



Ondřej Žára

JavaScript

Programátorské
techniky
a webové technologie

- Poutavý výklad na příkladu hry
- Moderní postupy vývoje funkční aplikace
- Od základů po pokročilé techniky

computer
press®

Ondřej Žára

JavaScript

Programátorské techniky a webové technologie

**Computer Press
Brno
2015**

JavaScript

Programátorské techniky a webové technologie

Ondřej Žára

Obálka: Martin Sodomka

Odpovědný redaktor: Martin Herodek

Technický redaktor: Jiří Matoušek

Objednávky knih:

<http://knihy.cpress.cz>

www.albatrosmedia.cz

eshop@albatrosmedia.cz

bezplatná linka 800 555 513

ISBN 978-80-251-4573-9

Vydalo nakladatelství Computer Press v Brně roku 2015 ve společnosti Albatros Media a. s.
se sídlem Na Pankráci 30, Praha 4. Číslo publikace 19 278.

© Albatros Media a. s. Všechna práva vyhrazena. Žádná část této publikace nesmí být kopírována
a rozmnožována za účelem rozšiřování v jakékoli formě či jakýmkoli způsobem bez písemného
souhlasu vydavatele.

1. vydání

 **ALBATROS** MEDIA a.s.

Obsah

Úvod	9
Zpětná vazba od čtenářů	9
Zdrojové kódy ke knize	9
Errata	10
KAPITOLA 1	
Začínáme	11
Motivace	11
Java-co?!	11
Odbočka (historická)	12
Metodika	13
KAPITOLA 2	
Pravidla hry	15
Herní strategie	16
Varianty pravidel	17
Existující implementace	17
KAPITOLA 3	
Než se dáme do práce	21
Jak pracovat s knihou	21
Terminologie	22
Opáčko základů JavaScriptu	22
Pár slov o prohlížečích	23
Coding style	24
Test na závěr kapitoly	26

KAPITOLA 4

První výkop **27**

Odbočka (historická) 27

Jdeme na to 28

Důvod k menší oslavě! 31

KAPITOLA 5

Modularizace **33**

Návrhové vzory a API 33

Odbočka (terminologická) 34

SoC, SRP 34

Hlavní komponenty 35

Odbočka (historická) 39

KAPITOLA 6

Plnohodnotné kreslení **41**

Odbočka (terminologická) 41

HTML <canvas> 42

Klíčové slovo this a pseudoprivátní vlastnosti 43

Příprava canvasu a mřížka 45

Buňky a atomy v nich 47

Ostatní soubory 50

KAPITOLA 7

Řetězová reakce **53**

Úpravy v rámci hrací plochy 53

Volání sebe sama 56

Ještě k vykreslování 58

KAPITOLA 8

Reagujeme per partes **61**

Intimní vztah rekurze a cyklu 61

Jen to trochu zpomalit... 62

Čas sáhnout do kódu	65
Asynchronní finále	67
Co tam dělá ten bind?	69

KAPITOLA 9

Brouci, mouchy a další hmyz **71**

Jehla v kupce sena	72
Past sklapla	74
Lépe a radostněji?	75
Odbočka (terminologická)	78
Druhé kolo oprav	78

KAPITOLA 10

Více hráčů, skóre a konec hry **81**

Dva, tři, moc	81
Řídí to tu někdo?	82
Nejprve existující kód	82
Překvapivé události	85
Nový objekt Score	87

KAPITOLA 11

Datový typ souřadnic **91**

Odbočka (terminologická)	91
Objekty ex nihilo a další	92
Operátor new	95
Co umí souřadnice?	96
Souřadnice všude tam, kde se s nimi pracuje	98

KAPITOLA 12

Umělá inteligence **103**

Tři druhy hráčů	104
Odbočka (historická)	108
Pročištění zbylých komponent	109
Tak kde je ta AI?	114

KAPITOLA 13

Hrubě silná umělá inteligence **117**

Teoretická příprava algoritmu	118
Pozor na DRY	118
Jen to trochu přiohnout...	120
Simulujeme díky klonům	122
Odbočka (terminologická)	124

KAPITOLA 14

Bonus #1: Ukládání hry **127**

Kamarádi WebStorage a JSON	127
Úpravy v HTML a koncept save/load	129
Stav objektu Board	132
Stav objektu Player	133

KAPITOLA 15

Bonus #2: Zvučíme **135**

Stavební kameny Web Audio API	135
Generování zvuku	136
Kam se zvukem v Atomech?	137
Bind pro starší a pokročilé	138
Refactoring exploze	138
Komponenta Audio	139

KAPITOLA 16

Bonus #3: Testování **143**

Čeho všeho se to týká?	144
Testovací prostředí Jasmine	145
Testování kódu třinácté kapitoly	146
Tak jak si stojíme?	151

KAPITOLA 17

Bonus #4: Kreslení s WebGL **153**

Jak funguje WebGL 154

Co se musíme naučit 155

Nejprve kontext a shadery 156

WebGL buffery a vykreslení 160

KAPITOLA 18

Bonus #5: Více hráčů po síti **165**

Firebase 166

Firebase API 166

Návrh konceptu hry po síti 169

Počáteční synchronizace 169

Jak přesně funguje Player.Remote? 172

Rejstřík **175**

Úvod

Jen těžko se lze pohybovat v oblasti tvorby webu a nepovšimnout si, jakému zájmu se JavaScript těší. Spolu s oblíbeností a rozšířeností logicky roste také poptávka po kvalifikovaných vývojářích. Najít zkušeného JavaScriptového programátora přitom vůbec není jednoduché; základy si na Internetu najde každý, ale skutečně hlubokého pochopení dojde jen málokdo. Je k tomu zapotřebí mnoho praxe, zvědavosti i obecného algoritmického rozhledu.

Na oblíbeném programátorském webu *stackoverflow.com* je položeno přes tři čtvrtě milionu otázek na téma JavaScript. To je dobrým dokladem zájmu i nedostatečného vzdělání mezi webovými vývojáři. Ambicí této knihy je poskytnout podklady pro systematické pochopení JavaScriptu a jeho praktického používání.

Zpětná vazba od čtenářů

Nakladatelství a vydavatelství Computer Press, které pro vás tuto knihu připravilo, stojí o zpětnou vazbu a bude na vaše podněty a dotazy reagovat. Můžete se obrátit na následující adresy:

Computer Press
Albatros Media a.s., pobočka Brno
IBC
Příkop 4
602 00 Brno

nebo

sefredaktor.pc@albatrosmedia.cz

Computer Press neposkytuje rady ani jakýkoli servis pro aplikace třetích stran. Pokud budete mít dotaz k programu, obraťte se prosím na jeho tvůrce.

Zdrojové kódy ke knize

Z adresy <http://knihy.cpress.cz/K2209> si po klepnutí na odkaz Soubory ke stažení můžete přímo stáhnout archiv s ukázkovými kódy. Najdete je na serveru GitHub na adrese <https://github.com/ondras/javascript>.

Errata

Přestože jsme udělali maximum pro to, abychom zajistili přesnost a správnost obsahu, chybám se úplně vyhnout nelze. Pokud v některé z našich knih najdete chybu, ať už chybu v textu nebo v kódu, budeme rádi, pokud nám ji oznámíte. Ostatní uživatelé tak můžete ušetřit frustrace a pomoci nám zlepšit následující vydání této knihy.

Veškerá existující errata zobrazíte na adrese <http://knihy.cpress.cz/K2209> po klepnutí na odkaz Soubory ke stažení.

Začínáme

V této kapitole:

- Motivace
- Java-co?
- Metodika

Motivace

Předvídaní budoucnosti je vždy ošemetná věc, u informačních technologií tomu není jinak. S ohledem na vývoj posledních zhruba deseti let se však zdá, že málokterá výpočetní oblast zažívá takový opravdový boom jako World Wide Web. Důvodů je mnoho: dostupný a mobilní Internet, miniaturizace chytrých zařízení, překotný rozvoj webových prohlížečů, snadno přístupné jazyky a rozhraní. Troufám si proto tvrdit, že sázka na webové technologie se rozhodně vyplatí – a to nejen těm, kdo potřebují nabízet své produkty elektronickou cestou, ale i vývojářům všeho druhu, technologickým fandům i zájemcům o hlubší setkání s výpočetní technikou.

Konkrétní nástroje, postupy a knihovny přicházejí a s různorodým odstupem zase mizí. Základní koncepty a jazyky WWW však zůstávají a postupným vylepšováním upevňují web na předním místě každodenních životních činností. Práce s HTML stránkami, jejich design a postupné ožívování pomocí skriptů jsou tak více a více považovány za základní IT vzdělání; mnoho vývojářů se s prvními algoritmy a jazyky potká právě v rámci tvorby webu.

S rostoucí nabídkou je na místě dodávat i užitečné studijní podklady. Zde přichází ke slovu tato kniha: poskytuje krátký exkurz do světa pod pokličkou HTML stránek, ukazuje principy vývoje a seznamuje s typickými postupy. Na své by si tak měli přijít jak začátečníci, tak i zkušenější vývojáři z ostatních oborů, kteří by rádi obhlédli, jak že se to pracuje s tím HTML, CSS a JavaScriptem.

Java-co?

Specifické typografické znaménko v nadpisu (tzv. *interrobang*, kombinace vykřičníku a otazníku) koresponduje s údivem, který tento jazyk – včetně svého jména – nezdídky vyvolává ve vývojářích. Oprávněně očekávají klientskou verzi jazyka Java, jenže

ouha. Jak říká tradiční anglické úsloví, „Java is to JavaScript as ham is to hamster“ (slovní hříčka přeložitelná možná jako „Java a JavaScript, to je jako párek a kašpárek“), tedy podobnost mezi těmito jazyky je jen v názvu.

Pozoruhodnou souhrou historických událostí se stalo, že jazyk JavaScript, navržený právě pro skriptování v prostředí webového prohlížeče, dostal do vínku nezvyklou kombinaci vlastností. Jeho syntaxe vychází z jazyka C, některá standardní rozhraní (vestavěné funkce, objekty, proměnné) se podobají Javě, objektový a funkcionální model je inspirován jazyky Scheme a Self. Je dynamicky typovaný (tj. o datových typech proměnných se rozhoduje až za běhu programu) a funkce jsou v něm k dispozici jako běžný datový typ. JavaScript staví na takzvané *prototypové dědičnosti* (budeme se jí věnovat v jedenácté kapitole), která bývá velmi obtížně uchopitelná pro programátory vyškolené v tradičním *třídním* objektovém modelu.

Během několika prvních let života dokázal JavaScript odsunout na vedlejší kolej svého jediného konkurenta, VBScript, navržený společností Microsoft jako alternativní jazyk pro skriptování webových stránek, a stal se tak de facto jediným skutečným programovacím jazykem použitelným v rámci HTML dokumentů. Implementace JavaScriptu je dnes nedílnou součástí všech webových prohlížečů a bez nadsázky lze říci, že se jedná o jednu z nejrozšířenějších technologií vůbec.

Odbočka (historická)

Vývoj a verze JavaScriptu

Archeologičtí nadšenci se mohou detailnější informace dozvědět mimo jiné na stránkách https://www.w3.org/community/webbed/wiki/A_Short_History_of_JavaScript a <http://en.wikipedia.org/wiki/JavaScript#History> – tato odbočka je zkrácenou kombinací textů z několika zdrojů, zejména však těch dvou odkazovaných.

První verze jazyka JavaScript byla navržena a naimplementována během pouhých deseti dní (!) Brendanem Eichem ve společnosti Netscape. Psal se rok 1995 (mimo jiné též rok vzniku jazyka PHP) a Eich dostal za úkol v této kruté časové lhůtě navrhnout *menšího a hloupějšího sourozence Javy* – jako konkurenční krok v tvrdém souboji s prohlížečem Internet Explorer. Tato verze jazyka se jmenovala Mocha a záměrně, aby nevypadala příliš jako Java, nabízela namísto tříd prototypy. Ještě v roce 1995 došlo k přejmenování na LiveScript v rámci vydání v prohlížeči Netscape Navigator 2. Brzy poté přibyla do Netscape Navigatoru podpora pro spouštění appletů Java a zároveň s tím dostal klientský skriptovací jazyk své dnešní jméno.

V roce 1996 se objevuje první konkurenční implementace – JScript v Internet Exploreru 3. Zároveň dochází k formální standardizaci jazyka, tentokrát pod názvem ECMAScript (ECMA je název standardizační organizace *European Computer Manufacturers Association*, zodpovědné mimo jiné za specifikace formátů dat na CD či jazyka C#).

Poslední zajímavá zmínka ve dvacátém století pochází z konce roku 1999, kdy světlo světa spatřuje třetí verze specifikace jazyka, ECMA-262. Ta je na dlouhou dobu poslední, a JavaScript se tak v této podobě během následujících deseti let rozšiřuje do podvědomí webových vývojářů.

Pro svůj vlastní prohlížeč Chrome se společnost Google rozhodla vytvořit zbrusu novou a výkonnou implementaci JavaScriptu. V roce 2008 se objevuje Chrome 1.0 a implementace dostává název V8. Tím dochází k oživení myšlenky, že jazyk jako takový nemá důvod být používán jen v rámci klient-ského skriptování a lze jej využít pro programování serverového kódu. Tyto úvahy v roce 2009 ústí v založení neformální organizace *CommonJS*, která si za cíl stanovila prozkoumat a standardizovat chybějící serverová JavaScriptová rozhraní.

Ještě v roce 2009 se ale objevuje projekt Node.js: sada rozhraní a knihoven, obalující implementaci V8, a dovolující tak jednoduše spouštět JavaScriptový kód i mimo webový prohlížeč. Node.js si velmi rychle získal silnou uživatelskou základnu, zastínil CommonJS, a stal se tak de facto standardním řešením pro serverové vykonávání JavaScriptu.

Koncem roku 2009 konečně vychází nová, pátá verze specifikace jazyka ECMAScript (ES5). Obsahuje podporu pro tzv. *striktní režim*, funkce pro práci s datovým formátem JSON a další.

Dnes, v době psaní knihy (počátek roku 2015), je ECMAScript v pátém vydání nejrozšířenější verzí, a proto se v knize využívá právě tento standard. Organizace ECMA však zavrhuje práci na šesté verzi, která přinese razantní novinky jak syntaktické, tak funkční. Tato šestá verze se zpravidla označuje ES6 či ES2015. Do budoucna je pak v plánu verze sedmá, o které toho zatím není mnoho známo.

Metodika

O jazyku JavaScript bylo napsáno mnoho knih; běžný popis či referenční příručka by tak byla jen nošením dříví do lesa. V této knize není cílem čtenáře seznámit se všemi aspekty jazyka, protože v praxi se vývojář jen málokdy potká s nutností ovládat JavaScript na tak komplexní úrovni. Namísto toho chceme na praktických úlohách naznačit, kudy by mohlo vést řešení a jaké konstrukce (syntaktické a algoritmické) se nabízejí a hodí.

V některých vývojářských komunitách se čas od času konají události nazvané *Code Retreat*: jedná se o krátká jednodenní školení či tréninky, při kterých se účastníci zdokonalují ve svých dovednostech opakovaným řešením té samé úlohy. Za den ji zvládnou vyřešit třeba desetkrát. Pokaždé s trochu jinými postupy, někdy s ohledem na rychlost, jindy bez použití té či oné vlastnosti jazyka a podobně.

Lehce podobný přístup je k dispozici čtenáři v této knize. Ústředním tématem je konkrétní úloha, která je v průběhu kapitol studována, rozvíjena a vylepšována. Po několika prvních kapitolách je hotové základní řešení; s ním se ale nespokojíme a postupným přidáváním funkcionality nahlédneme i do dalších partií webových technologií, jako je testování, hledání a odstraňování chyb a podobně. Čtenář se proto setká s postupem, který je velmi blízký skutečnému vývoji v praxi – s tím rozdílem, že jednotlivá architektonická a implementační rozhodnutí dopředu zvolil autor knihy, aby na nich demonstroval různé formy vývoje a zajímavé prostředky jazyka.

Stejně tak jako u opravdového vývojářského úkolu ani zde nebudou probrány a ozkoušeny veškeré dostupné vlastnosti jazyka a souvisejících technologií. Naše úloha však byla zvolena tak, aby pokryla co možná nejširší spektrum běžných programátorských činností, a měla by tak být přínosná pro všechny zájemce o JavaScript.

Pravidla hry

V této kapitole:

- Herní strategie
- Varianty pravidel
- Existující implementace

V této kapitole si popíšeme úlohu, jejíž řešení je náplní knihy. Jedná se o implementaci poměrně málo známé deskové hry, která nemá jednoznačný název – objevuje se pod anglickými jmény *Chain reaction*, *Atoms*, *Atomic reaction* a dalšími. My budeme hře říkat prostě *Atomy*.

Koncept hry coby programátorské úlohy je zvolen záměrně. Úkoly z kurzů programovacích jazyků jsou často nudné a nezábavné; jde o algoritmizační či matematické hříčky a problémy, které dokážou zaujmout nadšence, ale ne široké publikum. Bývají totiž zpravidla buď příliš abstraktní („napište program hledající miliontou cifru desetinného rozvoje Ludolfova čísla“), nebo neužitečné či příliš specifické („napište program řešící úlohu obchodního cestujícího“). Naše herní úloha má jasně definované a dobře uchopitelné zadání; zároveň se jedná o užitečný úkol, protože jeho výstupem je fungující aplikace. Lze ji hrát ve dvou i více hráčích, experimentovat s pravidly, hledat dobrou strategii a podobně. V pozdějších kapitolách knihy se též objeví téma umělé inteligence, a tak i možnost naprogramovat si vlastního protivníka (a buď jej se slávou porazit, nebo mu – s nemenší radostí – podlehnout).

Rigorózní pravidla pro *Atomy* neexistují, v průběhu let vzniklo nespočet variant a žádná kanonická verze není k dispozici. My se budeme držet těchto základních stavebních kamenů hry:

1. Hry se účastní alespoň dva hráči, kteří se pravidelně střídají v tazích.
2. Hrací plocha má podobu čtvercové mřížky velikosti 6×6 polí (buněk).
3. Buňka hracího pole může být buď prázdná (tak je tomu u všech buněk na začátku hry), nebo může obsahovat několik *atomů* (kuliček). Každý atom má barvu podle hráče, kterému přísluší. **V buňce smí být jen atomy stejné barvy.**
4. Hráč, který je na tahu, musí do jedné buňky vložit tzv. *atom* (kuličku) své barvy. Atom lze vložit buď do prázdné buňky, nebo do buňky obsahující atomy hráče na tahu. Nelze tedy atom přidat do buňky s atomy jiné barvy.

5. Pokud hráč nemůže táhnout (tj. na hrací ploše neexistují žádné prázdné buňky a v žádné buňce hráč na tahu nemá své atomy), prohrál. Při hře dvou hráčů to značí vítězství oponenta; u více hráčů se pokračuje do té doby, než zbude jediný.
6. Pro každou buňku hracího pole existuje takzvané *kritické množství* atomů, které je definováno jako počet sousedů (v čtyřsousednosti). Čtyři rohové buňky mají tedy kritické množství 2, šestnáct buněk po straně hrací plochy má kritické množství 3, ostatní (vnitřní) buňky pak 4.
7. Pokud v rámci tahu hráč vložím atomu překoná kritické množství atomů v buňce, dojde okamžitě k jejímu *rozpadu*: do každé sousední buňky přeskočí jeden atom z té rozpadající se. V původní buňce tak (většinou) zůstane jeden atom. Během rozpadu se může stát, že sousední buňka již obsahuje atomy – stejné či odlišné barvy. Pokud je barva odlišná, v rámci přeskočení atomu dojde k *přebarvení* všech atomů v sousední buňce na barvu hráče na tahu. Znamená to, že tento rozpad je jediným způsobem, jak může hráč získat buňku, která patří jeho soupeři.
8. Po dokončení rozpadu buňky mohou na herní ploše zůstat další nadkritické buňky (nadkritická množství atomů se v nich objevila v rámci přeskočení). I tyto buňky se nyní musí rozpadnout (je jedno, v jakém pořadí); tento proces se opakuje do té doby, než všechny buňky obsahují nejvýše kritické množství atomů.

Herní strategie

Autorovi knihy není znám žádný výzkum, který by se věnoval otázce pravidel hry a její rozhodnutelnosti, výherní strategie a podobně. Cílem hry je přirozeně zabránit všech buněk hracího pole, ale není patrné, jestli by se tak mělo postupovat už od začátku (tj. nejprve vložit do co nejvíce buněk po jednom atomu), či naopak nejprve co nejlépe zaplnit menší počet buněk a expandovat až v rámci (řetězových) reakcí.

Stejně tak se nabízí otázka, jestli je lepší oddalovat interakci s dalšími hráči až do explozivního finále hry, nebo vyhledávat časté šarvátky již v jejím průběhu. Pravidla hry totiž dovolují pokládat atomy zcela nesouvisle do vzdálených (prázdných) buněk a pokrýt tak značnou část hrací plochy.

Další prostor pro herní taktiku nabízí odlišná kritická množství atomů v buňkách. Zabíráním rohových polí se dá docílit rozpadu již po třech tazích a obecně pole u krajů hrací plochy dovolují rychlejší *zbrojení* a přípravu na případnou reakci v porovnání s buňkami *ve vnitrozemí*.

Atomy jistě neobsahují tolik taktické hloubky jako některé jiné hry, například dáma či šachy. I tak je k dispozici dostatek možností pro budování strategie a hra ani po mnoha partiích nepůsobí *ohraným* dojmem – zejména zvážíme-li možnost účasti více hráčů.

Varianty pravidel

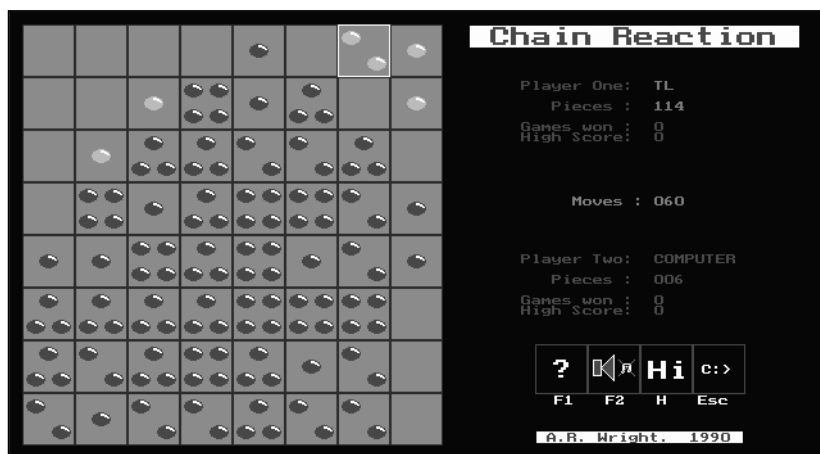
Kromě základních pravidel výše se nabízejí i další drobné varianty, které mohou hru zpestřit či upravit. V knize se na ně nebudeme soustředit, ale zvědavý čtenář si je může zkusit – povětšinou velmi snadno – naimplementovat *za domácí úkol*.

- Hrací plocha může mít libovolné rozměry (i obdélníkové); často se objevuje velikost 8×8 . Při minimální hrací ploše 2×2 mají všechny buňky stejné kritické množství a tento *degenerovaný* případ se může hodit pro testovací a ladící účely.
- V některých implementacích se objevují kritická množství o jedničku nižší. Při rozpadu buňky se tak většinou stane, že centrální pole zůstane prázdné (tj. opět volné pro zabránění libovolným hráčem).
- Podobně je též možné uvažovat osmisousednost, tedy i rozpad do diagonálních směrů. Kritická množství pak adekvátně narostou.
- Kromě vkládání nových atomů lze v rámci tahu povolit i přesun atomu z jedné buňky do jiné. Tuto formu tahu můžeme omezit například jen na přesun do sousední buňky.
- Při aplikaci předchozího pravidla je možné hráče vybavit pevným (konečným) množstvím atomů. Jakmile jim atomy dojdou, jsou odkázáni jen na přesouvání po hrací ploše.

Existující implementace

Pojďme se na konec této kapitoly v rychlosti podívat na některé existující implementace konceptu hry Atomy. Jedná se zpravidla o starší programy, protože elektronické verze deskových her zažívaly svůj boom spíše v dřevních dobách počítačové éry.

- **Chain reaction** (autor A. R. Wright) z roku 1990 je patrně první existující verzí pro počítače PC. Program je k dohledání ke stažení na Internetu.



Obrázek 2.1 Chain reaction, A. R. Wright