

Front-end

Environment setup

Best practices

Debugging

Profiling

Введение

Этапы работы

1. Выбор решений, подходов
2. Выбор инструментов согласно нуждам
3. Развертка рабочего окружения
4. Отладка
5. Профилирование
6. Тестирование
7. Поддержка

Выбор среды разработки

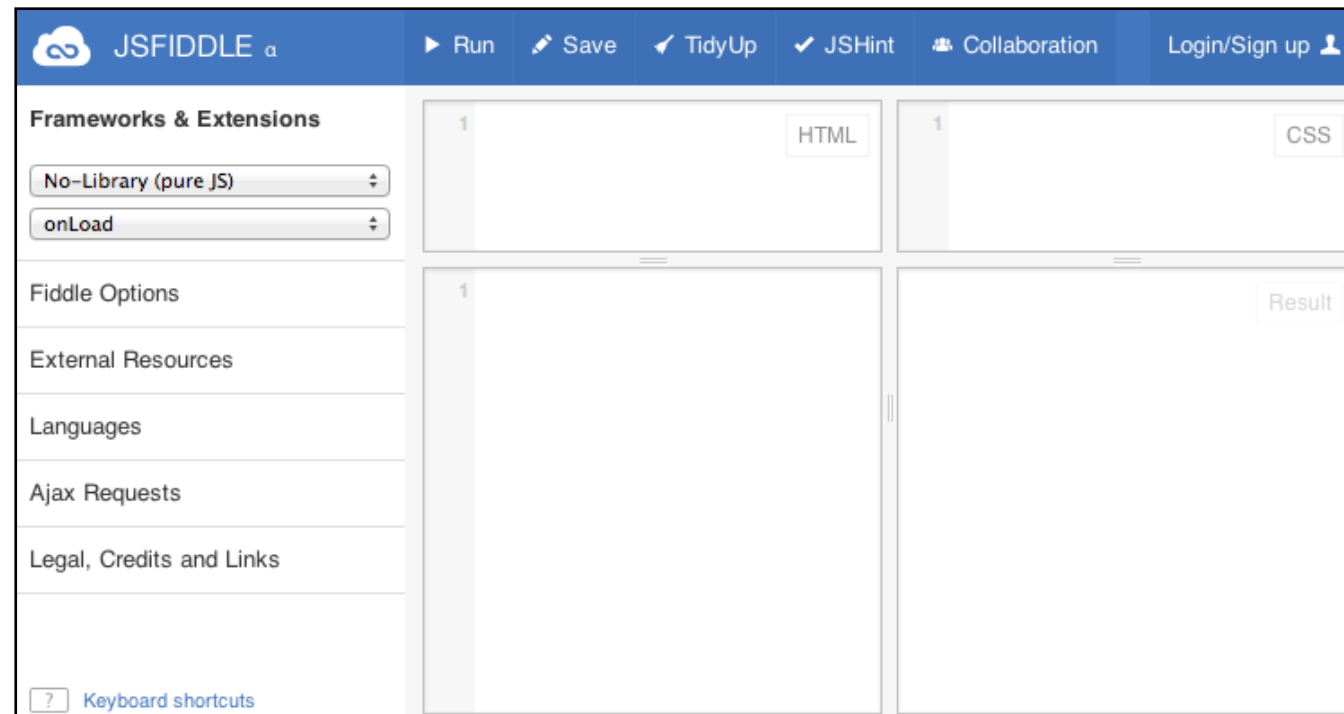
- Продукты JetBrains:
 - WebStorm – JavaScript IDE (Integrated Development Environment) – платная, но есть лицензия для студентов и преподавателей
 - IntelliJ IDEA – Java IDE – платная, но также есть лицензия для студентов и преподавателей
- NetBeans (бесплатная)
- Eclipse (бесплатная)
- Sublime Text
- Atom

Возможности:

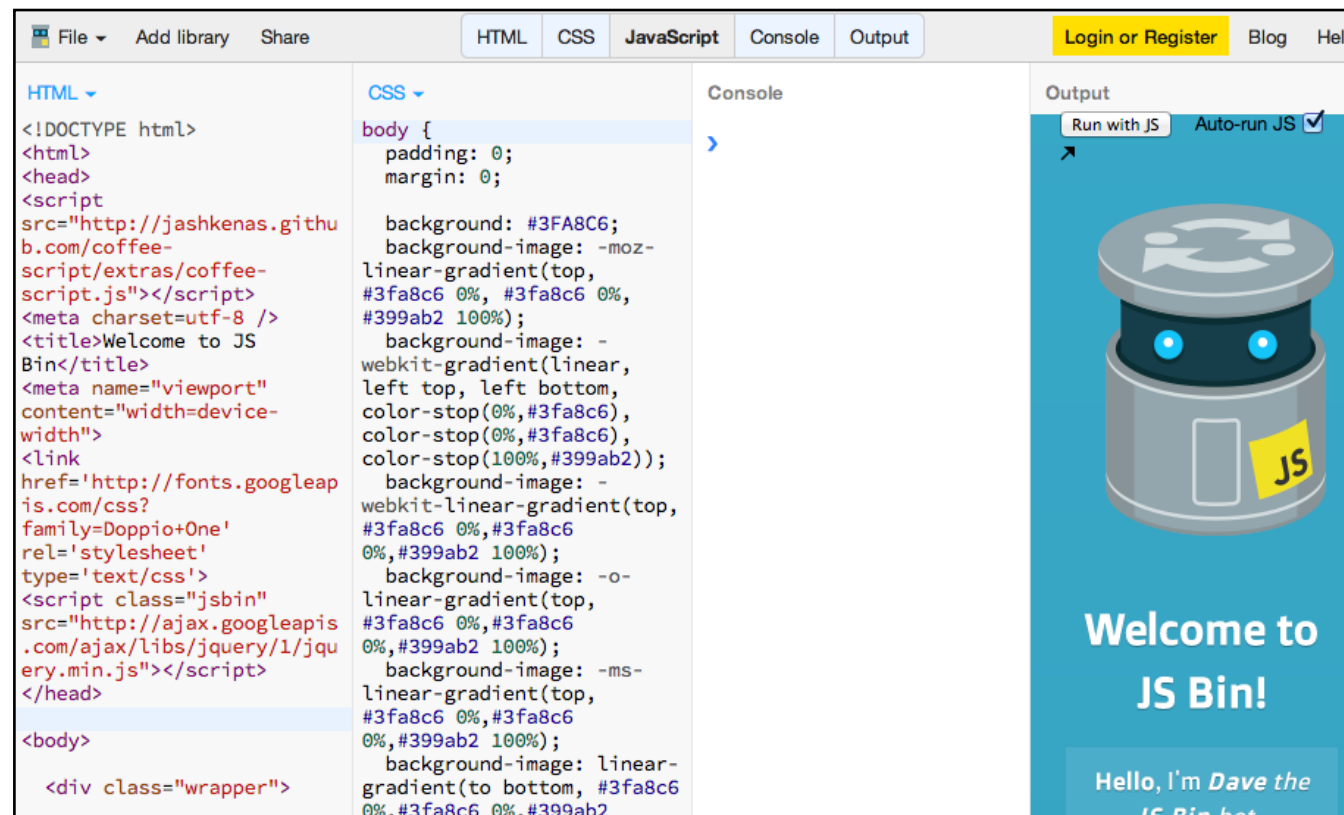
- поддержка VCS (Version Control System)
 - поддержка всевозможных инструментов / фреймворков
 - code complete
 - поддержка множества языков программирования
 - встроенные инструменты для мобильной разработки
 - автоформатирование
- и многое другое

Выбор среды разработки

<http://jsfiddle.net/>



<http://jsbin.com/>



Система контроля версий

— система для работы с изменяющейся информацией

Этапы работы:

- 1) Обновление рабочей копии
- 2) Модификация проекта
- 3) Фиксирование изменений

Subversion (SVN), Git – наиболее популярные

TortoiseSVN – бесплатный SVN клиент для Windows
GUI клиенты для работы с Git.

Сборка проекта

Grunt, Maven, Ant— инструменты для сборки JavaScript проектов

- в основе работы — задачи / цели
- существует множество плагинов, содержащих инструкции:
 - по объединению файлов
 - по слежению за файлами
 - по проверке кода (напр. JSHint)
 - по тестированию
 - по минификации/обфускациии другое

Grunt написан на JavaScript

Сборка проекта

Grunt

```
concat: {
  options: {
    separator: ';'
  },
  dist: {
    src: ['src/**/*.js'],
    dest: 'dist/main.js'
  }
}

uglify: {
  dist: {
    files: {
      'dist/main.min.js': ['<%=
concat.dist.dest %>']
    }
  }
}
```

```
qunit: {
  files: ['test/**/*.html']
}

jshint: {
  files: ['Gruntfile.js', 'src/**/*.js',
'test/**/*.js'],
  options: {
    // override JSHint defaults
    globals: {
      jQuery: true,
      console: true,
      module: true,
      document: true
    }
  }
}

watch: {
  files: ['<%= jshint.files %>'],
  tasks: ['jshint', 'qunit']
}
```

Полезные инструменты

Emmet — инструмент для ускорения разработки HTML и CSS. В его основе — динамические аббревиатуры, которые преобразуются в код.

#page>div.logo+ul#navigation>li*5>a{Item \$}

преобразуется в

```
<div id="page">
  <div class="logo"></div>
  <ul id="navigation">
    <li><a href="">Item 1</a></li>
    <li><a href="">Item 2</a></li>
    <li><a href="">Item 3</a></li>
    <li><a href="">Item 4</a></li>
    <li><a href="">Item 5</a></li>
  </ul>
</div>
```

Emmet встроен в IntelliJ IDEA и WebStorm

Существует множество плагинов для различных IDE

Анализаторы кода

JavaScript

Анализаторы кода выявляют:

- потенциальные ошибки
- неэффективный код
- неправильный синтаксис
- стилистические нарушения кода

Как используются?

- как плагин для редактора для моментального анализа кода
- на этапе сборки проекта
- перед комитом

JSLint — самый старый из анализаторов

JSHint

- ответвление JSLint
- более настраиваемый

ESLint

- правила подключаются как плагины
- любое правило можно включить и выключить
- каждое правило настраивается индивидуально (предупреждение или ошибка)
- позволяет создавать собственные правила

Анализаторы кода

CSS

CSS Lint — интегрируется с системой сборки

(<https://github.com/CSSLint/csslint/wiki/Command-line-interface>)

Примеры возможных правил:

- запретить использование !important;
- запретить использование ID в селекторах #menu { ... }
- запретить пустые правила .sidebar { }
- запретить единицы измерения для нулевых значений .sidebar { width: 0px; }

CSSCSS

— анализирует повторяющиеся правила

`{.contact .content .primary} and {article, #comments} share 5 rules`

Анализаторы кода

HTML

HTML Inspector

- написан на JavaScript и запускается в браузере
- правила можно изменять, добавлять новые
- интегрируется с системой сборки

Проверяет:

- валидность:
 - правильное использование элементов, атрибутов
 - отсутствие повторяющихся ID
- хорошие практики:
 - отсутствие обработчиков событий, установленных как атрибуты
`<input id="b1" value="Win gold medal" onclick="alert('Phelps');" type="button"/>`
 - нет не используемых классов, элементов

и др.

- | |
|---|
| ⚠ The 'bgcolor' attribute is no longer valid on the <body> element and should not be used. ▶ body |
| ⚠ <style> elements outside of <head> must declare the 'scoped' attribute. ▶ style |
| ⚠ The <hgroup> element is obsolete and should not be used. ▶ hgroup |
| ⚠ The class 'post' is used in the HTML but not found in any stylesheet. ▶ article.post |
| ⚠ The 'alt' attribute is required for elements. ▶ img |
| ⚠ The <il> element is not a valid HTML element. ▶ il |
| ⚠ The 'action' attribute is required for <form> elements. ▶ form |

Разработка

Некоторые общие правила

- Не изобретать колесо! Может быть, эту работу уже делал кто-то до вас
- Руководствоваться устоявшимися практиками, применять шаблоны проектирования
- Писать поддерживаемый код
- Писать читаемый код
- Придерживаться единого стиля написания кода
- Придерживаться единого стиля именования переменных, функций, файлов и т.д.
- Писать комментарии
- Писать тесты
- Излишняя оптимизация — плохо
- Минифицировать

Разработка JS кода

Хорошие практики

- Минимизировать число глобальных переменных

```
// antipattern
coffee_sort = "Arabica";
function myFunc() {
    accidentally_global_var = 15; //
    forgetting var
    var a = b = 0; // b is global
}
```

- === ВМЕСТО ==

```
var zero = "0";
if (zero == 0) {
    // executing...
}
if (zero === 0) {
    // not executing
}
```

- eval() is evil

- не использовать for-in там, где подойдет обычный цикл
for (var x = 0; x < 10; x++) {
};

- ИСПОЛЬЗОВАТЬ ;

- {} вместо new Object(), [] вместо new Array()
var arr1 = [];
var obj = {};
obj.a = 1;

Принципы хорошей архитектуры CSS

Предсказуемость

Повторное использование

Поддержка

Масштабируемость

Распространенные ошибки

- Изменение компонентов в зависимости от родителя

```
.widget {  
    background:yellow;  
    border:1px solid black;  
    color:black;  
    width:50%;  
}  
#sidebar .widget {  
    width:200px;  
}  
body.homepage .widget {  
    background:white;  
}
```

- Слишком сложные селекторы

```
#main-nav ul li ul li div { }  
#content article h1:first-child { }  
#sidebar > div > h3 + p { }
```

- Слишком общие имена селекторов

```
.widget {}  
.widget .title {}  
.widget .contents {}  
.widget .action {}
```

- Правило делает слишком много

```
.widget {  
    position:absolute;  
    top:20px;  
    left:20px;  
    background-color:red;  
    font-size:1.5em;  
    text-transform:uppercase;  
}
```

Принципы хорошей архитектуры CSS

CSS должен как можно меньше «знать» про структуру HTML.

- Точное описание

```
/* Граната */           /* Снайперская винтовка */  
#main-nav ul li ul { }  .subnav { }
```

- Разделение ответственности

`background`, `color` и `font` в одном правиле с `position`, `width`, `height` и `margin`

- Использование пространства имен

```
/* Плохо */ .widget { } .widget .title { }  
/* Хорошо */ .widget { } .widget-title { }
```

- Расширение компонентов модификаторами классов

```
/* Плохо */ .widget { } #sidebar .widget { }  
/* Хорошо */ .widget { } .widget-sidebar { }
```

Принципы хорошей архитектуры CSS

- Логическая структура в организации кода
- Использование классов только для оформления, обозначение «неоформительских» классов
- Логическая структура в именах классов

/* Плохо! */

```
/* Компонент */ .button-group { }  
/* Модификатор компонента */ .button-primary { }  
/* Вложенный объект (внутри .button) */ .button-icon { }  
/* Компонент или глобальный элемент */ .header { }
```

/* Вариант, как сделать лучше */

```
/* Правила компонентов */  
.component-name  
.component-name--modifier-name  
.component-name__sub-object  
.component-name__sub-object--modifier-name  
/* Правила раскладки (модульные сетки, глобальные элементы) */  
.l-layout-method  
.grid  
/* Правила состояний */ .is-state-type  
/* Классы для JavaScript без оформления */ .js-action-name
```


Принципы хорошей архитектуры CSS

Препроцессоры

- помогают писать CSS быстрее
- помогают писать плохой CSS быстрее ;)

SASS, LESS, Stylus

- Переменные

```
$primary-color: #333;  
body { color: $primary-color; }
```

- Вложенность

```
nav {  
    color: #000;  
    ul { margin: 0; }  
}
```

=>

```
nav { color: #000; }  
nav ul { margin: 0; } // результат
```

- Примеси

```
@mixin border-radius($radius) {  
    -webkit-border-radius: $radius;  
    -moz-border-radius: $radius;  
    -ms-border-radius: $radius;  
    border-radius: $radius;  
}  
.box { @include border-radius(10px); }
```

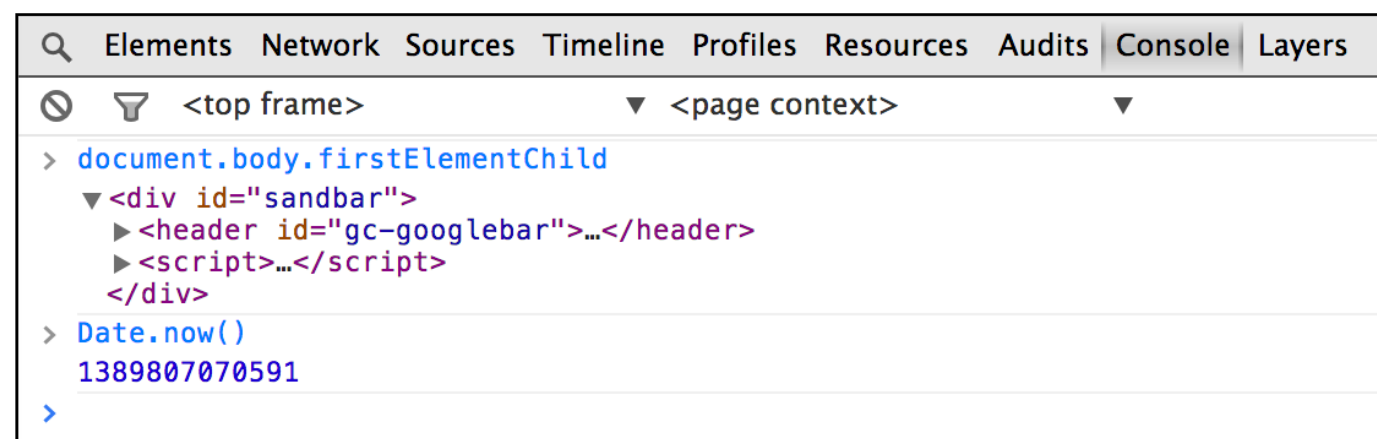
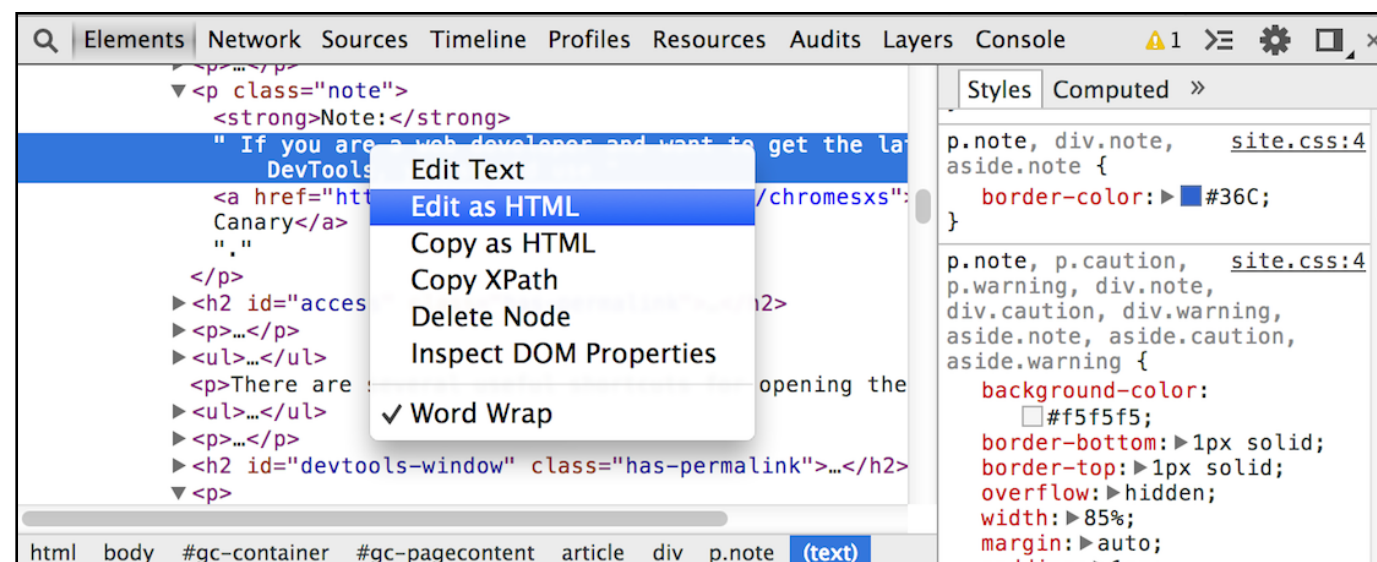
- Другое

Отладка

Chrome Developer Tools

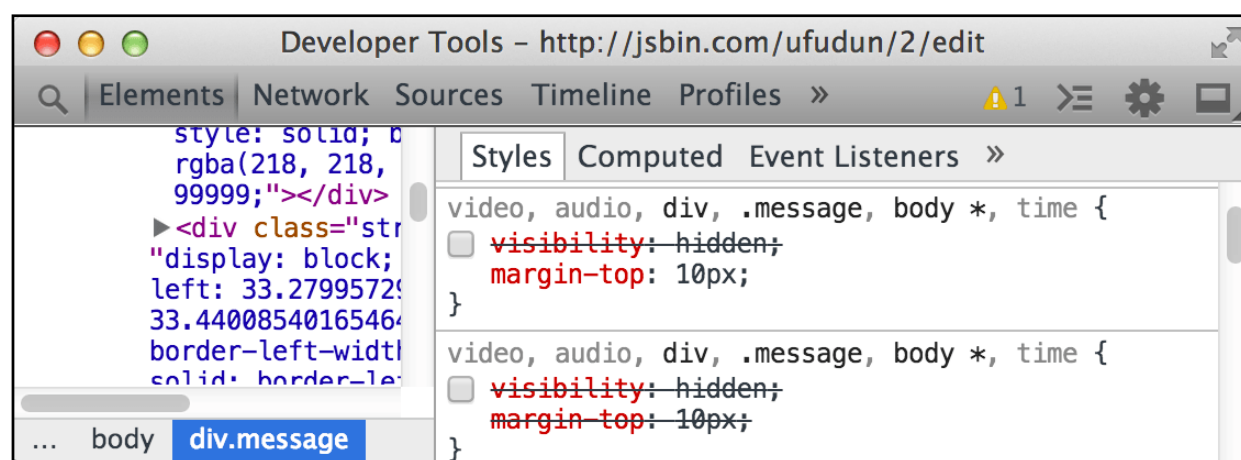
DOM-дерево

- Инспектировать элементы
 - Редактировать элементы и атрибуты
 - Передвигать узлы в дереве
 - Удалять узлы
- и др.

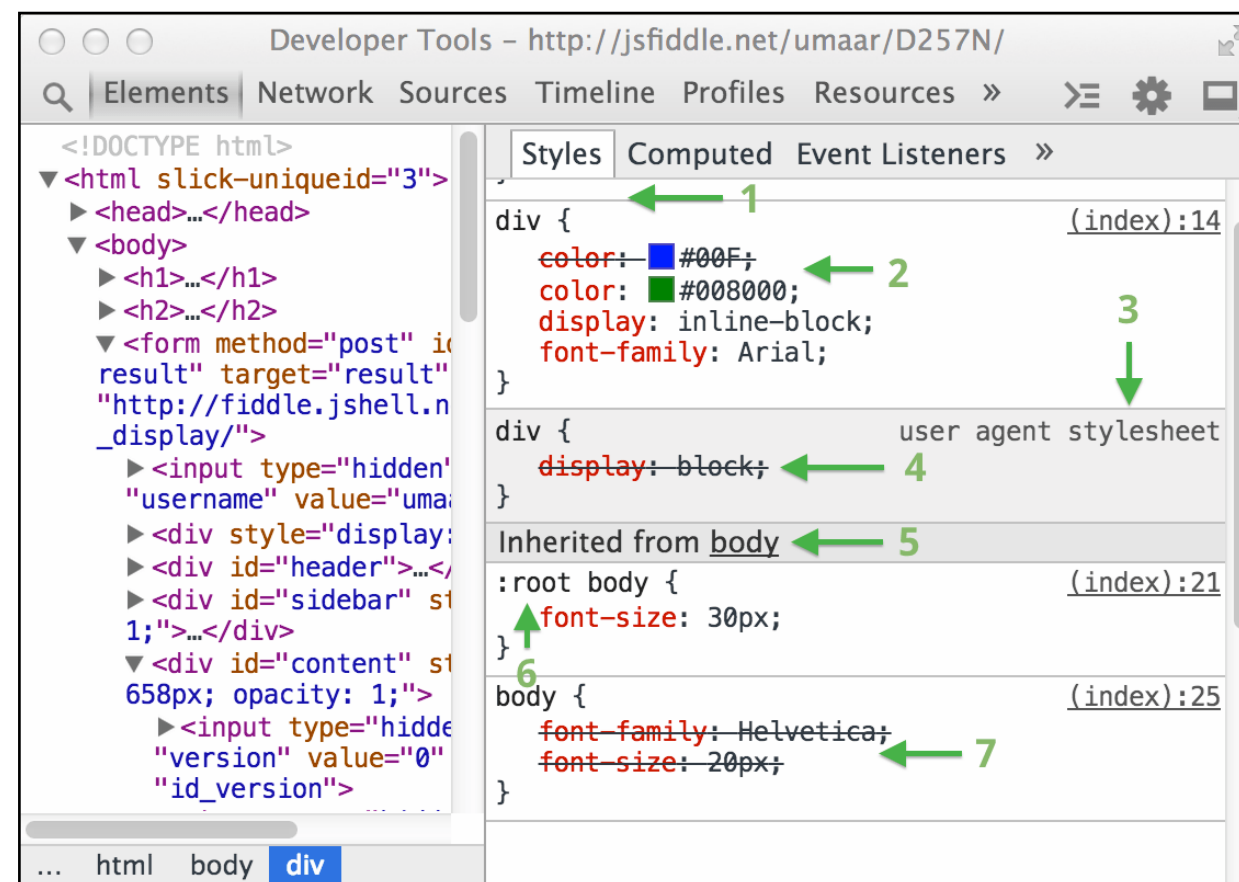


Отладка

CSS



width: 86%; width: 86%;



Отладка

Firebug

HTML

The first screenshot shows the HTML panel with a tree view of the document structure. The second screenshot shows the DOM tree with a specific element selected. The third screenshot shows the Inspect tool being used to examine an element. The fourth screenshot shows the Inspect tool's context menu with options like 'Copy HTML', 'Copy innerHTML', 'Copy XPath', 'Copy CSS Path', and 'Log Events'.

JS

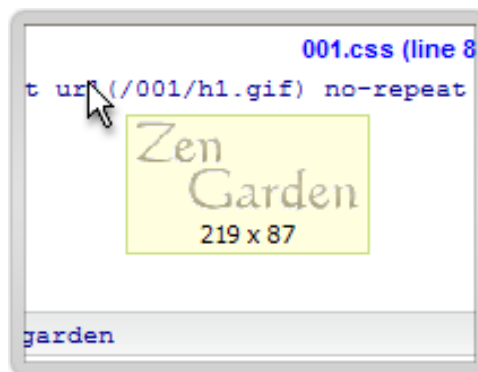
The first screenshot shows the JS panel with a list of scripts. The second screenshot shows a breakpoint set in the JS code. The third screenshot shows the Watch panel with variables being monitored. The fourth screenshot shows the Console panel with a log message. The fifth screenshot shows the Console panel with a log message.

Отладка

Firebug

CSS

```
li {  
    margin-bottom: 0.3em;  
}  
  
Inherited from ul.news-digg  
  
.news-digg {  
    text-align: center;  
    font-size: 85%;  
    list-style-type: none;  
    list-style-image: none;  
}
```

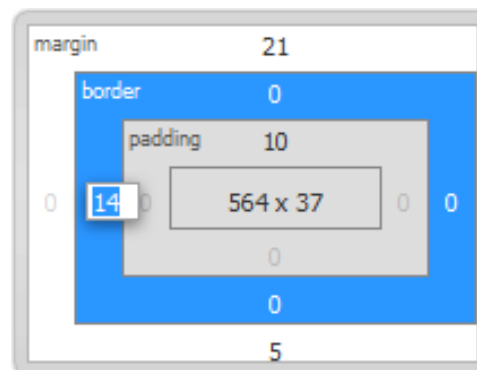


```
Style ▾ Computed Layout DC  
#boasting h2 {  
    font-family: Georgia  
    font-size: 1.6em;  
    font-weight: bold;  
}  
h2 {  
    font-size: 2em;  
}
```

```
Firefox .features h3 a {  
    padding: 2px 0pt 6px 34px  
    background: transparent url  
    display: block;  
    text-decoration: none;  
}  
Firefox .features p {  
    margin: 0pt 50px 1em 32px;  
}
```

```
border: medium none;  
}  
#firefox .features h3 a {  
    padding: 2px 0pt 6px 35px  
    background: transparent u  
    display: block;  
    text-decoration: underlin  
}  
#firefox .features p {
```


```
Style ▾ Computed Layout DC  
input {  
    :hover  
    :active  
    :focus  
    0 2px 2px -1px #FFFFFF  
    text-shadow: 1px 1px rgl  
}
```



Анализ производительности

YSlow

[Home](#) | [Grade](#) | [Components](#) | [Statistics](#)

Grade  Overall performance score 78 Ruleset applied: YSlow(V2) URL: http://h22154.www2.hp.com/

ALL (23) FILTER BY: [CONTENT \(6\)](#) | [COOKIE \(2\)](#) | [CSS \(6\)](#) | [IMAGES \(2\)](#) | [JAVASCRIPT \(4\)](#) | [SERVER \(6\)](#)

C Make fewer HTTP requests

F Use a Content Delivery Network (CDN)

A Avoid empty src or href

F Add Expires headers

B Compress components with gzip

A Put CSS at top

B Put JavaScript at bottom

B Avoid CSS expressions

n/a Make JavaScript and CSS external

E Reduce DNS lookups

B Minify JavaScript and CSS

A Avoid URL redirects

A Remove duplicate JavaScript and CSS

A Configure entity tags (ETags)

A Make AJAX cacheable

A Use GET for AJAX requests

C Reduce the number of DOM elements

A Avoid HTTP 404 (Not Found) error

A Reduce cookie size

F Use cookie-free domains

A Avoid AlphaImageLoader filter

A Do not scale images in HTML

A Make favicon small and cacheable

Grade C on Make fewer HTTP requests

This page has 8 external javascript scripts. Try combining them into one.
This page has 7 external background images. Try combining them with CSS sprites.

Decreasing the number of components on a page reduces the number of HTTP requests required to render the page. The following components include: combine files, combine multiple scripts into one script, combine multiple CSS files into one CSS file.

[»Read More](#)

Copyright © 2014 Yahoo! Inc. All rights reserved.

Анализ производительности

PageSpeed

https://www.yahoo.com/АНАЛИЗИРОВАТЬ

Для мобильных

Для компьютеров

84 / 100 Рекомендации

Исправьте обязательно:

Удалите из верхней части страницы код JavaScript и CSS, блокирующий отображение

Как исправить?

Исправьте по возможности:

Оптимизируйте видимое содержание

Как исправить?

Сократите HTML

Как исправить?

Оптимизируйте изображения


Как исправить?

Сократите JavaScript

Как исправить?

Выполнено правил: 5

Подробности



**Результаты кешируются на 30 секунд. Если вы изменили страницу, подождите 30 секунд, прежде чем запускать тест снова.*

Оптимизация

- Уменьшать число запросов к серверу
 - Склеивать файлы (JavaScript, CSS)
 - Использовать спрайты
- Использовать CDN (Content Delivery Network)
- Кэшировать ресурсы
- Использовать компрессию (gzip)
- Помещать стили сверху страницы
- Помещать скрипты в конец страницы
- Помещать JavaScript и CSS в отдельные файлы
- Минифицировать / обфусцировать JavaScript и CSS
- Оптимизировать изображения
 - Не использовать изображения большего размера, чем нужно
``
 - Не использовать изображения большего размера, чем нужно
- Избавиться от ненужных html элементов
- Упрощать css-селекторы



Оптимизация

- Кэшировать результат поиска по DOM-дереву:

```
var el = document.getElementById('id');
if (el.attr1) {
    el.attr2 = val2;
    el.attr3 = val3;
}
```
- Объединять операции с элементами DOM дерева, использовать **DocumentFragment**
- Использовать CSS-классы, не стили

```
// 5 перерисовок
jQuery('#dialog-window').width(600).height(400).css('position':
'absolute').css('top', '200px').css('left', '200px');

// Объединили, 1 перерисовка
jQuery('#dialog-window').css({
    width: '600px', height: '400px',
    position: 'absolute',
    top: '200px', left: '200px' });

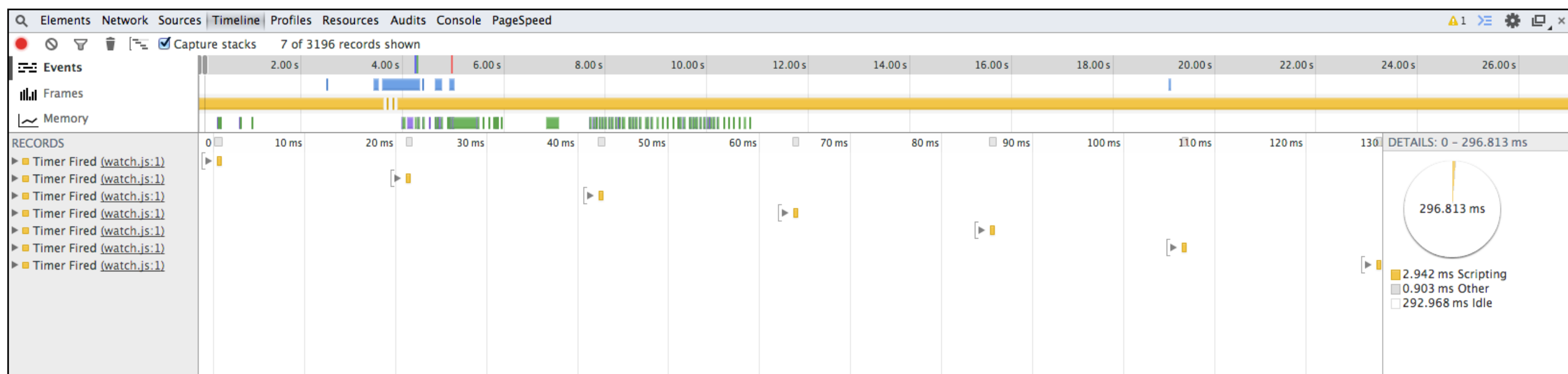
// Еще лучше
jQuery('#dialog-window').addClass('mask-aligned-window');
```

Профилирование

Chrome Developer Tools

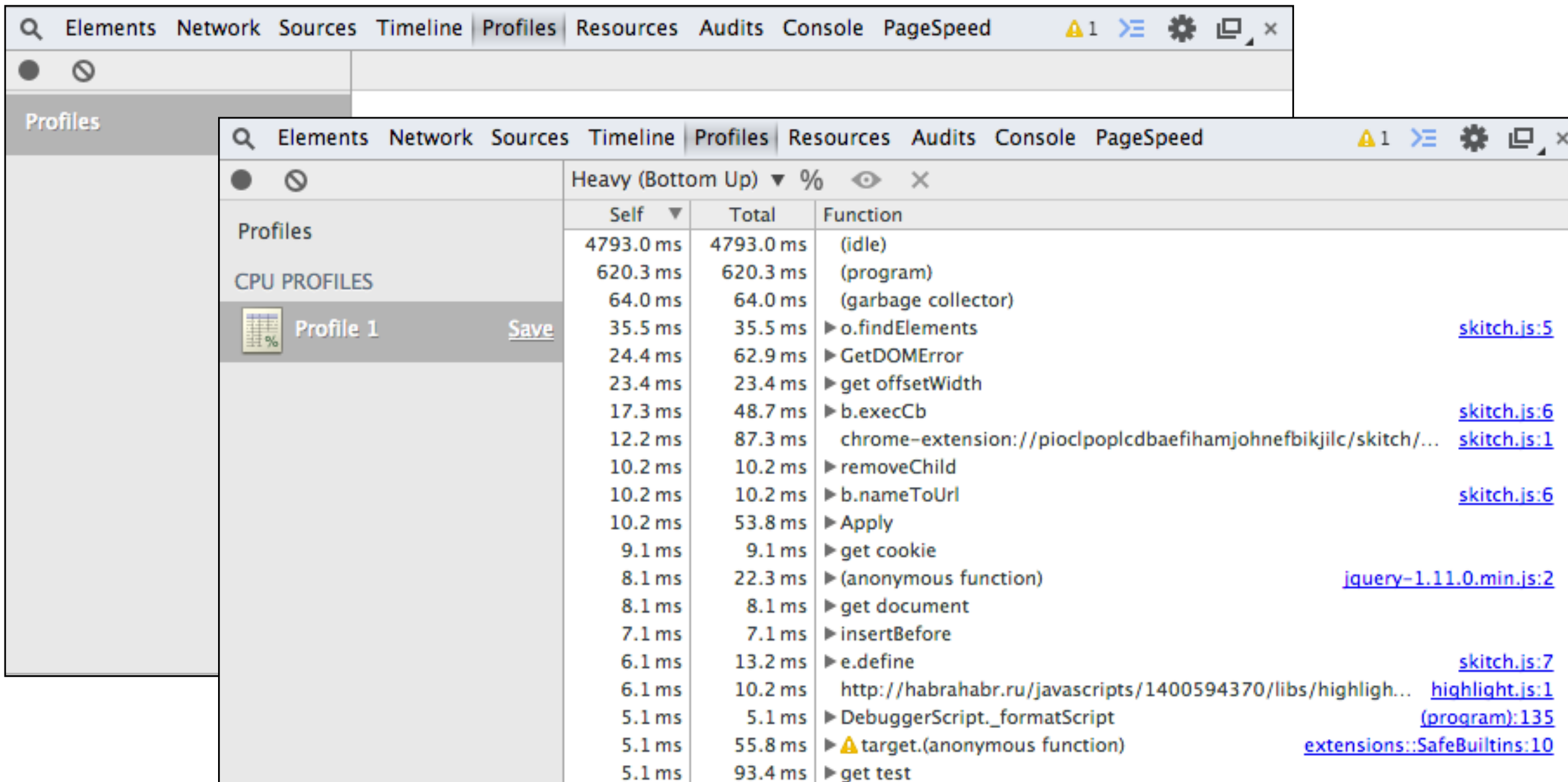
<https://developer.chrome.com/devtools/index>

Вкладка Timeline показывает время работы кода



Профилирование

Вкладка Profiles показывает, как работает код



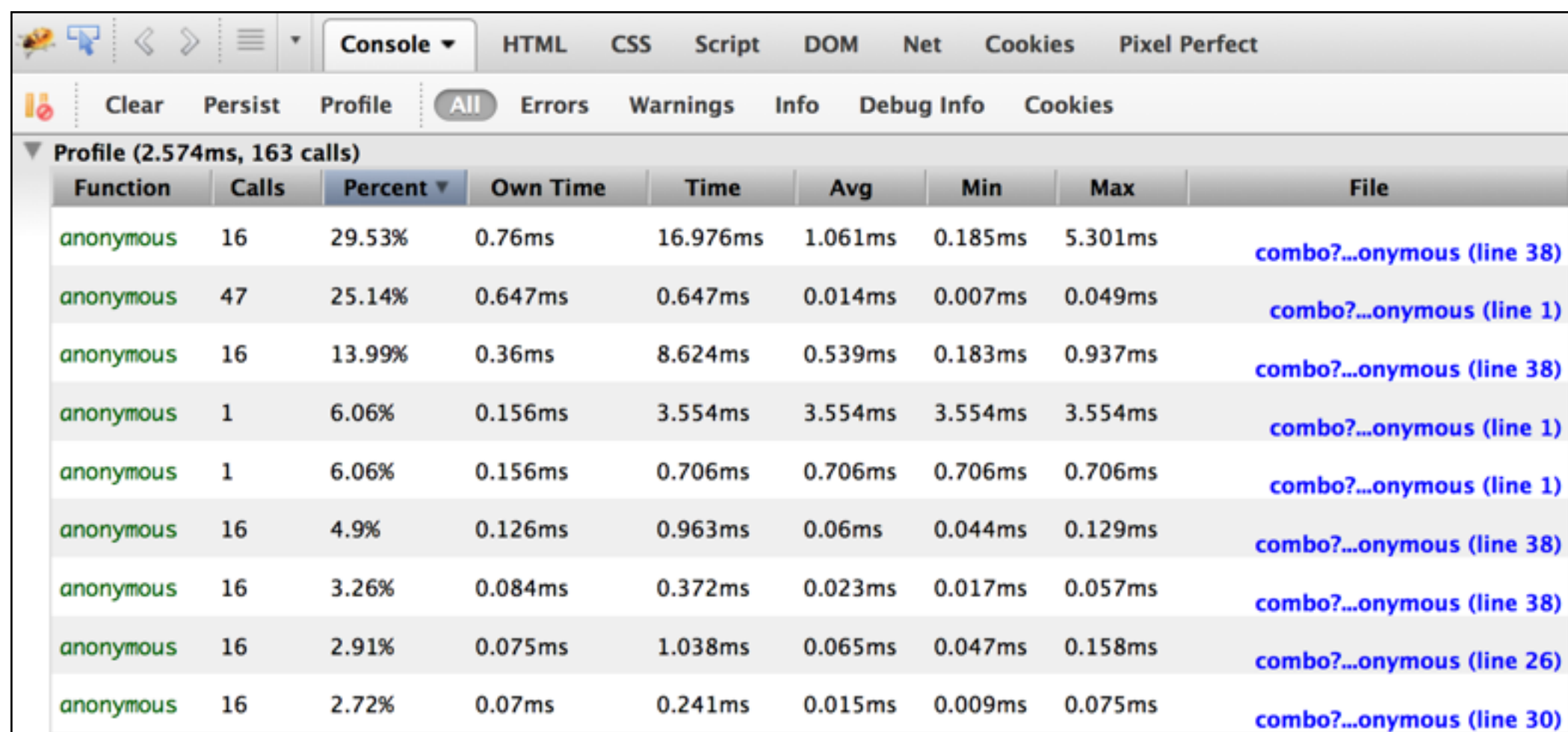
The screenshot displays the Chrome DevTools Profiles tab, showing a list of CPU profiles. The 'Profile 1' profile is selected, and its details are shown in the main panel. The table below represents the data shown in the 'Function' column of the profile details.

Self	Total	Function
4793.0 ms	4793.0 ms	(idle)
620.3 ms	620.3 ms	(program)
64.0 ms	64.0 ms	(garbage collector)
35.5 ms	35.5 ms	▶ o.findElements skitch.js:5
24.4 ms	62.9 ms	▶ GetDOMError
23.4 ms	23.4 ms	▶ get offsetWidth
17.3 ms	48.7 ms	▶ b.execCb skitch.js:6
12.2 ms	87.3 ms	chrome-extension://pioclpoplcldbaefihamjohnefbikjilc/skitch/... skitch.js:1
10.2 ms	10.2 ms	▶ removeChild
10.2 ms	10.2 ms	▶ b.nameToUrl skitch.js:6
10.2 ms	53.8 ms	▶ Apply
9.1 ms	9.1 ms	▶ get cookie
8.1 ms	22.3 ms	▶ (anonymous function) jquery-1.11.0.min.js:2
8.1 ms	8.1 ms	▶ get document
7.1 ms	7.1 ms	▶ insertBefore
6.1 ms	13.2 ms	▶ e.define skitch.js:7
6.1 ms	10.2 ms	http://habrahabr.ru/javascripts/1400594370/libs/highligh... highlight.js:1
5.1 ms	5.1 ms	▶ DebuggerScript._formatScript (program):135
5.1 ms	55.8 ms	▶ ⚠ target.(anonymous function) extensions::SafeBuiltins:10
5.1 ms	93.4 ms	▶ get test

Профилирование

Firebug

<https://getfirebug.com/>



Console HTML CSS Script DOM Net Cookies Pixel Perfect								
Clear Persist Profile All Errors Warnings Info Debug Info Cookies								
▼ Profile (2.574ms, 163 calls)								
Function	Calls	Percent ▼	Own Time	Time	Avg	Min	Max	File
anonymous	16	29.53%	0.76ms	16.976ms	1.061ms	0.185ms	5.301ms	combo?...onymous (line 38)
anonymous	47	25.14%	0.647ms	0.647ms	0.014ms	0.007ms	0.049ms	combo?...onymous (line 1)
anonymous	16	13.99%	0.36ms	8.624ms	0.539ms	0.183ms	0.937ms	combo?...onymous (line 38)
anonymous	1	6.06%	0.156ms	3.554ms	3.554ms	3.554ms	3.554ms	combo?...onymous (line 1)
anonymous	1	6.06%	0.156ms	0.706ms	0.706ms	0.706ms	0.706ms	combo?...onymous (line 1)
anonymous	16	4.9%	0.126ms	0.963ms	0.06ms	0.044ms	0.129ms	combo?...onymous (line 38)
anonymous	16	3.26%	0.084ms	0.372ms	0.023ms	0.017ms	0.057ms	combo?...onymous (line 38)
anonymous	16	2.91%	0.075ms	1.038ms	0.065ms	0.047ms	0.158ms	combo?...onymous (line 26)
anonymous	16	2.72%	0.07ms	0.241ms	0.015ms	0.009ms	0.075ms	combo?...onymous (line 30)

Сжатие ресурсов

- Минификация — убираются лишние символы: пробелы, табуляция, комментарии и т.д.
- Обфускация — код запутывается, имена переменных заменяются на более короткие там, где это возможно

```
function MyClass(){
    this.foo = function(argument1, argument2){
        var addedArgs = parseInt(argument1)+parseInt(argument2);
        return addedArgs;
    }
    var anonymousInnerFunction = function(){
        // do stuff here!
    }
}
```

может превратиться в

```
function MyClass(){this.foo=function(c,b){var d=parseInt(c)
+parseInt(b);return d};var a=function(){}};
```

Самые популярные: YUI Compressor и Closure Compiler — поддерживают сжатие и обфускацию JavaScript, сжатие CSS

Сжатие ресурсов

CSSO (CSS-optimizer) – минимизатор CSS, в том числе выполняет структурную минимизацию

- Минимизация без изменения структуры:
 - Удаление пробелов
 - Удаление концевых ‘;’
 - Удаление комментариев
 - Минимизация margin и padding
 - Минимизация font-weight
 - и др.
- Минимизация с изменением структуры:
- Слияние блоков с одинаковыми селекторами
- Удаление перекрываемых свойств
- и др.

Пример:

```
.test { color: red }  
.test { color: green }
```

Обработка с минимизацией структуры:

```
.test{color:green}
```

Тестирование

TDD / BDD

- 1) написание теста до внесения желаемого изменения → тест падает с ошибками
- 2) написание кода → тест отработывает без ошибок
- 3) рефакторинг кода

- Уменьшение количества ошибок
- Улучшение поддерживаемости кода
- Улучшение дизайна кода

TDD = Test Driven Development
ориентирован на код

BDD = Behavior Driven Development

Ориентирован на поведение (думаем не функциями и возвращаемыми значениями, а поведением). Данный подход помогает:

- понять с чего начать
- понять что тестировать
- сколько тестов нужно написать за один подход
- как структурировать тесты

Наиболее популярные библиотеки: JUnit, Jasmine, Mocha

Литература

С чего начать?

[JavaScript: The Good Parts](#)

[Professional JavaScript for Web Developers](#)

[Eloquent JavaScript. A Modern Introduction to Programming](#)

[JavaScript: The Definitive Guide](#)

[JavaScript Patterns](#)

Шаблоны проектирования: [Learning JavaScript Design Patterns](#)

Улучшаем производительность

[High Performance JavaScript](#)

[Best Practices for Speeding Up Your Web Site](#)

[High Performance Browser Networking](#)

Блоги:

www.nczonline.net/blog/

<http://addyosmani.com/blog/>

<http://www.paulirish.com/>

The End

Exadel[®]