How to approach the web platforms

Peter-Paul Koch
http://quirksmode.org
http://twitter.com/ppk

Nordic Competence Conference, 12 September 2015

The problem

Web platforms

I feel back-end developers underestimate the web platform, and thus front-end development

because they misunderstand one crucial aspect.

The web is not one platform.

Environments

- Back-enders usually work for one known environment, where languages, libraries, power and memory, and tools are pre-defined.
- They expect front-end to be similar: one known environment that they have to learn the ins and outs of, but that’s not fundamentally different
- But it *is* fundamentally different

Suppose your application…

- must run on the five most common Java server environments, all of which bring out a new version every six weeks
- uses four CDNs, two of which have bad APIs
- uses three sets of incompatible libraries, one of which is still in beta
- needs two root certificates that are deliberately incompatible

Welcome to my world

And …

- good documentation exists only for two Java server environments and one CDN - the rest is only sketchily documented by other developers
- kernel panics occur *on your users' computers*, and not on your servers
- this entire landscape changes every six months

That's the front-end experience

Assumptions

- The web is one monolithic platform
- A mature toolchain is available

These two assumptions reinforce each other.

The web is one platform because one toolchain is available.

One toolchain is available because the web is one platform.

Both assumptions are wrong

2 Toolchains

Toolchains

- The web platform toolchain is not mature
- Every *week* there are new libraries and frameworks that promise to solve slight problems in existing ones
- Sure, there are the big ones, such as Angular, React, Ember, etc.
- but they're pretty new as well
- and they have their share of problems

Anatomy of library use

1. Browser must download the library. This is not a huge problem, since authors have been well aware of this fact for years
2. Then the library must be initialised. This is commonly forgotten

Library initialisation

- Take an Android phone and measure its battery power consumption
- Load a mobile Wikipedia page. It uses jQuery for its show/hide script.
- Then replace jQuery by a five-line custom script.
- This saves 30% of the power

http://crypto.stanford.edu/~dabo/papers/browserpower.pdf

Anatomy of library use

1. Browser must download library. This is not a huge problem, since authors have been well aware of this fact for years
2. Then the library must be initialised. This is commonly forgotten

Anatomy of library use

1. Browser must download library. This is not a huge problem, since authors have been well aware of this fact for years
2. Then the library must be initialised. This is commonly forgotten
3. Then the library might need to parse the DOM.

Library DOM parsing

```
<span id=“complexId1”>{{item.name}}</span>
{{item.value}}<br>{{item.price}}
```

- This sort of code belongs on the server. If a library uses it it’s not suited for front-end development.
- Why not? The library has to parse DOM text nodes to find these fragments, and that is the most costly operation imaginable.
- (Looking at you, Angular.)

Library DOM parsing

```
<span id=“itemName”>Placeholder</span> <span
id=“itemValue”>Placeholder</span><br><span
id=“itemPrice”>Placeholder</span>
```

- Instead, libraries should require HTML elements with specific IDs, so that they can be found without parsing the entire DOM.
- (Or any other scheme based on HTML attributes. They’re a LOT easier to find than text.)

Anatomy of library use

1. Browser must download library. This is not a huge problem, since authors have been well aware of this fact for years
2. Then the library must be initialised. This is commonly forgotten
3. Then the library might need to parse the DOM.

Anatomy of library use

1. Browser must download library. This is not a huge problem, since authors have been well aware of this fact for years
2. Then the library must be initialised. This is commonly forgotten
3. Then the library might need to parse the DOM
4. Now the library is ready and waits for user input - which might cause costly operations all over again, but we're aware of that

The true JavaScripter

- uses libraries and frameworks when he needs
- but studies them in detail before doing so
- and prefers to use a single one per project
- is able to write a medium-complex application without any libraries or frameworks
- which gives him the technical background to change a library or framework if necessary

Toolchain conclusion

- There are plenty of tools, but the front-end toolchain is immature. Too much is going on.
- Many tools are fairly heavy; either in initialisation or in needless DOM parsing
- State of the toolchain proves nothing about the web platform
- Make sure you *can* do without tools (which is not the same as never using tools)

3 Browsers

Browsers are the most hostile development platforms in the world

Douglas Crockford

Hostile

The web is not a platform.

It is a multitude of platforms, most of which you’ll never test on.

4 The Android example

Android WebKit

- Originally the Android default browser was Android WebKit
- Each device vendor modified it slightly
- So we’ve got HTC Android WebKit, and Samsung Android WebKit, and LG Android WebKit, and … well, you get the idea.

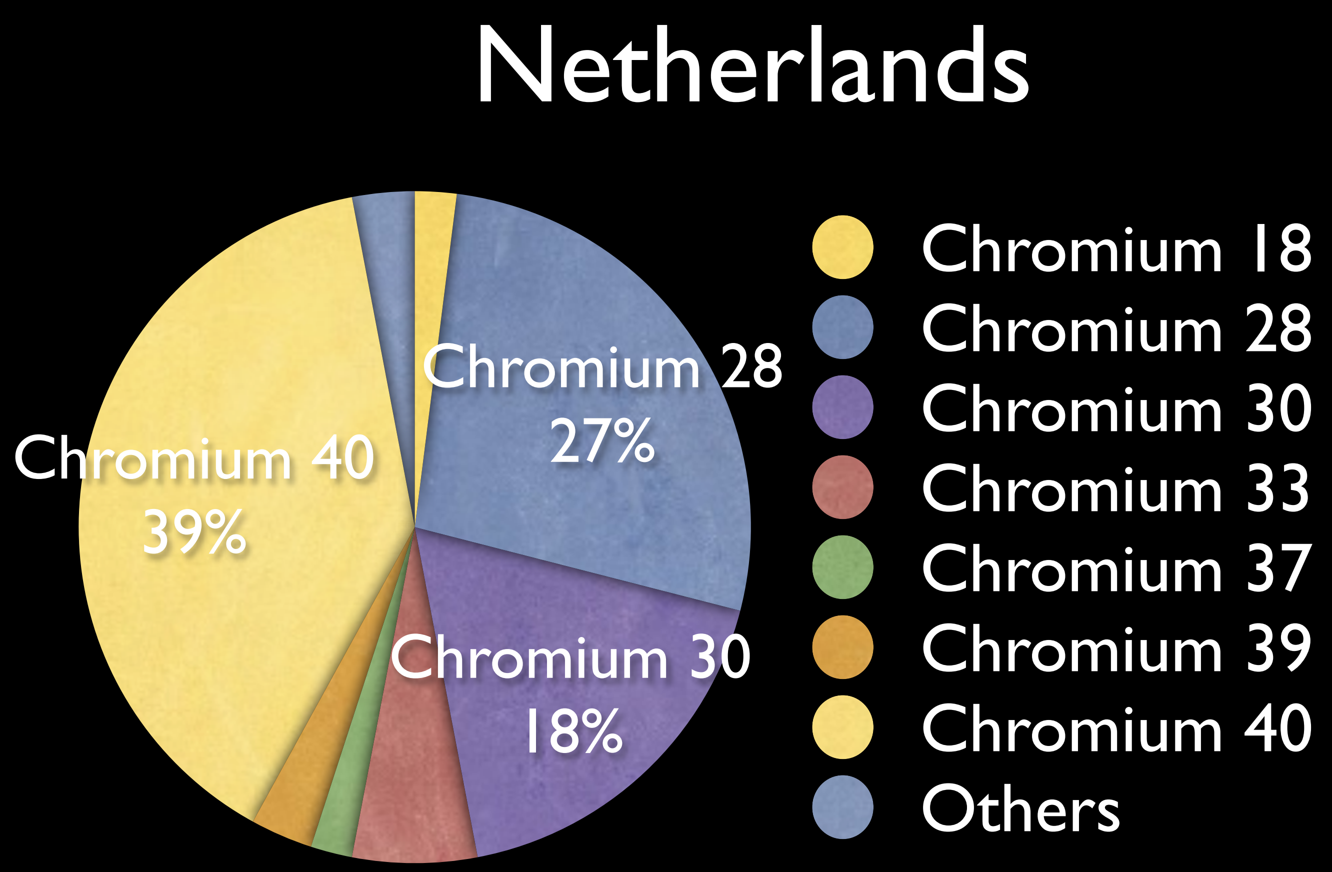
Change of browsers

- Google wants Chrome to become the default browser on Android.
- It ceased development of Android WebKit after Android 4.3
- Device vendors can continue to use it, or can switch to Chrome.
- OR …
- they can build their own. Chromium is free to download, after all.

Chromia

- Now the Android default browser is Chrome
- Each device vendor can either use Google Chrome, or download and adjust its own
- So we’ve got HTC Chromium, and Samsung Chromium, and LG Chromium, and … well, you get the idea.

Chromia shares



Source: about 46K Chromium-impressions from a Dutch ad network; 31st Jan 2015

See also: http://www.quirksmode.org/blog/archives/2015/02/counting_chromi.html

Google Services

- The situation is clearly not complicated enough.
- Google Services. A package of crucial Google apps such as Maps, YouTube, Play … and Chrome.
- Device vendors may opt in to or out of the entire package. No picking and choosing!
- Opting in requires them to install Google Chrome on their devices
- but *not* to make it the default browser

Google Services

- Thus the Samsung Galaxy S4 and up has two browsers pre-installed:
- Samsung Chrome 28, which is under the “Internet” icon on the home screen
- Google Chrome 45 (and counting), which can be found in the Apps menu

Google Services

- Thus the HTC M8 has two browsers pre-installed:
- HTC Chromium 33, which is under the "Internet" icon on the home screen
- Google Chrome 45 (and counting), which can be found in the Apps menu

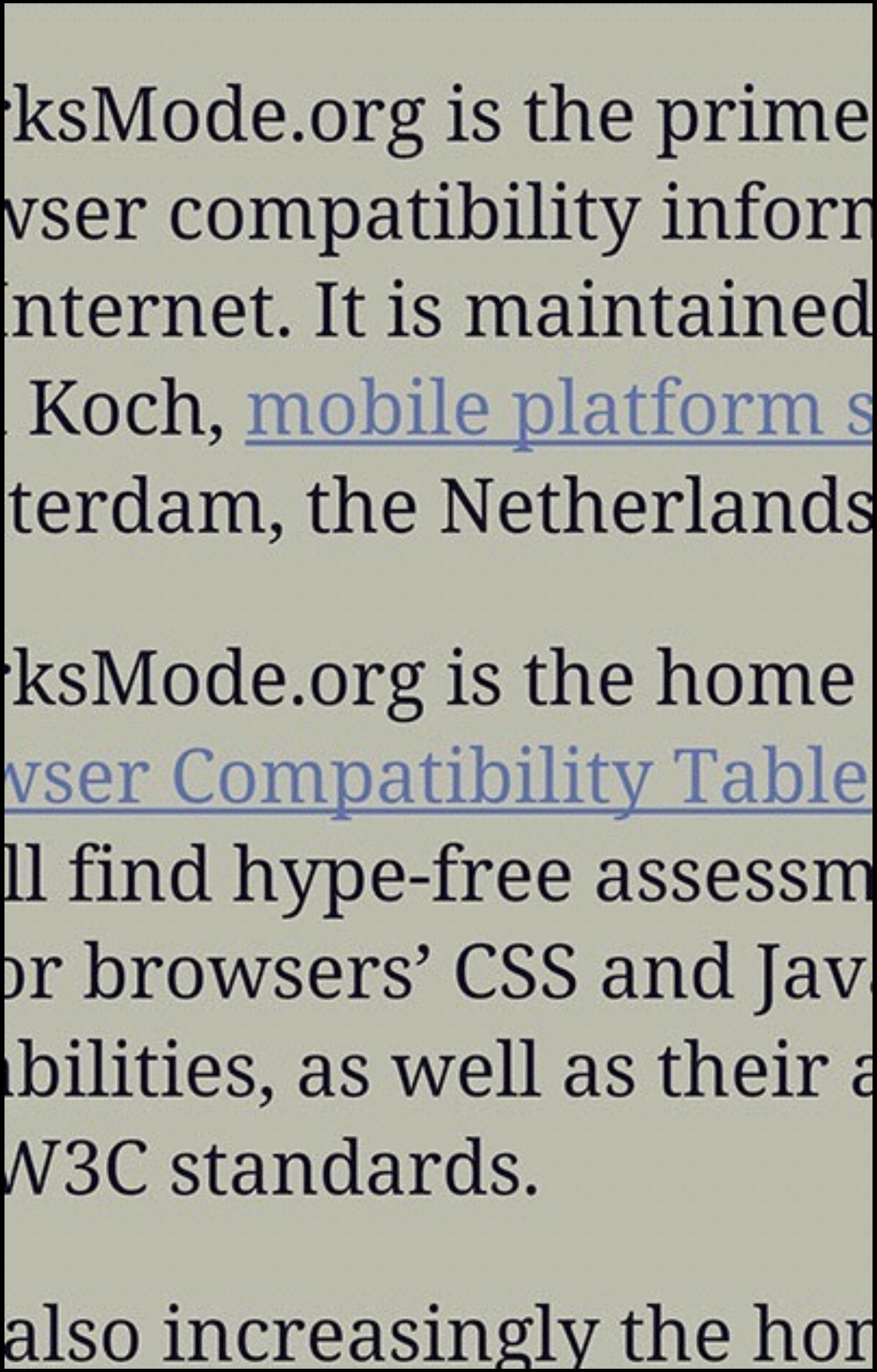
Google Services

- Some vendors opted out of Google Services
- Amazon (competitor of Google on service level)
- the Chinese vendors such as Xiaomi, Huawei, ZTE, etc. (China has its own services)
- These devices must create their own default browser (based, if they wish, on Chromium)
- and they do NOT have Google Chrome pre-installed.

Android default browsers

Samsung	Chromium 28 or 34
HTC	Chromium 33
LG	Chromium 30 or 34
Xiaomi	Chromium 34 or 35
Cyanogen	Chromium 33
Huawei	Chromium 30
Sony	Chrome current
Nexus	Chrome current

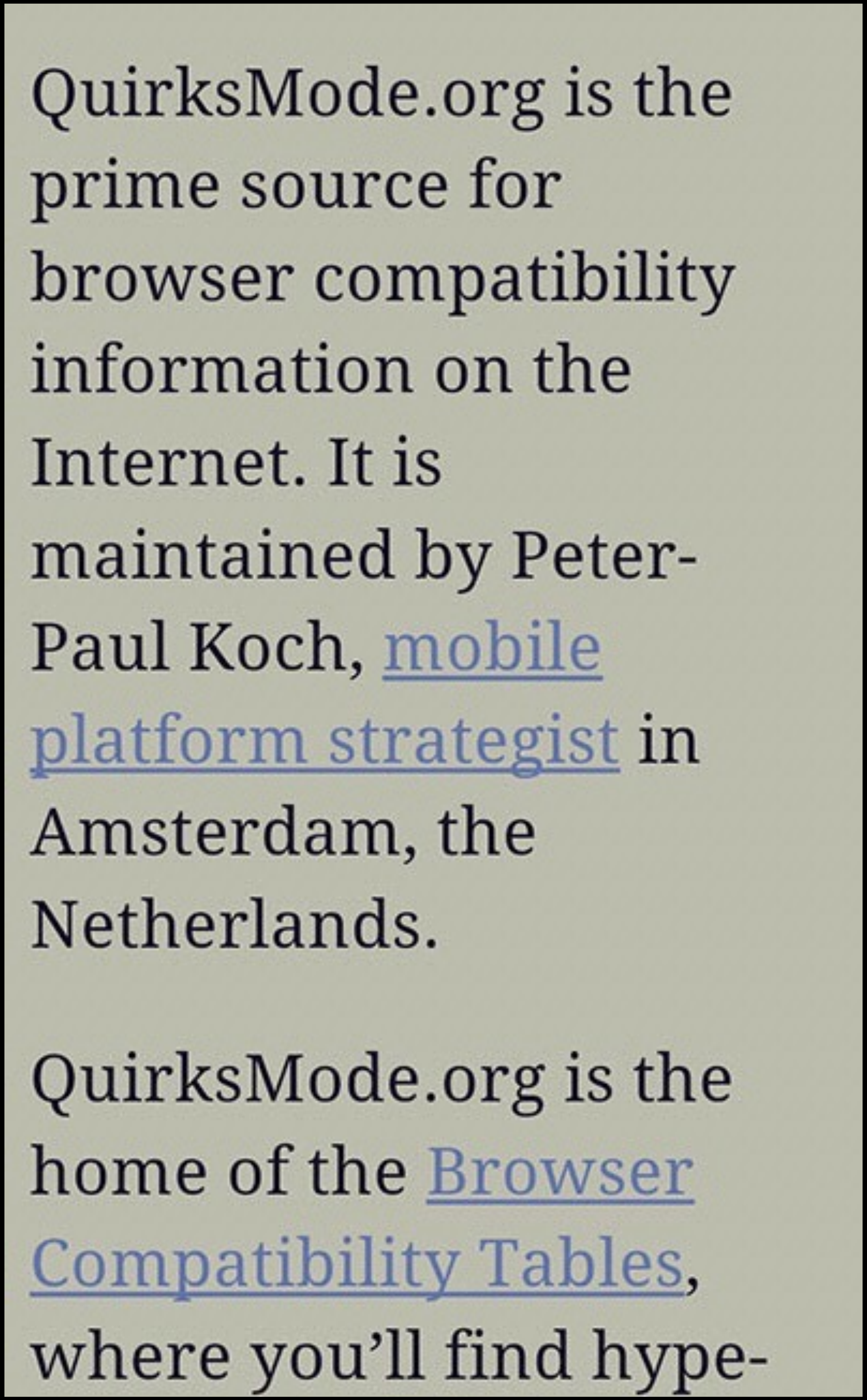
(Incidentally, Chrome on iOS is not Chrome.
It's Safari.)


ksMode.org is the prime
vser compatibility inforn
nternet. It is maintained
Koch, mobile platform s
terdam, the Netherlands

ksMode.org is the home
vser Compatibility Table
ll find hype-free assessm
or browsers' CSS and Jav
bilities, as well as their a
N3C standards.

also increasingly the hor

Most browsers do this



QuirksMode.org is the prime source for browser compatibility information on the Internet. It is maintained by Peter-Paul Koch, mobile platform strategist in Amsterdam, the Netherlands.

QuirksMode.org is the home of the Browser Compatibility Tables, where you'll find hype-

HTC default browsers do this

Future:
Moar Chromia!

5 Handling browsers

Supported browser list

- But that browser isn’t on our Supported Browser List!
- Not fair.
- First, an environment (including toolchain) is defined
- and the Supported Browser List consists of browsers that run that environment.
- Circular argument

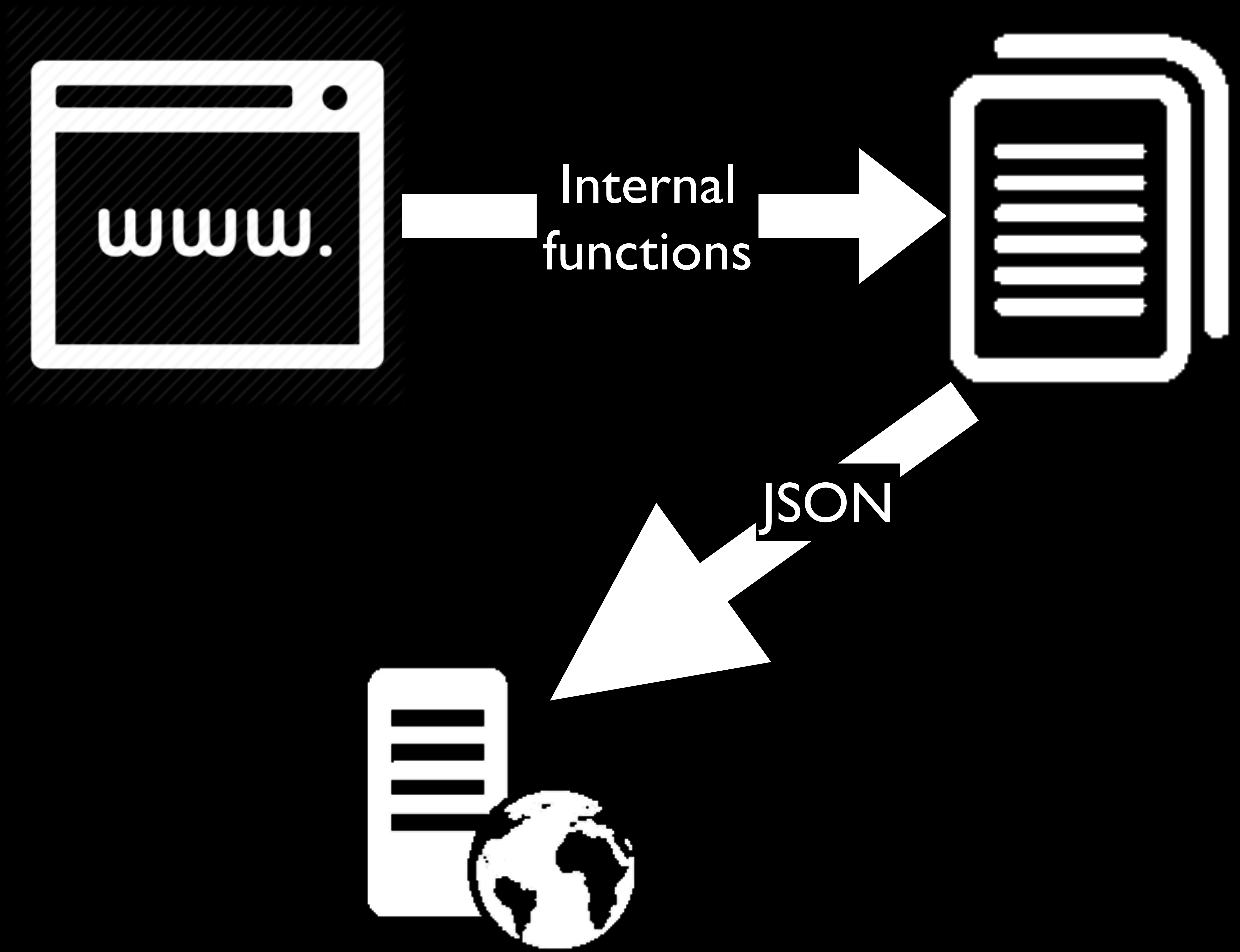
Supported browser list

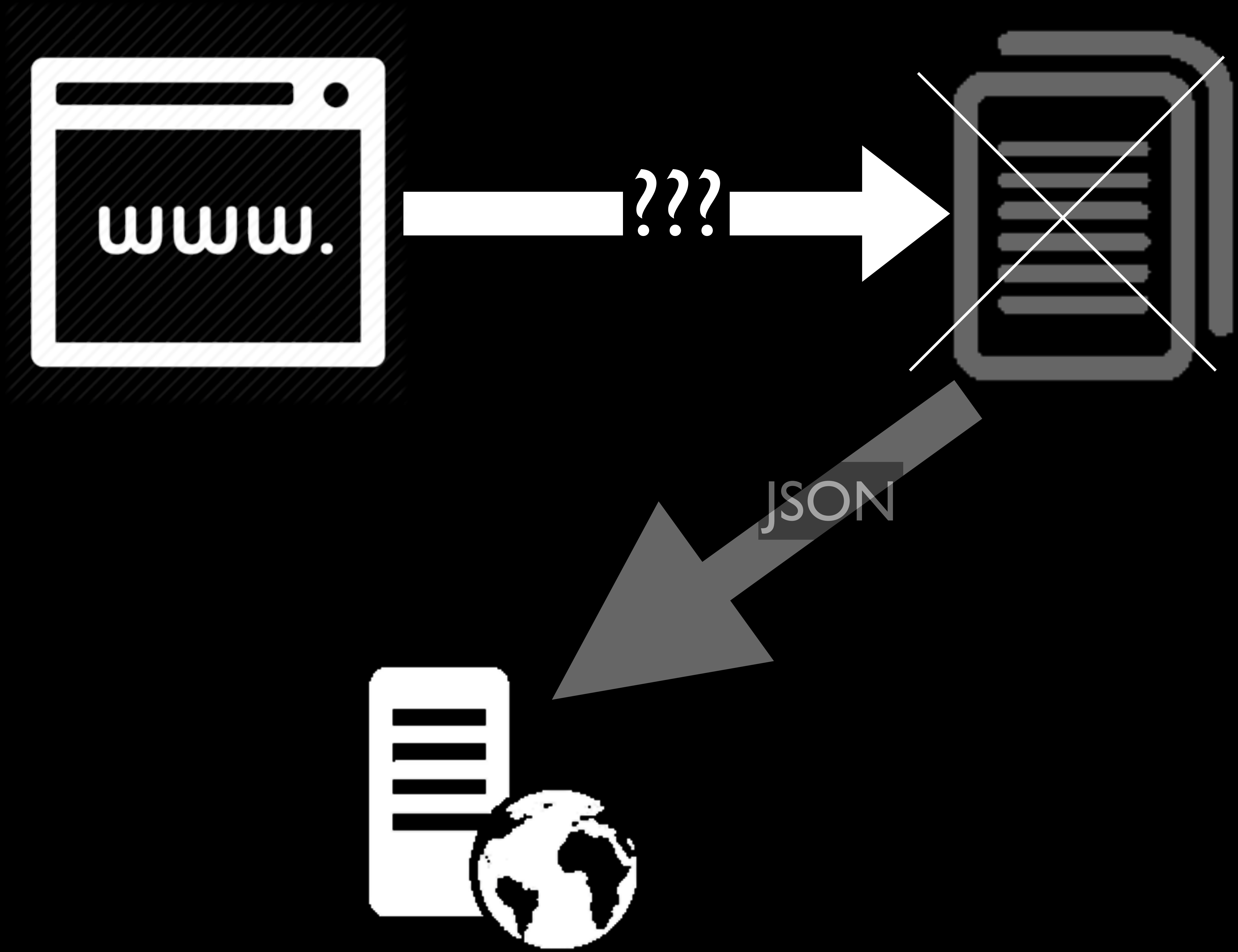
- This is not the right way to approach the web.
- You pretend there is only one platform
- but as we saw there are countless ones.

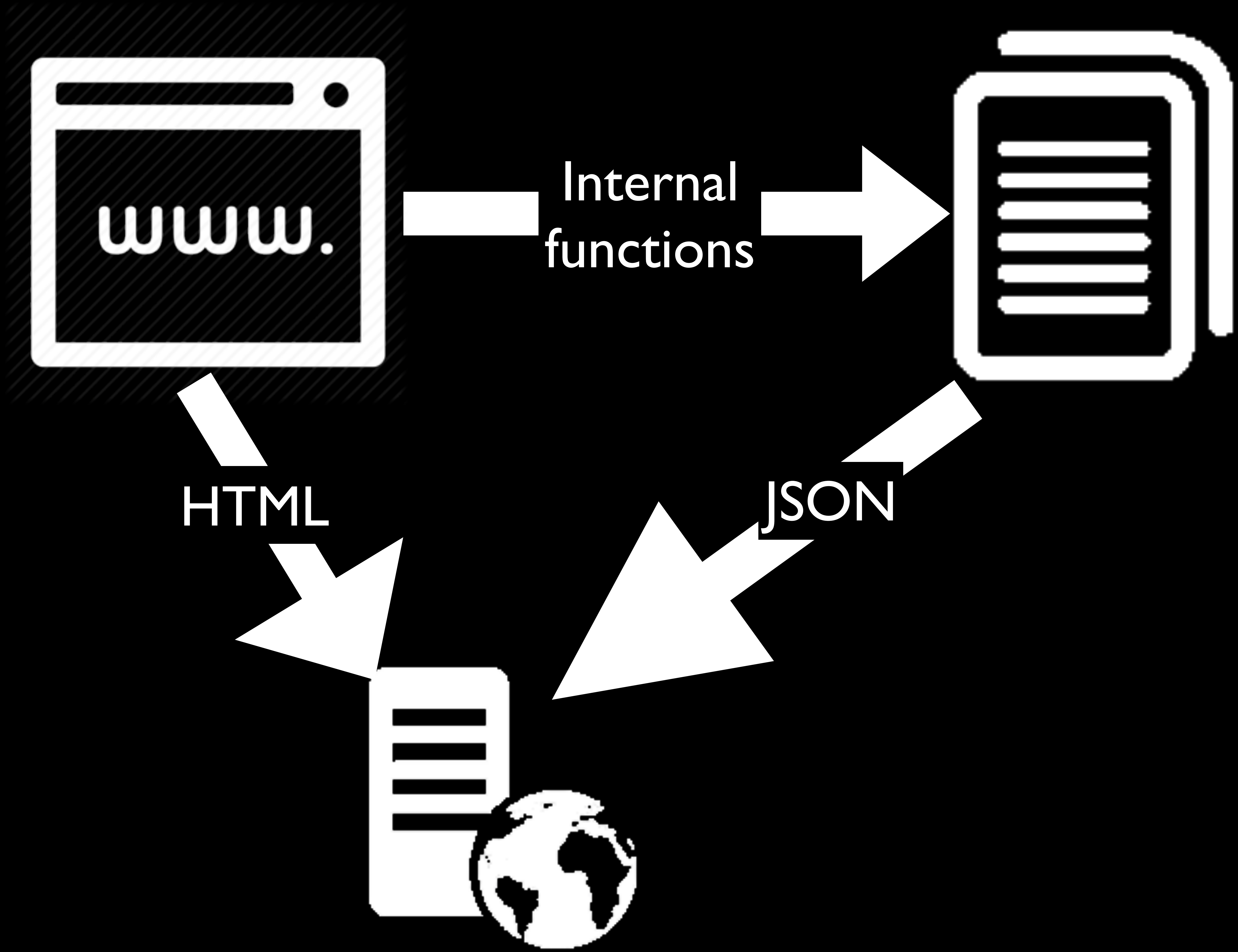
Supported browser list

- Instead, take an A-list, B-list, C-list approach.
- A-list browsers are the ones you guarantee the application to work in.
- B-list browsers (optional) are the ones you consider likely to run the entire application and/or they lack a few non-essential features.
- C-list browsers are all other browsers; you don't make any guarantees, but they'll at least be able to view the HTML.

Solution: build multiple versions of your application


www.
Internal functions
JSON


www.
???
JSON


www.
Internal functions
HTML
JSON

Noscript version

- The point of building a noscript version of your application is NOT users without JavaScript.
- Few users disable JavaScript entirely.
- Instead, users may be on a crappy device that takes ages to initialise the libraries
- or on a crappy network where the library just doesn't load
- A monitor script could figure that out and switch to the noscript version

Noscript version

- The noscript version is aimed at different platforms than the scripted one.
- In fact, because it aims at the lowest common denominator (HTML), it is certain to work in ALL browsers
- Thus it’s an excellent fallback when unforeseen problems occur

Several versions

- This is in fact a general rule.
- Creating an extra version of your application allows you to target more web platforms.
- In addition to noscript, maybe you want a version that doesn’t use animations
- or one that leaves out the secondary page content

6 Conclusion

The web platforms

- There are countless web platforms
- and the web toolchain is in chaos
- and browsers are in chaos, too, especially on Android
- The optimal strategy is to create several versions of your application, each aimed at certain web platforms
- and to be able to write JavaScript without libraries

Learning and teaching

- Java developers hold that they're better than front-end developers at structuring large applications.
- Front-enders recognise that
- Front-enders hold that they're better than Java developers at dealing with browsers.
- Back-enders don't recognise that
- Imbalance; insisting on your version of the truth while casually dismissing someone else's comes across as arrogant

Thank you

I'll put these slides online

Questions?

Peter-Paul Koch
http://quirksmode.org
http://twitter.com/ppk
Nordic Competence Conference, 12 September 2015