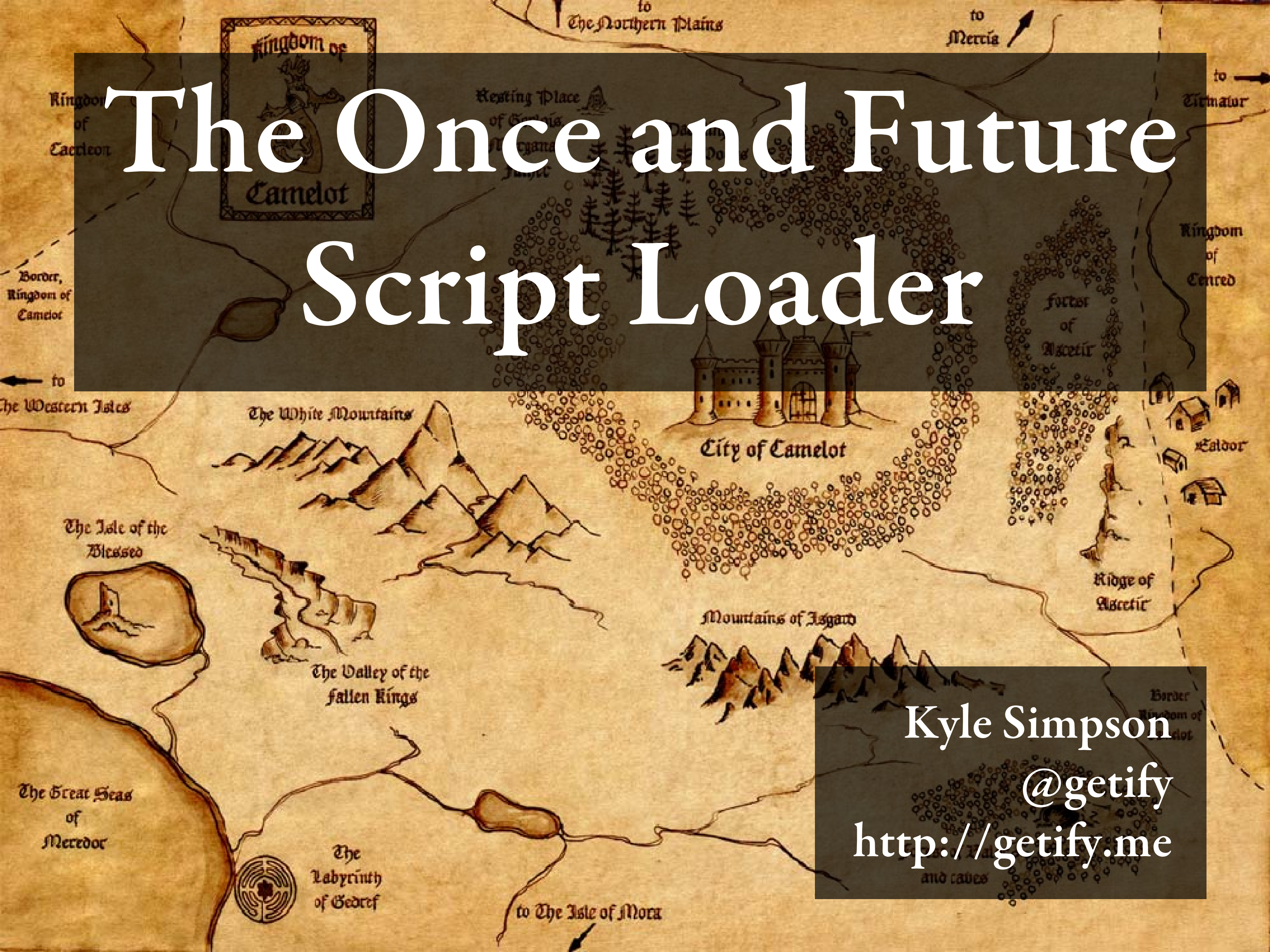
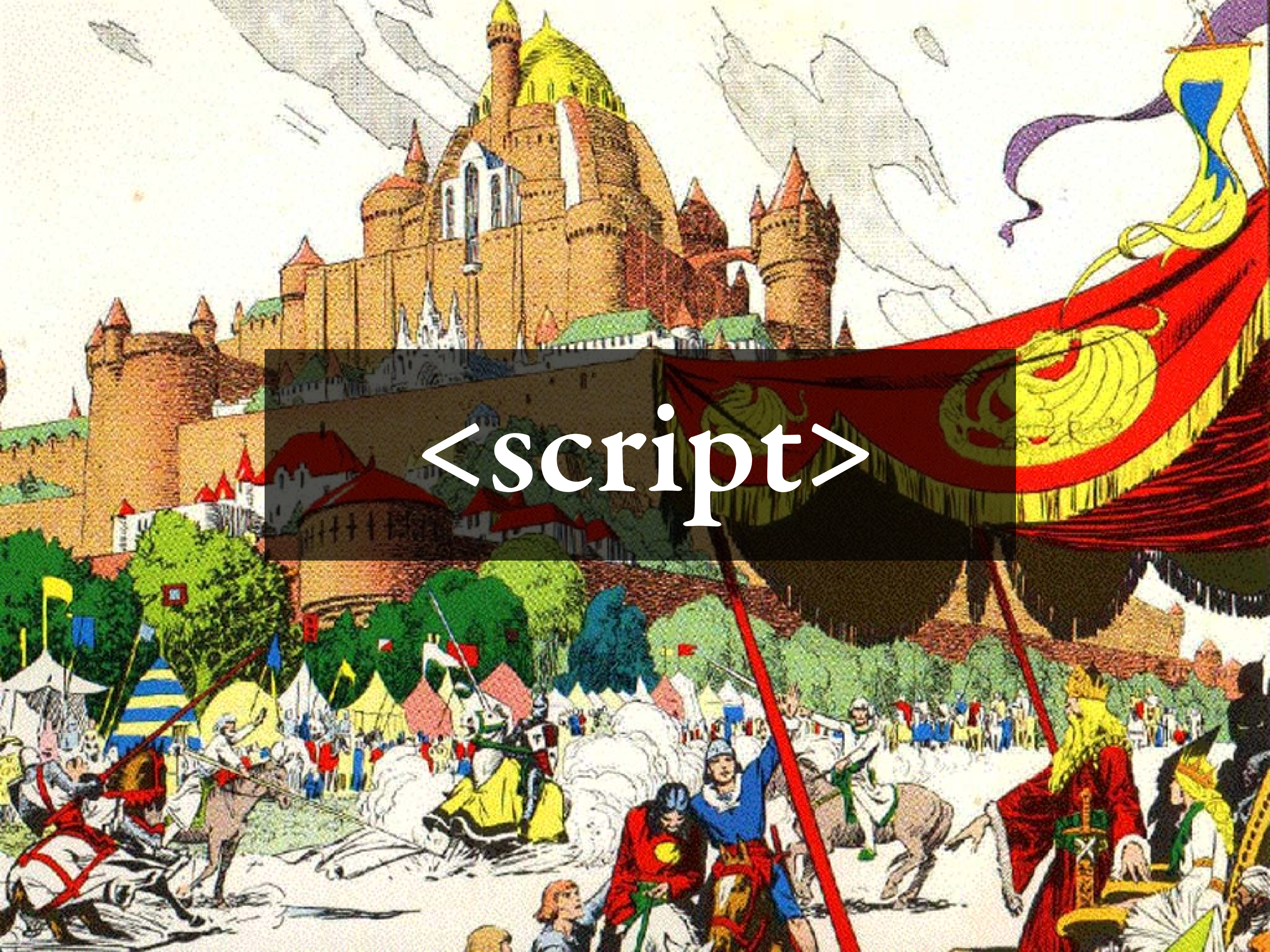
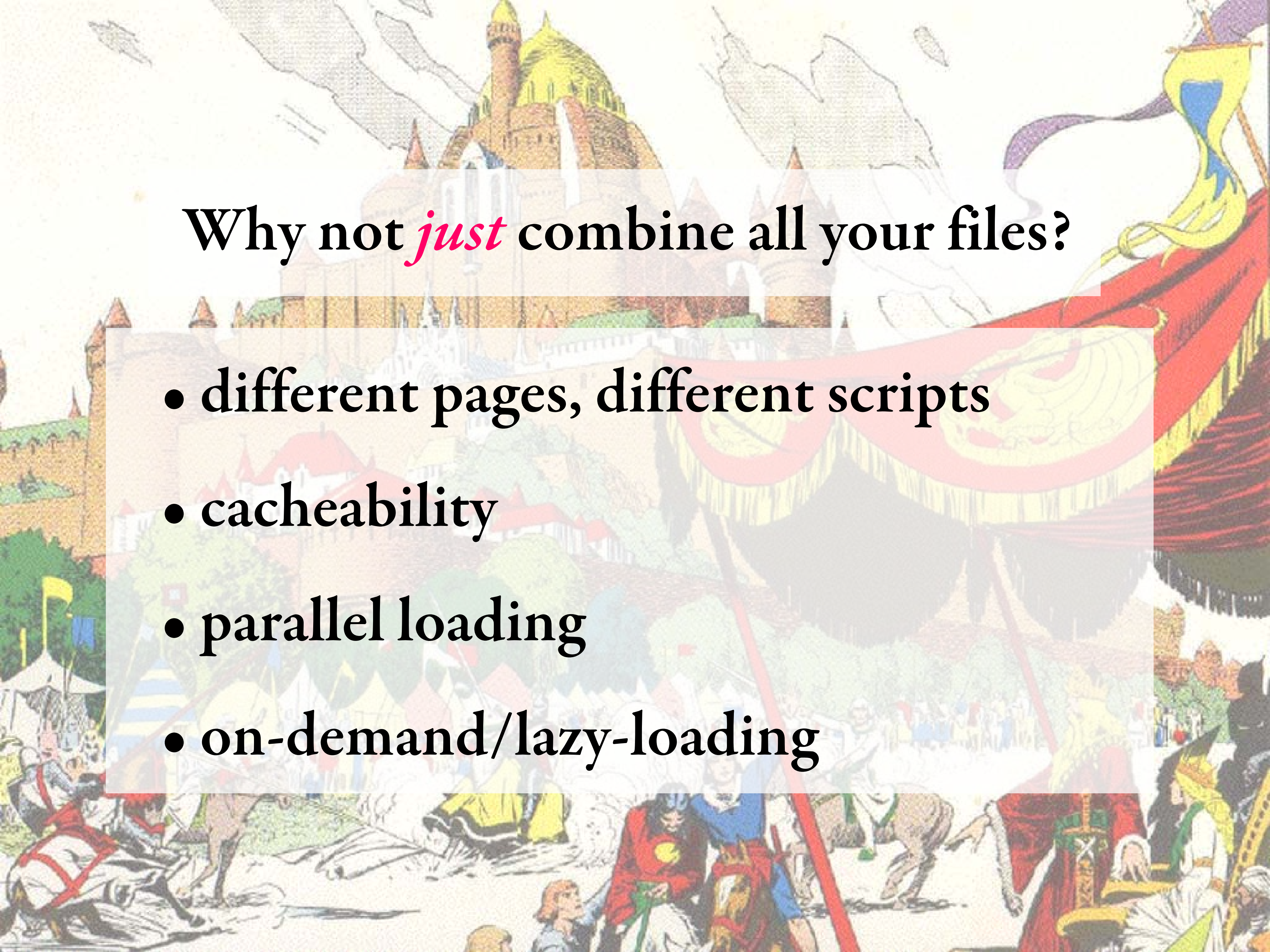
The Once and Future Script Loader

Kyle Simpson
@getify
http://getify.me



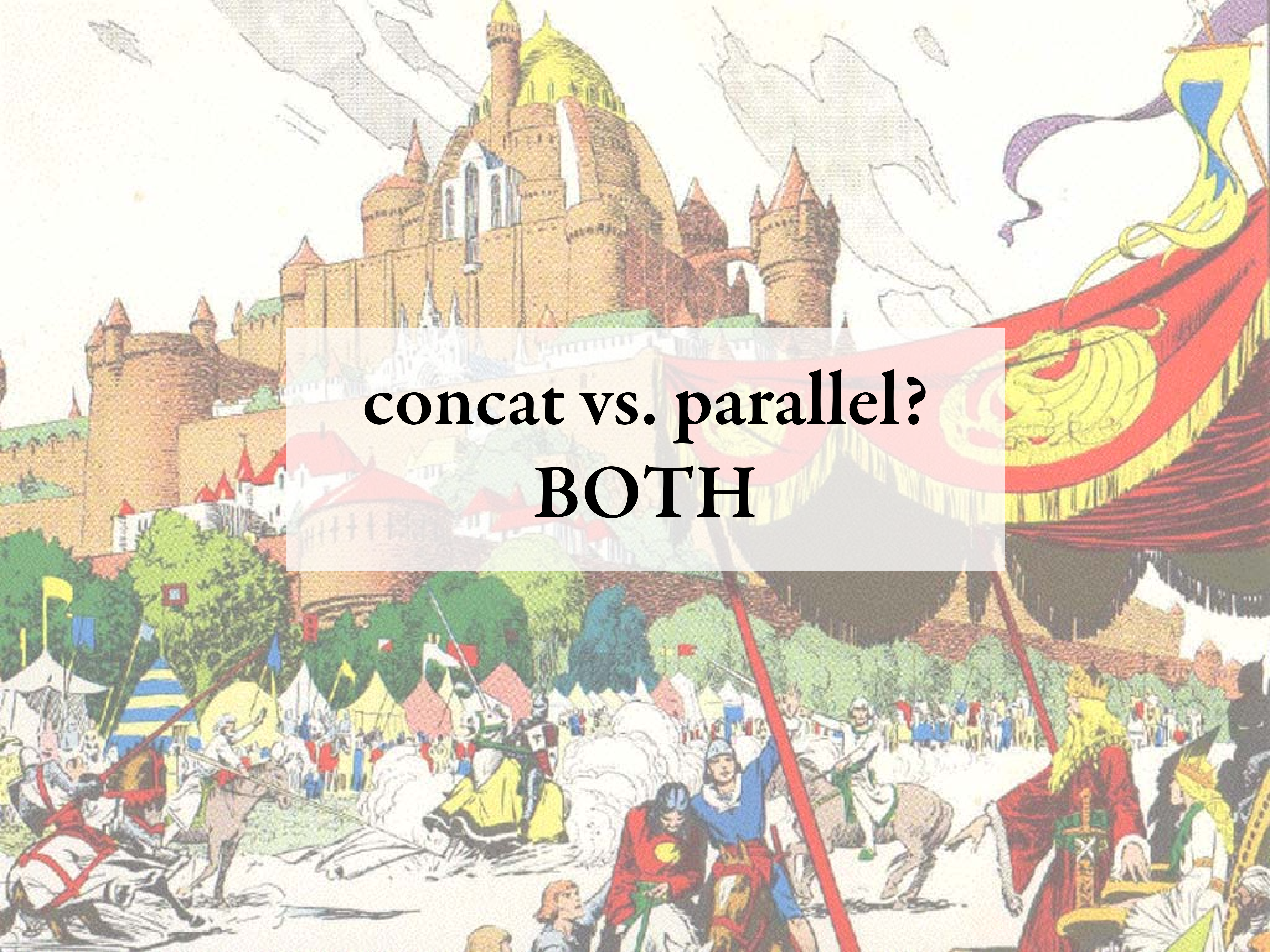


<script>



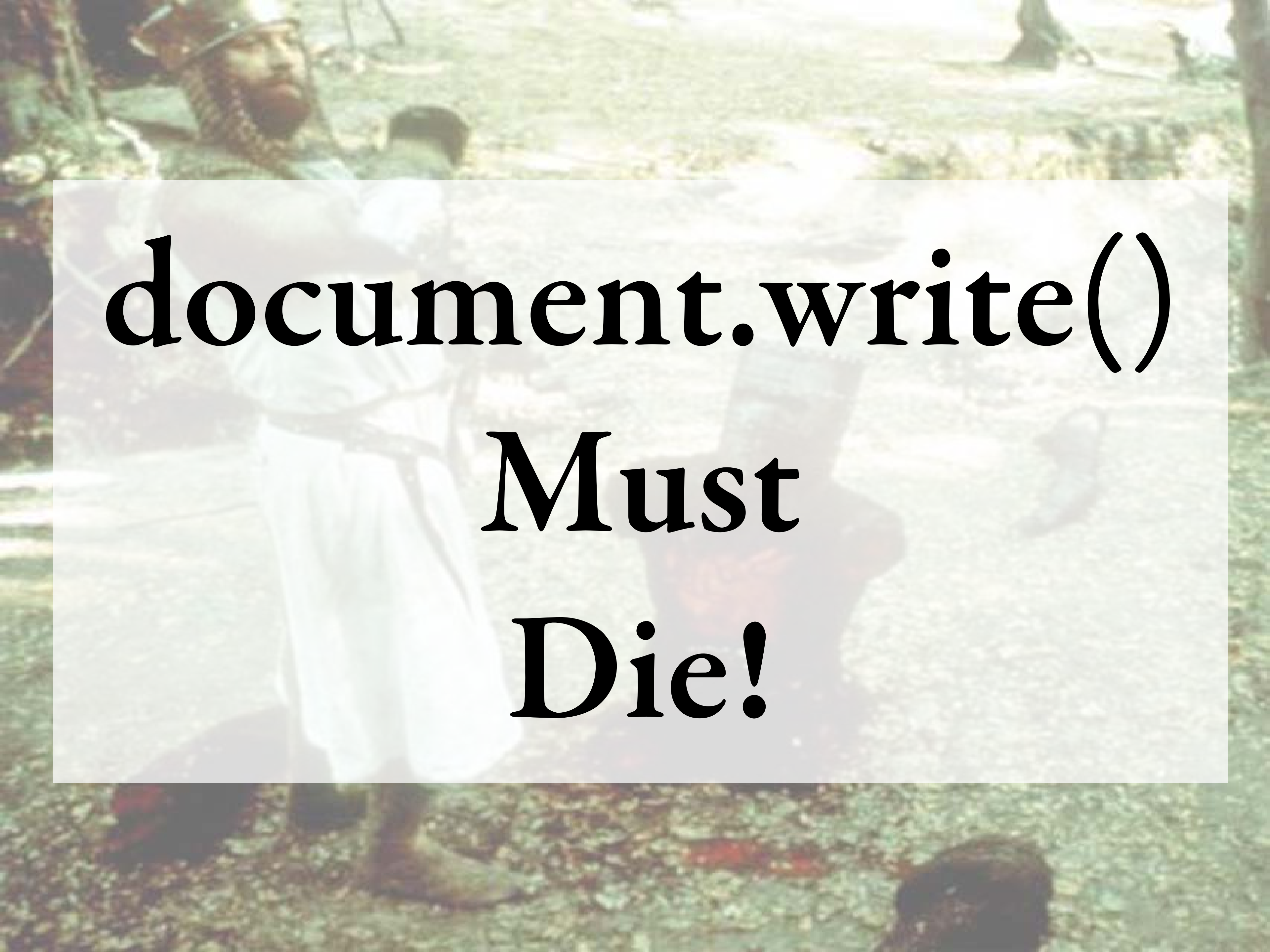
Why not *just* combine all your files?

- different pages, different scripts
- cacheability
- parallel loading
- on-demand/lazy-loading



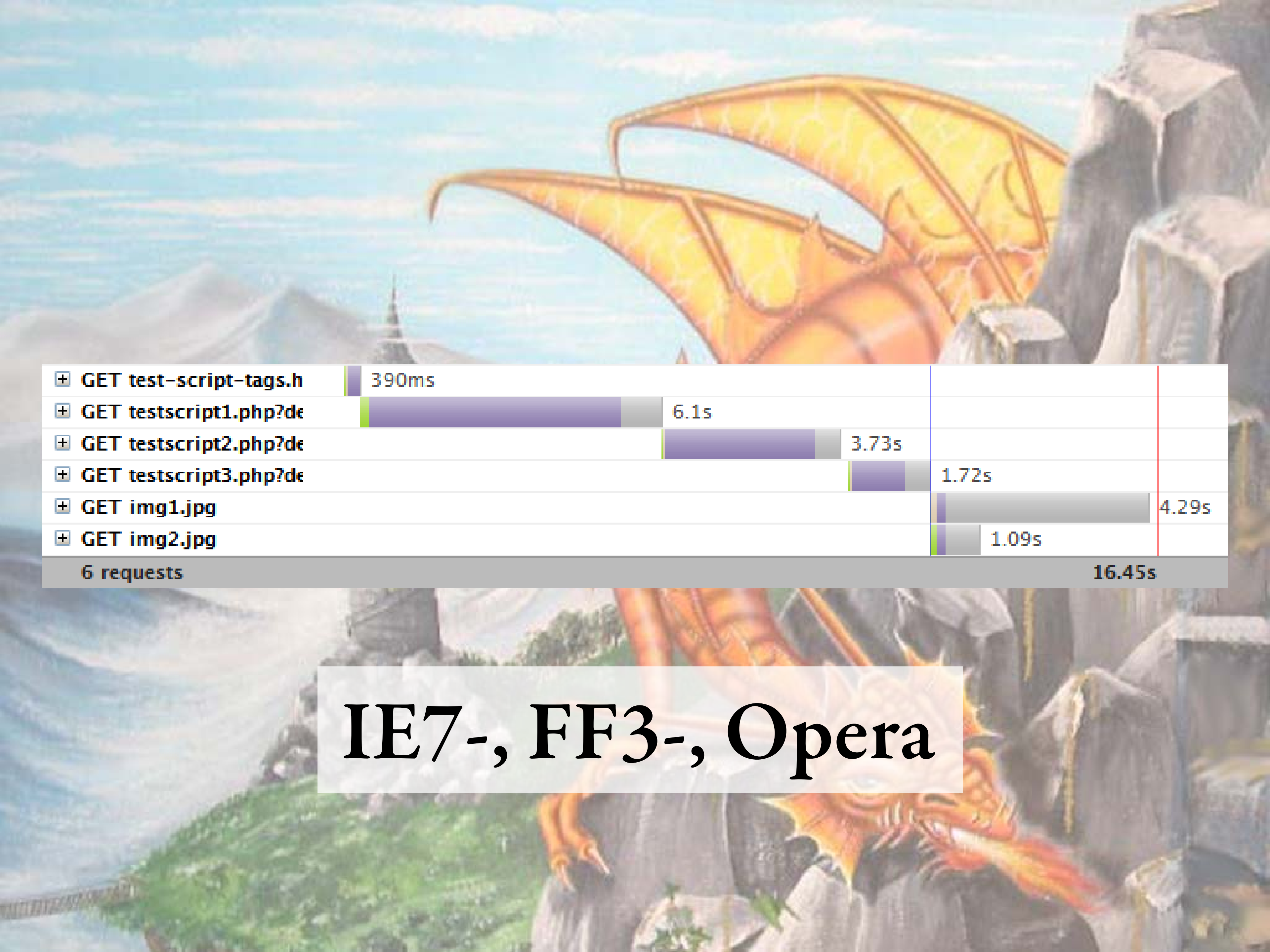
concat vs. parallel?
BOTH

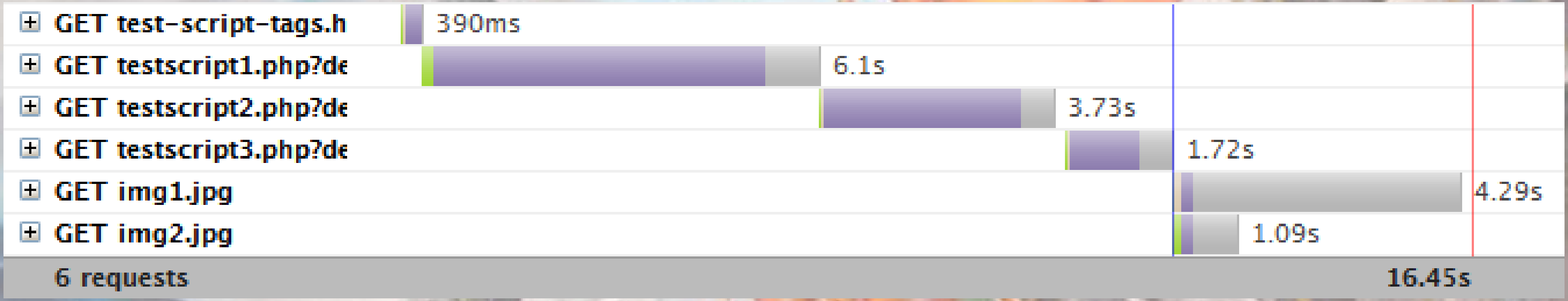
document.write()



document.write() Must Die!

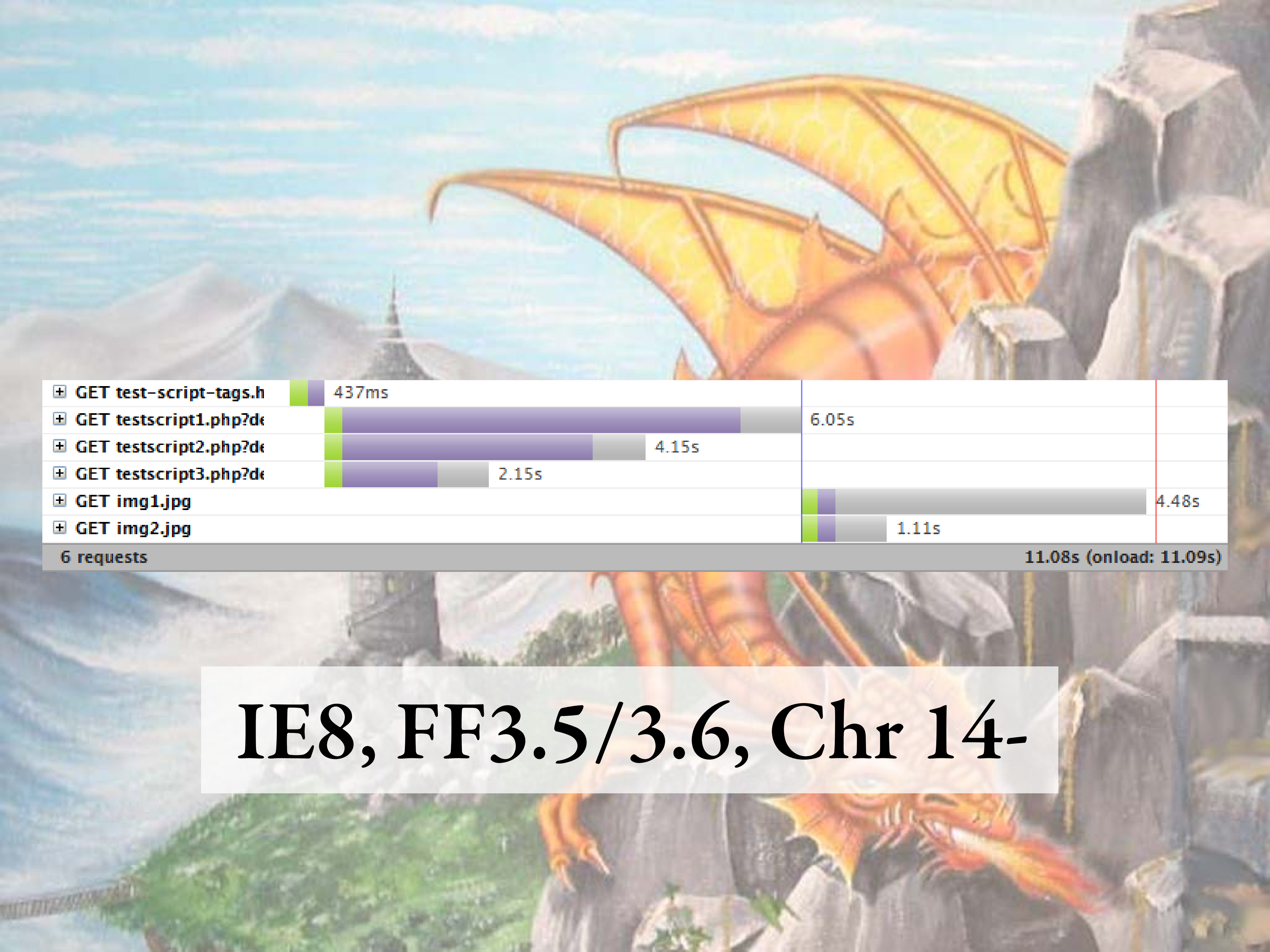
Performance

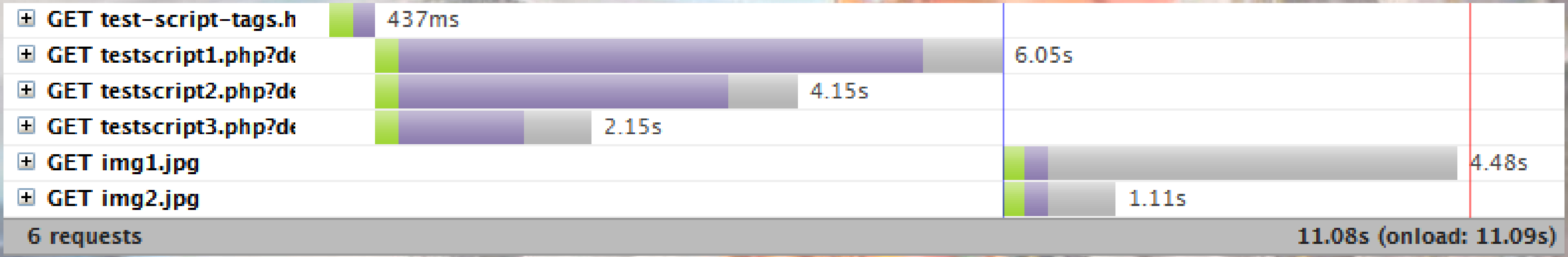




```
+ GET test-script-tags.h      390ms
+ GET testscript1.php?de      6.1s
+ GET testscript2.php?de      3.73s
+ GET testscript3.php?de      1.72s
+ GET img1.jpg                4.29s
+ GET img2.jpg                1.09s
  6 requests                  16.45s
```

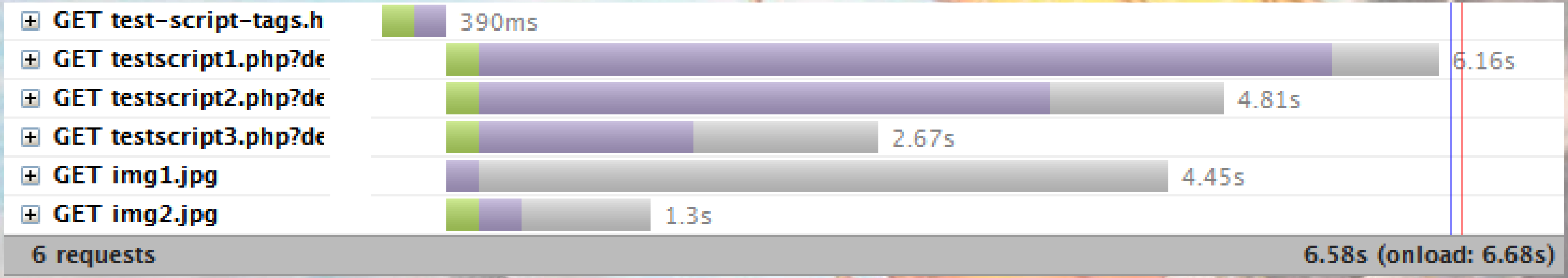
IE7-, FF3-, Opera





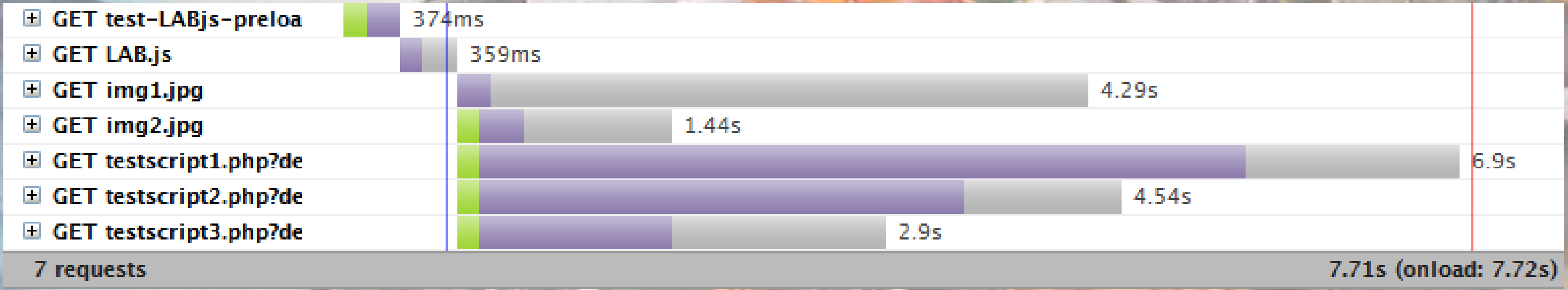
```
⊞ GET test-script-tags.h      437ms
⊞ GET testscript1.php?de                      6.05s
⊞ GET testscript2.php?de               4.15s
⊞ GET testscript3.php?de         2.15s
⊞ GET img1.jpg                                              4.48s
⊞ GET img2.jpg                                    1.11s
6 requests                                   11.08s (onload: 11.09s)
```

IE8, FF3.5/3.6, Chr 14-



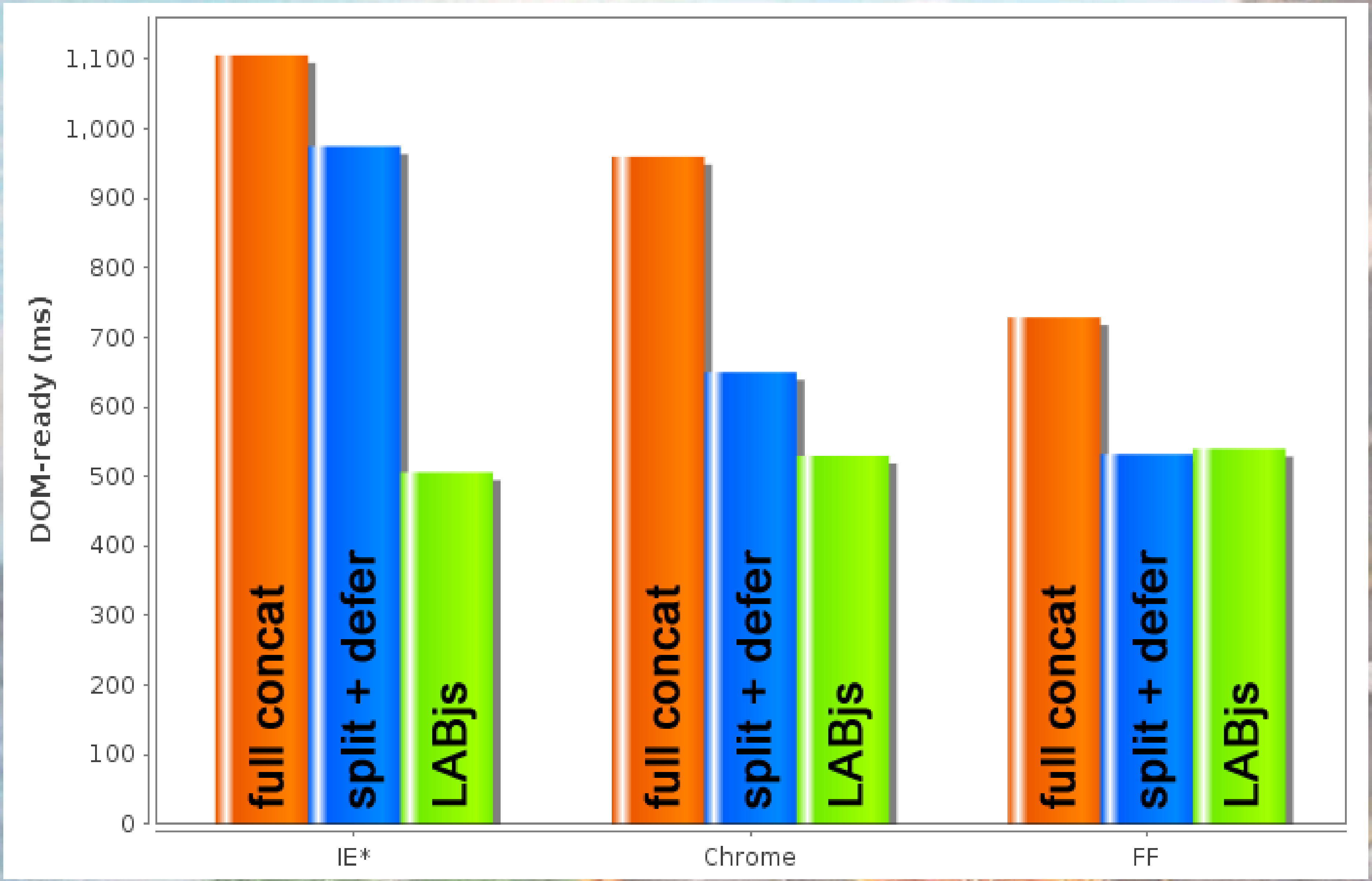
```
⊞ GET test-script-tags.h      390ms
⊞ GET testscript1.php?de      6.16s
⊞ GET testscript2.php?de      4.81s
⊞ GET testscript3.php?de      2.67s
⊞ GET img1.jpg                4.45s
⊞ GET img2.jpg                1.3s
6 requests                    6.58s (onload: 6.68s)
```

IE9+, FF4+



```
⊞ GET test-LABjs-preloa       374ms
⊞ GET LAB.js                  359ms
⊞ GET img1.jpg                4.29s
⊞ GET img2.jpg                1.44s
⊞ GET testscript1.php?de      6.9s
⊞ GET testscript2.php?de      4.54s
⊞ GET testscript3.php?de      2.9s
7 requests                    7.71s (onload: 7.72s)
```

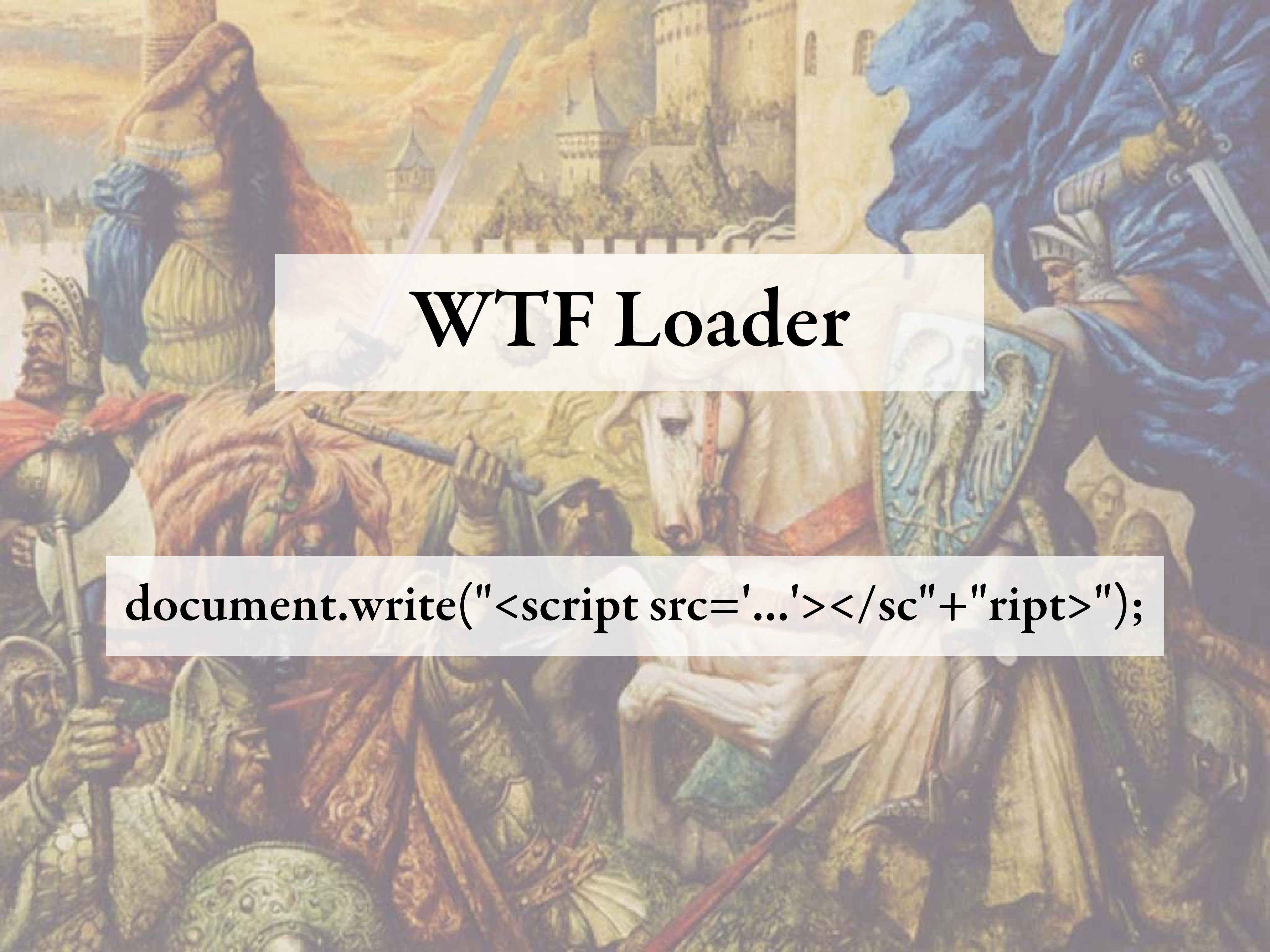
Script Loaders



comparing DOM-ready times across loading techniques

<script> tags also suck because...

- browser-specific scripts
- conditional loading/URLs
- event handling

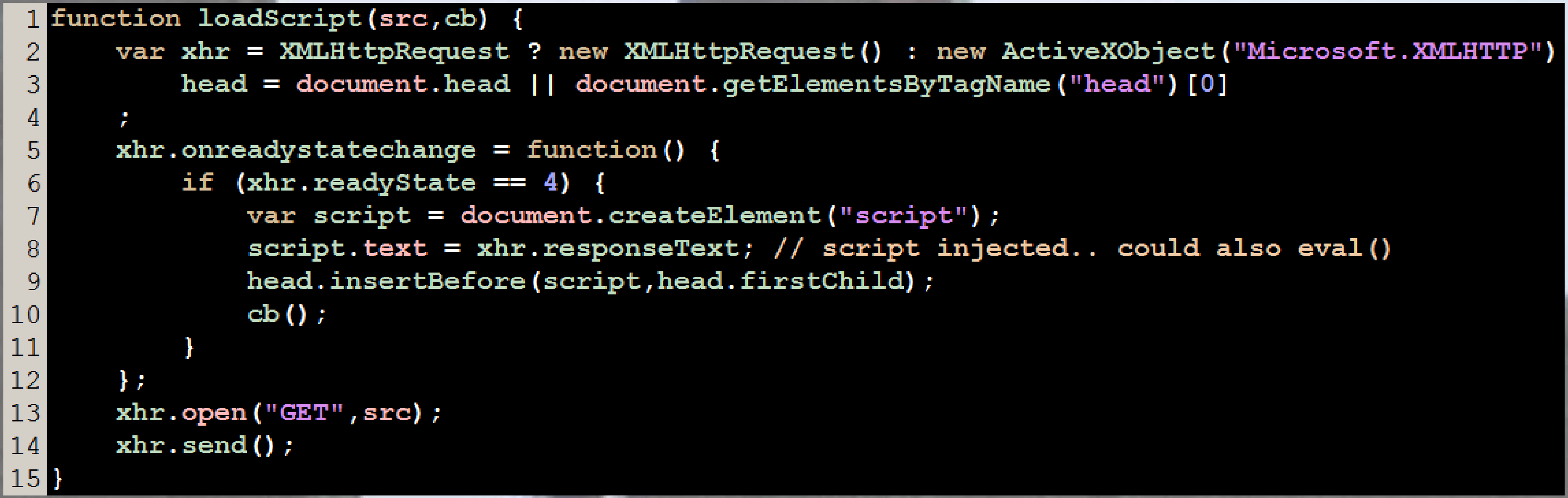


WTF Loader

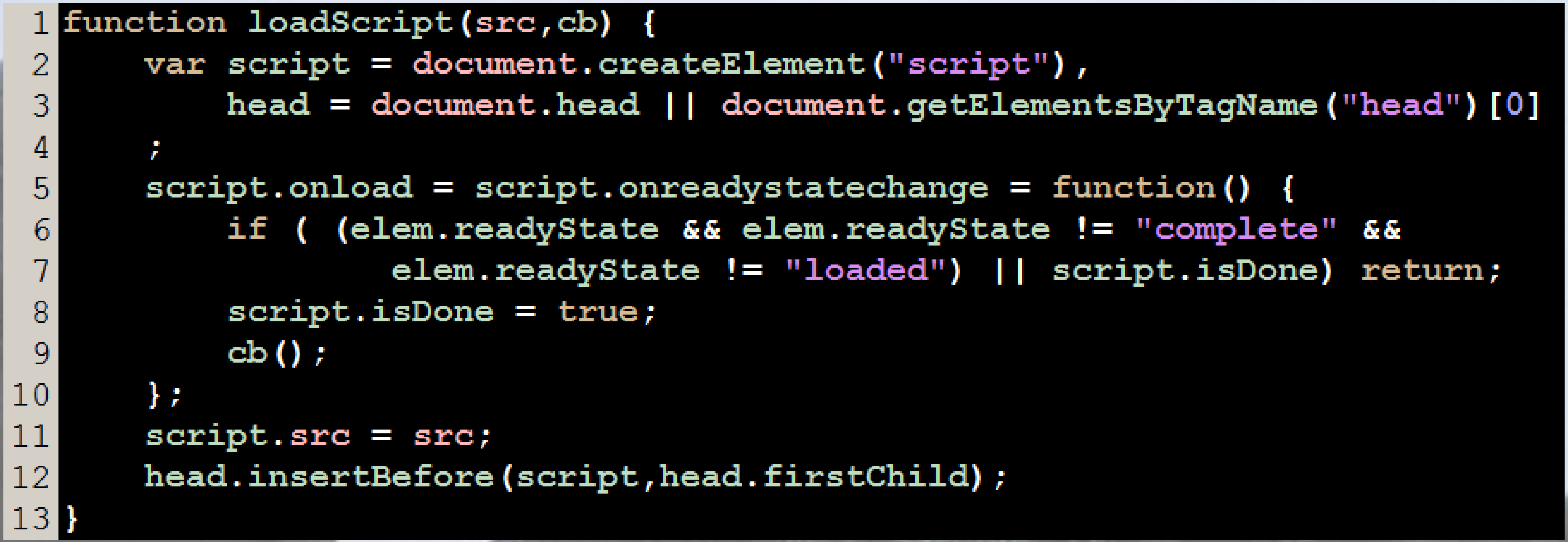
```
document.write("<script src='...'></sc"+"ript>");
```

Surely we can do better!?

XHR?

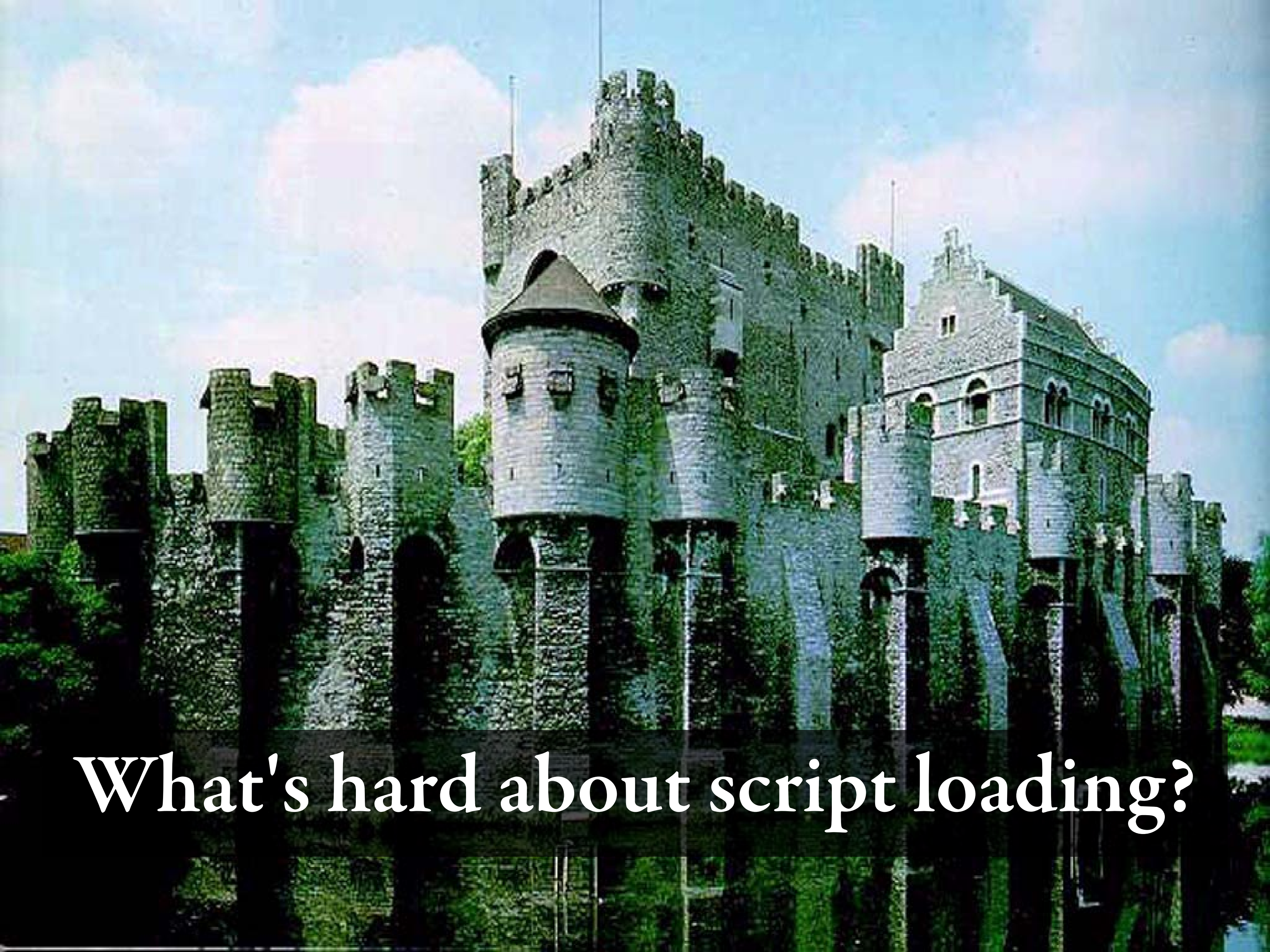


```
function loadScript(src,cb) {
    var xhr = XMLHttpRequest ? new XMLHttpRequest() : new ActiveXObject("Microsoft.XMLHTTP")
        head = document.head || document.getElementsByTagName("head")[0]
    ;
    xhr.onreadystatechange = function() {
        if (xhr.readyState == 4) {
            var script = document.createElement("script");
            script.text = xhr.responseText; // script injected.. could also eval()
            head.insertBefore(script,head.firstChild);
            cb();
        }
    };
    xhr.open("GET",src);
    xhr.send();
}
```



```
function loadScript(src,cb) {
    var script = document.createElement("script"),
        head = document.head || document.getElementsByTagName("head")[0]
    ;
    script.onload = script.onreadystatechange = function() {
        if ( (elem.readyState && elem.readyState != "complete" &&
                elem.readyState != "loaded") || script.isDone) return;
        script.isDone = true;
        cb();
    };
    script.src = src;
    head.insertBefore(script,head.firstChild);
}
```

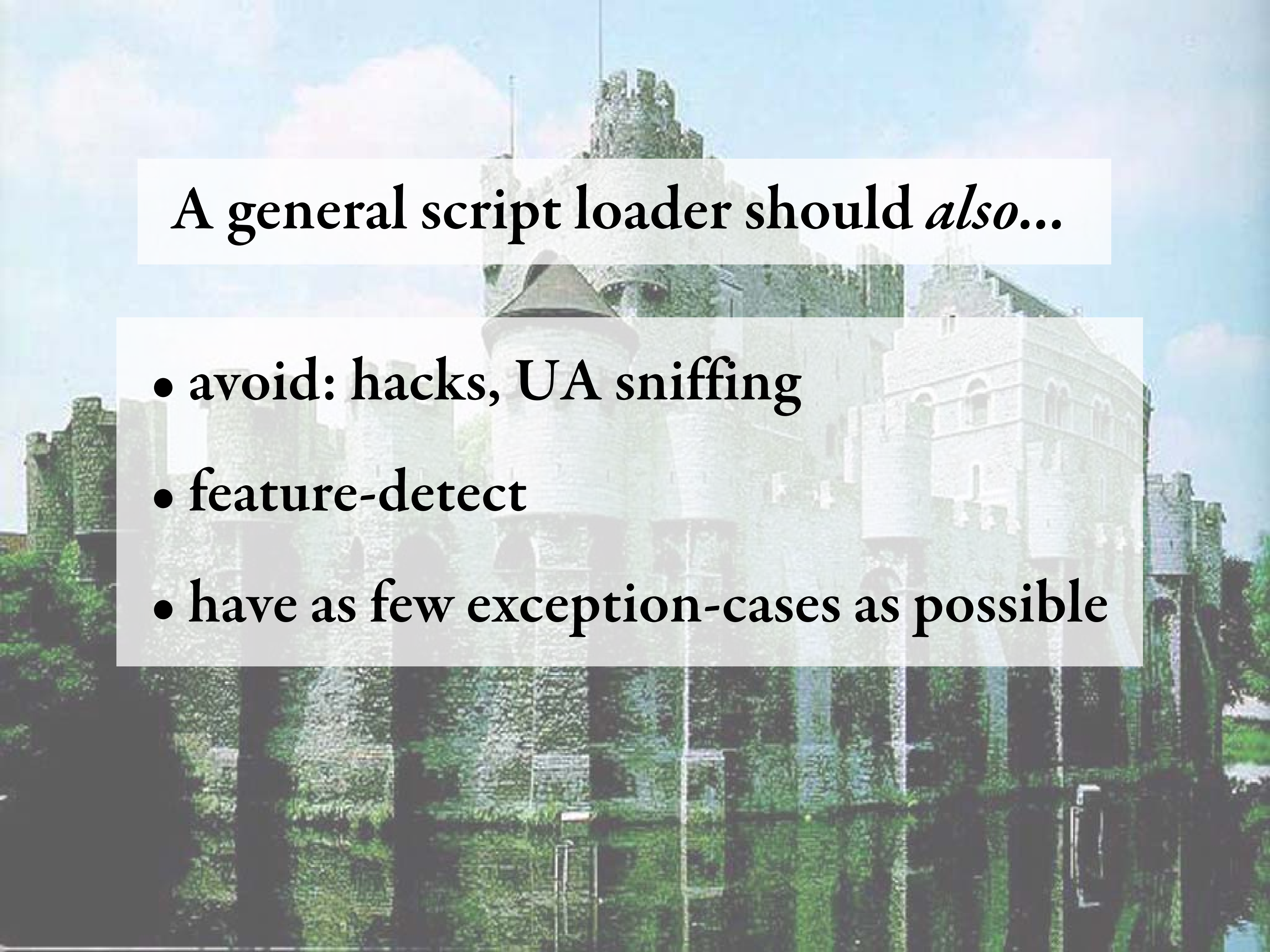
Making progress… but not *there* yet



What's hard about script loading?

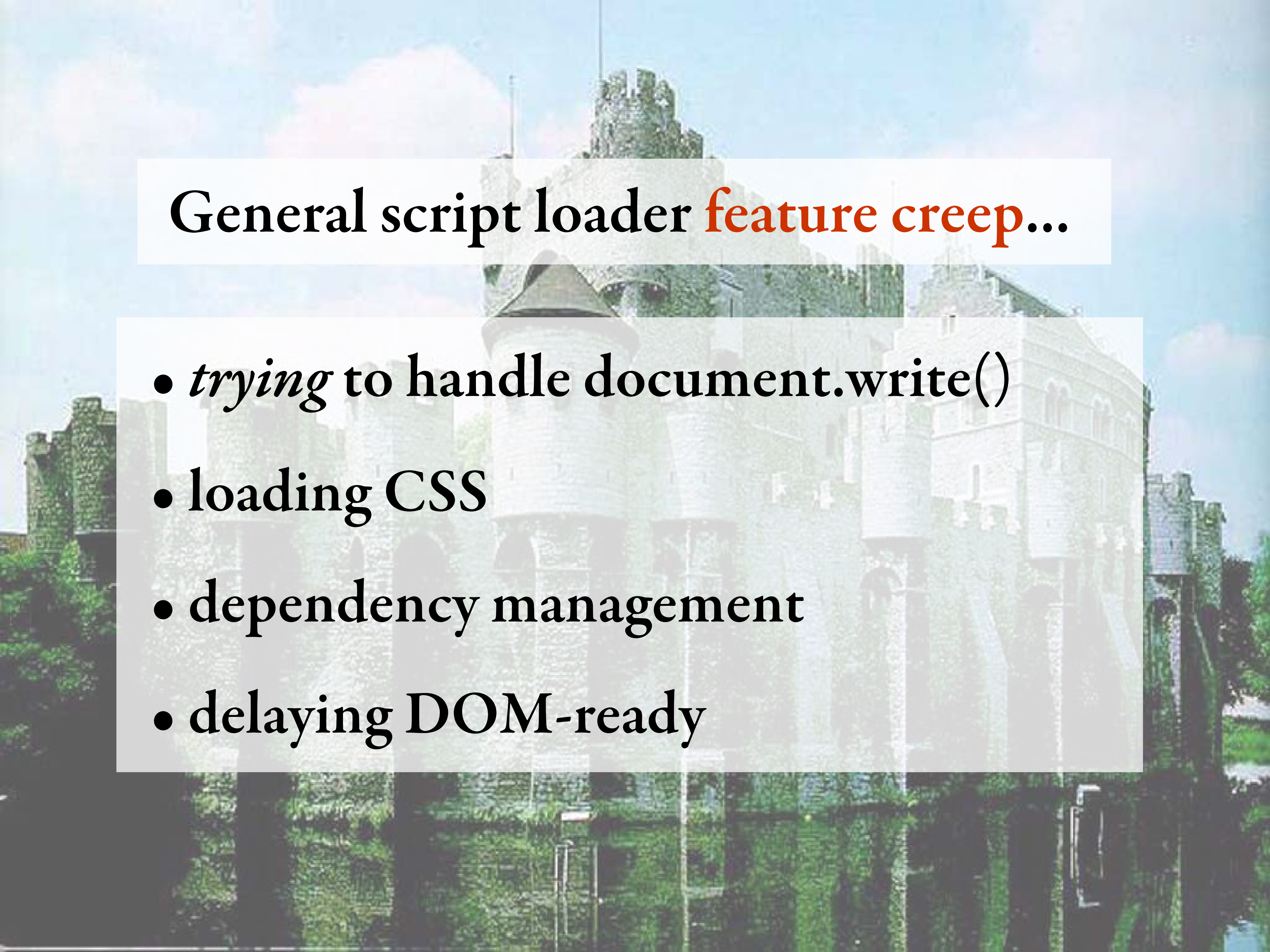
A general script loader *should*...

- load any script, at any time, from anywhere... except document.write()
- have performance at or better than <script> in modern browsers
- equivalent behavior in *all* browsers



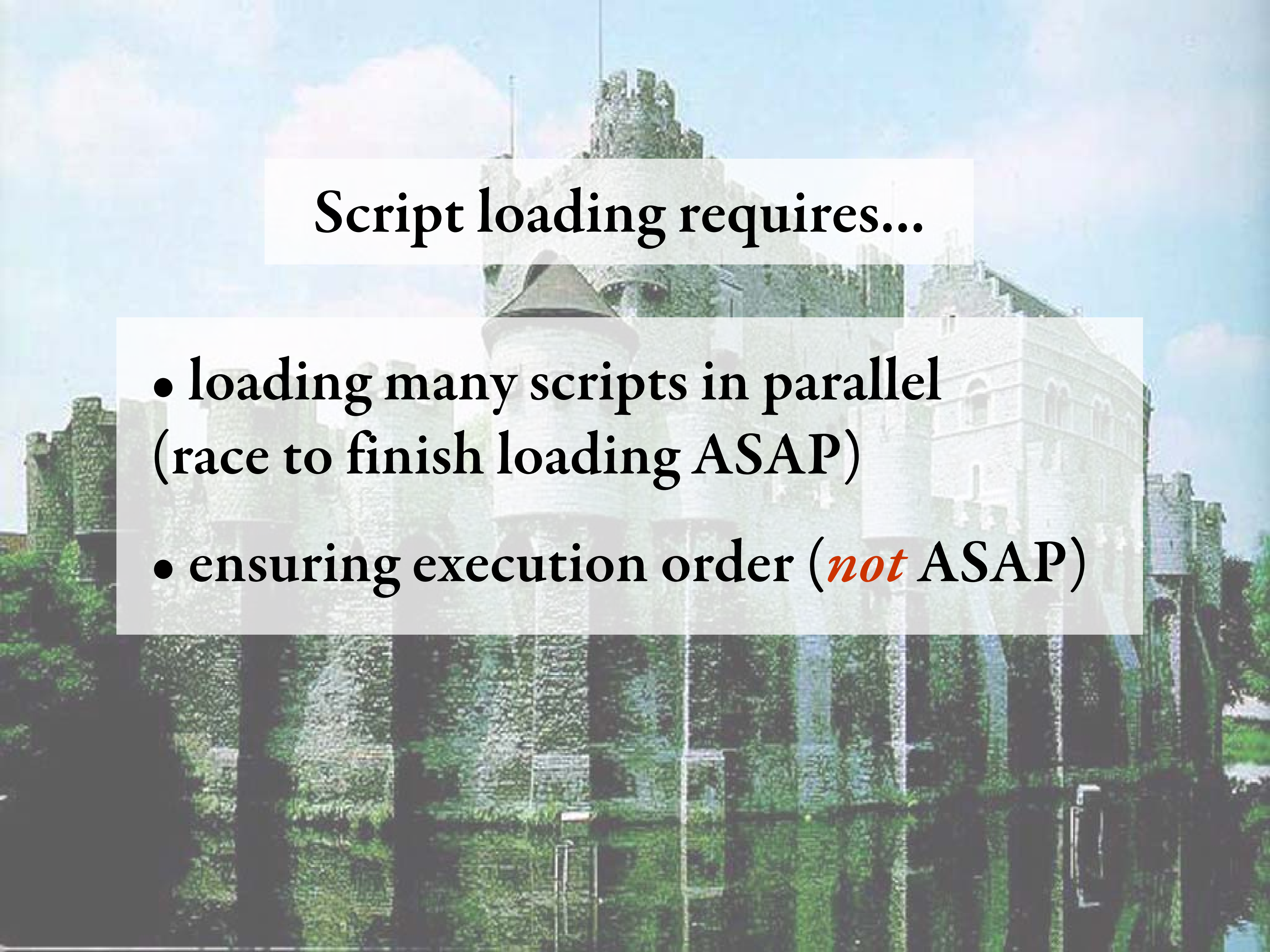
A general script loader should *also*...

- avoid: hacks, UA sniffing
- feature-detect
- have as few exception-cases as possible



General script loader feature creep...

- *trying* to handle document.write()
- loading CSS
- dependency management
- delaying DOM-ready



Script loading requires...

- loading many scripts in parallel (race to finish loading ASAP)
- ensuring execution order (*not* ASAP)

LABjs

LABjs: *performance* script loader

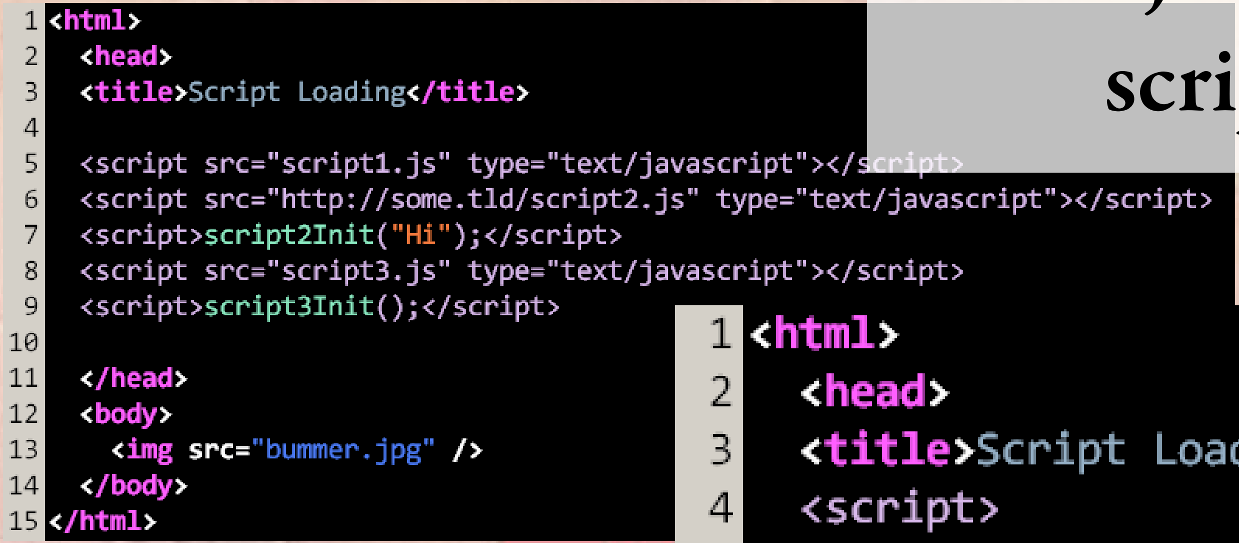
```
<html>
  <head>
  <title>Script Loading</title>

  <script src="script1.js" type="text/javascript"></script>
  <script src="http://some.tld/script2.js" type="text/javascript"></script>
  <script src="script3.js" type="text/javascript"></script>

  </head>
  <body>
    <img src="bummer.jpg" />
  </body>
</html>
```

```
<html>
  <head>
  <title>Script Loading</title>
  <script>
    $LAB
    .script("script1.js")
    .script("http://some.tld/script2.js")
    .script("script3.js");
  </script>
  </head>
  <body>
    <img src="awesome.jpg" />
  </body>
</html>
```

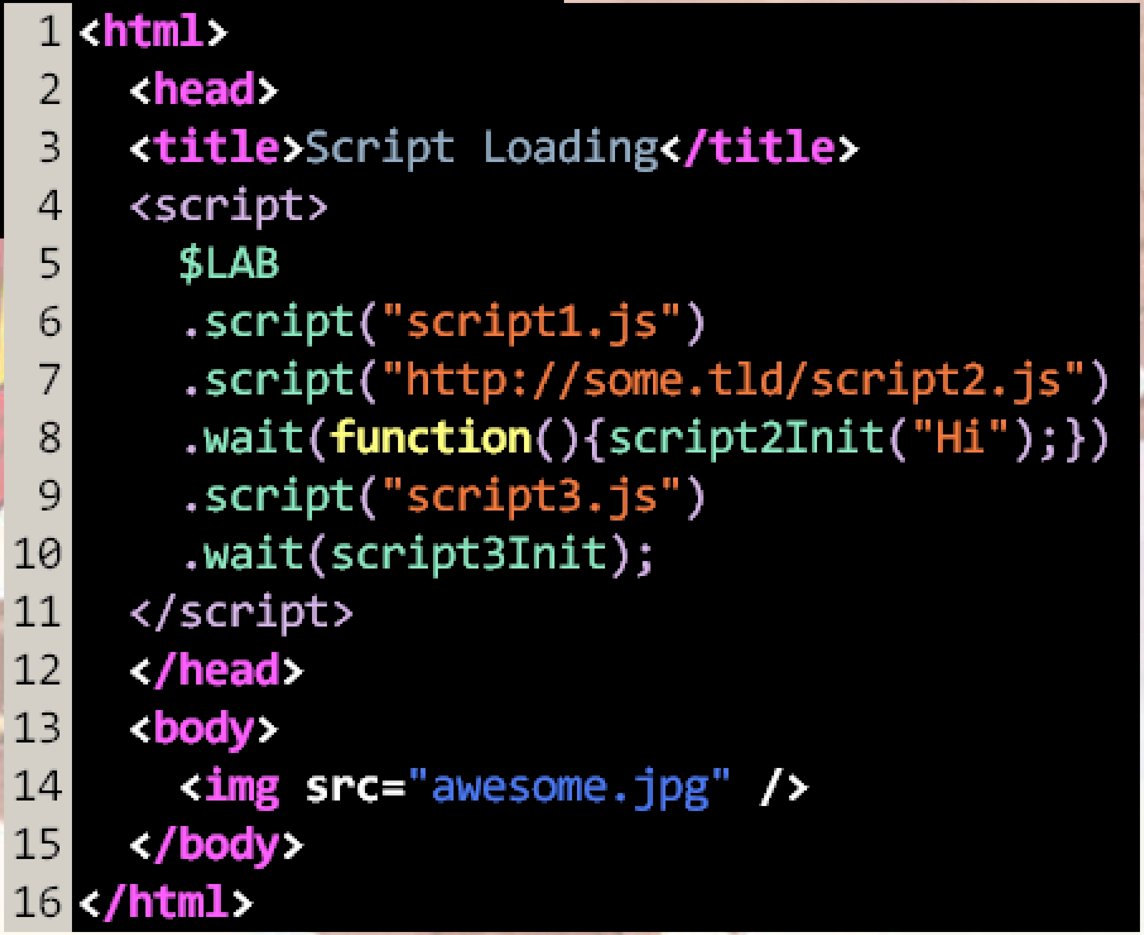
LABjs: *performance* script loader



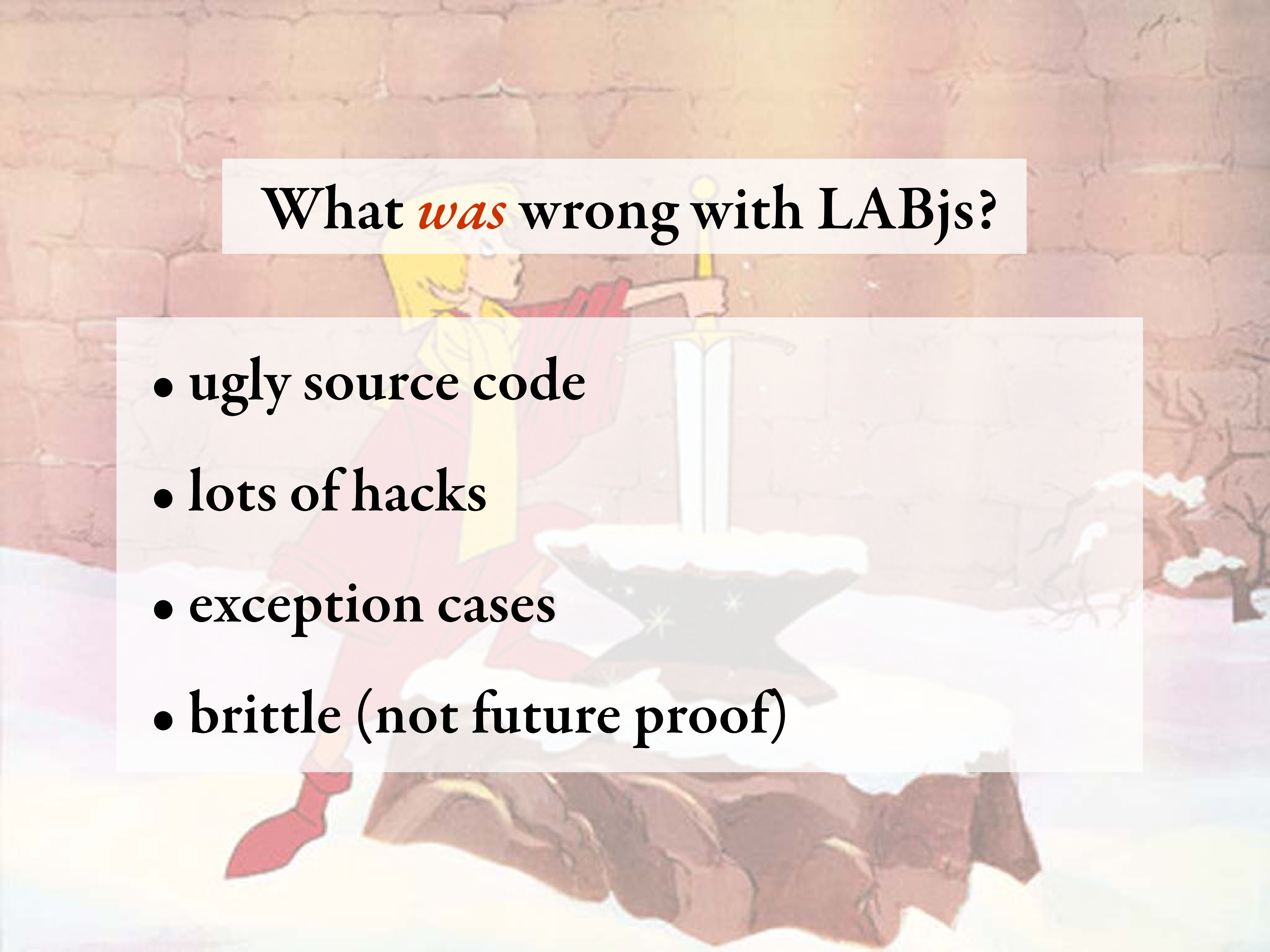
```
<html>
  <head>
  <title>Script Loading</title>

  <script src="script1.js" type="text/javascript"></script>
  <script src="http://some.tld/script2.js" type="text/javascript"></script>
  <script>script2Init("Hi");</script>
  <script src="script3.js" type="text/javascript"></script>
  <script>script3Init();</script>

  </head>
  <body>
    <img src="bummer.jpg" />
  </body>
</html>
```



```
<html>
  <head>
  <title>Script Loading</title>
  <script>
    $LAB
    .script("script1.js")
    .script("http://some.tld/script2.js")
    .wait(function(){script2Init("Hi");})
    .script("script3.js")
    .wait(script3Init);
  </script>
  </head>
  <body>
    <img src="awesome.jpg" />
  </body>
</html>
```



What *was* wrong with LABjs?

- ugly source code
- lots of hacks
- exception cases
- brittle (not future proof)

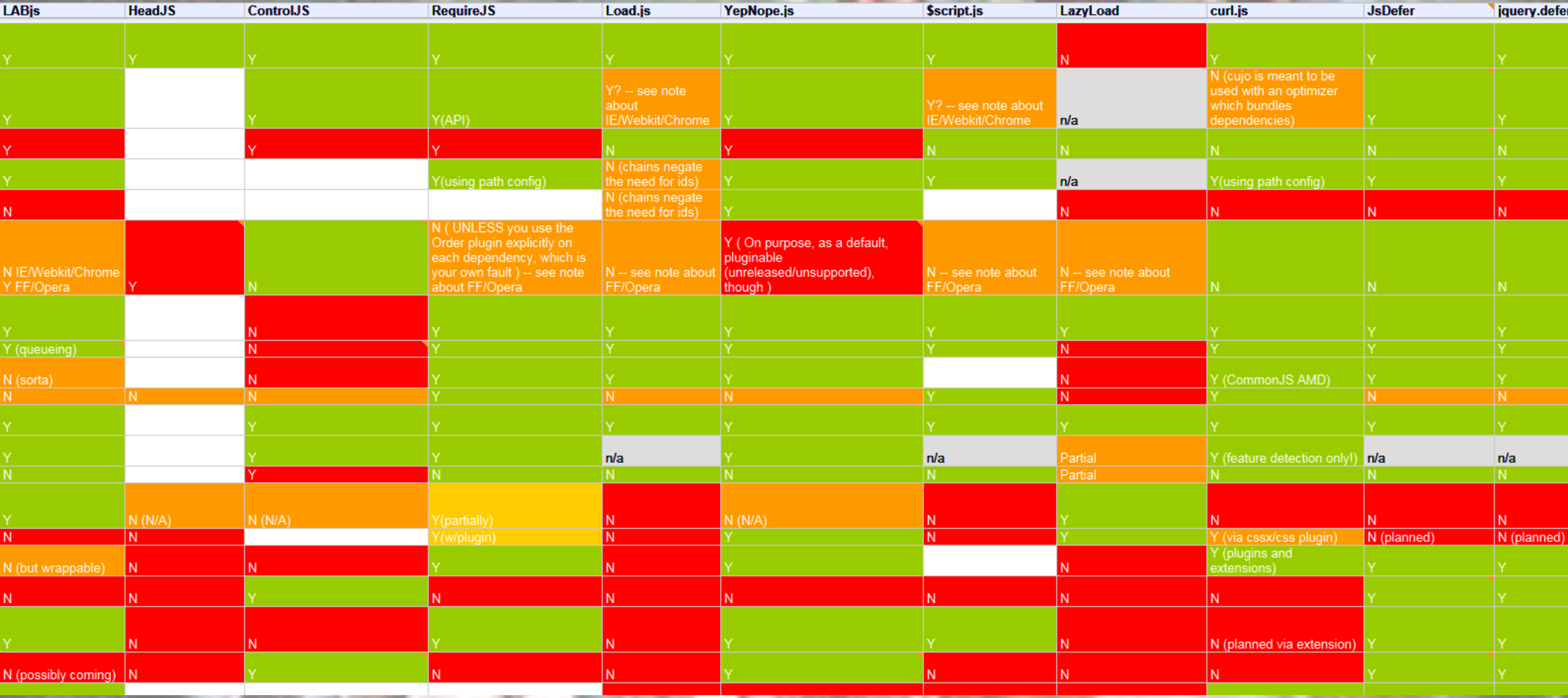
LABjs 2.0

- *readable* source code
- "future proof" feature detects
- *fewer* exception cases
- better performance

http://labjs.com

Anyone can write a loader!

(and *many* devs have)



LABjs	HeadJS	ControlJS	RequireJS	Load.js	YepNope.js	$script.js	LazyLoad	curl.js	JsDefer	jquery.defe
Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y
Y		Y	Y(API)	Y? -- see note about IE/Webkit/Chrome	Y	Y? -- see note about IE/Webkit/Chrome	n/a	N (cujo is meant to be used with an optimizer which bundles dependencies)	Y	Y
Y		Y	Y	N	Y	N	N	N	N	N
Y			Y(using path config)	N (chains negate the need for ids)	Y	Y	n/a	Y(using path config)	Y	Y
N				N (chains negate the need for ids)	Y		N	N	N	N
N IE/Webkit/Chrome Y FF/Opera	Y	N	N (UNLESS you use the Order plugin explicitly on each dependency, which is your own fault) -- see note about FF/Opera	N -- see note about FF/Opera	Y (On purpose, as a default, pluginable (unreleased/unsupported), though)	N -- see note about FF/Opera	N -- see note about FF/Opera	N	N	N
Y		N	Y	Y	Y	Y	Y	Y	Y	Y
Y (queueing)		N	Y	Y	Y	Y	N	Y	Y	Y
N (sorta)		N	Y	Y	Y		N	Y (CommonJS AMD)	Y	Y
N	N	N	Y	N	N	Y	N	Y	N	N
Y		Y	Y	Y	Y	Y	Y	Y	Y	Y
Y		Y	Y	n/a	Y	n/a	Partial	Y (feature detection only!)	n/a	n/a
N		Y	N	N	N	N	Partial	N	N	N
Y	N (N/A)	N (N/A)	Y(partially)	N	N (N/A)	N	Y	N	N	N
N	N		Y(w/plugin)	N	Y	N	Y	Y (via cssx/css plugin)	N (planned)	N (planned)
N (but wrappable)	N	N	Y	N	Y		N	Y (plugins and extensions)	Y	Y
N	N	Y	N	N	N	N	N	N	Y	Y
Y	N	N	Y	N	Y	Y	N	N (planned via extension)	Y	Y
N (possibly coming)	N	Y	N	N	Y	N	N	N	Y	Y

https://spreadsheets0.google.com/spreadsheet/ccc?key=tDdcrv9wNQRCNCRCflWxhYQ

Loaders are competing more on APIs than on features or performance

They're also copying each other (good and bad)

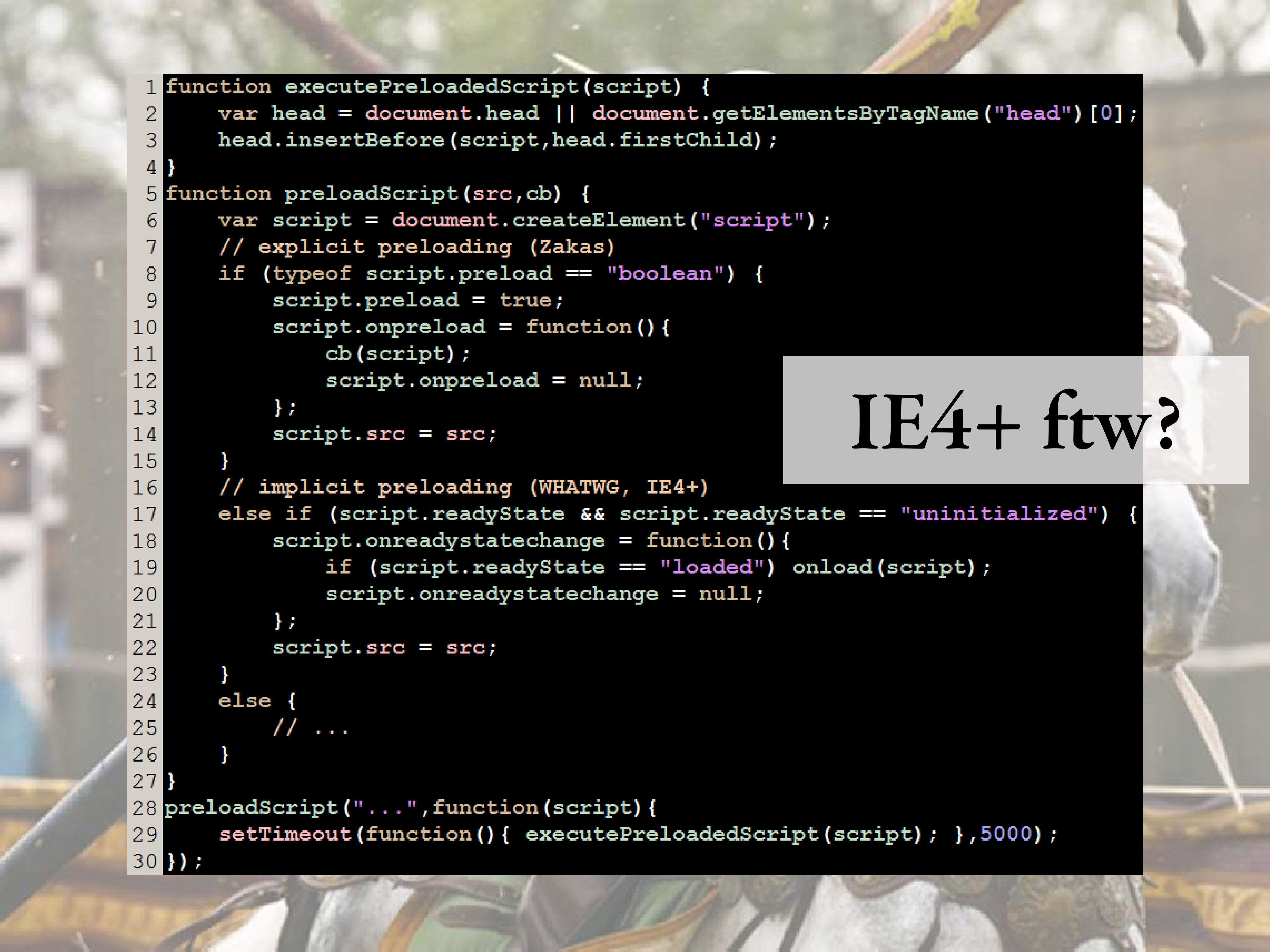
But, can their *functionality* be trusted?

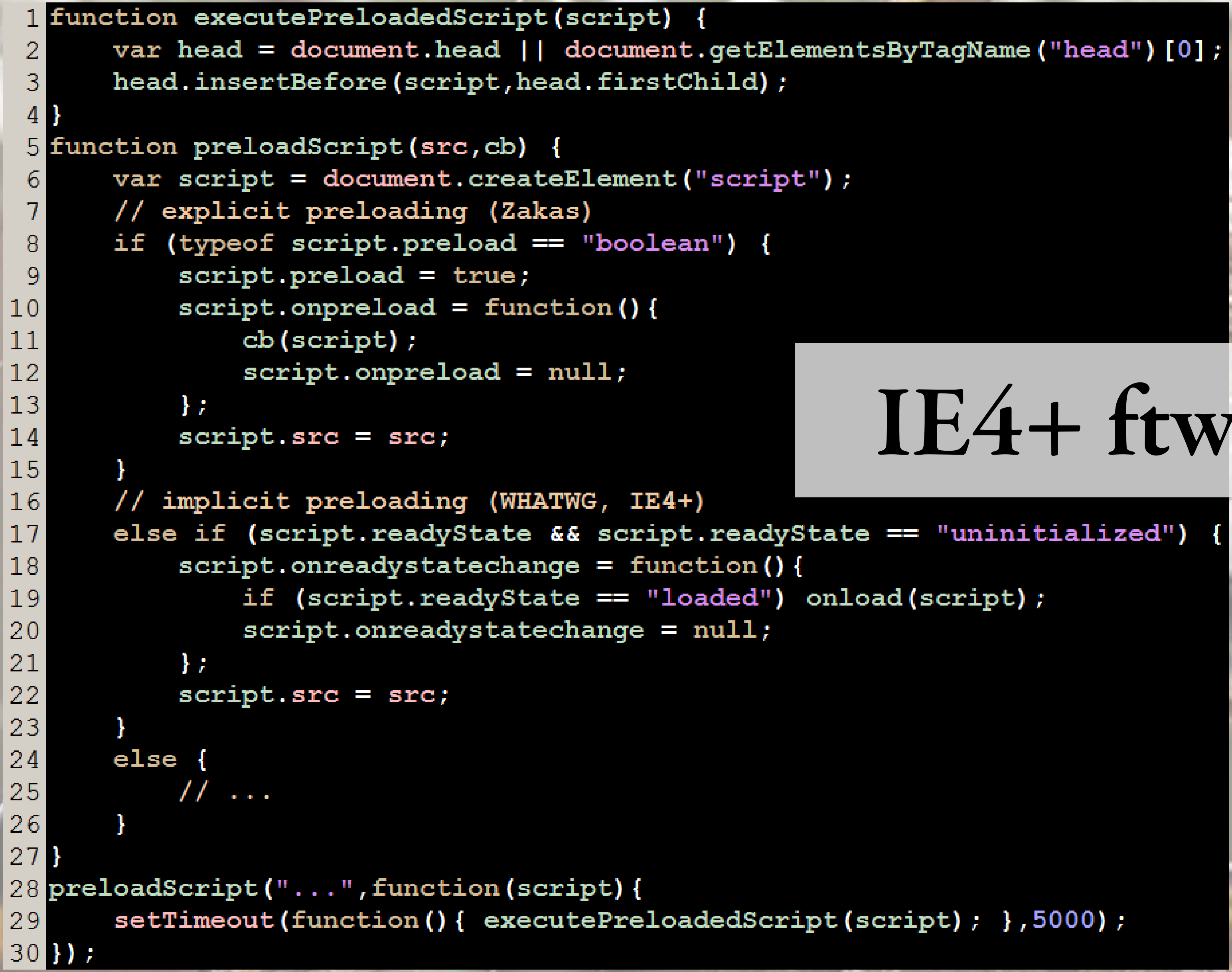
Testing is *much* harder than most realize

What *should* a loader do?

Real Preloading

http://wiki.whatwg.org/wiki/Script_Execution_Control





```
function executePreloadedScript(script) {
    var head = document.head || document.getElementsByTagName("head")[0];
    head.insertBefore(script,head.firstChild);
}
function preloadScript(src,cb) {
    var script = document.createElement("script");
    // explicit preloading (Zakas)
    if (typeof script.preload == "boolean") {
        script.preload = true;
        script.onpreload = function(){
            cb(script);
            script.onpreload = null;
        };
        script.src = src;
    }
    // implicit preloading (WHATWG, IE4+)
    else if (script.readyState && script.readyState == "uninitialized") {
        script.onreadystatechange = function(){
            if (script.readyState == "loaded") onload(script);
            script.onreadystatechange = null;
        };
        script.src = src;
    }
    else {
        // ...
    }
}
preloadScript("...",function(script){
    setTimeout(function(){ executePreloadedScript(script); },5000);
});
```

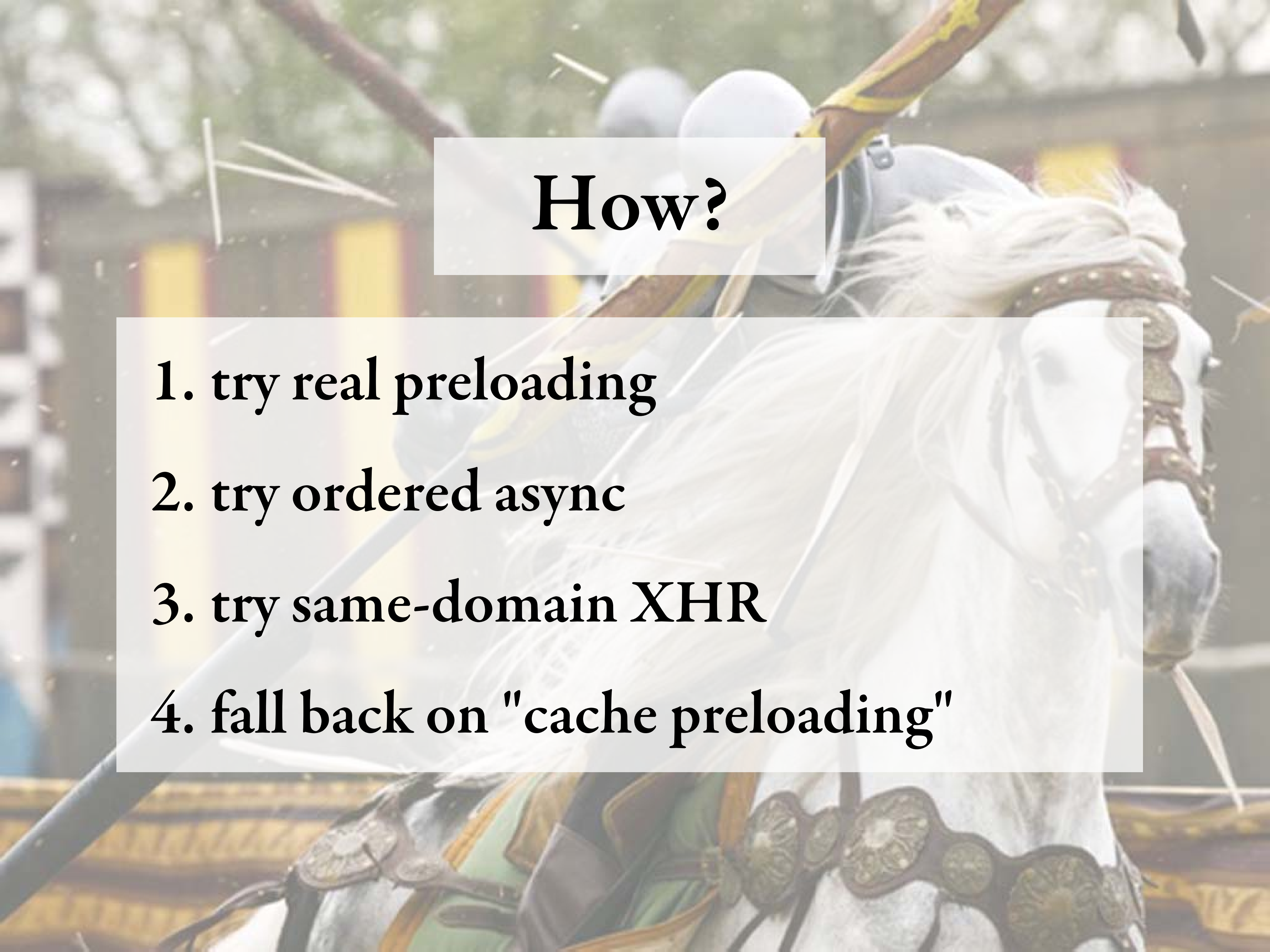
IE4+ ftw?

Ordered Async

async=false

FF4+, Chr 12+, IE10p2+, Webkit/Safari, Opera (soon!)

http://wiki.whatwg.org/wiki/Dynamic_Script_Execution_Order



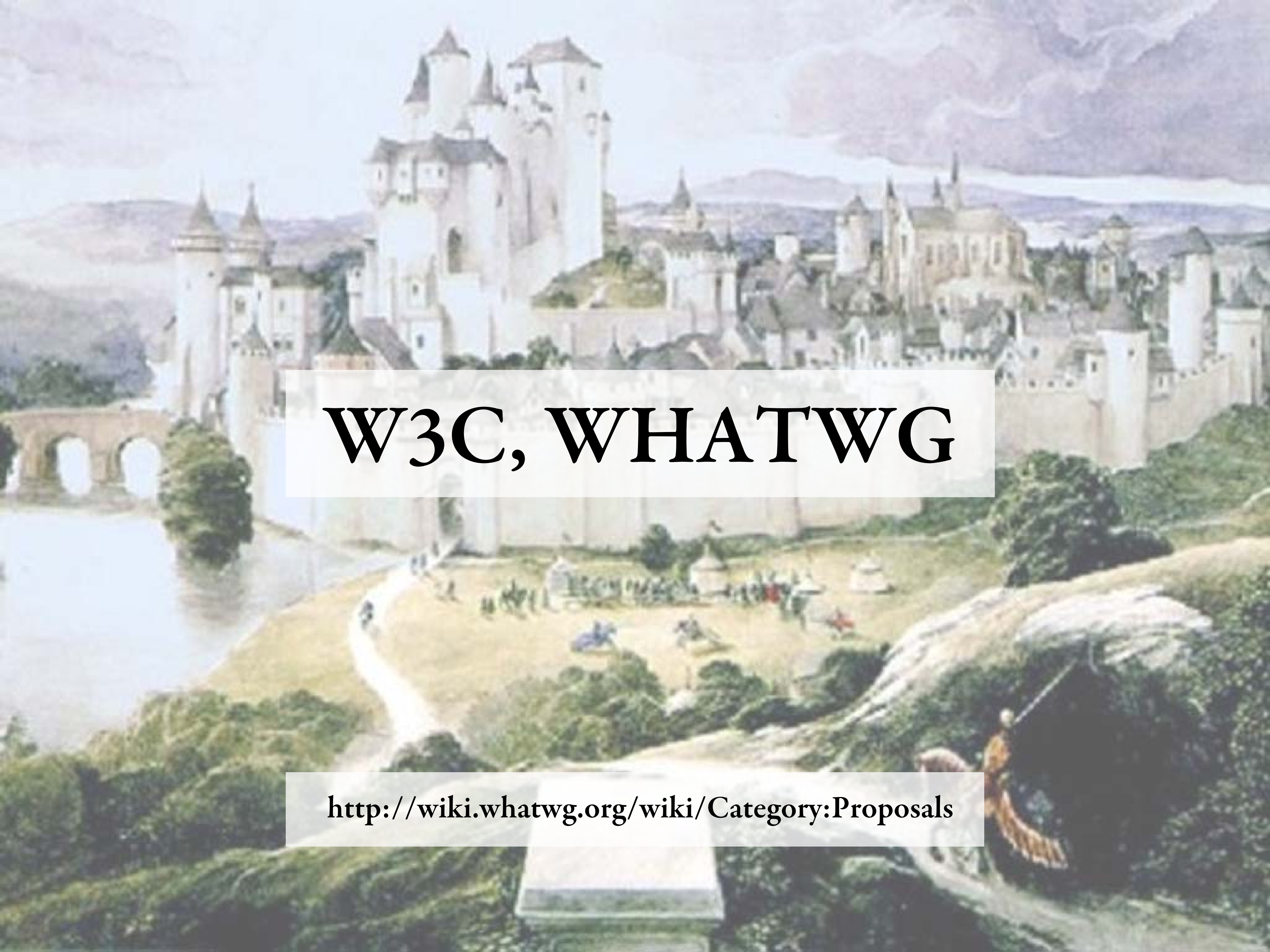
How?

1. try real preloading
2. try ordered async
3. try same-domain XHR
4. fall back on "cache preloading"

Competition is good, *only* if community is educated

Browsers do not *yet* give us what we *really* need

Co-opetition is much healthier for the community



W3C, WHATWG

http://wiki.whatwg.org/wiki/Category:Proposals

W3C, WHATWG

http://ygp.go.ly/script-preloading

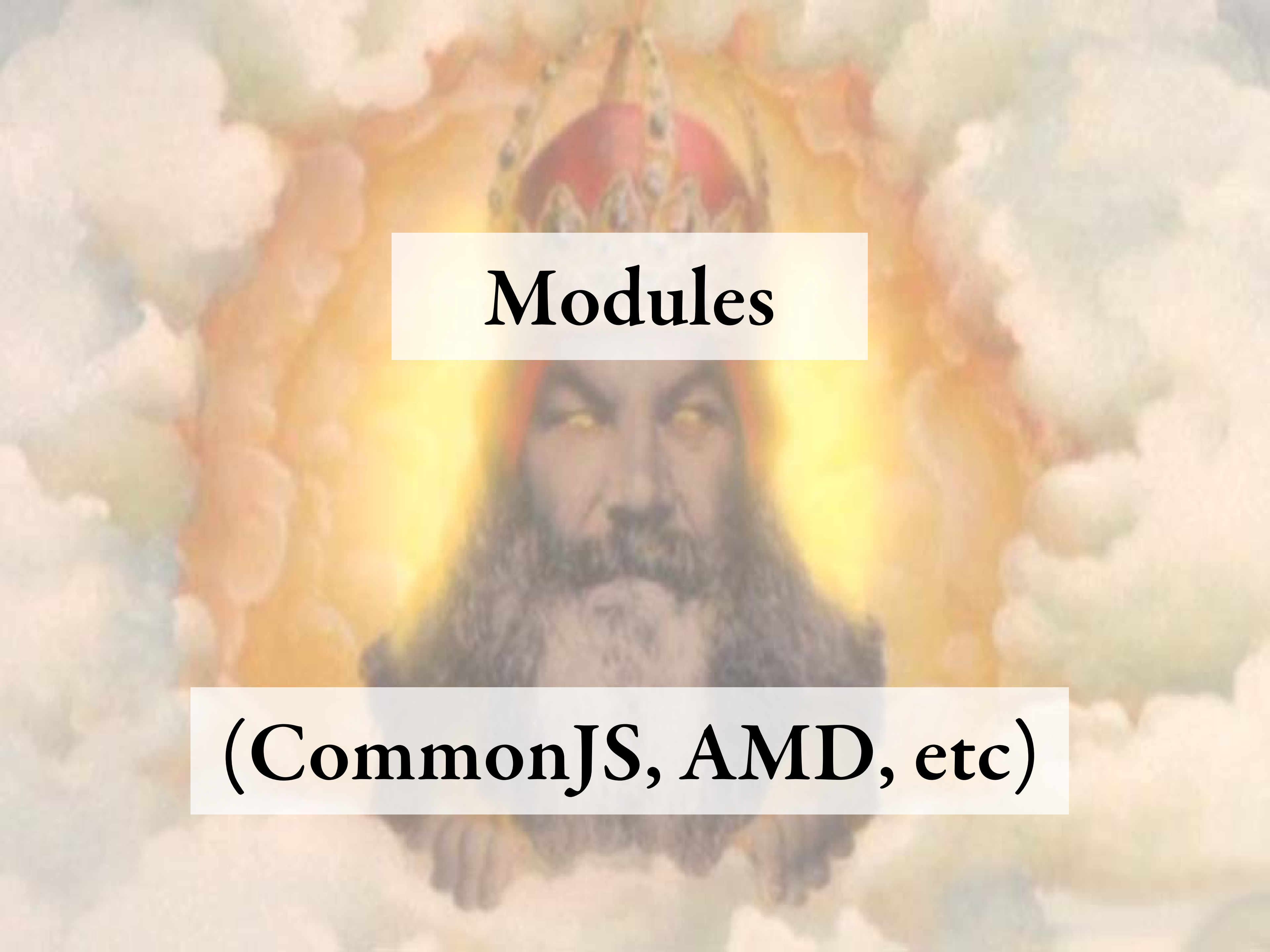
http://odq.go.ly/load-error-events

We'll accomplish more if we work together

Future, The Script Loader

Preloading

(deferred execution)

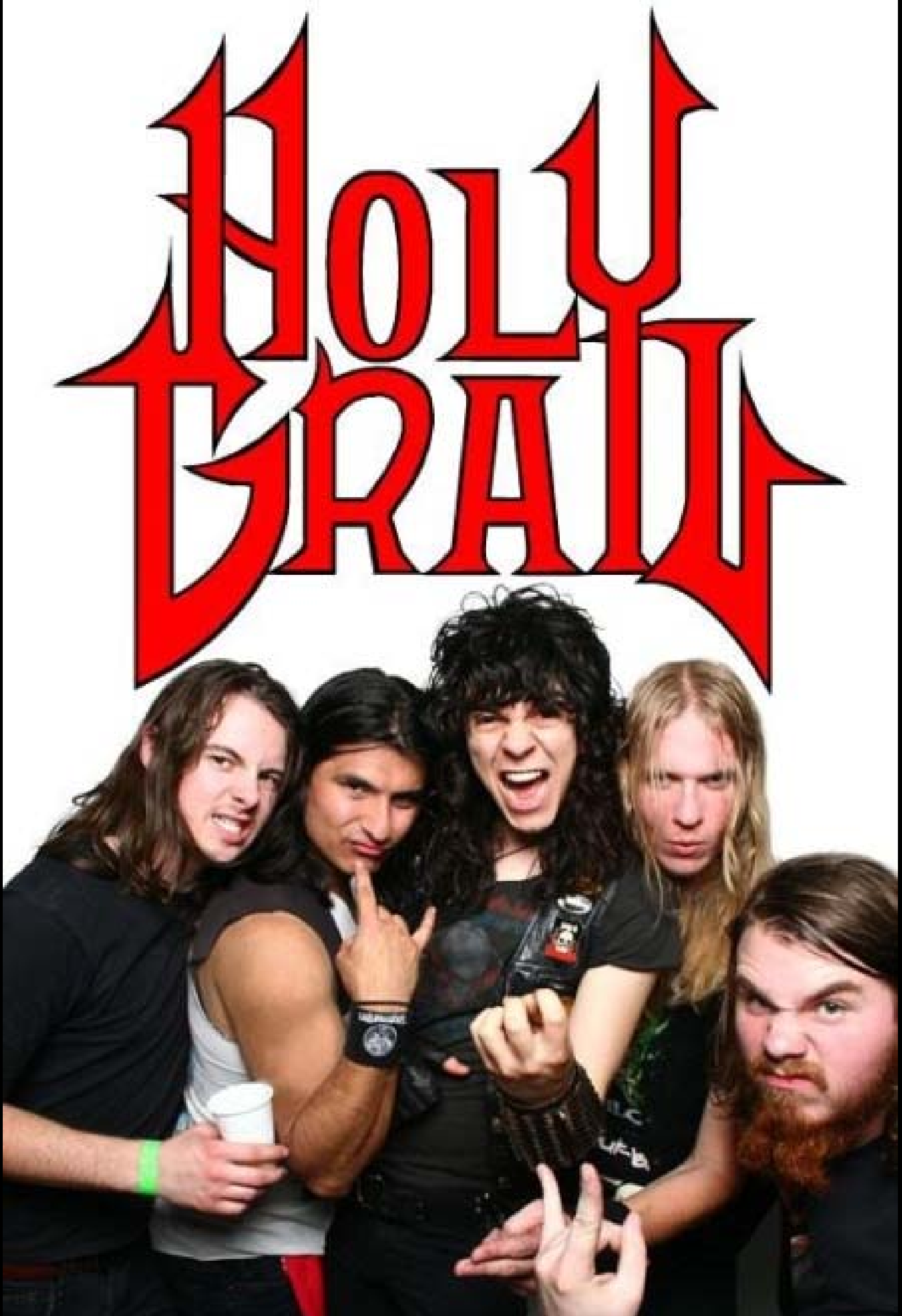


Modules

(CommonJS, AMD, etc)

Native Modules

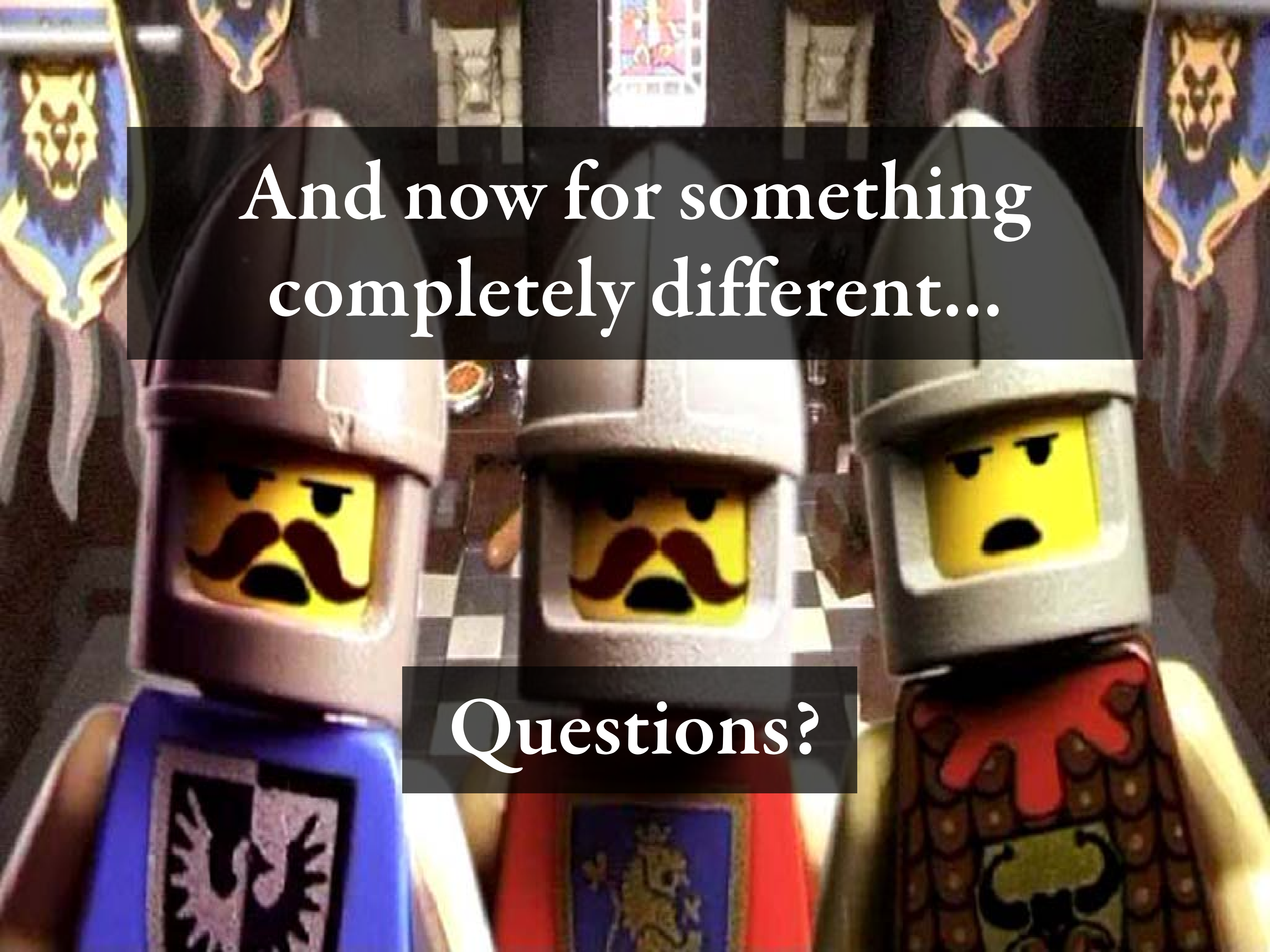
(ES-Harmony?)


HOLY GRAIL

"Script Loader of my dreams"

What else?

- timeouts
- load abort
- error handling
- load priority



And now for something completely different…

Questions?

Kyle Simpson
@getify
http://getify.me

http://wiki.whatwg.org/wiki/Category:Proposals

http://labjs.com