WRITING MAINTAINABLE AND PERFORMANT JS FOR DRUPAL

Who am I?

- Backend developer in past 

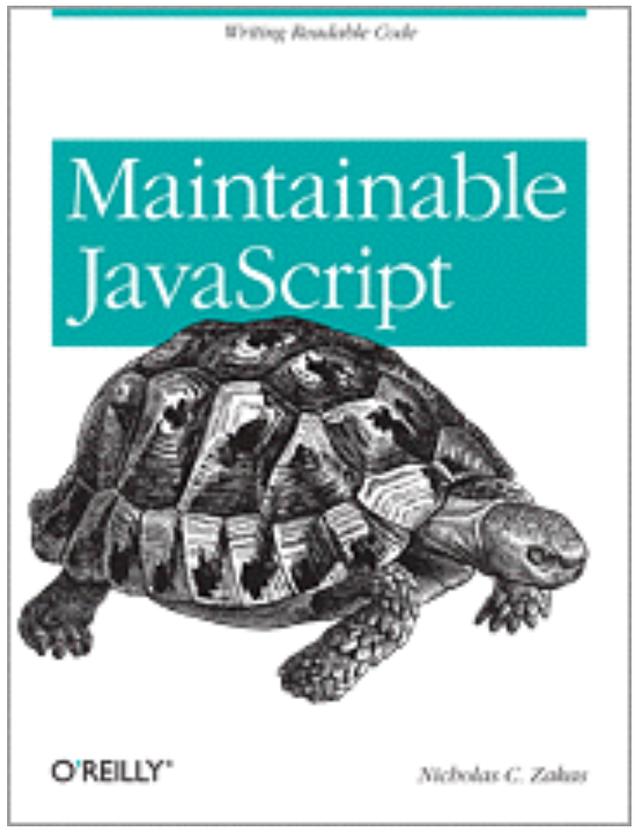
- Drupal Frontend developer now
- Senior Software Engineer at 

  @sergesemashko

 sergey.semashko

What is maintainable JS?

- Intuitive/Readable
- Understandable/Documented
- Adaptable
- Extendable
- Testable
- Debuggable



Nickolas C. Zakas

What is maintainable JS speaking in terms of Drupal?

- Written according to Drupal JS code standards
- Written according to jQuery code standards and best practices
- Integrated with Drupal environment and API

Wrapping JS code by anonymous function

Look! All variables are accessible from global scope:

```
var $ = jQuery.noConflict();
var sharedVar;
function foo() {
  sharedVar = `NJCAMP2015`;
}
function bar() {
  if (typeof sharedVar === `undefined`) {
    sharedVar = `defined!`;
  }
}
```

Wrapping JS code in anonymous function

How about now:

```
// last param is always undefined
(function (window, Modernizr, D, $, undefined)
  var sharedVar;
  function foo() {
    sharedVar = `NJCAMP2015`;
  }
  function bar() {
  // the same as typeof sharedVar === `undefined`
    if (sharedVar === undefined) {
      sharedVar = `defined!`;
    }
  }
})(window, Modernizr, Drupal, jQuery)
```

Wrapping JS code by anonymous function

Wins:

- Prevents extracting variables to global scope. *Global variables are never cleared by garbage collector.*
- Replacement of $ = jQuery.noConflict().
- Helps to minify files. JS Minifiers don't compress variable names for global scope.
- Good way to describe dependencies. It helps to avoid calling anything from global scope.

How we usually init JS…

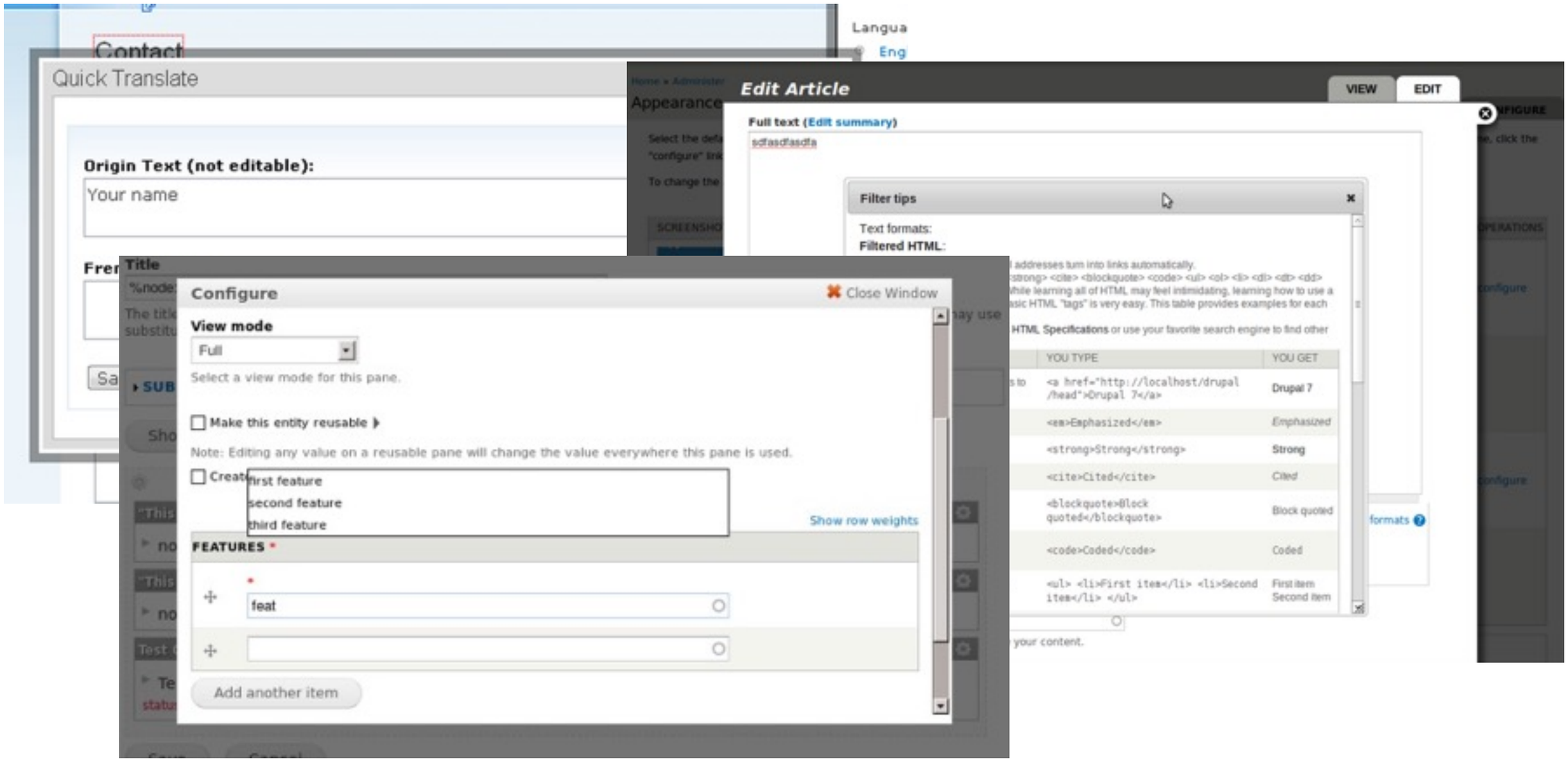
```
$(function() {

  $(`.autocomplete`).autocomplete(…);

});
```

And it’s fine until…

…we have dynamically inserted content



`$(`.autocomplete`).autocomplete(…);` - executed only once on DomContentLoaded. Dynamic content require to run the same code again.

Drupal Behaviors

- `attach()` - called on DOMContentLoaded, Modal popups, AJAX/AHAH.

  drupal.js:

```
$(function () {
 Drupal.attachBehaviors(document, Drupal.settings);
})
```

- `detach()` - called on `destroy` type of events. Ex.: element has been deleted from DOM, popup is closed, etc

- *Ok, that's it?*

- *No, don't repeat "Initialize" step*

`Drupal.attachBehaviours()` can be called multiple times on the same elements.

We need something like:

```
$(`.block:not(.processed)`).addClass(`processed`).doSomething();
```

… and we have jQuery.once():

```
$(`.block`).once(doSomething);
```

Behaviors: attach() and detach() + once() usage:

```
(function ($) {
var SELECTOR = `.my-block`;
  Drupal.behaviours.myBlock = {
    attach: function (context, settings) {
      $(SELECTOR, context).once(`my-block`).myPlugin();
    },
    detach: function (context, settings, trigger) {
      // 1. Unbind event handlers
      // 2. Remove elements you don’t need anymore
      // 3. Reset to the state before attach()
      $(SELECTOR, context).myPlugin(`destroy`);
    }
  }
})(jQuery)
```

Behaviors: attach() and detach() . Tips

1. Passing and using `context` is extremely important. Drupal always passes context to when calling Drupal.attachBehavior().

2. Prefer to use local `settings` variable passed to attach handler rather then Drupal.settings. AJAX/AHAH may pass different settings from Drupal.settings

Calling behaviors

- We have behaviors, so let's use them! Call Drupal.attachBehaviors() for dynamically inserted content:

```
function MyController (element, settings) {

  var $element = $(element);

  var ajaxUrl = Drupal.settings.basePath + $element.data(`url`);

  $.get(ajaxUrl, function (newBlock) {

     $element.append(newBlock);

     // apply all behaviors

     Drupal.attachBehaviors($element);

  });

}
```

Base url

Somewhere in the code…

```
$.ajax(`/ajax/my-module/some-action`);
```

Then you moved from example.com to example.com/subsite and the code stops working.

Use `Drupal.settings.basePath` :

```
var ajaxUrl = Drupal.settings.basePath + 'ajax/my-module/
some-action';
```

String output

Helpers available both on backend and frontend:

- Drupal.t('text'); - translates strings
- Drupal.checkPlain(name); - check for HTML entities
- Drupal.formatPlural(count, singular, plural, args, options); - translates strings with proper plural endings for multilingual sites

Javascript logic

There are several options how to organize JS logic.
You can use:

- Contrib library (jQuery plugin, etc)
- Drupal library
- custom controller / Library

Contrib libraries

Manageable by Bower

bower.json:

```
{
  "name": "project",
  "version": "0.0.1",
  "dependencies": {
    "masonry": "~3.1.0",
    "jquery.lazyload": "~1.9.3",
    "media-match": "~2.0.2",
    "shufflejs": "~2.1.2",
    "jquery.validation": "~1.13.0",
    "fastclick": "1.0.3",
    "jquery-sticky": "1.0.1"
  },
  "devDependencies": {
    "responsive-indicator": "~0.2.0",
  }
}
```

Drupal Library API

- Install Libraries API
- Add the library to sites/all/libraries
- Create a very short custom module that tells Libraries API about the library
- Add the library to the page where you want it
- Use it!

Drupal hook_library_info()

```php
/**
 * Implements hook_libraries_info().
 */
function MYMODULE_libraries_info() {
 $libraries['flexslider'] = array(
    'name' => 'FlexSlider',
    'vendor url' => 'http://flexslider.woothemes.com/',
    'download url' => 'https://github.com/woothemes/FlexSlider/zipball/master',
    'version arguments' => array(
      'file' => 'jquery.flexslider-min.js',
      // jQuery FlexSlider v2.1
      'pattern' => '/jQuery FlexSlider v(\d+\.+\d+)/',
      'lines' => 2,
    ),
    'files' => array(
      'js' => array(
        'jquery.flexslider-min.js',
      ),
    ),
  );
  return $libraries;
}
// Include library on the page
libraries_load('flexslider');
```

Writing JS controller. Principles.

- One controller per element
- Encapsulation - extract public methods, don’t tweak from outside of controller
- Controller must operate only in context of element

Writing reusable controllers

- One controller per element:

```
var DEFAULT_SETTINGS = {…};
$(SELECTOR, context).once(`calendar`, function () {
  var $this = $(this); // cache $(this) call
  // Store new instance of controller for future access in data-controller attribute.
  $this.data(`controller`, new Calendar(
     this,
     settings[$this.data(`nodeId`)] || DEFAULT_SETTINGS)
  );
})
```

jQuery plugins works according the same principle:

```
// jQuery plugin iterates over array of elements and initialize logic using same settings
$(SELECTOR, context).once(`myBehavior`).calendar({…});
```

Writing reusable controllers

- Incapsulate controllers, extract and use public methods, hide private ones (unless testing of private methods is obligatory):

```
// Calling public method of jQuery plugin
$(`.calendar`, context).calendar(`show`);

// Custom Calendar constructor
function Calendar(…) {
  …
  function _privateMethod() {…}
  function show() {…}
  this.show = show;
  return this;
}
// store reference to object after initialization
$(element).data(`calendar-instance`, new Calendar(…));
// get stored object and call public method
$(element).data(`calendar-instance`).show();
```

Writing reusable controllers

- Controller must operate only in context of element

```
function Calendar(element) {
  var $element = $(element);

  // bad, all .calendar-link from the page will be selected
  var $link = $(`.calendar-link`);

  // good, only items within context of $element will be selected
  var $button = $(`.button`, $element);
}
```

Communication between controllers. Pub/Sub pattern.

Publisher/Subscriber can be used for passing data, notifications. Ex. jQuery custom events:

```
// module1
$(document).trigger(`tooltipOpened`, [param1, param2]);
// module2
$(document).on(`tooltipOpened`, function (event, param1, param2) {…})
```

JS Code style Tips

- Check you code style with JSHint on writing or post-commit hook. Drupal 8 is shipped with ESHint, Yay!

- Avoid DOM traversable methods like .children(), .closest(), .is(), .next(), .prev() and etc. as they are slow and make code less readable. Get items directly by selector.

- Custom controllers should live in the same files with behaviors. Put behaviors on top of your file.

Naming tips

Use:

- UPPERCASED letters for constants:
  `var RESPONSE_TIMEOUT = 5000;`

- underscore before for private methods: `_privateMethod()`

- $ for jQuery objects to: `var $items = $(`.item`);`

- camelCase for variables and function names: `myVariable;` `myFunction()`

- Capitalized words for constructor names: `MyController()`

JS performance.

Don’t guess it - test it!

WEBPAGETEST

HOME | TEST RESULT | TEST HISTORY | FORUMS | DOCUMENTATION | ABOUT

Web Page Performance Test for

www.drupalcampnj.org

From: Dulles, VA - **IE 9** - **Cable**
1/28/2015, 10:06:48 PM

Need help improving?

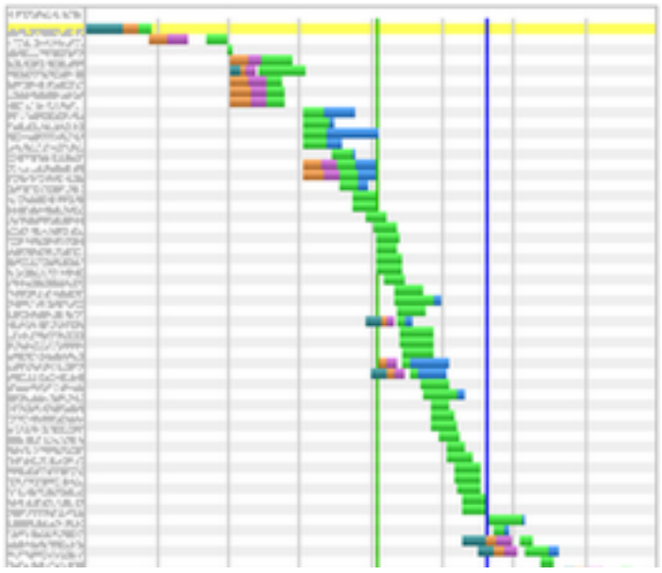
F	A	A	F	F	X
First Byte Time	Keep-alive Enabled	Compress Transfer	Compress Images	Cache static content	Effective use of CDN

Summary | Details | Performance Review | Content Breakdown | Domains | Screen Shot

Tester: IE9202-192.168.102.92
Re-run the test

Raw page data - Raw object data
Export HTTP Archive (.har)
See in ShowSlow
View Test Log

	Load Time	First Byte	Start Render	Speed Index	DOM Elements	Document Complete: Time	Document Complete: Requests	Document Complete: Bytes In	Fully Loaded: Time	Fully Loaded: Requests	Fully Loaded: Bytes In
First View	2.805s	0.978s	2.032s	2032	318	2.805s	47	585 KB	3.972s	62	742 KB
Repeat View	1.568s	0.471s	0.955s	1014	318	1.568s	8	30 KB	2.451s	18	85 KB

	Waterfall	Screen Shot	Video
First View (2.805s)			Filmstrip View - Watch Video

Google Developers

Products > PageSpeed Insights

PageSpeed Insights +1

http://www.drupalcampnj.org/ ANALYZE

Mobile Desktop

60 / 100 Speed

! Should Fix:

Eliminate render-blocking JavaScript and CSS in above-the-fold content

▸ Show how to fix

Optimize images

▸ Show how to fix

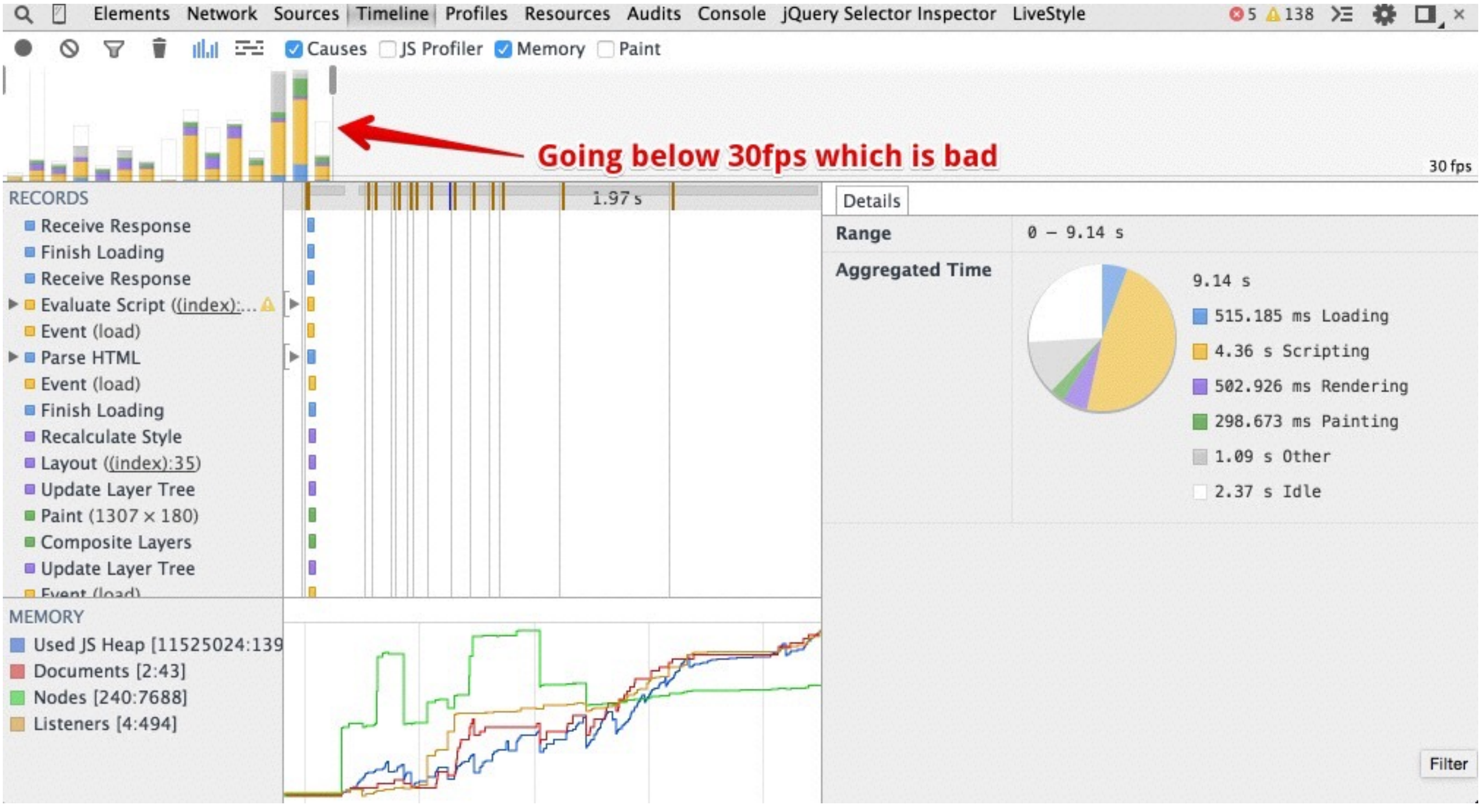
Leverage browser caching

▸ Show how to fix

! Consider Fixing:

Minify JavaScript

Use Timeline devTool to identify issues with rendering



Going over 60fps

- Replace jQuery.animate() by CSS3 animation. Check out GSAP or velocity.js CSS3 based animation libraries.

- Reduce layout thrashing

- For sticky elements use CSS3 position: sticky; (if supported by browser) instead of listening window.scroll()

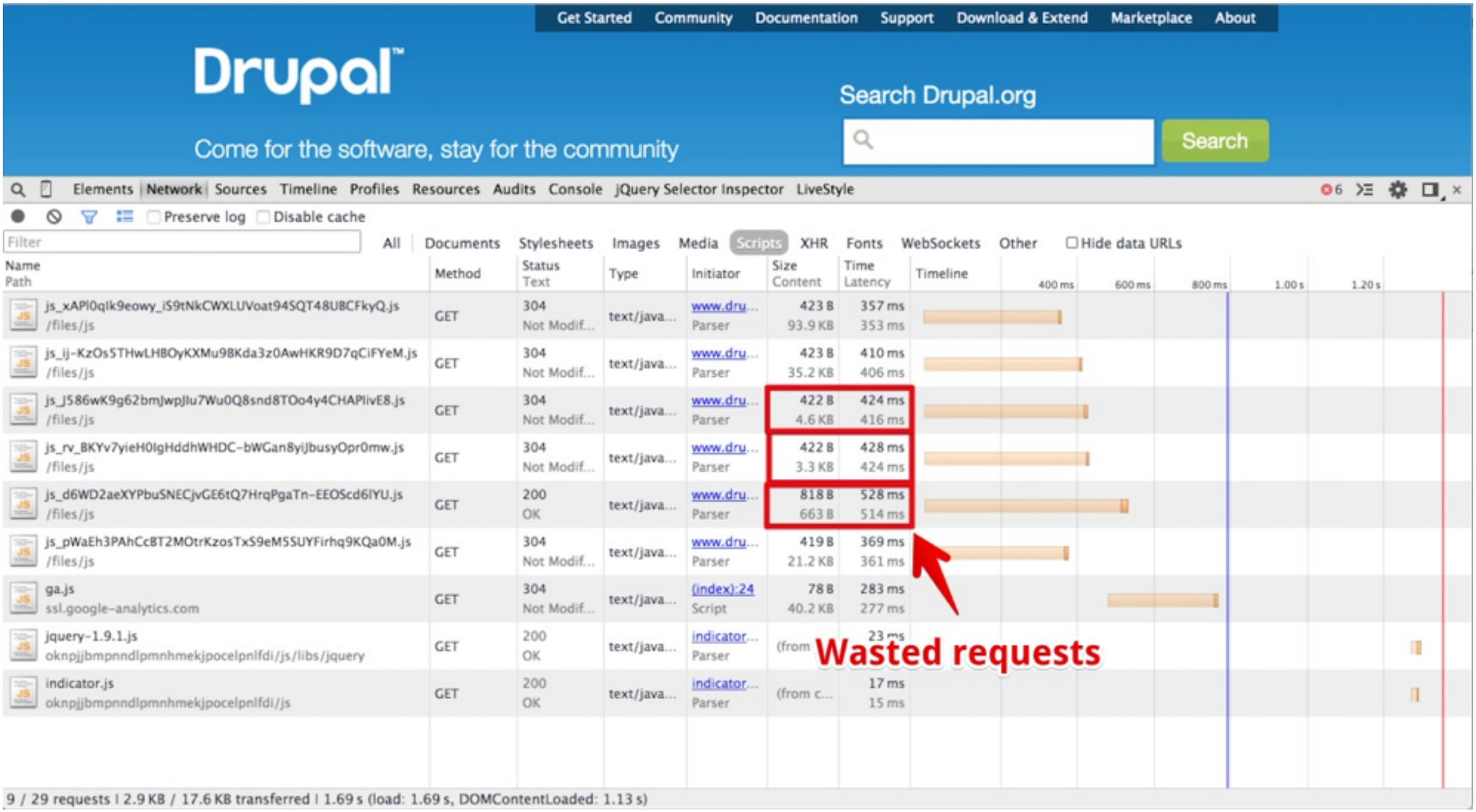
Going over 60fps: handling window.resize() & .scroll()

- Do as less as possible operations in resize()/scroll() handlers. Use non-blocking and light handlers.

- Use setTimeout to reduce unnecessary resize() processing:

```
var timer;
$(window).resize(function () {
  clearTimeout(timer);
  // call resizeHanlder only once after resize complete
  timer = setTimeout(resizeHanlder, 300);
})
```

Use Network tab to identify blocking Javascript



What breaks files grouping?

----- new group -----

file1.js, weight: 0, scope: header , group: JS_LIBRARY, every_page: true

file2.js, weight: 0.001, scope: header , group: JS_LIBRARY, every_page: true

----- new group -----

file3.js, weight: 0.002, scope: header , group: JS_LIBRARY, **every_page**: false

----- new group -----

file4.js, weight: 0.003, scope: header , group: JS_LIBRARY, every_page: false, **type**: inline

----- new group -----

file5.js, weight: 0.004, **scope**: footer, group: JS_THEME, every_page: false

file6.js, weight: 0.005, scope: footer, group: JS_THEME, every_page: false

----- new group -----

file7.js, weight: 0.006, scope: footer, group: JS_THEME, every_page: false, **type**: external

----- new group -----

file8.js, weight: 0.007, scope: footer, group: JS_THEME, every_page: false, **type**: file

Smarter JS grouping

- Advanced CSS/JS aggregation module
- Cache External files module
- hook_alter_js() - group files manually to reduce amount of JS requests:

```
/**
 * Implements hook_js_alter().
 */
function mymodule_js_alter(&$javascript) {
  $js_to_sort = array(`js/example.js`,`misc/drupal.js`, …)
  foreach ($js_to_sort as $name) {
    $javascript[$name]['weight'] = $i++;
    $javascript[$name]['group'] = JS_DEFAULT;
    $javascript[$name]['every_page'] = FALSE;
  }
}
```

Fun fact: Drupal 8 has more lines of Javascript than Drupal 5 had lines of PHP.

— Jeff Eaton (@eaton) April 6, 2013

drupal8.local:8888

Manage | Shortcuts | admin

Content | Structure | Appearance | Extend | Configuration | People | Reports | Help

drupal8.local:8

Elements | Network | Sources | Timeline | Profiles | Resources | Audits | Console | jQuery Selector Inspector | LiveStyle

Preserve log | Disable cache

Filter | All | Documents | Stylesheets | Images | Media | Scripts | XHR | Fonts | WebSockets | Other

Name / Path	Method	Status / Text	Type	Initiator	Size / Content	Time / Latency	Timeline
/core/modules/quickedit/js/models	GET	OK	applicati...	Parser	693 B	288 ms	
AppView.js?v=8.0.0-dev /core/modules/quickedit/js/views	GET	200 OK	applicati...	drupal8.l... Parser	23.4 KB 22.9 KB	314 ms 291 ms	
FieldDecorationView.js?v=8.0.0-dev /core/modules/quickedit/js/views	GET	200 OK	applicati...	drupal8.l... Parser	10.4 KB 9.8 KB	316 ms 292 ms	
EntityDecorationView.js?v=8.0.0-dev /core/modules/quickedit/js/views	GET	200 OK	applicati...	drupal8.l... Parser	1.2 KB 741 B	324 ms 299 ms	
EntityToolbarView.js?v=8.0.0-dev /core/modules/quickedit/js/views	GET	200 OK	applicati...	drupal8.l... Parser	17.3 KB 16.8 KB	327 ms 300 ms	
ContextualLinkView.js?v=8.0.0-dev /core/modules/quickedit/js/views	GET	200 OK	applicati...	drupal8.l... Parser	2.1 KB 1.5 KB	325 ms 302 ms	
FieldToolba... /core/modules/quickedit/js/views	GET	200 OK	applicati...	drupal8.l... Parser	5.9 KB 5.4 KB	328 ms 303 ms	
EditorView.js?v=8.0.0-dev /core/modules/quickedit/js/views	GET	200 OK	applicati...	drupal8.l... Parser	12.1 KB 11.6 KB	330 ms 308 ms	
theme.js?v=8.0.0-dev /core/modules/quickedit/js	GET	200 OK	applicati...	drupal8.l... Parser	5.8 KB 5.3 KB	328 ms 307 ms	
jquery-1.9.1.js oknpjjbmpnndlpmnhmekj...pnlfdi/js/libs/jquery	GET	200 OK	text/java...	indicator... Parser	(from c...	11 ms 11 ms	
indicator.js oknpjjbmpn...lpmnhme...pocelpnlfdi/js	GET	200 OK	text/java...	indicator... Parser	(from c...	4 ms 4 ms	

71 / 102 requests | 591 KB / 744 KB transferred | 3.69 s (load: 2.91 s, DOMContentLoaded: 2.19 s)

Drupal 8: 591KB of JS

JS minification

- UglifyJS - for minifying custom JS. Setup a job using Grunt or Gulp.

- Speedy module - for minifying Drupal core JS

Future

- AMD for JS architecture - loading scripts on demand in Drupal 9, Drupal 8?

- Remove dependency on jQuery - logical code clean up. Already in Drupal 8.
- Deprecation of drupal_add_js(). Drupal 8 uses #attached library approach.
- New libraries in Drupal 8: Underscore, Backbone, jQuery UI Touch Punch, Modernizr, domReady, html5shiv & classList
- Far future: wide usage of HTTP2 - no need to aggregate files

Useful Links

- Drupal JS Manual
- Drupal JS coding standards
- Google style guide
- jQuery contributing style guide
- Google I/O 2013 - True Grit: Debugging CSS & Render Performance
- Rendering Without Lumps
- Performance Tooling

Thanks!
Questions?

@sergesemashko

siarhei_semashko@epam.com