Zadání soutěžních úloh

Kategorie tvorba webu

19. až 21. dubna 2018

Soutěž v programování – 32. ročník

Krajské kolo 2017/2018

Úlohy můžete řešit v libovolném pořadí a samozřejmě je nemusíte vyřešit všechny. Počet bodů za každou úlohu je uveden přímo v jejím zadání. Hodnotí se shoda se zadáním, funkčnost, dodržování webových standardů a přehlednost zdrojového kódu.

Na řešení úloh máte 4 hodiny čistého času.

Před zahájením soutěže vám pořadatel oznámí, kde najdete testovací soubory a kam máte ukládat řešení úloh. Kompletní řešení každé úlohy (soubory HTML, CSS, obrázky, případně Javascript) uložte do samostatného podadresáře nazvaného jménem úlohy (např. `had`). Stránku s řešením vždy pojmenujte `index.html` a uložte v kódování UTF-8.

Při zápisu HTML a CSS kódu dodržujte webové standardy. Součást hodnocení vašich řešení je i kontrola zdrojových kódů pomocí validátorů. Doporučujeme používat HTML5, ale můžete použít i starší verze – HTML4 nebo XHTML1.

Při řešení je povoleno používat knihovnu jQuery bez pluginů. Ostatní frameworky a knihovny nejsou dovoleny.

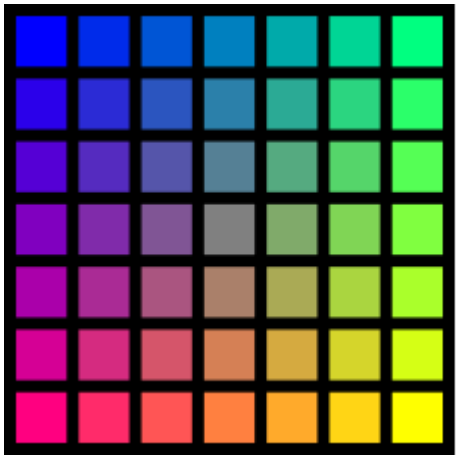
Had

max. 40 bodů

V této úloze je zakázáno používat předem připravené rastrové obrázky. Významné zjednodušení řešení (tj. pouhé zobrazení obrázku) znamená ohodnocení úlohy 0 body.

Statická část

Po otevření stránky zobrazte mřížku z čtverečků jako na obrázku. Pro plný počet bodů by mřížka měla mít rozměry 7×7 políček. Mřížka je vždy umístěna uprostřed obrazovky (jak horizontálně, tak vertikálně).

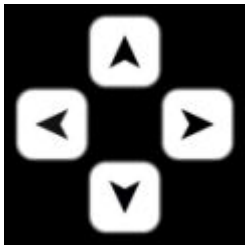


Pozadí stránky je černé. Rozměr čtverečku je 40 px, mezi čtverečky je 10 px mezera. Barva čtverečků je určena jejich pozicí. Každá složka (červená, modrá, zelená) je určena svým vlastním pravidlem:

- Červená roste rovnoměrně po řádcích. V prvním řádku je 0 %, v posledním 100 %.
- Zelená roste rovnoměrně po sloupcích. V prvním sloupci 0 %, v posledním 100 %.
- Modrá klesá rovnoměrně po diagonálách. V levém horním rohu je 100 %, v pravém dolním 0 %.

Pro kontrolu: prostřední pole je šedé, `#808080`.

Vytvořte na stránce, vpravo od herního pole, vertikálně uprostřed stránky, ovládací panel podle obrázku:



Šipka je Unicode znak s hexadecimálním kódem `27a4`. Tlačítka mají šedý zaoblený okraj, černou šipku a bílé pozadí.

V dalších odstavcích budeme mluvit o různých stavech políčka. Je-li políčko *rozsvícené*, má barvu přidělenou popsanými pravidly. Pokud je políčko *zhasnuté*, každá ze složek jeho barvy je 20 % původní hodnoty. Pokud je *inverzní*, má všechny složky barev převrácené, tedy například z modré se stane žlutá.

Mřížka reprezentuje herní pole hry *Had*. Políčka zabraná hadem jsou *rozsvícená*. Políčka, na kterých se vyskytuje jídlo, jsou *inverzní*. Ostatní políčka jsou *zhasnutá*.

Nebudete-li programovat dynamickou část, upravte mřížku, aby vypadala jako na obrázku:

V našem příkladě je had dlouhý sedm políček (fialová až šedá), jídlo se nachází na pastelově zeleném políčku.

Dynamická část

Kliknutím na zhasnuté políčko se na jeho místě objeví jídlo. Kliknutím na jídlo se na tomto políčku jídlo zruší. Klikání na hada nemá žádný efekt.

Naprogramujte pohyb hada pomocí šipek ovládacího panelu. Na začátku had stojí uprostřed plochy. Při najetí myší na šipku had změní směr. Šipka změní okraj a barvu na červenou a hlava hada se začne pohybovat daným směrem. Had se pohne o jedno políčko každou půlsekundu. Šipka zůstává červená, dokud se had pohybuje daným směrem.

Pokud by měl had vyjet z herní plochy, objeví se na protilehlé straně.

Na začátku hry je had 3 políčka dlouhý, ale je svinut na prostředním políčku. Až když se začne hýbat, vysune se do své plné délky.

Pokud se hlava hada přesune na políčko s jídlem, had se v tu chvíli prodlouží o jedno pole (v tomto časovém kroku se neposune ocas hada). Jídlo je hadem snědeno, z herního plánu tedy mizí.

Pokud by had narazil hlavou do sebe sama, hra končí. Zobrazte zprávu o ukončení hry. Pozor na to, že had se hýbe celý najednou. Například pokud je had dlouhý 7 políček, může bez zásahu uživatele procházet stěnou a kroužit dokola, aniž by si ukousnul ocas.

Had nemůže „couvat“, neboli si ukousnout políčko těsně za hlavou. Najetí na šipku takového směru prostě ignorujte.

Během hry počítejte skóre (počet snědených jídel) a viditelně ho zobrazte. Zobrazte ho také ve zprávě o ukončení hry. Nabídněte restart hry.

S každým snědeným jídlem zkraťte časový krok o 0.01 sekundy. Nemusíte řešit okrajový případ.

Umožněte uživateli zvolit si parametry hry pomocí parametrů požadavku v URL. Zohledněte následující parametry (výchozí hodnoty v závorkách):

- `width`: šířka pole (7)
- `height`: výška pole (7)
- `length`: počáteční délka hada (3)
- `timestep`: počáteční čas na jedno kolo v milisekundách (500)
- `timedelta`: úbytek času na bod skóre v milisekundách (viz níže)

Není-li uveden úbytek času explicitně, dopočítá se vzorcem `timestep / (width * height)`. Nemusíte kontrolovat smysluplnost parametrů, předpokládejte, že jsou vždy rozumné.

Například kdybychom chtěli hrát těžkou hru na mřížce 30×15:

```
had.html?width=30&height=15&timestep=300&timedelta=5
```

Dynamické vyhledávání

max. 20 bodů

Právě jste se z čista jasna ocitli s Felisem na vesmírné lodi. Bohužel ani jeden neumíte vesmírnou loď řídit a kolem vás je blížící se meteorit, dále spousta světelných let vesmíru a ještě také několik desítek příruček k raketě. Záchranou by mohlo být pouze to, kdyby se vám podařilo včas najít správnou příručku s informacemi podstatnými pro vychýlení směru vesmírné lodi a vyhnout se tak jinak téměř jisté srážce s meteoritem.

Vytvořte aplikaci, která vám bude poskytovat grafické rozhraní s funkcí vyhledávání ve zdrojích specifikovaných v přiloženém souboru.

Přiložený soubor obsahuje seznam knih, které máte k dispozici a bude úkolem je zobrazit. Se souborem můžete dělat co uznáte za vhodné.

Formát souboru

- Soubor obsahuje jednotlivé knihy v elementu `knihovnicka`.
- Jednotlivé knihy jsou uzavřeny v elementu `article`.
- Element `details` obsahuje element `summary` obsahující název knihy.
- Informace o knize – název, autor, kategorie jsou uzavřeny samostatně v elementu `var` a jejich název je určen atributem `data-type`.
- Dále element `details` obsahuje ještě element `figure`, ve kterém je vložen obrázek.
- Příklad souboru naleznete ve `vyhledavani/data.xml`.

Příklad

```
<?xml version="1.0" encoding="UTF-8"?>
<knihovnicka>
  <article>
    <figure>
      <img src="http://ecx.images-amazon.com/images/I/41kSLEoswsL.jpg"
           alt="044310073X.jpg image" width="304" height="228"/>
      <figcaption>044310073X.jpg image</figcaption>
    </figure>
    <details>
      <summary>Oral and Maxillofacial Surgery: An Objective-Based
                  Textbook, 2e</summary>
      <var data-type="indexId">044310073X</var>
      <var data-type="author"></var>
      <var data-type="categoryId">16</var>
      <var data-type="category">Medical Books</var>
    </details>
  </article>
</knihovnicka>
```

Grafická část aplikace:

- Webová stránka bude rozdělena na tři části – záhlaví, hlavní obsah a zápatí.
- V záhlaví bude pole pro zadání textu, vyjížděcí seznam obsahující položky „Název“, „Autor“, „Autor a název“, „Kategorie“ a „Vše“, checkbox s textem „Použít regex“ a tlačítko s nápisem „Vyhledat“.
- Hlavní část stránky bude obsahovat seznam knih z výše popisovaného souboru.
- Seznam knih bude zobrazen ve třech sloupcích.
- Knihy budou mít postupně zobrazeny pouze své názvy. Po kliknutí na název se kniha zobrazí přes všechny sloupce, pod jejím názvem se objeví zbylé informace. V levé polovině budou textové informace – bude zde zobrazen pouze autor knihy a název kategorie, v pravé polovině bude zobrazen obrázek. Zároveň zbylé knihy budou zobrazeny nad a pod touto knihou.
- Zápatí a záhlaví bude mít stejnou (rozumnou) výšku a bude viditelné po celou dobu skrolováním stránkou.
- Zápatí bude obsahovat text „Počet nalezených položek X/Y“ kde X je hodnotu počtu aktuálně zobrazených (tj. nalezených) položek a Y je celkový počet knih. Po spuštění aplikace budou zobrazeny všechny knihy, viz funkční část aplikace.

Funkční část aplikace:

- Aplikace funguje jako filtrování nalezených knih. Po otevření stránky či jejím aktualizováním (a zároveň tedy i pro prázdné vyhledávací pole) jsou zobrazeny všechny výsledky seřazené dle abecedy.
- Poté, co uživatel zadá do vyhledávacího pole textový řetězec, budou zobrazeny výsledky, které začínají nebo obsahují zadaný řetězec v názvu knihy.
- Ve výsledcích budou zobrazeny prioritně knihy, které zadaným řetězcem začínají.
- Vylepšete implementaci tak, že řetězec bude vyhledáván v poli vybraném ve vyjížděcím seznamu (pro položku „Vše“ bude prohledáváno „Autor“, „Název“ a „Kategorie“).
- Implementujte i vyhledávání pomocí regulárních výrazů (část znaků v řetězci tedy bude moci být nahrazena např. znaky jako jsou * . ? či jinými regexovými patterny). Vyhledávání pomocí regexu bude použito v případě, že uživatel zaškrtne checkbox v záhlaví.
- Vyhledávání bude fungovat dynamicky – nebude nutné stisknout tlačítko vyhledat a výsledky budou filtrovány průběžně v průběhu psaní dotazu uživatelem.
- Na základě počtu aktuálně nalezených knih bude aktualizována hodnota v zápatí.

Oprava aplikace

max. 30 bodů (10 grafická část, 20 funkční část)

Pejsek s kočičkou se rozhodli si zase něco vyrobit. Protože pejsek měl rád dopravní prostředky a kočička Javascript, tak si v Javascriptu napsali simulátor dopravních prostředků. Má ale nějaké mouchy, kočičku už to přestalo bavit a pejsek si to spravit neumí, tak by potřeboval pomoc.

Součástí zadání je aplikace, spustíte jí otevřením souboru `index.html`. Nahoře je „ovládací panel“, kde můžete ovládat rychlost simulace, resetovat jí a hlavně nahrát soubor (JSON, formát je popsaný níže). Testovací soubory najdete v stejné složce, je tam jednoduchý `test1.json`, a komplexnější `test2.json` (sice vám může připadat povědomý, ale je to dost zjednodušené, tak si podle toho neplánujte cestu domů).

Aplikace je napsaná v Typescriptu, což je jazyk transpilovaný do Javascriptu; přidává téměř jenom typovou kontrolu (aby tam nebyly hloupé chyby). Navíc aplikace používá některé nové konstrukce Javascriptu (ES6), tak se nelekněte. Pro případ, že by vám to nevyhovovalo, nebo byste se jenom chtěli podívat, do čeho se určitá konstrukce kompiluje, máte vedle souboru `script.ts` také `script.js` (ES6) a `script.es5.js` (ES5, "starý Javascript"), které jsou ekvivalentní (je to výstup Typescriptového kompilátoru). I pokud nechcete upravovat Typescriptovou verzi, může vám trochu pomoci se podívat na definice typů, se kterými aplikace pracuje – jsou všechny hned nahoře a jsou u nich komentáře. Chyby stačí opravit jenom v jednom z nich, ale aplikace musí používat ten opravený, aby fungovala (takže pokud si vyberete, že opravíte Typescript, musíte ho zkompilovat do Javascriptu, aby to browser pochopil). Opravený výsledek včetně všech dalších potřebných souborů uložte do adresáře s názvem úlohy stejně jako u ostatních úloh.

Formát dat

Data aplikace jsou v JSONu a kromě toho, že jsou data v tomto formátu uložená, tak s ekvivalentním modelem simulátor interně pracuje – uložený soubor je v podstatě jenom serializovaný objekt typu `SimulationData`. Struktura vypadá nějak takto:

- `map` – Mapa světa, ve kterém se pohybuje
  - `nodes` – pole bodů v mapě (vrcholy, tj. křižovatky)
    - `pos` – souřadnice bodu `[x, y]`
  - `connections` – propojení mezi body (hrany, tj. silnice/koleje)
    - `from`, `to` – mezi jakými body vede (číslo je index v poli `nodes`). Cesta je vždy obousměrná.
    - `distance` – jak dlouho trvá tudy projet (v simulovaných sekundách)
    - `curve` – křivka (resp. lomená čára) cesty, bod [0,0] je na začátku, prázdné pole udělá rovnou čáru
  - `stations` – pole stanic
    - `node` – číslo bodu, kde se stanice nachází
    - `name` – její jméno
- `lines` – pole linek, které jezdí po mapě
  - `stops` – na kterých stanicích spoje staví (`id` je index v poli stanic)
  - `timeTable` – jízdní řád výchozích stanic

        - `direction` – směr, jakým spoj jede (1 nebo -1)
        - `time` – kdy vyjíždí
        - `from` – odkud vyjíždí. Pokud není uvedeno, tak je to konečná určená podle směru
    - `name` – jméno linky

Jednoduchý příklad, který si můžete dle libosti editovat pro otestování, čeho budete chtít, je v souboru `test1.json`.

Co je potřeba upravit

Aplikace má několik chyb, které je potřeba opravit:

- Menu je ošklivé – kočička má sice ráda Javascript, ale CSS moc ne… Tak by to chtělo trochu vyčistit. Tlačítka v menu by měla být zarovnaná, kliknutí na pauzu by nemělo změnit velikost, „Uložit mapu“ by mělo vypadat stejně jako „Otevřít mapu“, popisek „Rychlost“ by měl být vizuálně svázaný s posouvátkem rychlosti a ne náhodně pohozený v panelu, čas by měl být součástí panelu (vpravo, prosím) a ne pohozený nad ním.
- Aplikace by neměla mít scrollbar a měla by využít celou plochu obrazovky (menu by mělo být nahoře, tag `svg` by měl pokrýt zbytek obrazovky a neměl by přetékat).
- Po pauze už nejde simulace znovu spustit.
- Čas se nezobrazuje tak, jak jsme zvyklí – pokud je sekund méně než 10, zobrazují se jenom jednociferně. Místo „4:7“ by se mělo zobrazovat „4:07“.
- Chtěli bychom, aby se po najetí myší na zelený bod zobrazil název linky, kterou tento dopravní prostředek obsluhuje. Mělo by to fungovat podobně, jako když najedete na zastávku (červený bod). Tip: během pauzy se na bod najíždí lépe.