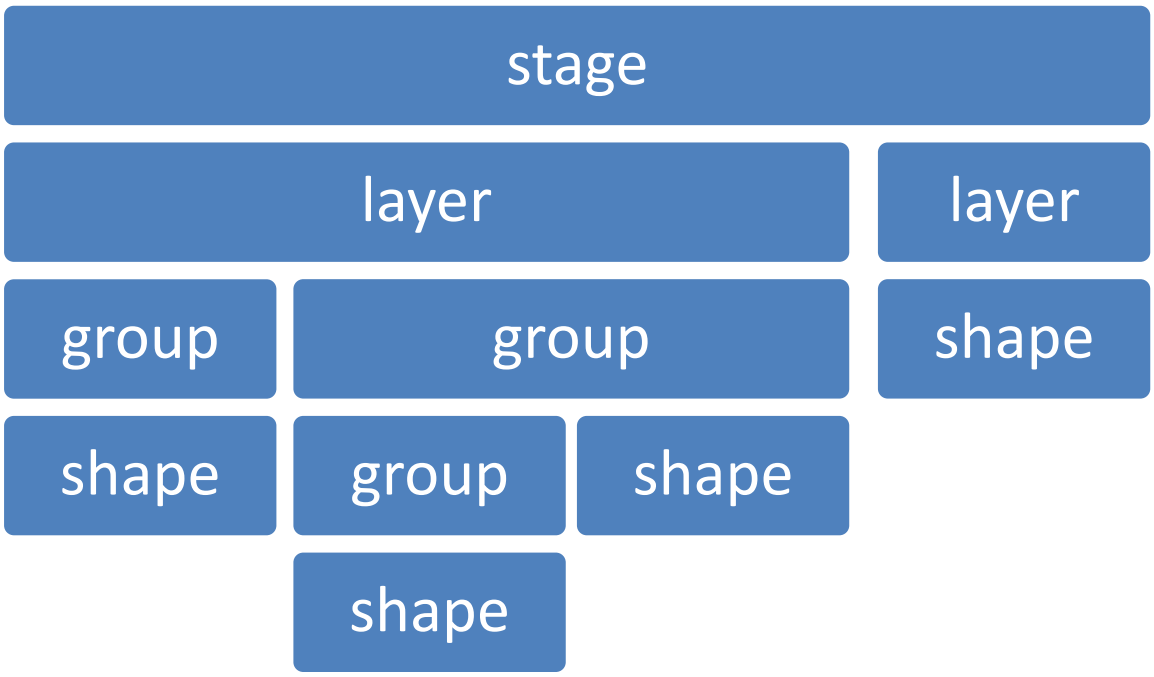
Nekaj primerov frameworkov: YUI, Dojo, AmpleSDK

## 6. KineticJS

KineticJS je HTML5 Javascript framework za prikazovanje kanvasov. Omogoča hitre animacije, prehode, uporablja layerje oziroma plasti in dogodke. Del na katerem se vse izvaja, izrisuje se imenuje »Stage«.

Na ta del lahko dodajamo različne stvari (vse lahko najdemo v njihovi dokumentaciji, kjer je spisano vse kar potrebujemo da jih v celoti uporabljamo). Tem stvarem lahko nastavljamo dogodke, jih premikamo, večamo/manjšamo, rotiramo,…

Stage je sestavljen iz različnih, poljubno definiranih layerjev (plasti), na katere dodajamo zgoraj omenjene elementi, lahko pa ustvarimo tudi grupe.




Slika 1: Hierarhični graf, ki prikazuje kako je LAHKO sestavljena aplikacija

## 7. ExeCute

Pred nekaj leti je nastalo odprtokodno avtorsko orodje eXe, ki je ponudilo enostaven uporabniški vmesnik, ki je omogočal enostavno izdelavo e-gradiv, ki vsebujejo besedilo, zvok, sliko, video, javanske programčke, datoteke, različne kvize in hierarhično zasnovo strani. Tudi v Sloveniji je v zadnjem obdobju nastalo precejšnje število gradiv v tem orodju, največ v okviru razpisov MŠŠ. Nekatere prednosti orodja eXe pred ostalimi orodji iz vidika slovenskih avtorjev e-gradiv so:

- poslovenjen uporabniški vmesnik,
- podobno delovno okolje kot v učnih okoljih LMS,
- samostojno orodje, ni potrebne internetne povezave za delovanje,
- zastonj.

## 8. Primer knjižnice

Z vsemi temi sredstvi se sedaj da povsem enostavno sestaviti enostavni modul za ExeCute. Za začetek najprej k knjižnici, ki programerju pomaga pri razvoju takšnih modulov za program ExeCute.

Torej ustvaril sem javascript datoteko, ki jo uporabnik vključi v svoj dokument/program, ki se imenuje komponente.js. Kar omogoča je to, da lahko enostavno v svoj program vključimo (zaenkrat) gumb in formo. To vse bo lažje zaradi dodatnih funkcij in nastavitvami.

Primer dodajanja gumba in forme v program:

```
var gumb_1 = new mBtn("gumb_1", "Gumb", 30, 30,200,200, btnLayer, true);
var form_1 = new mForm("form_1", "Form / Obrazec", 150, 30, formLayer, true);
formLayer.draw();
```

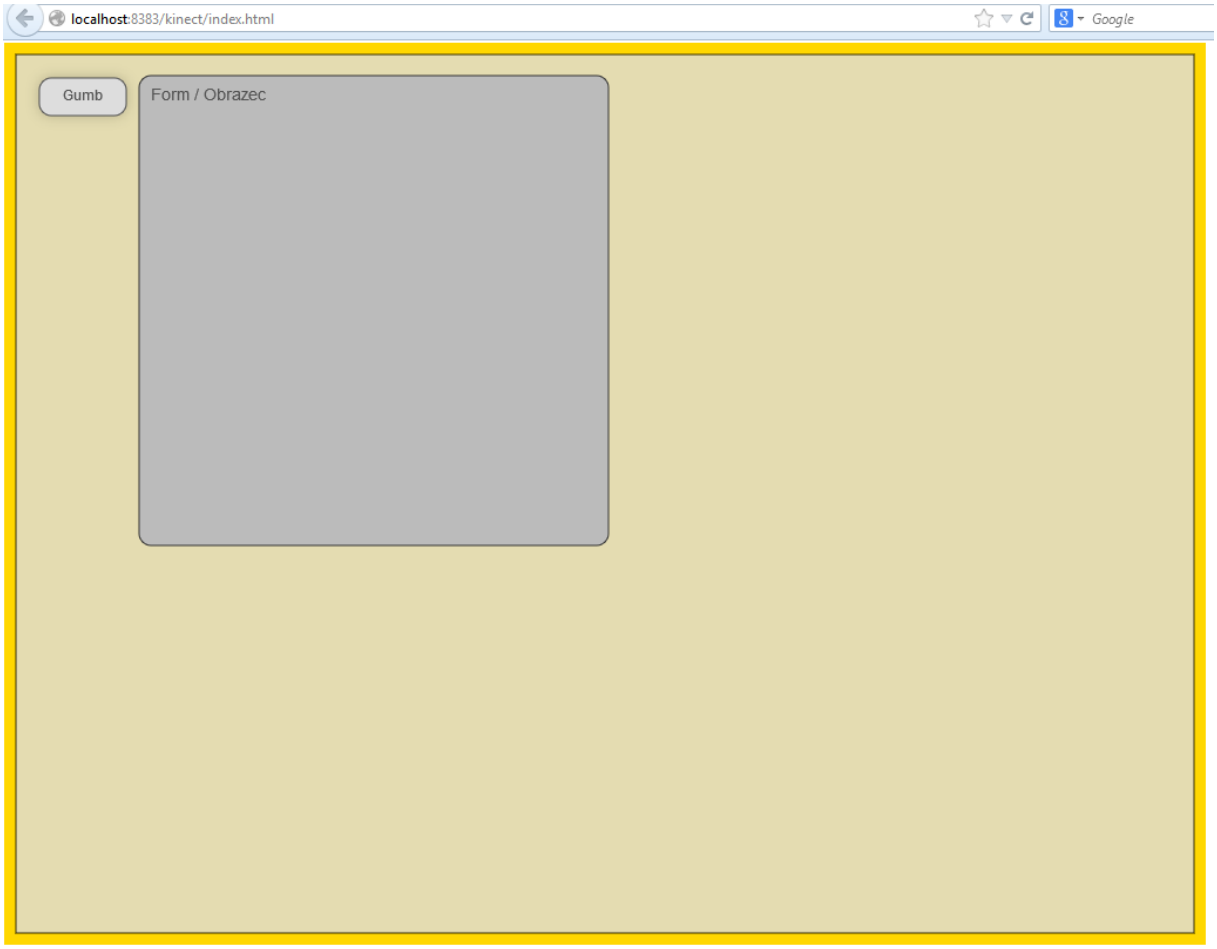
Te vrstice ustvarijo 2 nova objekta »mBtn« in »mForm«. V njega lahko dodamo svoje lastnosti kot so:

- Ime gumba (gumb_1)
- Napis ki bo prikazan v gumbu (Gumb)
- Poziciji x in y (30,30)

- Širino in dolžino (200,200)
- Layer na katerem bo ta gumb (btnLayer)
- In nastavitev true ali pa false za lastnost »drag« ki omogoča premikanje gumba poljubno

Za form je podobno le da jo dodamo na svoj layer.

Sedaj pa še prikaz kaj bi naj to izrisalo.



Ta primer teče na lokalnem strežniku, in je dodan v index.html datoteko.

Gumbu pa lahko s metodo »set_click« (ali tudi »set_drag_end«, »set_drag_start«, »set_mouse_over«) določimo funkcijo ki se izvrši ob kliku. Ta mora biti napisana naprej, njeno ime pa podamo v metodo.

```
var gumb_1 = new mBtn("gumb_1", "Gumb", 30, 30,200,200, btnLayer, true);
gumb_1.set_click(test);
```

Izvedlo bo funkcijo »test«, ki bo v alertu izpisala moje ime (»Jaz sem Žan«)

```
function test() {
alert("Jaz sem Žan. ");
}
```

Tako pa izgleda gradnik, gumba in forma.

```
function mBtn(name, text, x, y, w, h, layer, drag) {
    this.name = name;
    this.text = text;
    this.x = x;
    this.y = y;
    this.layer = layer;
    this.d = 20; //razmik besedilo-gumb
    this.drag=drag;
    this.width = w;
    this.height = h;
    this.o = this;
    this.movePosition = movePosition;
    this.setPosition = setPosition;

    this.btnGroup = new Kinetic.Group({
        draggable: this.drag
    });
    //grafične lastnosti gumba
    this.btnText = new Kinetic.Text({...10 lines });
    this.btnShape = new Kinetic.Rect({...13 lines });
    var asd = this;
    this.click = empty_click;

    this.set_click = function set_click(f) {
        this.click = f;
    };


    this.o.btnGroup.on('click', function() {
        asd.click();
    });
```

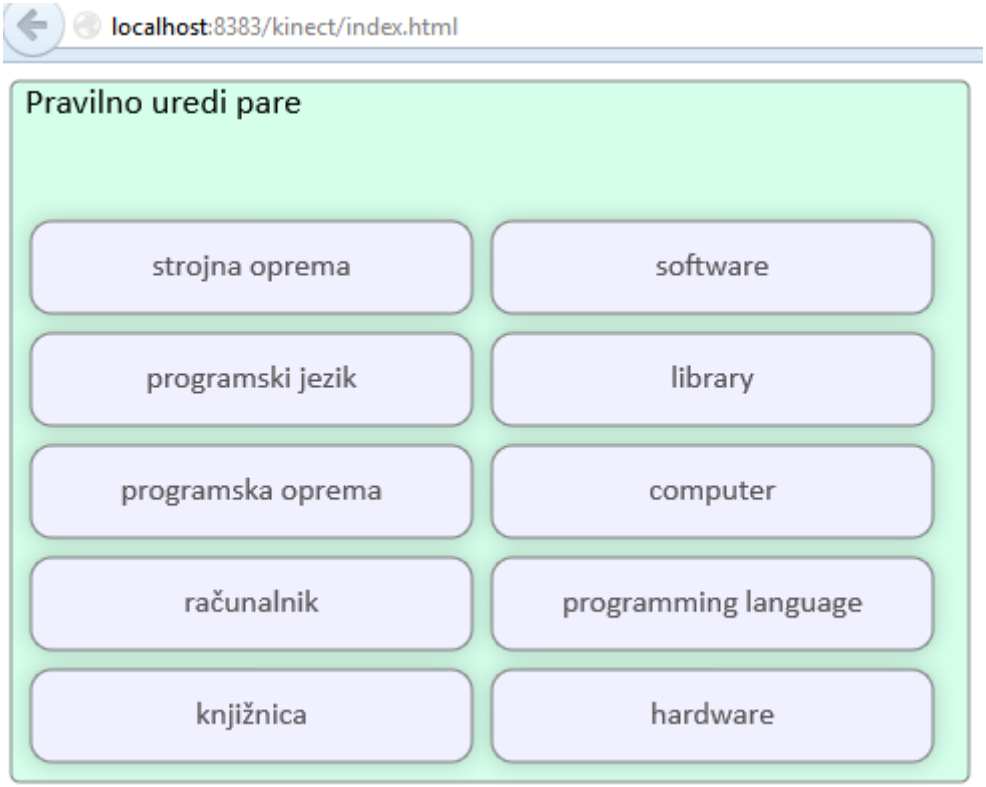
```
function mForm(name, text, x, y, layer, drag) {
    this.name = name;
    this.text = text;
    this.x = x;
    this.y = y;
    this.layer = layer;
    this.o = this;
    this.d = 20; //razmik besedilo-gumb
    this.drag=drag;

    this.formGroup = new Kinetic.Group({
        draggable: this.drag
    });
    //grafične lastnosti gumba
    this.formTxt = new Kinetic.Text({
        x: x,
        y: y,
        text: text,
        fontSize: 14,
        text: text,
        fill: '#555',
        padding: 10,
        align: 'center'
    });
    this.formShape = new Kinetic.Rect({
        x: x,
        y: y,
        stroke: '#333',
        strokeWidth: 1,
        fill: '#bbb',
        width: 400,
        height: 400,
```

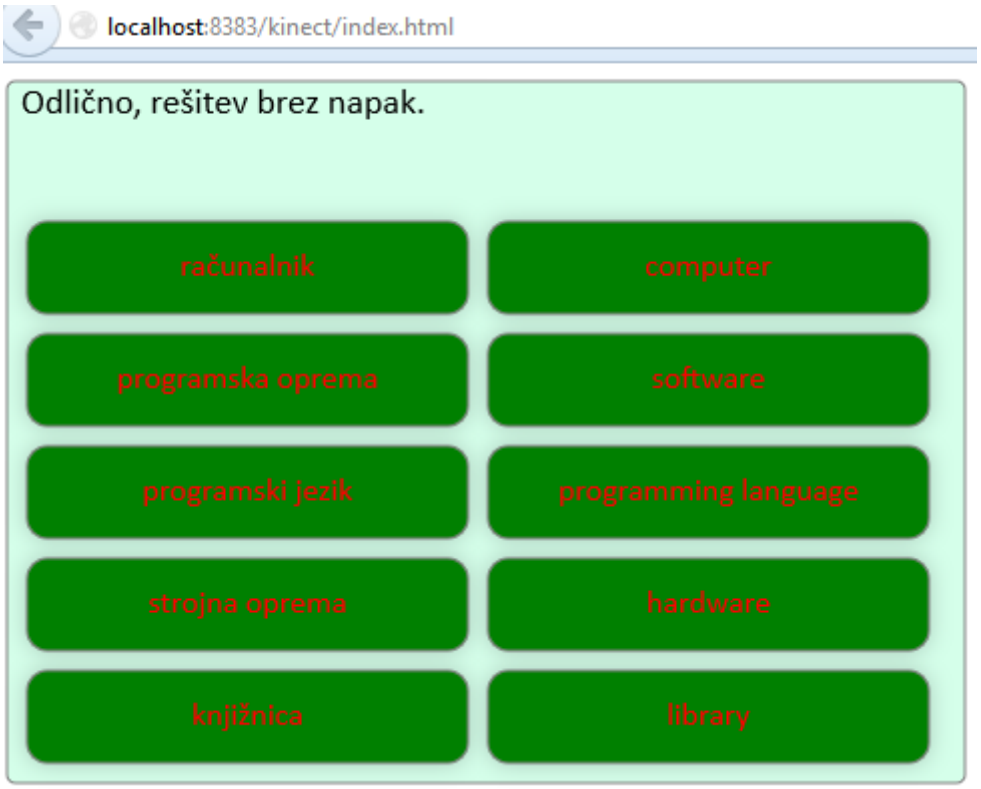
## 9. Primer modula

To knjižnico sem sedaj vključil v projekt, kjer sem ustvaril »povezovalno igro«, kot e-gradivo in jo nato dodal v program ExeCute.

Program ustvari dva stolpca gumbov, ki so shranjeni v polju (oz. Arrayju), namen je da povežeš odgovore iz prvega stolpca v drugega, program pa na koncu preveri če je rešitev pravilna.




Potem pa z miškinim premikom (»drag«) postavimo pojme na svoje pravo mesto. Ko vse pojma postavimo pravilno se pojavi gumb preveri, njegov klik pa vrne pravilne oziroma nepravilne rezultate.




Igra deluje tako da so pojmi prvega stolpca shranjeni v polju1, pojmi drugega stolpca pa v polju2. Lahko jih poljubno dodajamo – več kot jih je daljša je dolžina vendar moremo paziti da so pravilno zapisani saj, pojmi ki so na enakem mestu (z enakim indeksom) spadajo skupaj in so pravilni. Razporedijo se popolnoma naključno na začetku programa.

Aplikacijo vključimo v dokument in kreiramo objekt organizator.

```
var pojmi_levo = new Array("računalnik", "programski jezik",
"strojna oprema", "programska oprema", "knjižnica");
var pojmi_desno = new Array("computer", "programming language",
"hardware", "software", "library");
organizator(pojmi_levo, pojmi_desno, animalLayer);
```

Organizator sprejme polje1, polje2 in plast (layer) na katerem se gumbi/forma izrisuje.

```
function organizator(pojmi_levo, pojmi_desno, layer) {

    dodatna_visina_gumba = 10;
    y_zgornjega_roba_za_gumbe = 50 + 2 * dodatna_visina_gumba;
    obrazec_1 = new mForm("Obrazec_1", "Pravilno uredi pare", x_okna, y_okna, sirina_okna,
    (pojmi_levo.length * (y_zgornjega_roba_za_gumbe)), '#D5FFEA', layer, false);

    gumb_preveri = new mBtn("Gumb_Preveri", "Preveri", x_okna + 360,
    y_okna + 10, 100, 10, '#00FFCC', layer, false);
    gumb_preveri.set_click(preveri);
    //gumb_preveri.click("Test");
    gumb_preveri.shape.hide();
    gumb_preveri.captionText.hide();

    var ji = 0;
    for (var i = 0; i < pojmi_levo.length; i++) {...13 lines }

    // mix (shuffle) texts
    for (var key in pojmi_levi) {...13 lines }
    for (var key in pojmi_desni) {...13 lines }

    for (var i = 0; i < pojmi_levo.length; i++) {...10 lines }

    poravnaj_gumbe();
}
```

Na začetku so nastavljene vrednosti za bolj optimalen izgled gumbov, potem se ustvari form na kateri se gumbi ustvarjajo. Ustvari se »gumb_preveri« za preverjanje, ki mu določimo ustrezno funkcijo in se hkrati tudi skrije, ker se pokaže šele ko vse pojme povežemo. Nato sledi nekaj zank.

Zanka za zapolnjevanje polja ki bo služil na ravni organizatorja.

```
    for (var i = 0; i < pojmi_levo.length; i++) {
        var text_1 = {
            t: pojmi_levo[i],
            poz: i + 1
        };
        var text_2 = {
            t: pojmi_desno[i],
            poz: i + 1
        };
        pojmi_levi[i] = text_1;
        pojmi_desni[i] = text_2;
        ji = i;
    }
```

Zanki za naključno razporejanje elementov.

```
    for (var key in pojmi_levi) {
        var i = ji, rand, t, p;
        while (i) {
            rand = Math.floor(i * Math.random());
            t = pojmi_levi[rand].t;
            p = pojmi_levi[rand].poz;
            pojmi_levi[rand].t = pojmi_levi[i].t;
            pojmi_levi[rand].poz = pojmi_levi[i].poz;
            pojmi_levi[i].t = t;
            pojmi_levi[i].poz = p;
            i--;
        }
    }
    for (var key in pojmi_desni) {
        var i = ji, rand, t, p;
        while (i) {
            rand = Math.floor(i * Math.random());
            t = pojmi_desni[rand].t;
            p = pojmi_desni[rand].poz;
            pojmi_desni[rand].t = pojmi_desni[i].t;
            pojmi_desni[rand].poz = pojmi_desni[i].poz;
            pojmi_desni[i].t = t;
            pojmi_desni[i].poz = p;
            i--;
        }
    }
```

In zanka za ustvarjanje gumbov iz imen pojmov.

```
    for (var i = 0; i < pojmi_levo.length; i++) {
        var but_1 = new mBtn("1Button_" + (i + 1), pojmi_levi[i].t,
        x_okna + x_stolpec_1, y_okna + y_novega_gumba_1, sirina_gumba,
        visina_gumba, začetna_barva_gumba, layer, true);
        y_novega_gumba_1 = y_novega_gumba_1 + but_1.height + 10;
        //but.shape.setWidth(200);
        gumbi_1[i] = but_1;
        var but_2 = new mBtn("2Button_" + (i + 1 + (ji + 1)),
        pojmi_desni[i].t, x_okna + x_stolpec_2, y_okna + y_novega_gumba_2,
        sirina_gumba, visina_gumba, začetna_barva_gumba, layer, true);
        y_novega_gumba_2 = y_novega_gumba_2 + but_2.height + 10;
        //but.shape.setWidth(200);
        gumbi_2[i] = but_2;
    }
```

Zraven tega pa obstajajo še različne funkcije, za pomik pojmov na pravilno mesto, barvanje gumbov in poravnavo. Vse te funkcije pa se izvajajo ob premikanju gumba.

```
function obarvaj_gumbe() {...6 lines }
function izvedi_povezavo(stolpec_z_gumbom, lega) {...76 lines }
function pokazi_povezavo(stolpec_z_gumbi, y) {...20 lines }
function poravnaj_gumbe() {...27 lines }
function preveri(niz) {...47 lines }
```

# 10. Zaključek

S tem inovacijskim predlogom bi rad dosegel da se učna gradiva, v računalniški obliki, ne bi prenehala izdelovati, in programi ne bi zastareli, ter bi se uporabil njihov polni potencial. S to knjižnico se lahko enostavno izdela tudi kakšna bolj napredna aplikacija na internetu, kjer bi lahko morda nastala tudi kakšna učna gradiva v »web« obliki. To bi bilo dobro tudi za pametne telefone, ker ga skoraj vsak učenec/dijak/študent že ima in velika večina jih podpira tudi HTML5, ki je osnova te knjižnice oz. frameworka »KineticJS«. Čeprav je »KineticJS« precej razširjen, ta knjižnica delo z njim olajša. V prihodnosti bi lahko dodal tudi še kakšne druge elemente, kot je na primer polje za vnašanje in bi še dodatno pripomoglo k razvijanju, saj bi to omogočalo delo z vnašanjem podatkov. Vesel sem, da sem lahko naredil to nalogo, saj sem s tem moje znanje programiranja nadgradil, na nekoliko večjo raven kot samo v šoli, dobil pa sem tudi ogromno idej za programiranje v prihodnosti, ker mi jih je do sedaj nekoliko primanjkovalo.

## 11. Družbena odgovornost

Za ta inovacijski predlog sem prišel do ugotovitev da so pomembni predvsem naslednji členi :

- ODGOVORNOST ZA VPLIV

ker sem jaz kot razvijalec te aplikacije odgovoren za to kakšen vpliv oz. odziv bo to imelo na družbo in na ljudi, ki bi jo morebiti uporabljali.

- ETIČNO OBNAŠANJE

v primeru da bi se aplikacija prodajala (čeprav je odprtokodna in na voljo vsem), ne bi prišlo do izkoriščanja ljudi in bi bil odnos pošten.

- SPOŠTOVANJE INTERESOV DELEŽNIKOV

moram biti pozoren tudi na mnenja, interese uporabnikov, ne samo na lastne

- SPOŠTOVANJE ČLOVEKOVIH PRAVIC

da z vsebino ki jo ponujam, ne bi kršil katero od pravic vsakega človeka

# 12. Viri


1. Javascript kot jezik - http://en.wikipedia.org/wiki/JavaScript angleška in slovenska različica
2. Razvojno okolje NetBeans - http://en.wikipedia.org/wiki/NetBeans
3. JS Framework KineticJS – http://www.kineticjs.com/
4. Programsko orodje ExeCute - http://execute.fnm.uni-mb.si/