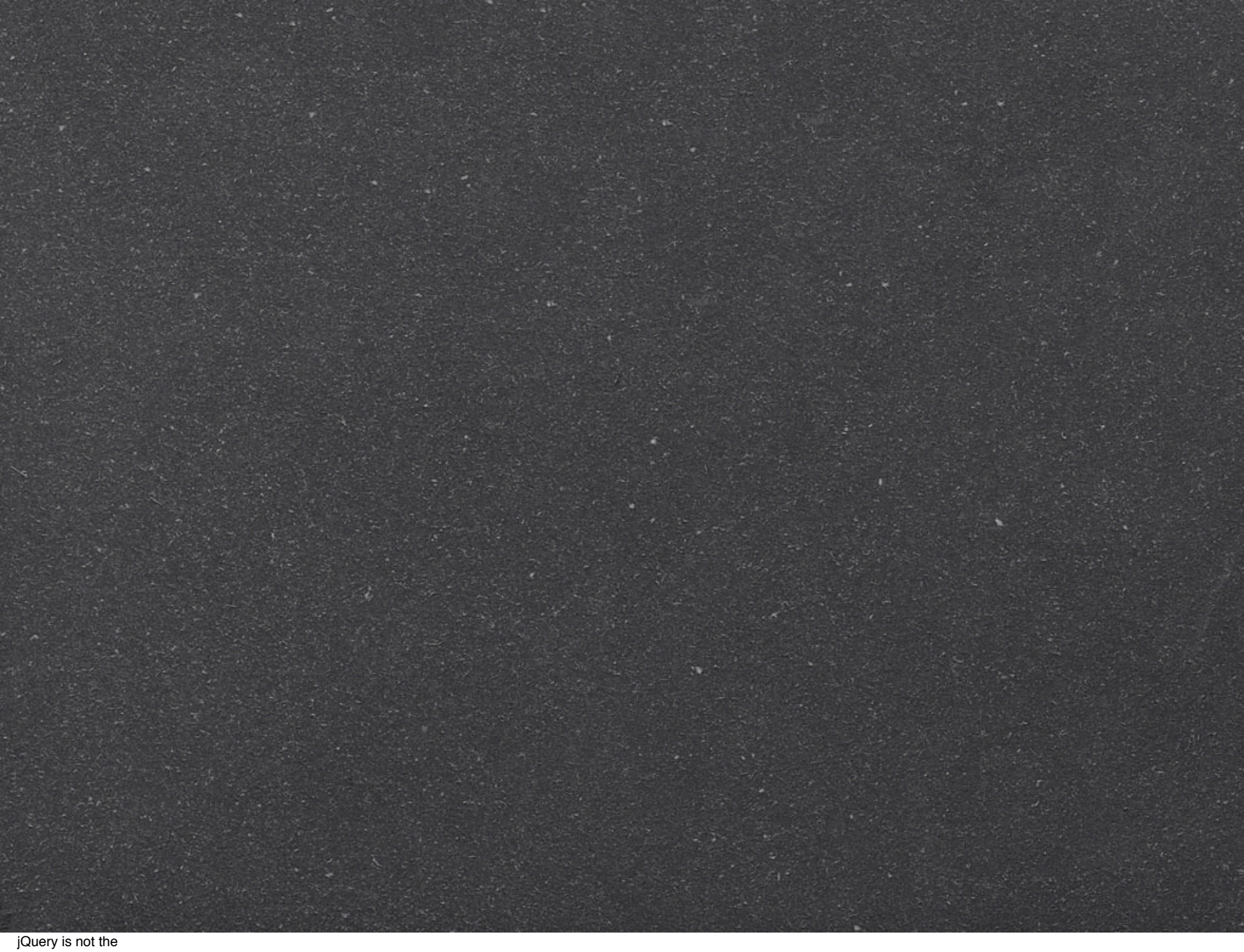


jQuery is not the answer!

stackoverflow

Questions | Tags | Users | Badges

Add a number to another number in JavaScript

0

hallo

I have got a number in my JavaScript variable! Now how do I add another number to it? Please

javascript

3 Answers

oldest

jQuery is not the answer!

Add a number to another number in JavaScript

0

hallo

I have got a number in my JavaScript variable! Now how do I add another number to it? Please

javascript

3 Answers

oldest

22

You should definitely use jQuery. It's really great and does all things

link | edit | flag

answered

I agree, jQuery is really the best, it solves all kinds of browser problems and is good, as well – ||sumc0

+1 jquery is best quality code ever, if you don't use your a idiot – Werry_Togan 4 mins ago

add comment

I think there's a jQuery plugin for that. Google for jQuery basic arithmetic plugin.

answered

jQuery is not the answer!

22

link | edit | flag

answered

I<3

1,

I agree, jQuery is really the best, it solves all kinds of browser problems and is good, as well – ||sumc0

+1 jquery is best quality code ever, if you don't use your a idiot – Werry_Togan 4 mins ago

add comment

4

I think there's a jQuery plugin for that. Google for jQuery basic arithmetic plugin.

link | edit | flag

answered

Tin

4,

yeah, jQuery is definately the way to go – fishnipples 5 mins ago

I used the jQuery diet plugin and lost 10kg in a week – jfatty 4 mins ago

add comment

-2

To add numbers together you should use the + operator, for example:

```
var a= 1;
var b= a+2;
alert(b);  // 3
```

jQuery is not the answer!

I think there's a jQuery plugin for that. Google for jQuery basic arithmetic plugin.

4

link | edit | flag

answered

Tim

4,

yeah, jQuery is definately the way to go – fishnipples 5 mins ago

I used the jQuery diet plugin and lost 10kg in a week – jfatty 4 mins ago

add comment

-2

To add numbers together you should use the + operator, for example:

```
var a= 1;
var b= a+2;
alert(b);  // 3
```

link | edit | delete | flag

answered

bo

so

-1 not enough jQuery – ||sumc0da 30 secs ago

you suck – Timothy Goatse 3 secs ago

jQuery is not the answer!

http://www.doxdesk.com/updates/2009.html

jQuery is not the answer!

NO !

JavaScript libraries are not “silver bullets”

(but some of them can help)

Real life experiences with Client-based Web Programming

Karl Krukow
LessPainful
karl@lesspainful.com

About me

- PhD, University of Aarhus, Trust Management: Security and Distributed Systems.
- Working at Trifork for about 6 years on Web, JavaScript, Java/JEE, Ruby/Rails, Clojure
- Last two years on iOS app programming for Trifork in the financial sector.
- Recently part of a start-up doing mobile automated testing: LessPainful ApS (http://www.lesspainful.com)

Servo-based rotation

LessPainful's test lab

Slightly more organized

Today's Theme

Today's Theme

- Not <u>every</u> client-based web project uses Flapjax with jQuery to let mobile devices speak WebSockets to cloud-deployed NodeJS servers via SocketIO.

Today's Theme

- Not <u>every</u> client-based web project uses Flapjax with jQuery to let mobile devices speak WebSockets to cloud-deployed NodeJS servers via SocketIO.
- (Although that would be fun...)

Today's Theme

- Not <u>every</u> client-based web project uses Flapjax with jQuery to let mobile devices speak WebSockets to cloud-deployed NodeJS servers via SocketIO.
- (Although that would be fun...)
- Today "Real Life" means problems that I've had to deal with in two larger JS-based project with customers.

Today's Theme

- Semi-large project (10.000+ LOC)
  - With JavaScript, how can we stay (fairly) sane?
- Developing for maintainance by customer
  - They may not know what a closure is!
- Context: Ubiquitous intranet web component, cross-site service integration, IE7+, FF3+
- Ubiquitous => quite performance sensitive.
- Ubiquitous => Critical search/navigation component
  - How to ensure quality?

Important Architectural Concerns

- Maintainability, Understandability, Extensibility [=> Developer focus]
- Interoperability
- Security [=> Focus on integration]
- Quality
- Performance [=> End-user experience]

First Challenge

First Challenge

- Manage a 10+ KLOC JS-based component to be maintained by client

First Challenge

- Manage a 10+ KLOC JS-based component to be maintained by client
- JavaScript focuses on flexibility, dynamism and expressive power
  - Not well-suited for large projects
  - Not well understood by the client (or the world in general)

First Challenge

- Manage a 10+ KLOC JS-based component to be maintained by client
- JavaScript focuses on flexibility, dynamism and expressive power
  - Not well-suited for large projects
  - Not well understood by the client (or the world in general)
- Try to compensate:
  - Tools, Testing, Code-Reviews, Documentation, Design Patterns.

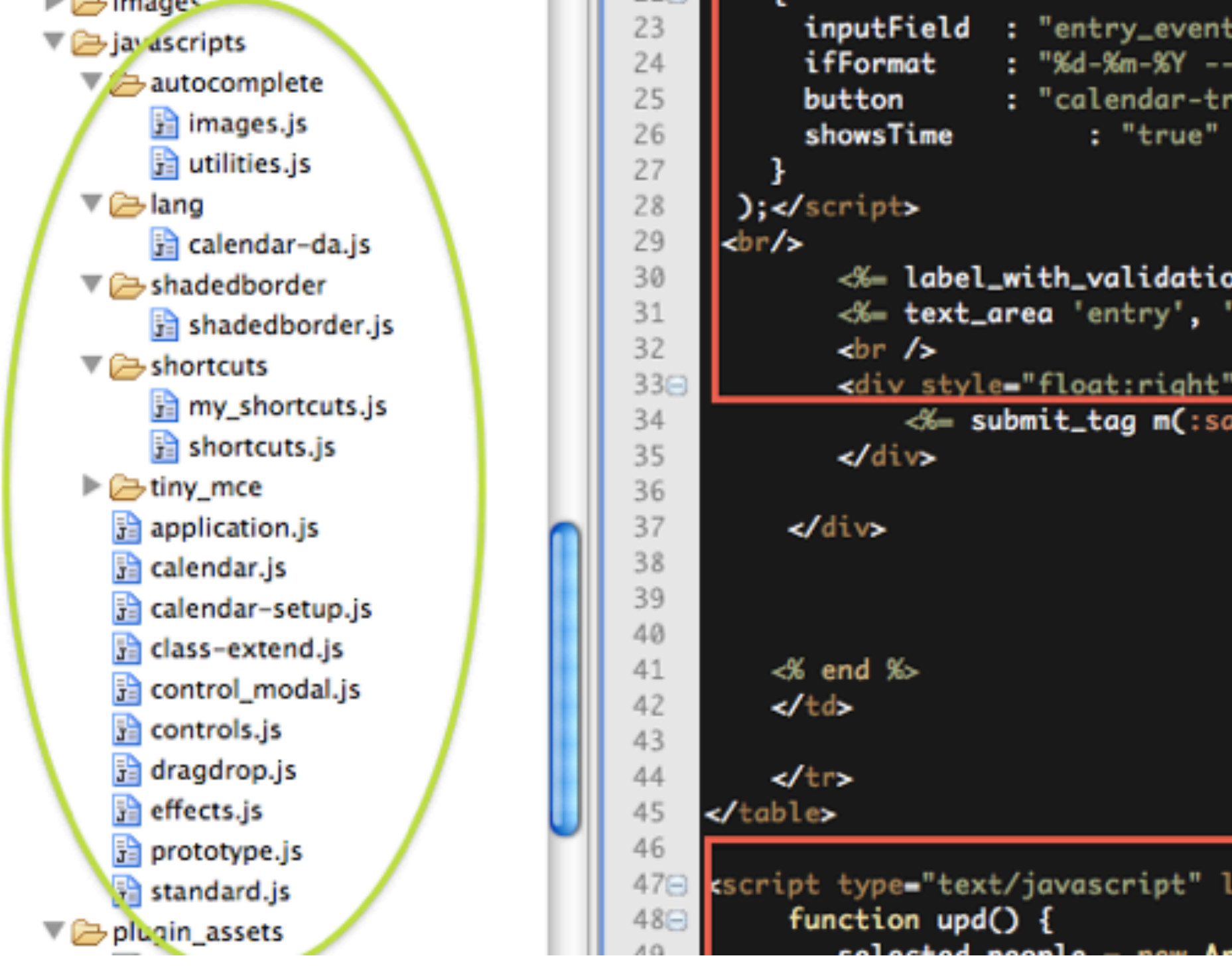
First Challenge:

Have you ever
found yourself
in a project like this?

First Challenge: Staying Sane

Have you ever
found yourself
in a project like this?

Development Time


images
javascripts
autocomplete
images.js
utilities.js
lang
calendar-da.js
shadedborder
shadedborder.js
shortcuts
my_shortcuts.js
shortcuts.js
tiny_mce
application.js
calendar.js
calendar-setup.js
class-extend.js
control_modal.js
controls.js
dragdrop.js
effects.js
prototype.js
standard.js
plugin_assets
23 inputField : "entry_event
24 ifFormat : "%d-%m-%Y --
25 button : "calendar-tr
26 showsTime : "true"
27 }
28);</script>
29
30 <%= label_with_validatio
31 <%= text_area 'entry', '
32
33 <div style="float:right"
34 <%= submit_tag m(:sa
35
36
37
38
39
40
41 <% end %>
42 </td>
43
44 </tr>
45 </table>
46
47 <script type="text/javascript" l
48 function upd() {

```
<br />
<script type="text/javascript">
Calendar.setup(
  {
    inputField  : "entry_event_occurred_at",
    ifFormat    : "%d-%m-%Y -- %H:%M",    // the date format
    button      : "calendar-trigger",
    showsTime        : "true"
  }
);</script>
<br/>
      <%= label_with_validation 'entry','contents', m(:new_note) %><br/>
      <%= text_area 'entry', 'contents',{:class=>"mceEditor", :value => par
      <br />
      <div style="float:right">
          <%= submit_tag m(:save_entry) %>  
      </div>

    </div>



  <% end %>
  </td>

  </tr>
</table>
```

```
    <% end %>
    </td>

    </tr>
</table>

<script type="text/javascript" language="javascript">
     function upd() {
        selected_people = new Array();
           var textElement = $('persons_persons');
           textElement.value.split(',').each(function(e) {
              e = e.strip().toLowerCase();
              var id = person_name_2_id[e];
              if (id != null) {
                    selected_people[id] = id;
              }
           });
           synchronizeImages(selected_people);
     }
     upd();
     Event.observe('persons_persons', 'keyup', upd);
     var now = new Date();
</script>
```

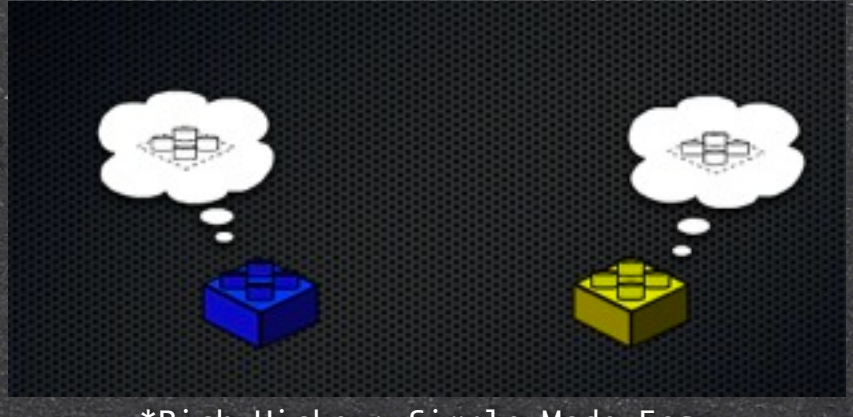
Development Time

Unstructured
No clear
architecture
No consistency

We need Structure!

Modularization

- Decompose a large and unstructured script into many small components.
- Each component is focused:
  - One concern
  - Public interface, private implementation
  - May (explicitly!) depend on other components



*Rich Hickey: Simple Made Easy
http://www.infoq.com/presentations/Simple-Made-Easy

AMD and RequireJS

http://requirejs.org/docs/whyamd.html

```
define(//define a module identified by path
 ["mod1", "mod2"], //Depends on mod1, mod2
 function (m1, m2) {//Local names of mod1 and mod2
   //Module pattern "built-in"
   var privateVar = 42,
       privateHelper = function () {};

    return {//Interface of defined module
      publicMethod: function(){
        //Can access private state and functions
        return privateVar + privateHelper();
      }
    };
});
```

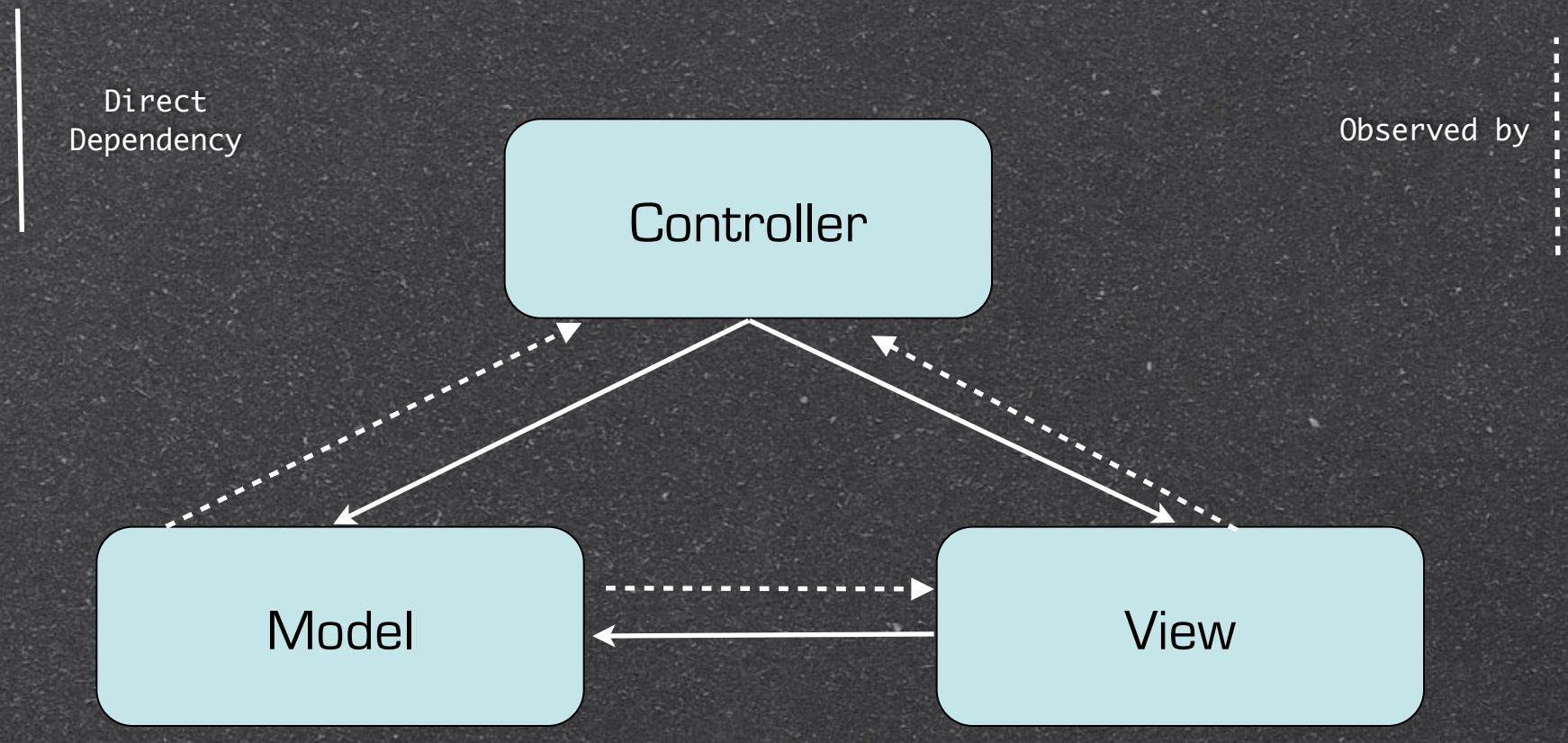
AMD

- AMD completely avoids global variables
- Dependencies are explicit
- Ordered script loading is managed
- Important in unknown environments
  - Portal Environment where each “portlet” contributes a part
  - JS-based component to exist in all intranet web systems.

Common Theme

- Model-View-Controller (or variation)
- Custom events / Observer
  - extend the DOM Level 2 event model to app- or component-specific events
  - Looser coupling, multiple listeners, bubbling, cancellation.
- Data binding

The triangle (again)



Document Modules: JSDoc

- Familiar for Java programmers (JavaDoc)
- Standardized => tool support
  - Some IDEs understand them, and can help with warnings, code-completion, inline documentation.
  - Generate HTML Doc.
  - Machine checkable contracts (Closure Compiler)
- Example IDEs: WebStorm, RubyMine, Spket, Aptana.

```
/**
 *
 * @param {string} str
 * @param {Array} arr
 */
function(str, arr) {
    arr.
}
```

Array	Array(ECMAScript.js2)	
concat	Array(system.js, .metadata/.plugins/org.ecl…	Array
every	Array(es3.js, closure-compiler/externs)	Boolean
filter	Array(es3.js, closure-compiler/externs)	!Array
forEach	Array(es3.js, closure-compiler/externs)	
index	Array(ECMAScript.js2)	Number
indexOf	Array(es3.js, closure-compiler/externs)	Number
input	Array(ECMAScript.js2)	Number
isArray	Array(es3.js, closure-compiler/externs)	Boolean
join	Array(system.js, .metadata/.plugins/org.eclip…	Array
lastIndexOf	Array(es3.js, closure-compiler/extern…	Number

Documentation for every

adv_js_sample

```
[ boolean ] Array.prototype.every( [ Function ] callback,
                                   [ Object, optional ] opt_thisobj )
```

Available in EcmaScript 5, Mozilla 1.6+.

Parameters:

callback
opt_thisobj

Returns:

@this {Object}

See also:

http://developer.mozilla.org/en/Core_JavaScript_1.5_Reference/Global_Objects/Array/every

Can be statically checked

```
/**
 * Represents a task with a name (i.e., title) and status.
 * @param {string} name the name of task
 * @param {boolean} done is this task finished?
 * @constructor
 */
Todo.data.Todo = function (name, done) {
    /**
     * A globally unique id generated at construct time.
     * @type {string}
     * @private
     */
    this.id = Todo.utils.uuid();
    /**
     * Name of this task.
     * @type {string}
     */
    this.name = name;
    /**
     * Done status.
     * @type {boolean}
     */
    this.done = done;
};
```

Can be statically checked

Second Challenge: Performance

Component to be used on EVERY page loaded on client intranet which is where most employees do most work.

Cost of modularization

Cost of modularization

Console HTML CSS Script DOM Net Cookies YSlow

Clear Persist All HTML CSS JS XHR Images Flash Media

GET table.css	200 OK	services.brics.dk	1 KB	130.225.2.179:443	124ms
GET auprint.css	200 OK	services.brics.dk	508 B	130.225.2.179:443	132ms
GET jquery-1.7.1.min.js	200 OK	services.brics.dk	91.7 KB	130.225.2.179:443	357ms
GET errorhandler.js	200 OK	services.brics.dk	190 B	130.225.2.179:443	41ms
GET jwigboost.js	200 OK	services.brics.dk	6.1 KB	130.225.2.179:443	114ms
GET table.js	200 OK	services.brics.dk	36.5 KB	130.225.2.179:443	233ms
GET dynamicContent.js	200 OK	services.brics.dk	11.7 KB	130.225.2.179:443	671ms
GET tiny_mce_src.js	200 OK	services.brics.dk	409.3 KB	130.225.2.179:443	645ms
GET submithandlers.js	200 OK	services.brics.dk	7.7 KB	130.225.2.179:443	120
GET popup.js	200 OK	services.brics.dk	2.8 KB	130.225.2.179:443	1
GET tooltip.js	200 OK	services.brics.dk	2 KB	130.225.2.179:443	
GET posting.js	200 OK	services.brics.dk	1.1 KB	130.225.2.179:443	
GET jquery.hoverIntent	200 OK	cs.au.dk	638 B	130.225.12.1:443	
GET au-functions.js	200 OK	cs.au.dk	3.6 KB	130.225.12.1:443	
GET RTEmagicC_forskn	200 OK	auinstallation28.cs.au.dk	9.8 KB	130.225.12.1:80	
GET RTEmagicC_talentu	200 OK	auinstallation28.cs.au.dk	14.5 KB	130.225.12.1:80	
GET RTEmagicC_videnu	200 OK	auinstallation28.cs.au.dk	17 KB	130.225.12.1:80	
GET RTEmagicC_uddan	200 OK	auinstallation28.cs.au.dk	21.2 KB	130.225.12.1:80	
GET RTEmagicC_map_n	200 OK	auinstallation28.cs.au.dk	15.2 KB	130.225.12.1:80	
GET jwiglogo.gif	200 OK	services.brics.dk	1.3 KB	130.225.2.179:443	
GET slimbox.2.0.4.yui.js	200 OK	cs.au.dk	2.6 KB	130.225.12.1:443	
GET sprite_repeaters.pr	200 OK	cs.au.dk	3.7 KB	130.225.12.1:443	
GET sprite_specials.png	200 OK	cs.au.dk	17.6 KB	130.225.12.1:443	
GET aupassata_rg-web	Aborted	cmsdesign.au.dk	0 B	130.225.12.2:443	
GET aupassata_rg-web	200 OK	cmsdesign.au.dk	21 KB	130.225.12.2:443	
29 requests			818.8 KB		

Cost of modularization

Script DOM Net Cookies YSlow

XHR Images Flash Media

services.brics.dk	1 KB	130.225.2.179:443	124ms
services.brics.dk	508 B	130.225.2.179:443	132ms
services.brics.dk	91.7 KB	130.225.2.179:443	357ms
services.brics.dk	190 B	130.225.2.179:443	41ms
services.brics.dk	6.1 KB	130.225.2.179:443	114ms
services.brics.dk	36.5 KB	130.225.2.179:443	233ms
services.brics.dk	11.7 KB	130.225.2.179:443	671ms
services.brics.dk	409.3 KB	130.225.2.179:443	645ms
services.brics.dk	7.7 KB	130.225.2.179:443	120ms
services.brics.dk	2.8 KB	130.225.2.179:443	115ms
services.brics.dk	2 KB	130.225.2.179:443	111ms
services.brics.dk	1.1 KB	130.225.2.179:443	119ms
cs.au.dk	638 B	130.225.12.1:443	47ms
cs.au.dk	3.6 KB	130.225.12.1:443	46ms
auinstallation28.cs.au.dk	9.8 KB	130.225.12.1:80	651ms
auinstallation28.cs.au.dk	14.5 KB	130.225.12.1:80	651ms
auinstallation28.cs.au.dk	17 KB	130.225.12.1:80	651ms
auinstallation28.cs.au.dk	21.2 KB	130.225.12.1:80	651ms
auinstallation28.cs.au.dk	15.2 KB	130.225.12.1:80	651ms
services.brics.dk	1.3 KB	130.225.2.179:443	736ms
cs.au.dk	2.6 KB	130.225.12.1:443	90ms
cs.au.dk	3.7 KB	130.225.12.1:443	802ms
cs.au.dk	17.6 KB	130.225.12.1:443	879ms
cmsdesign.au.dk	0 B	130.225.12.2:443	734ms
cmsdesign.au.dk	21 KB	130.225.12.2:443	149ms
	818.8 KB		4.5s (onload: 6.21s)

Blocked: 1 of 1 YSlow 6.219s

Cost of modularization

Script DOM Net Cookies YSlow

XHR Images Flash Media

services.brics.dk	1 KB	130.225.2.179:443	124ms
services.brics.dk	508 B	130.225.2.179:443	132ms
services.brics.dk	91.7 KB	130.225.2.179:443	357ms
services.brics.dk	190 B	130.225.2.179:443	41ms
services.brics.dk	6.1 KB	130.225.2.179:443	114ms
services.brics.dk	36.5 KB	130.225.2.179:443	233ms
services.brics.dk	11.7 KB	130.225.2.179:443	671ms
services.brics.dk	409.3 KB	130.225.2.179:443	645ms
services.brics.dk	7.7 KB	130.225.2.179:443	120ms
services.brics.dk	2.8 KB	130.225.2.179:443	115ms
services.brics.dk	2 KB	130.225.2.179:443	111ms
services.brics.dk	1.1 KB	130.225.2.179:443	119ms
cs.au.dk	638 B	130.225.12.1:443	47ms
cs.au.dk	3.6 KB	130.225.12.1:443	46ms
auinstallation28.cs.au.dk	9.8 KB	130.225.12.1:80	651ms
auinstallation28.cs.au.dk	14.5 KB	130.225.12.1:80	651ms
auinstallation28.cs.au.dk	17 KB	130.225.12.1:80	651ms
auinstallation28.cs.au.dk	21.2 KB	130.225.12.1:80	651ms
auinstallation28.cs.au.dk	15.2 KB	130.225.12.1:80	651ms
services.brics.dk	1.3 KB	130.225.2.179:443	736ms
cs.au.dk	2.6 KB	130.225.12.1:443	90ms
cs.au.dk	3.7 KB	130.225.12.1:443	802ms
cs.au.dk	17.6 KB	130.225.12.1:443	879ms
cmsdesign.au.dk	0 B	130.225.12.2:443	734ms
cmsdesign.au.dk	21 KB	130.225.12.2:443	149ms
	818.8 KB		4.5s (onload: 6.21s)

Blocked: 1 of 1 YSlow 6.219s

Cost of modularization

Script DOM Net Cookies YSlow

XHR Images Flash Media

services.brics.dk	1 KB	130.225.2.179:443	124ms
services.brics.dk	508 B	130.225.2.179:443	132ms
services.brics.dk	91.7 KB	130.225.2.179:443	357ms
services.brics.dk	190 B	130.225.2.179:443	41ms
services.brics.dk	6.1 KB	130.225.2.179:443	114ms
services.brics.dk	36.5 KB	130.225.2.179:443	233ms
services.brics.dk	11.7 KB	130.225.2.179:443	671ms
services.brics.dk	409.3 KB	130.225.2.179:443	645ms
services.brics.dk	7.7 KB	130.225.2.179:443	120ms
services.brics.dk	2.8 KB	130.225.2.179:443	115ms
services.brics.dk	2 KB	130.225.2.179:443	111ms
services.brics.dk	1.1 KB	130.225.2.179:443	119ms
cs.au.dk	638 B	130.225.12.1:443	47ms
cs.au.dk	3.6 KB	130.225.12.1:443	46ms
auinstallation28.cs.au.dk	9.8 KB	130.225.12.1:80	651ms
auinstallation28.cs.au.dk	14.5 KB	130.225.12.1:80	651ms
auinstallation28.cs.au.dk	17 KB	130.225.12.1:80	651ms
auinstallation28.cs.au.dk	21.2 KB	130.225.12.1:80	651ms
auinstallation28.cs.au.dk	15.2 KB	130.225.12.1:80	651ms
services.brics.dk	1.3 KB	130.225.2.179:443	736ms
cs.au.dk	2.6 KB	130.225.12.1:443	90ms
cs.au.dk	3.7 KB	130.225.12.1:443	802ms
cs.au.dk	17.6 KB	130.225.12.1:443	879ms
cmsdesign.au.dk	0 B	130.225.12.2:443	734ms
cmsdesign.au.dk	21 KB	130.225.12.2:443	149ms
	818.8 KB		4.5s (onload: 6.21s)

Blocked: 1 of 1 YSlow 6.219s

Performance

There is real empirical evidence that substantiates the fact that speed is more than a feature. It's a requirement.
-- Fred Wilson, http://thinkvitamin.com/web-apps/fred-wilsons-10-golden-principles-of-successful-web-apps/

- Why care?
  - Business perspective!
  - Resources
- Server-side, client-side, network
  - Why care about *front end* performance?
- Latency vs perceived latency
- Start-up time is key

- undersøgelse ved YAHOO der viste at et forsinkelse på 400 ms gav et drop off på 4-9% i antal page views.
- Mozilla så 15% forøgelse af Firefox download ved at gøre siden 2.2 sek. hurtigere!
- Interessant observation ved Google at de brugere der havde været udsat for pausen var på virket i seks uger før de var tilbage til normal opførsel. Bruger hukommelse længere end en session!
- Google's page rank der er deres scorings mekanisme tager performance i betragtning!

Server-side: People not much interest until last couple of years.
Why front'end? 80-90% of what you can speed up is front'end or network layer.

Steve Souders
http://stevesouders.com/

- The front-end performance Go-To-Guy.
- Books:
  - High Performance Web Sites
  - Even Faster Web Sites
- Talks (many more)
  - http://www.infoq.com/presentations/High-Performance-HTML5
- Tools
  - YSlow
  - Cuzillion
  - SpriteMe
  - HammerHead
  - ...

http://velocityconf.com/velocity2012

Caching and Compression

A Good Strategy

A Good Strategy

- Set a “far future” expires header for ALL loaded resources (JS,CSS,IMG).
  - Ten years in the future!

A Good Strategy

- Set a “far future” expires header for ALL loaded resources (JS,CSS,IMG).
  - Ten years in the future!
- Make resources external (not inline).

A Good Strategy

- Set a “far future” expires header for ALL loaded resources (JS,CSS,IMG).
  - Ten years in the future!
- Make resources external (not inline).
- Next problem is how to push updates?
  - Browsers cache by URL
  - Simple solution:
    - html is usually dynamically created
    - simply include version numbers for all URLs
    - to push update simply change URL!

Netflix case

Improving Performance: Gzip

- Payload reduced in some cases 15x (typically in half)

Empty Cache	Primed Cache
804.8K 1HTML/Text	804.8K 1HTML/Text
284.5K 3JavaScript Files	284.5K 3JavaScript Files
73.4K 2Stylesheet Files	73.4K 2Stylesheet Files
8.0K20CSS Images	0.0K20CSS Images
55.0K60Images	0.0K60Images
1225.8K**Total size**	1162.8K**Total size**
86HTTP requests	**86HTTP requests**

Empty Cache	Primed Cache
51.6K 1HTML/Text	
0.0K 1XMLHttpRequest	
63.9K 3JavaScript Files	51.6K 1HTML/Text
15.4K 2Stylesheet Files	0.0K 1XMLHttpRequest
21.9K23CSS Images	0.0K23CSS Images
56.3K61Images	0.0K61Images
209.3K**Total size**	51.6K**Total size**
91HTTP requests	**86HTTP requests**

- User experience performance improvement 13% - 25%
- Network outbound traffic cut in half

NETFLIX

GZip considerations

- Surprisingly a large survey of top websites show that 15% of all traffic did not contain
  - Accept-Encoding: gzip
  - WHY? (all browsers support it)
- Anti-virus and other software may actually remove or change HTTP headers!
  - Important that content is optimized also for this case.
- To be safe: Only GZip content if you actually receive an Accept-Encoding: gzip header.
  - (or use standard software like apache/nginx)

Minimize number of requests

Minimizing number of requests

Minimizing number of requests

- One of the most important factors in HTTP performance is the number of requests.

Minimizing number of requests

- One of the most important factors in HTTP performance is the number of requests.
- General idea:
  - serve ONE JavaScript file not many
  - serve ONE CSS file not many
  - serve as few images as possible

Minimizing number of requests

- One of the most important factors in HTTP performance is the number of requests.
- General idea:
  - serve ONE JavaScript file not many
  - serve ONE CSS file not many
  - serve as few images as possible
- How? Simply concatenate all JavaScript files into one file (similarly CSS).
  - What about images? -> Sprites

Connections and hostname

Important this is HOSTNAME not IP address so simple DNS tricks can work (or multiple servers of course).

Connections and hostname

- Some pages have all HTTP resource served from one domain, others ‘shard’.
  - Some browsers e.g.,IE6+7 limit number of connections per domain to two!
  - IE 8, FF3 increase to 6 conn/domain.

Important this is HOSTNAME not IP address so simple DNS tricks can work (or multiple servers of course).

Connections and hostname

- Some pages have all HTTP resource served from one domain, others ‘shard’.
  - Some browsers e.g.,IE6+7 limit number of connections per domain to two!
  - IE 8, FF3 increase to 6 conn/domain.
- Blocks for parallel download.

Important this is HOSTNAME not IP address so simple DNS tricks can work (or multiple servers of course).

Connections and hostname

- Some pages have all HTTP resource served from one domain, others ‘shard’.
  - Some browsers e.g.,IE6+7 limit number of connections per domain to two!
  - IE 8, FF3 increase to 6 conn/domain.
- Blocks for parallel download.
- Often a speed-up can be acheived by splitting resources across more domains.

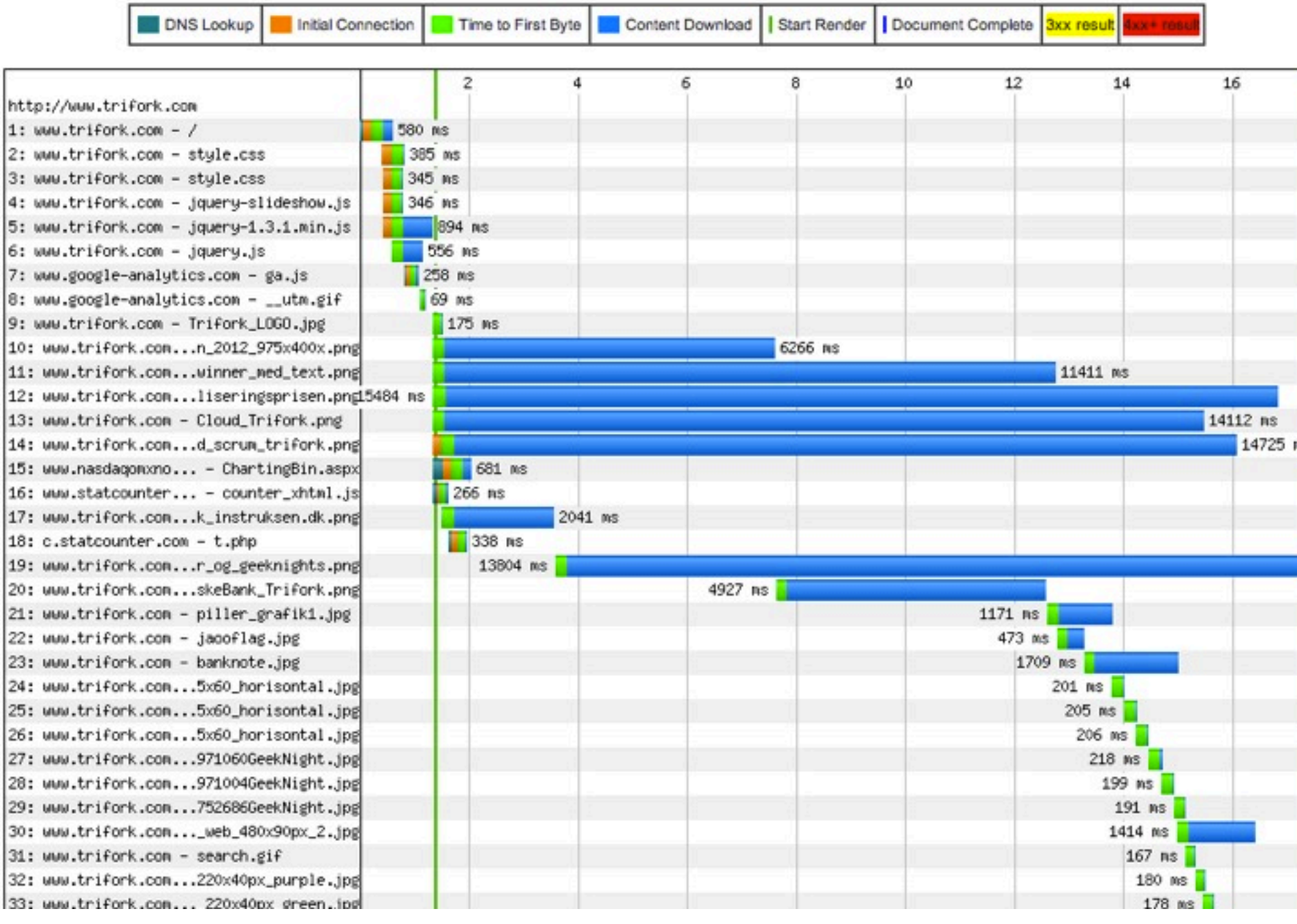
Important this is HOSTNAME not IP address so simple DNS tricks can work (or multiple servers of course).

Connections and hostname

- Some pages have all HTTP resource served from one domain, others ‘shard’.
  - Some browsers e.g.,IE6+7 limit number of connections per domain to two!
  - IE 8, FF3 increase to 6 conn/domain.
- Blocks for parallel download.
- Often a speed-up can be acheived by splitting resources across more domains.
- Research from Yahoo! shows 2 is optimal as a general rule of thumb.

Important this is HOSTNAME not IP address so simple DNS tricks can work (or multiple servers of course).

Waterfall View



Progressive rendering

Progressive rendering

Har mere at gøre med følelsen (perceived performance vs performance).
Der sker noget ("systemet virker"), giver en fornemmelse af hvor lang tid man skal vente, noget at lave mens man venter.
Progressive rendering: avoid flash-of-unstyled content: means that browsers that see links at bottom, may wait with rendering.

Progressive rendering

- Important: users get visual feedback quickly.
  - Progress indicator (HTML page, spinners for Ajax).

Har mere at gøre med følelsen (perceived performance vs performance).
Der sker noget ("systemet virker"), giver en fornemmelse af hvor lang tid man skal vente, noget at lave mens man venter.
Progressive rendering: avoid flash-of-unstyled content: means that browsers that see links at bottom, may wait with rendering.

Progressive rendering

- Important: users get visual feedback quickly.
  - Progress indicator (HTML page, spinners for Ajax).
- Browsers render content as quickly as possible, but at same time tries to improve user experience.
  - Yahoo experiment with <link> tags at bottom!

Har mere at gøre med følelsen (perceived performance vs performance).
Der sker noget ("systemet virker"), giver en fornemmelse af hvor lang tid man skal vente, noget at lave mens man venter.
Progressive rendering: avoid flash-of-unstyled content: means that browsers that see links at bottom, may wait with rendering.

Progressive rendering

- Important: users get visual feedback quickly.
  - Progress indicator (HTML page, spinners for Ajax).
- Browsers render content as quickly as possible, but at same time tries to improve user experience.
  - Yahoo experiment with <link> tags at bottom!
- Put stylesheets in the HEAD element at the top of the page!

Har mere at gøre med følelsen (perceived performance vs performance).
Der sker noget ("systemet virker"), giver en fornemmelse af hvor lang tid man skal vente, noget at lave mens man venter.
Progressive rendering: avoid flash-of-unstyled content: means that browsers that see links at bottom, may wait with rendering.

Script tags

Why no progressive?: script may document.write which dynamically changes the page.
Why no parallel down?: they might arrive out of order - order of script matters.

Script tags

- With Script tags the opposite rule applies!
  - Put script tags at the bottom of the page.

Why no progressive?: script may document.write which dynamically changes the page.
Why no parallel down?: they might arrive out of order - order of script matters.

Script tags

- With Script tags the opposite rule applies!
  - Put script tags at the bottom of the page.
- Why do it? Two reasons
  - Better progressive rendering
  - Better parallelization.

Why no progressive?: script may document.write which dynamically changes the page.
Why no parallel down?: they might arrive out of order - order of script matters.

Script tags

- With Script tags the opposite rule applies!
  - Put script tags at the bottom of the page.
- Why do it? Two reasons
  - Better progressive rendering
  - Better parallelization.
- Why does it work?
  - With scripts progressive render is blocked for all content below script tag.
  - Scripts block parallel downloading.

Why no progressive?: script may document.write which dynamically changes the page.
Why no parallel down?: they might arrive out of order - order of script matters.

http://stevesouders.com/hpws/js-middle.php

Rule 6 - Put Scripts at the Bottom

Rules 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Example 1 - Scripts in the Middle

Examples 1 2 3 4 5 6

This page contains a `<SCRIPT SRC="...">` tag in the middle of the page that takes 10 seconds to load. This illustrates two problems.

PROBLEM 1: Everything below the script won't render until the script is loaded.

The `<SCRIPT SRC="...">` *tag is here.*

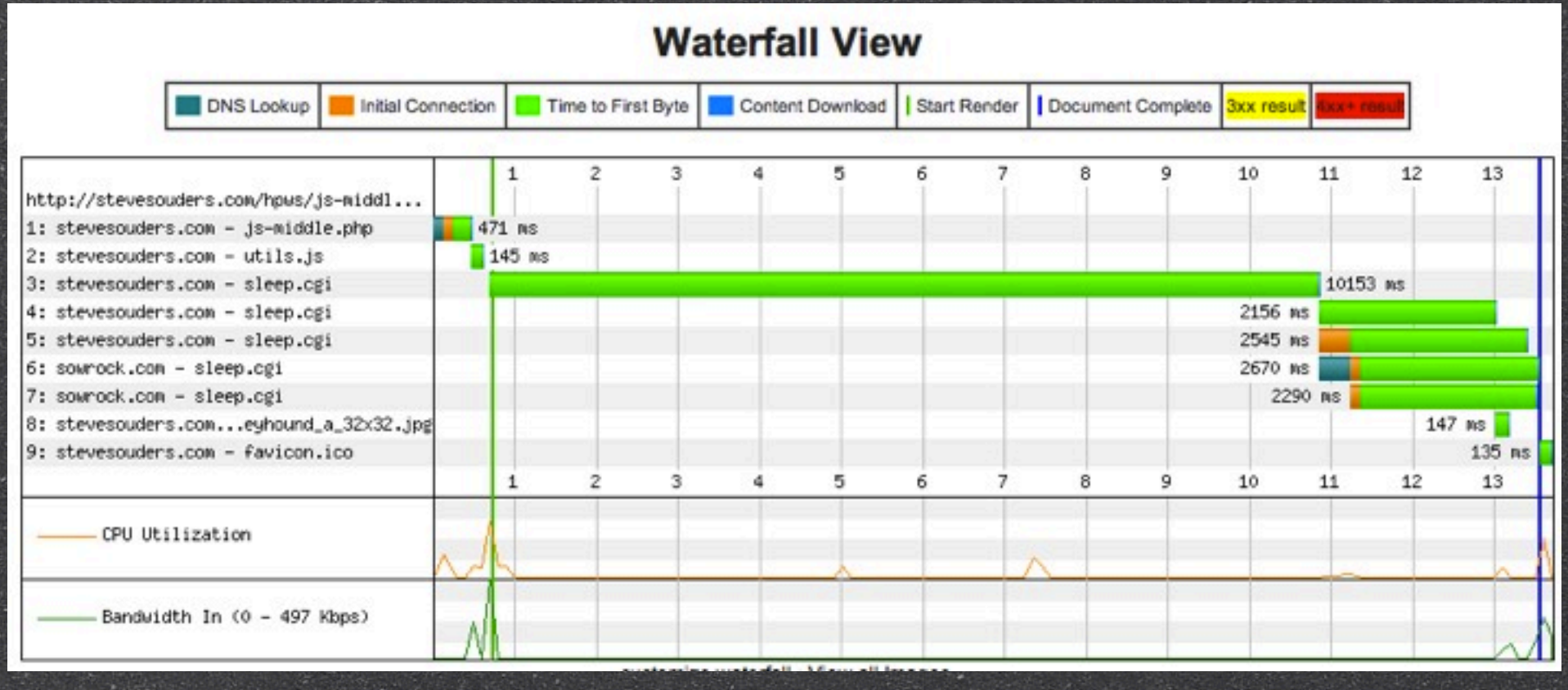
PROBLEM 2: All components below the script don't start downloading until the script is done.

The script blocks these images from being downloaded, so they're not available until much later during rendering.

It's even worse if scripts are in the document `HEAD`. Look at Example 2 - Scripts Top vs. Bottom to see a comparison of scripts at the top vs. the bottom of the page.

Example.

Waterfall: script in middle



Resource size

Minimizing size of scripts (+css)

Minimizing size of scripts (+css)

- Uncompressed size of scripts and resources is also very important.
  - Smaller scripts are faster!
  - 15% uncompressed traffic!
  - Fit in caches (mobile with small caches)
  - JavaScript has to be parsed and run!

Minimizing size of scripts (+css)

- Uncompressed size of scripts and resources is also very important.
  - Smaller scripts are faster!
  - 15% uncompressed traffic!
  - Fit in caches (mobile with small caches)
  - JavaScript has to be parsed and run!

Minimizing size of scripts (+css)

- Uncompressed size of scripts and resources is also very important.
  - Smaller scripts are faster!
  - 15% uncompressed traffic!
  - Fit in caches (mobile with small caches)
  - JavaScript has to be parsed and run!

- According to Yahoo!'s Exceptional Performance Team, 40% to 60% of Yahoo!'s users have an empty cache experience and about 20% of all page views are done with an empty cache (see this article by Tenni Theurer on the YUIBlog for more information on browser cache usage).

Minification

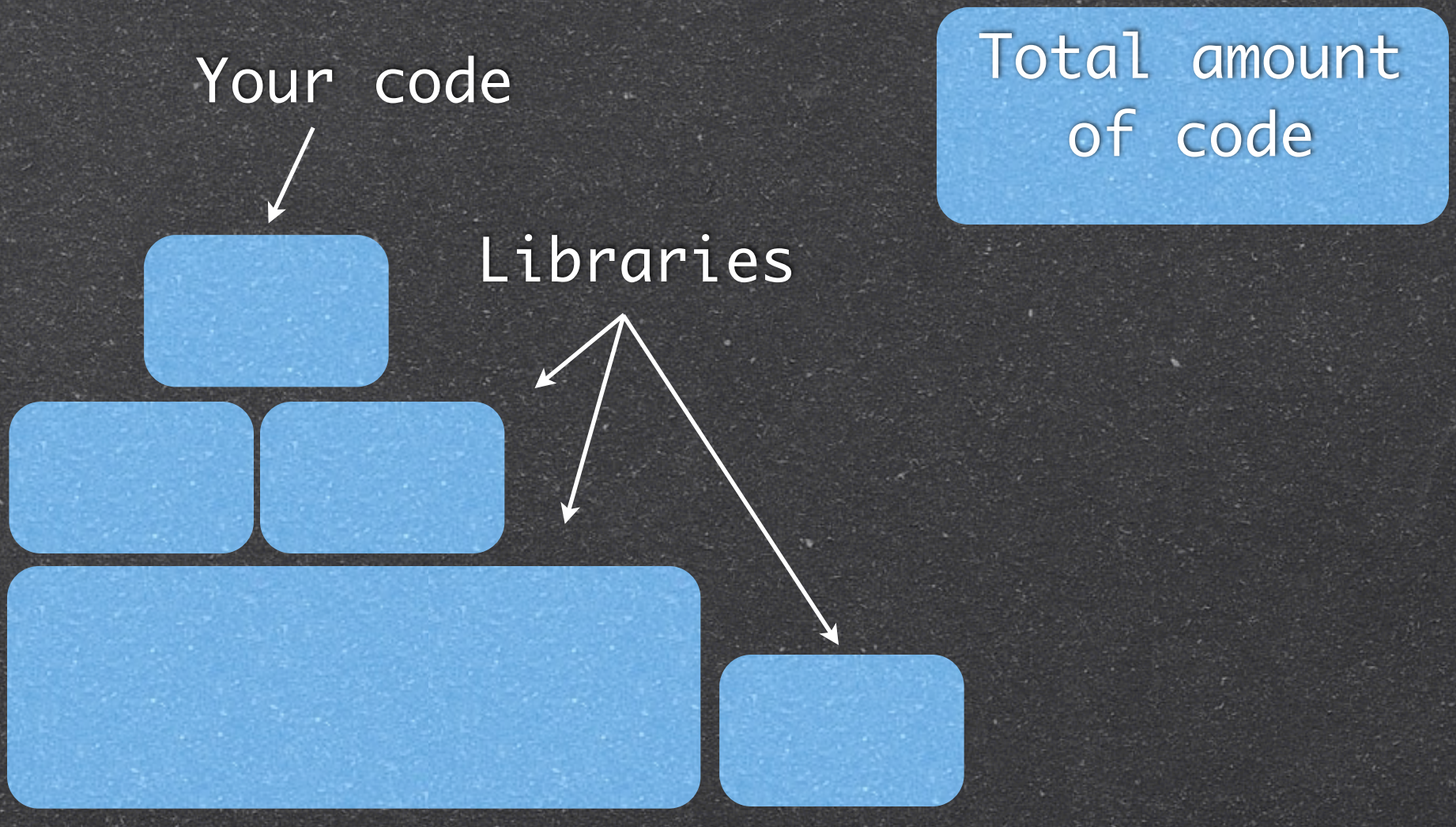
- There are many tools that take scripts, optimize and output a single script.
  - JSMin (comments, whitespace)
  - YUICompressor (+local optimization)
  - Dojo Shrinksafe (+local optimization)
  - UglifyJS (+local optimization)
  - Google Closure Compiler (+global optimization)
- Potential drawbacks:
  - tool may introduce errors
  - debugging
    (Note Source Maps: http://www.html5rocks.com/en/tutorials/developertools/sourcemaps/)
  - requires build process

Typical JS Application composition

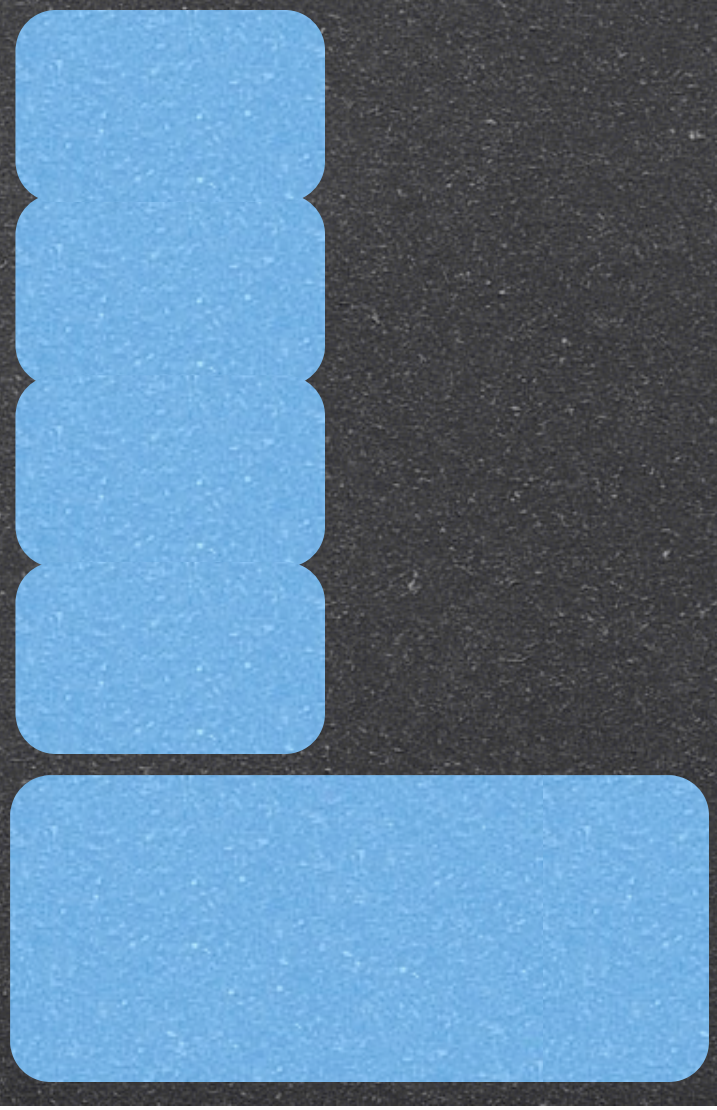
Total amount of code

Typical JS Application composition

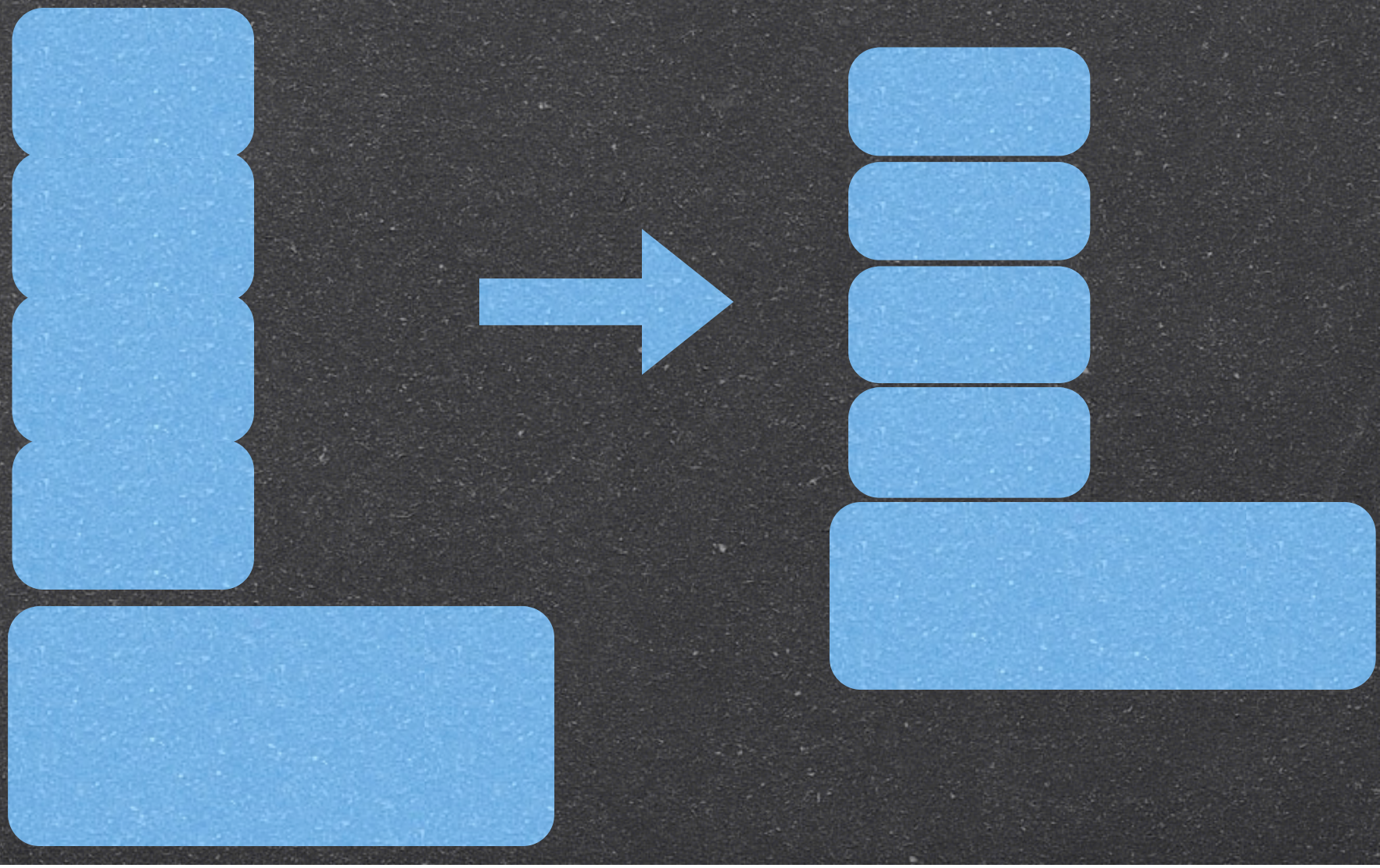
Typical JS Application composition



Local optimization



Local optimization



YUICompressor

Remove comments
Remove extra whitespace
Identifier replacement (var longname; => var a)
Micro optimizations

YUICompressor

Remove comments
Remove extra whitespace
Identifier replacement (var longname; => var a)
Micro optimizations

Closure Compiler

Closure Compiler

- Extensible optimizing JavaScript-to-JavaScript compiler

Closure Compiler

- Extensible optimizing JavaScript-to-JavaScript compiler
- Several modes - the interesting one is “advanced mode”.

Closure Compiler

- Extensible optimizing JavaScript-to-JavaScript compiler
- Several modes - the interesting one is “advanced mode”.
  - whole-program analysis (all js files + extern must be supplied).

Closure Compiler

- Extensible optimizing JavaScript-to-JavaScript compiler
- Several modes - the interesting one is “advanced mode”.
  - whole-program analysis (all js files + extern must be supplied).
  - optimizations (e.g., dead-code elimination, constant folding/propagation)

Closure Compiler

- Extensible optimizing JavaScript-to-JavaScript compiler
- Several modes - the interesting one is “advanced mode”.
  - whole-program analysis (all js files + extern must be supplied).
  - optimizations (e.g., dead-code elimination, constant folding/propagation)
  - optional type-checking, @private access, ...

Closure Compiler

- Extensible optimizing JavaScript-to-JavaScript compiler
- Several modes - the interesting one is “advanced mode”.
  - whole-program analysis (all js files + extern must be supplied).
  - optimizations (e.g., dead-code elimination, constant folding/propagation)
  - optional type-checking, @private access, ...
  - lazy-loading of modules

Closure Compiler

- Extensible optimizing JavaScript-to-JavaScript compiler
- Several modes - the interesting one is “advanced mode”.
  - whole-program analysis (all js files + extern must be supplied).
  - optimizations (e.g., dead-code elimination, constant folding/propagation)
  - optional type-checking, @private access, ...
  - lazy-loading of modules
- Requires writing JavaScript code in a particular way.

Closure Compiler

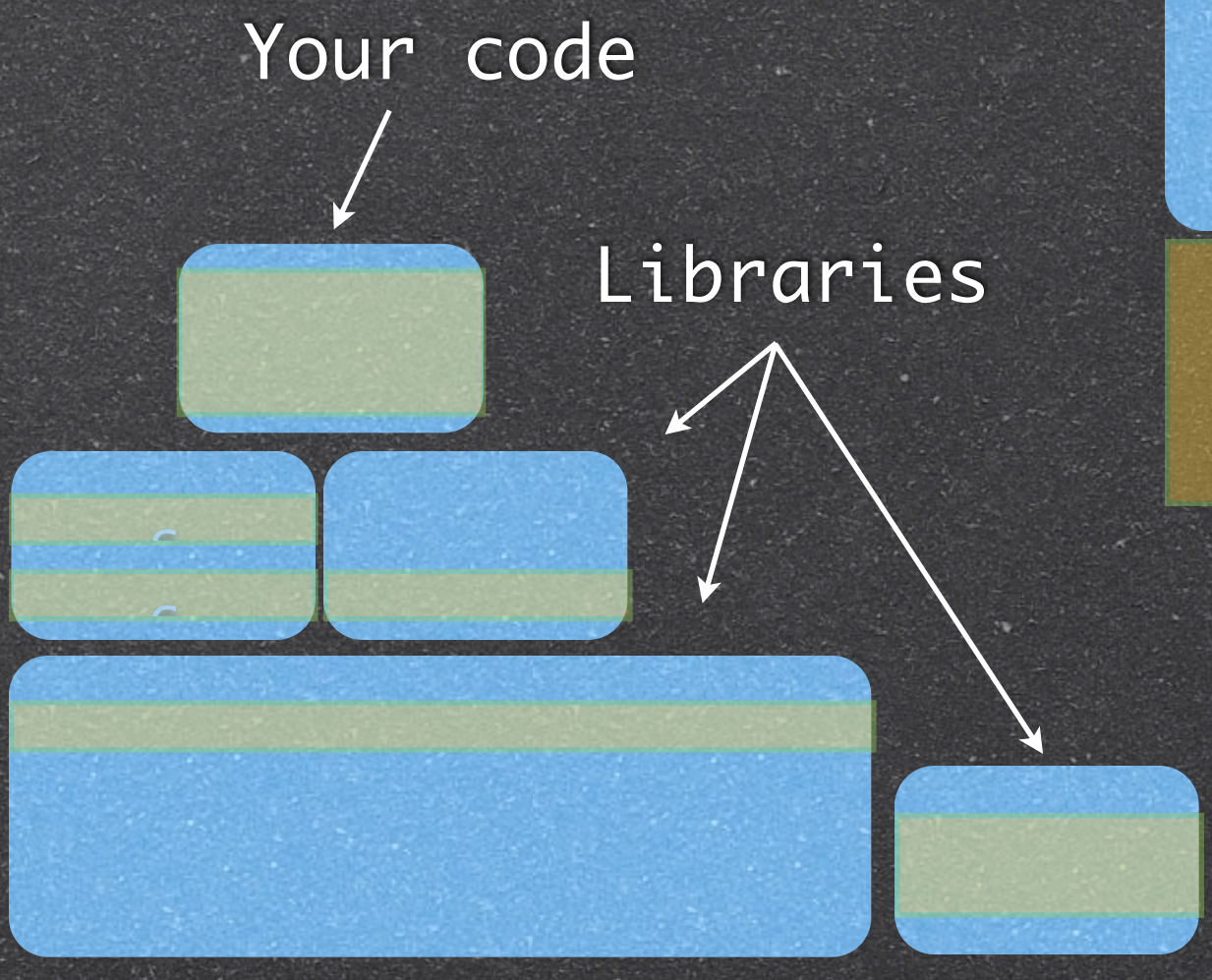
- Extensible optimizing JavaScript-to-JavaScript compiler
- Several modes - the interesting one is “advanced mode”.
  - whole-program analysis (all js files + extern must be supplied).
  - optimizations (e.g., dead-code elimination, constant folding/propagation)
  - optional type-checking, @private access, ...
  - lazy-loading of modules
- Requires writing JavaScript code in a particular way.
- Very under-appreciated! Not just yet another minifier!

Typical JS Application composition

Total amount of code

Code actually used at runtime

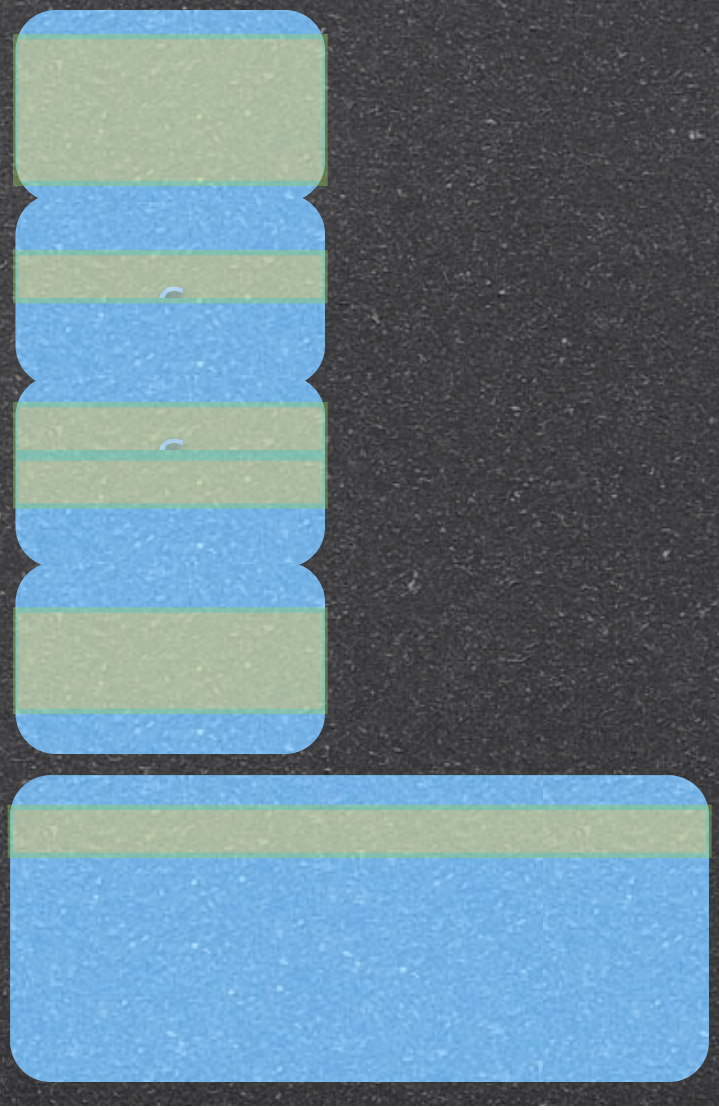
Typical JS Application composition



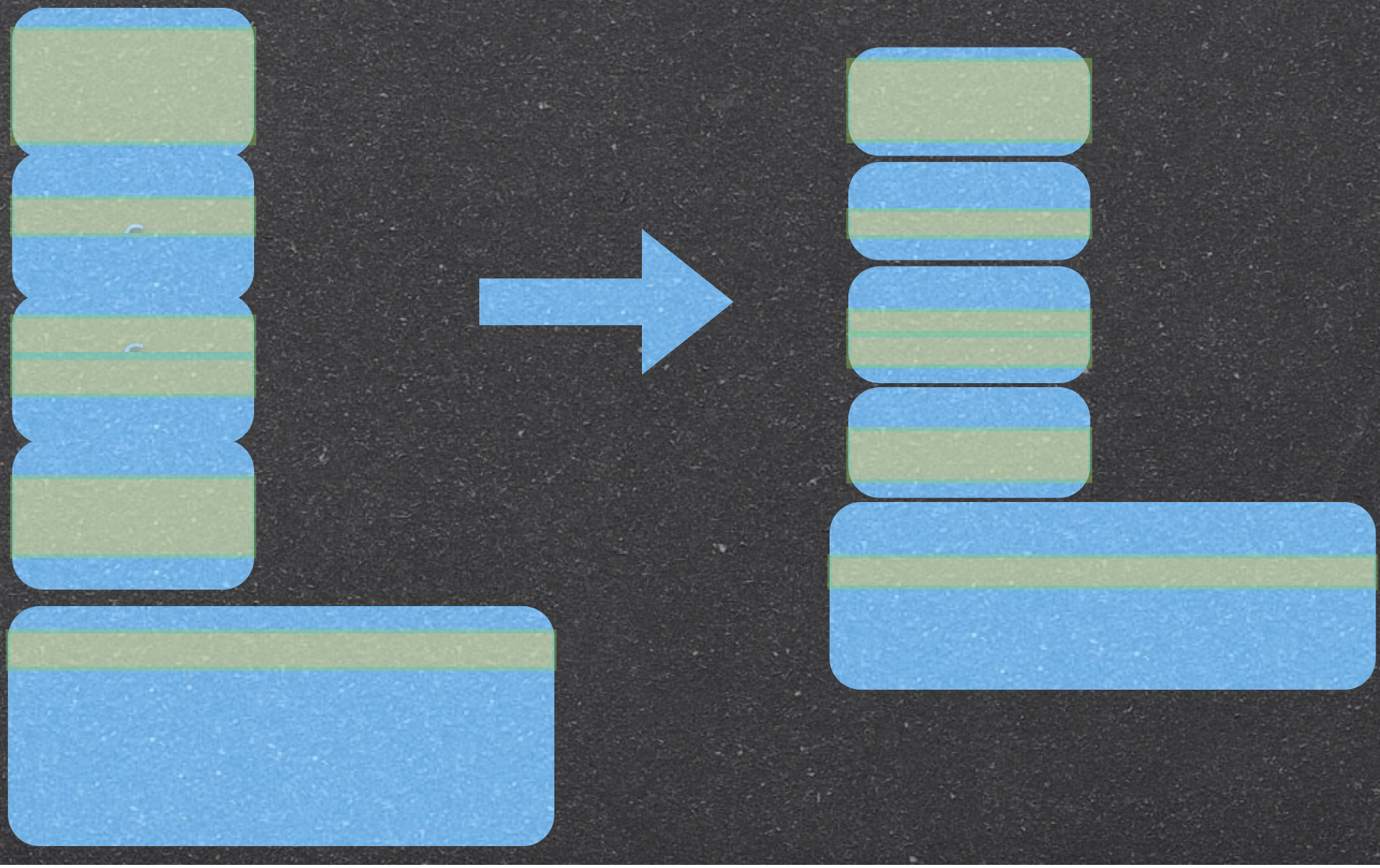
Total amount of code

Code actually used at runtime

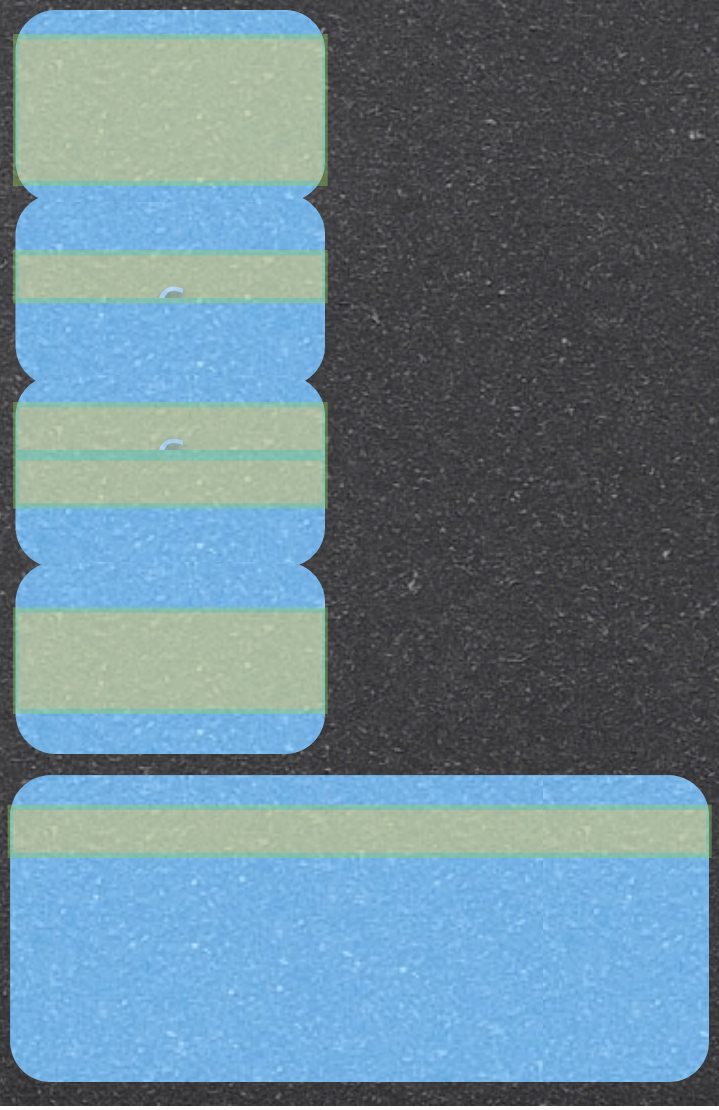
Local optimization



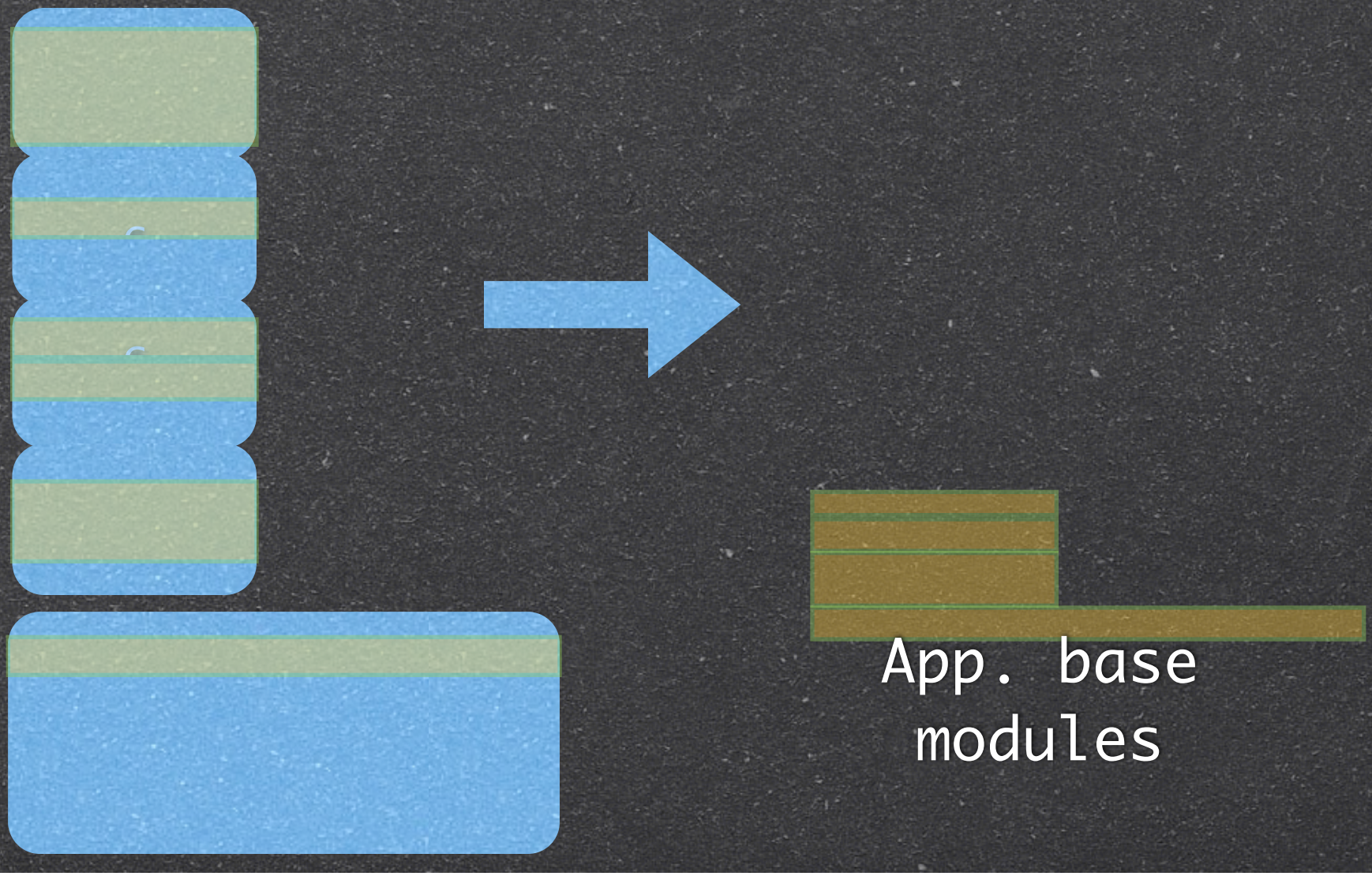
Local optimization



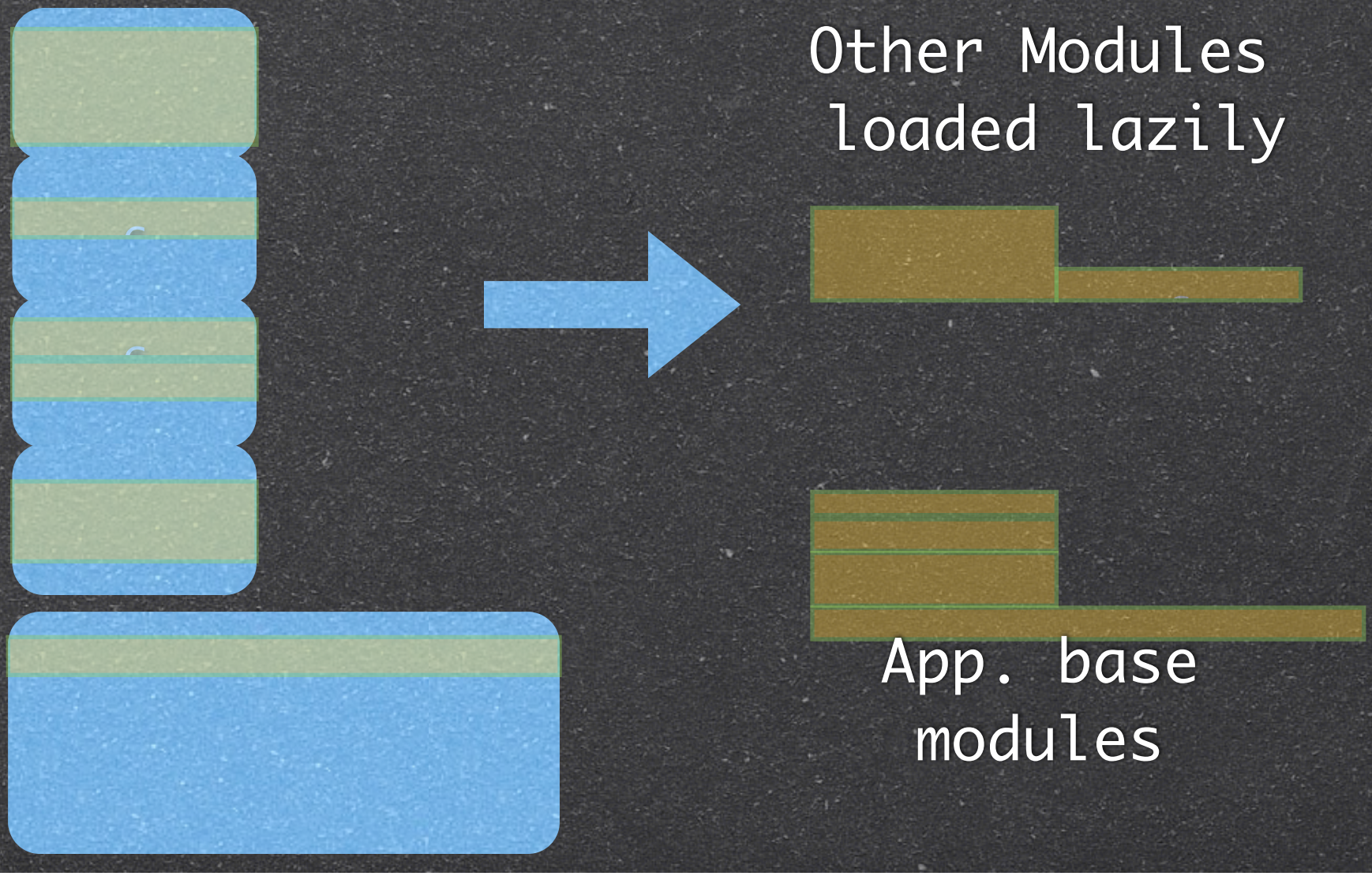
Closure Compiler



Closure Compiler



Closure Compiler



Development time

Total: 603,3 kB

app.js (two files) (6.3kB)

stilts-stomp.js (7kB)

jQuery-UI-min (367kB)

jQuery (233kB)

Total: ~ 1.3 MB

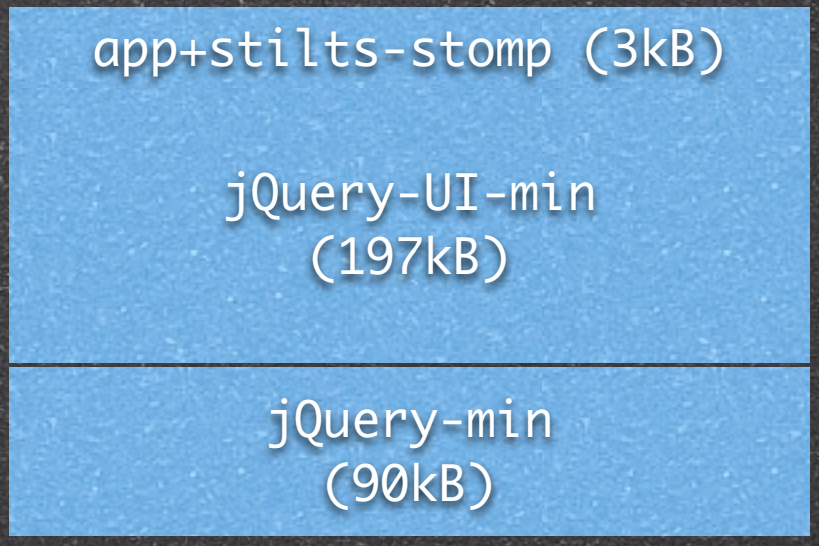
"App" (several files...) (35kB)

Stomple-0.99.js (39,6kB)

Closure Library (approx. 1.2MB)

Production

Total: 290 kB



Production

Total: 290 kB

Total: 71 kB

app+stilts-stomp (3kB)

jQuery-UI-min
(197kB)

jQuery-min
(90kB)

Closure compiled: 71kB

Production

Total: 290 kB

Total: 71 kB

app+stilts-stomp (3kB)

jQuery-UI-min
(197kB)

jQuery-min
(90kB)

or optionally

Webkit Closure: 64kB

Closure compiled: 71kB

Production

Total: 290 kB

Total: 71 kB

BEFORE GZIP

app+stilts-stomp (3kB)

jQuery-UI-min (197kB)

jQuery-min (90kB)

or optionally

Webkit Closure: 64kB

Closure compiled: 71kB

Closure Compiler

Closure Compiler

- The most aggressive optimizer

Closure Compiler

- The most aggressive optimizer
  - Whole-program analysis, inlining, etc

Closure Compiler

- The most aggressive optimizer
  - Whole-program analysis, inlining, etc
  - Used by Google: GMail, Docs,Google+, ...

Closure Compiler

- The most aggressive optimizer
  - Whole-program analysis, inlining, etc
  - Used by Google: GMail, Docs,Google+, ...
  - advanced support for splitting in modules

Closure Compiler

- The most aggressive optimizer
  - Whole-program analysis, inlining, etc
  - Used by Google: GMail, Docs,Google+, ...
  - advanced support for splitting in modules
  - Can significantly reduce startup time.

Closure Compiler

- The most aggressive optimizer
  - Whole-program analysis, inlining, etc
  - Used by Google: GMail, Docs,Google+, ...
  - advanced support for splitting in modules
  - Can significantly reduce startup time.
- Downsides to ADVANCED_OPTIMIZATIONS

Closure Compiler

- The most aggressive optimizer
  - Whole-program analysis, inlining, etc
  - Used by Google: GMail, Docs,Google+, ...
  - advanced support for splitting in modules
  - Can significantly reduce startup time.
- Downsides to ADVANCED_OPTIMIZATIONS
  - Can be quite hard to retrofit your JavaScript project to use ADVANCED_OPTIMIZATIONS

Closure Compiler

- The most aggressive optimizer
  - Whole-program analysis, inlining, etc
  - Used by Google: GMail, Docs,Google+, ...
  - advanced support for splitting in modules
  - Can significantly reduce startup time.
- Downsides to ADVANCED_OPTIMIZATIONS
  - Can be quite hard to retrofit your JavaScript project to use ADVANCED_OPTIMIZATIONS
  - I’ve had cases where the compiler performed unsafe transformations

Closure Compiler

- The most aggressive optimizer
  - Whole-program analysis, inlining, etc
  - Used by Google: GMail, Docs,Google+, ...
  - advanced support for splitting in modules
  - Can significantly reduce startup time.
- Downsides to ADVANCED_OPTIMIZATIONS
  - Can be quite hard to retrofit your JavaScript project to use ADVANCED_OPTIMIZATIONS
  - I’ve had cases where the compiler performed unsafe transformations
  - Quite cumbersome to setup

Third Challenge Quality Assurance (testing)

Component failure => Many intranet systems hard to access

Motivation: automated testing

- Higher-quality before app goes to production or manual QA.
  - Reduce need for manual testing.
  - Save time, effort and tedium of repetitive manual testing with each iteration.
- Faster feedback for developers.
- Prevent regression.
- Provide higher test coverage (e.g., OS versions, browser versions, languages).
- Extra form of documentation.

Types of tests

- Unit test
  - usually tests only one class/function
  - dependencies mocked out
- Integration
  - integration of several units
  - external environment (server, services)
- Acceptance tests
  - end-to-end test, based on user stories

Unit-testing

- There are several libraries for unit testing in JavaScript, but it is actually not so easy to find one that easily allows
  - Automated execution (i.e. from the command line)
  - Automated Reporting test outcome in a machine readable form (Continuous integration)
  - IDE integration
- Candidates:
  - JS-Testdriver http://code.google.com/p/js-test-driver/
  - buster.js http://docs.busterjs.org/en/latest/
  - Test'em https://github.com/airportyh/testem

Js-Testdriver

- Tests defined using TestCase
  - each test must start with ‘test’...
  - support for setUp and tearDown.
  - assertions: http://code.google.com/p/js-test-driver/wiki/Assertions

Js-Testdriver

- Tests defined using TestCase
  - each test must start with ‘test’...
  - support for setUp and tearDown.
  - assertions: http://code.google.com/p/js-test-driver/wiki/Assertions

```
TestCase("Test utils randomID", {
    testRandomIdShouldNotReturnNull:function () {
        assertNotNull(Todo.utils.uuid());
    },

    testRandomIdShouldHaveMoreThan8Chars: function () {
        assertTrue(Todo.utils.uuid().length > 8);
    }

});
```

Demo

Acceptance & Integration

- De-facto standard: Selenium
  - Automated
  - Easy to integrate with CI servers like hudson
- Selenium 2.0/WebDriver
  - supports multiple browsers (each has a specific Driver implementing technical details)
  - fine grain control
- API bindings for many languages: Java, Clojure, C#, Ruby,... http://seleniumhq.org/docs/03_webdriver.html

Selenium Java

Selenium Java

```
WebDriver driver = new FirefoxDriver();
driver.get("http://www.google.com");

// Find the text input element by its name
WebElement element = driver.findElement(By.name("q"));

// Enter something to search for
element.sendKeys("Cheese!");
element.submit();
// Check the title of the page
System.out.println("Page title is: " + driver.getTitle());
(new WebDriverWait(driver, 10)).until(new ExpectedCondition<Boolean>() {
    public Boolean apply(WebDriver d) {
        return d.getTitle().toLowerCase().startsWith("cheese!");
    }
});
System.out.println("Page title is: " + driver.getTitle());

driver.quit();
```

SauceLabs

- Test in various browsers on various operating systems and languages.
- In parallel
- Remotely
- Visual results and automated test results.
- DEMO.

Questions?