Device Detection

Dino Esposito
http://software2cents.wordpress.com
@despos

pluralsight
hardcore dev and IT training

Key Points

Feature detection…

…or **Device** detection

Client detection vs.
Server detection

Detecting devices is hard?

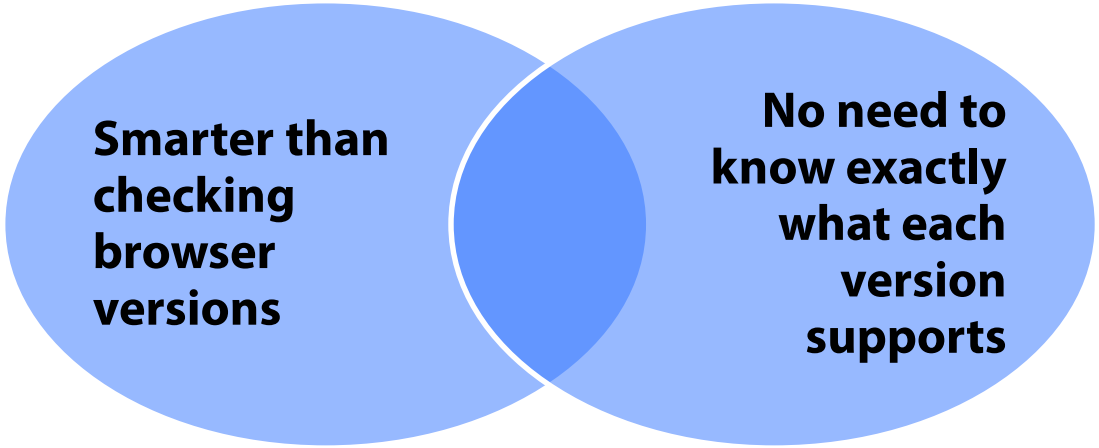
Don’t do that.
Be smart and just focus on browser’s features.

Via DOM/JavaScript.


Feature Detection
comes to the rescue.

Foundation of Feature Detection

- **Check the feature before you invoke it**
  - Most HTML5 API can be easily checked via JavaScript
  - Local storage, canvas, geolocation, events
- **Feature detection checks for a given feature**
  - Tracking evolution of browsers is just undoable
  - Supersedes browser detection

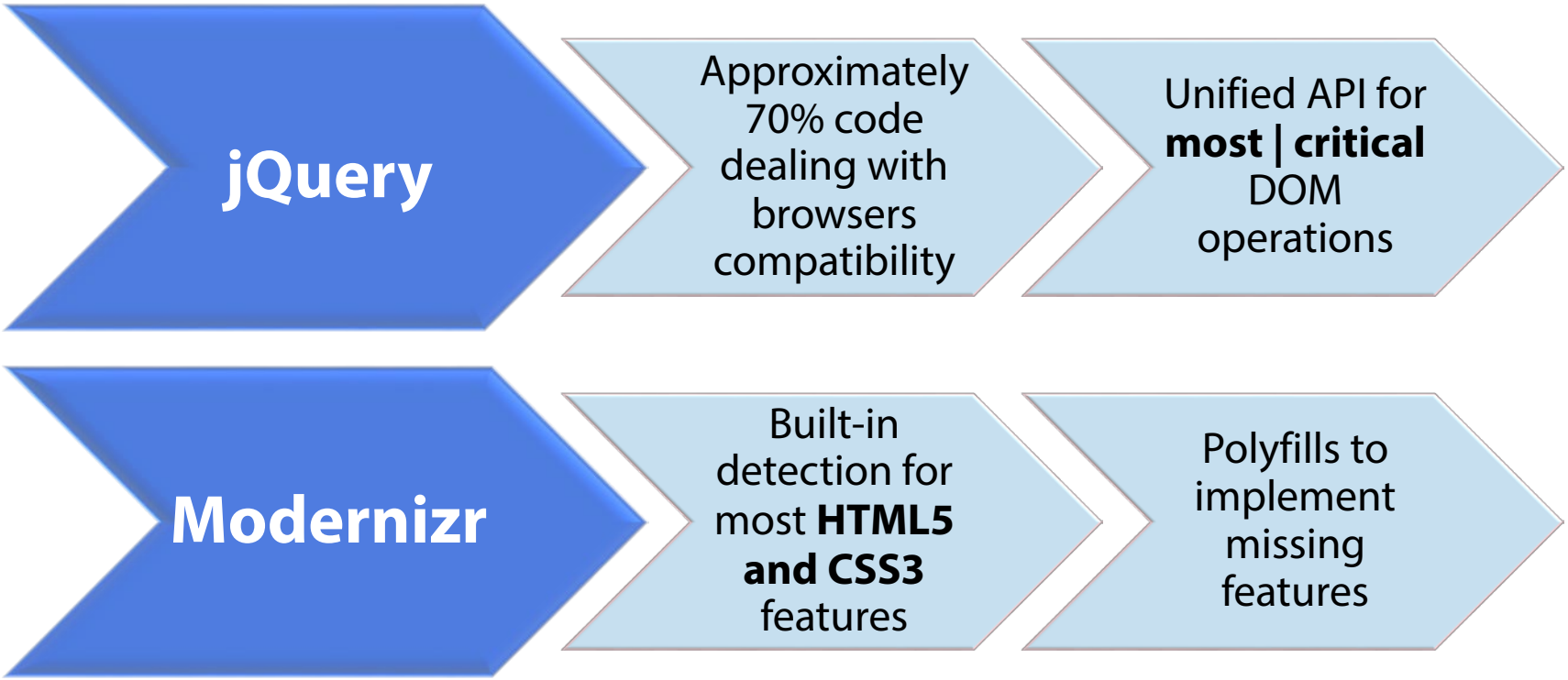


Feature Detection

Example

```
<script type="text/javascript">
  if(window.addEventListener)
  {
    // Check standard method first: addEventListener
    window.addEventListener("load", loadHandler, false);
  }
  else if(window.attachEvent)
  {
    // Check possible legacy forms: attachEvent
    window.attachEvent("onload", loadHandler);
  }
</script>
```

Feature Detection Libraries



Reliable?

- Sometimes , **the browser claims it supports a feature but …**
  - The browser lies
  - Support is only partial
  - Implementation is poor (read, you'd better find a workaround)
  - There's a bug and you need to detect it before you can bypass it
  - The bug is random (read, it's really hard to detect it)
- **Need an expert opinion**
  - Whether a given browser does support a given feature
  - How would you identify the browser?

Interesting reading:
http://www.sitepoint.com/javascript-feature-detection-fails/

So what's the point?

Feature detection is alternative to browser detection; not to device detection.

When It Comes to Devices…

- **Feature detection is the solution if**
  - You want to implement a feature **regardless** of the underlying browser
  - If you want the site to look **the same** on all browsers
- **Feature detection is NOT the solution if**
  - You want to provide a device-specific experience

Get to Know the

Form Factor of the Device.

To Be Known About Devices…

- **Form factor**
  - Size, shape, and style
  - Smartphone, (mini) tablet, phone, laptop, smart TV, wearable device
- **Platform**
  - iOS, Android, Windows Phone, Windows 8
  - Version number
  - WebView in an app or plain browser window?
- **Capabilities** **(optional)**
  - Physical: screen size, touch, phone capabilities, connectivity
  - Support for HTML/CSS specific elements
  - Video and image support, Flash/Silverlight

Form Factor

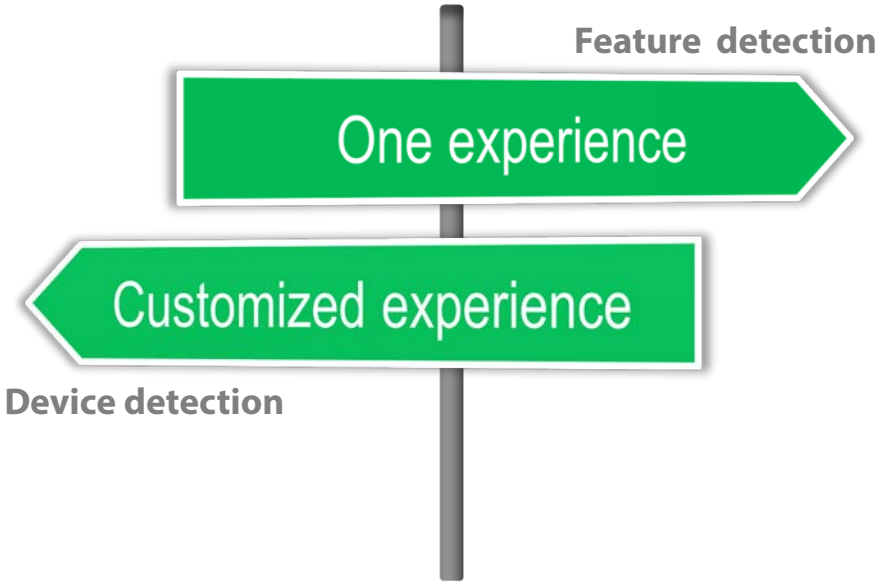
- **Analogous to visual breakpoints in RWD**
  - Smartphone is NOT like a resized browser window on a huge laptop
  - Devices may be connected over 3G

Wifi

Slow 3G on the go

Device vs. Feature Detection

- **Feature detection**
  - Depends on JavaScript and browsers
  - Aims at **replicating the same experience** through all devices
- **Device detection**
  - Depends on external code that reliably detects the device
  - Aims at **customizing experience/use-cases** on a per device basis



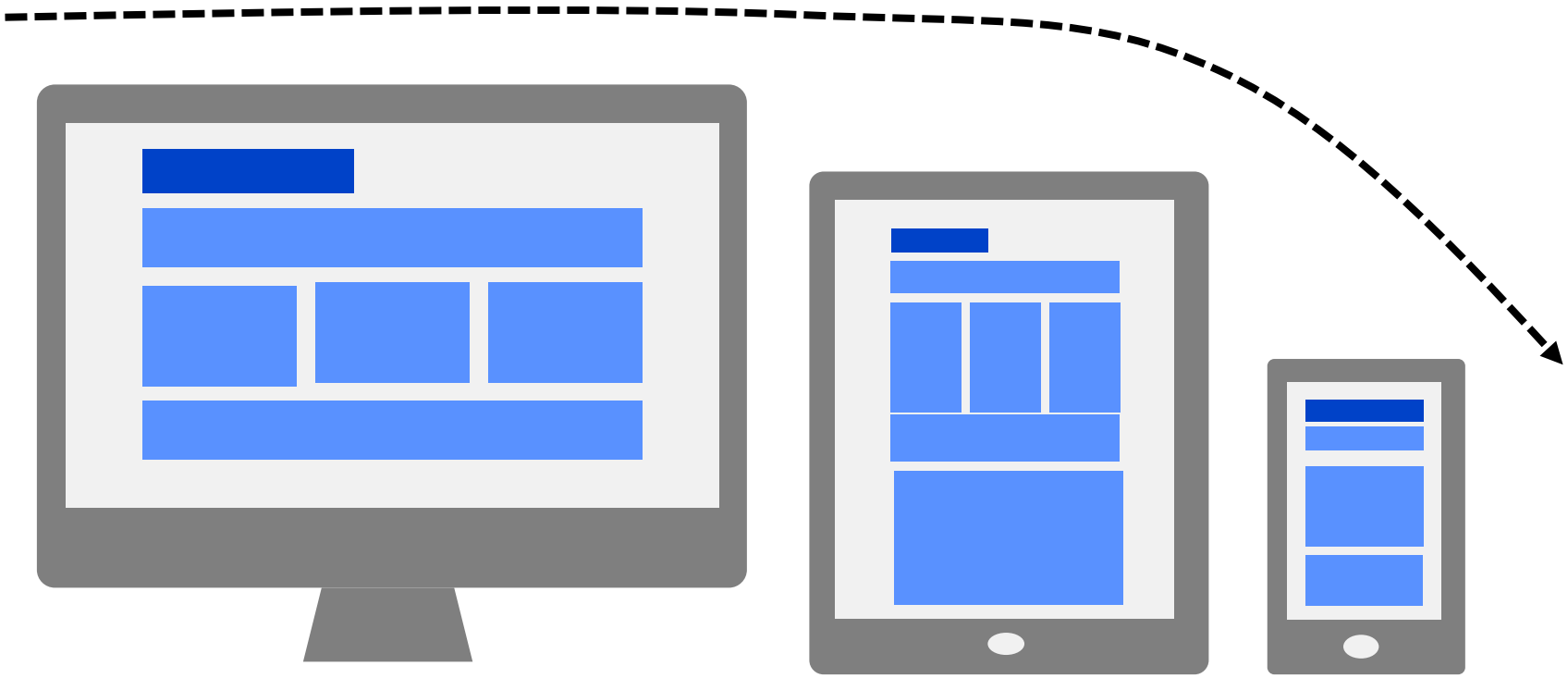
Use Form Factor to…

- **Select the most appropriate layout and UX**
  - An iPad is not a laptop
  - Users may expect different use-cases and hierarchy of content
- **Optimize requests and downloads**
  - An iPad or smartphone may be used on 3G
  - Minimize number of requests and markup downloaded
  - Smaller images, reduced JavaScript, minimal CSS
- **Fine-tune processing power**
  - A laptop is more powerful than tablets/smartphones in terms of CPU
  - Network latency
  - Do not assume 4G – or **make sure** your assumption is right

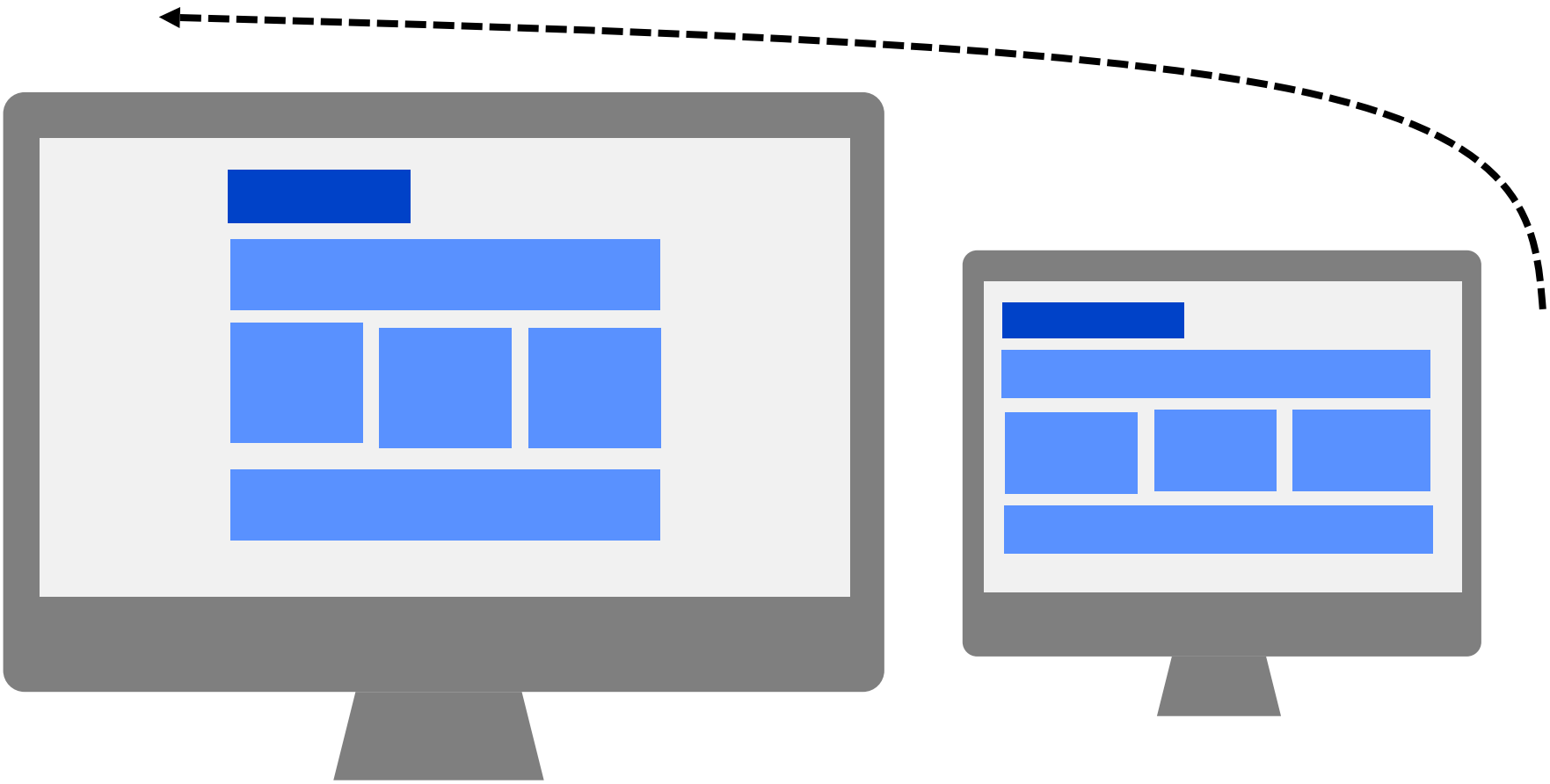
Ignoring Form Factor Leads to …

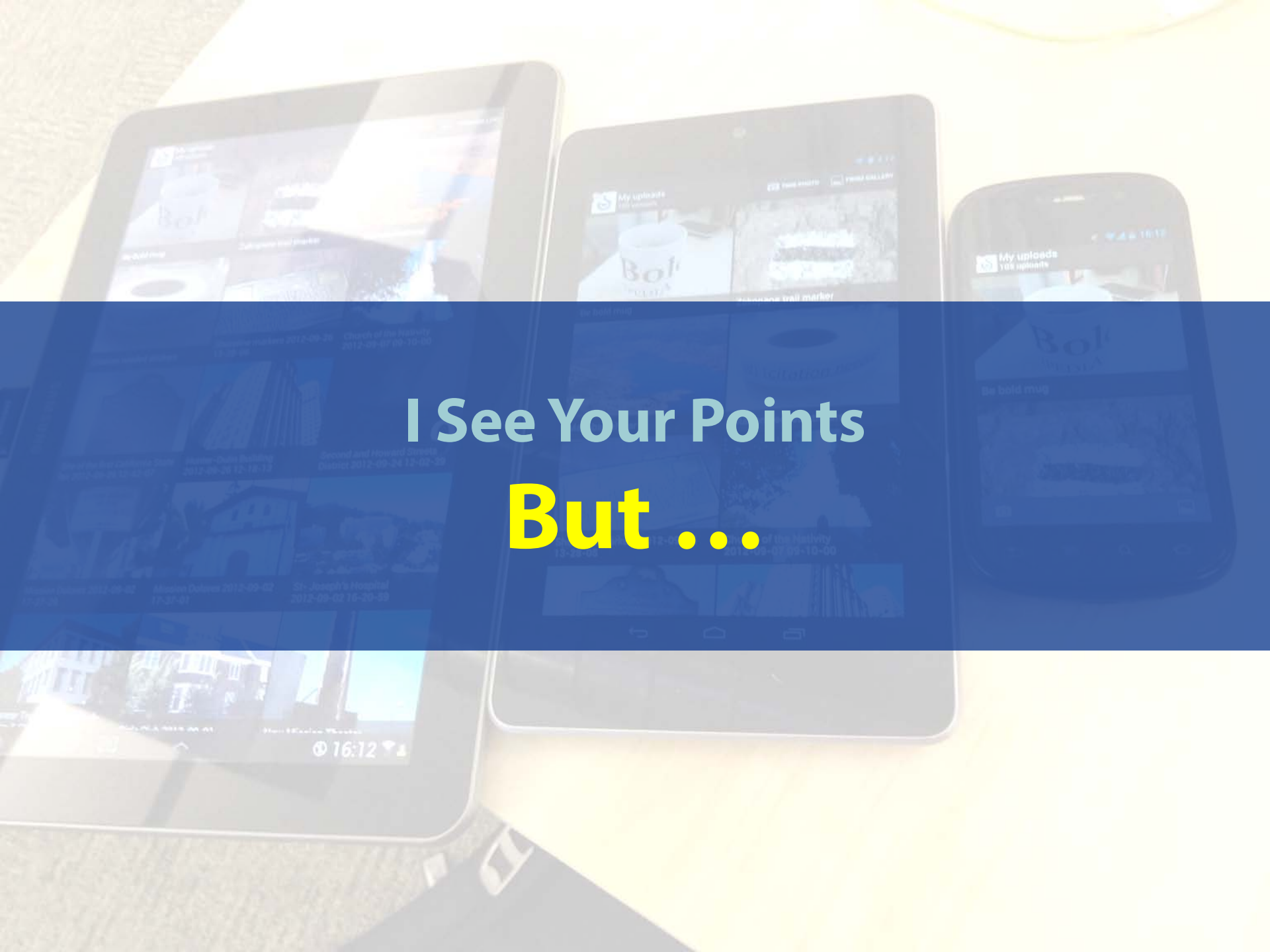
- **Treating all devices as one—mostly like the laptop**
- **Potential issues at various levels**
  - **Usability**: miss device-specific use-cases
  - **Performance**: download content that just gets hidden!
  - **Reachability**: your site is sold as device-friendly, but it is friend of just a few classes of high-end devices
- **Graphical issues too…**

Graphical :: Shrink



Graphical :: Stretch




I See Your Points
But …

Yes, But …

My users are required to use high-end devices	In two years, there will be no processing differences	**Commonplaces** (if not fallacies)
• **Great**! But sure this is really a requirement from customers?	• Maybe CPUs will be the same • Hardly a 4" screen will be **the same as** a 13" screen	• Nobody will use a **laptop** in five years • We're building software for the **next** generation of users • We do **mobile-first**

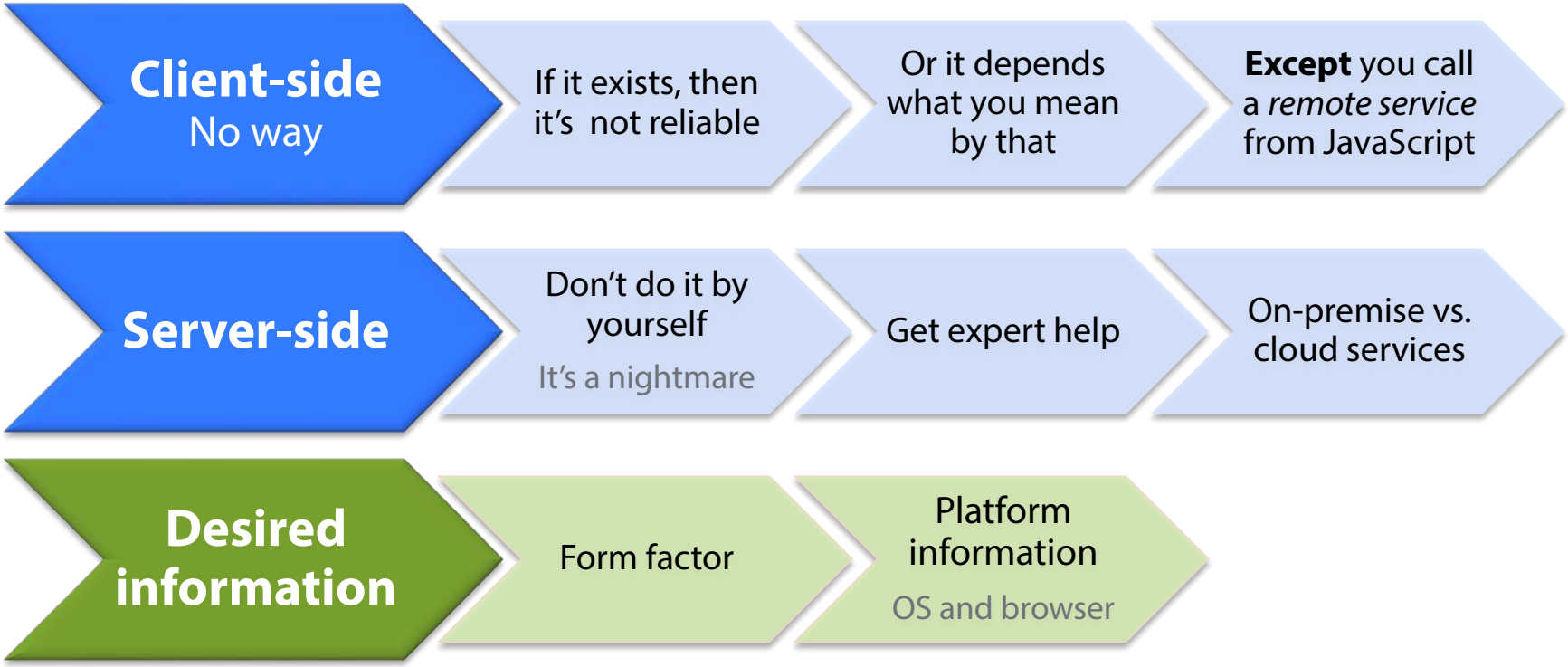
PS: **Mobile what? Mobile doesn't exist. It's devices! DEVICES! DEVICES! DEVICES!**

At the Very End of the Day

It's All About Device Detection.


If You're Building Software for Devices
You Can't Ignore Which Device You're On.

Foundation of Device Detection



Foundation of Client-side Detection

- **No information from browsers**
  - No standard coming up for form factor and platform detection
- **Navigator object maybe?**
  - Browser name properties: *appName*, *appVersion*
  - Operating system property: *platform* (ex: Win32, Linux)
  - Engine name property: *product*
- **Most reliable is userAgent**
  - Unique ID of the browser
  - Should incorporate **coded** information about platform, browser, versions, engine, host device name

User Agent (UA)

Value of the User-Agent HTTP header sent by the browser.

The value returned contains information about the name, version and platform of the browser.

A User Agent Looks Like This …

```
Mozilla/5.0 (Linux; Android 4.0.3;
GT-P3110 Build/IML74K)
AppleWebKit/535.19 (KHTML, like
Gecko) Chrome/18.0.1025.166
Safari/535.19
```

Poor Man's UA Sniffing

- **Searching for common substrings in the UA**
  - Using regular expressions for more sophisticated searches
  - Some JavaScript snippets/patterns exists

- **Handmade UA sniffing doesn't work**

At the current stage of web technology
UA sniffing is the only safe way to identify browsers

```
Mozilla/5.0 (Linux; Android 4.0.3; GT-P3110
Build/IML74K) AppleWebKit/535.19 (KHTML, like Gecko)
Chrome/18.0.1025.166 Safari/535.19
```

Who Can Do UA Sniffing Reliably?

Device Description Repository (DDR)

Database containing a list of known capabilities for thousands of devices. Devices are identified by their UA.

WURFL.js

- **HTTP endpoint of a remote service**
  - Backed by the WURFL framework—de facto standard in DDRs
- **Blinks at scenarios like**
  - Single-Page Applications
  - Client intensive apps where you use JavaScript to give pages a final touch after determining the form factor
- **JavaScript library**
  - Parses the UA
  - Injects a JavaScript object in the DOM describing the device
  - Free of charge: *http://www.scientiamobile.com/page/wurfl-js-license*

WURFL.js

Enable **WURFL.js** as below

```
<script type="text/javascript"
        src="http://wurfl.io/wurfl.js">
</script>
```

Here's what you get in the DOM

```
var WURFL = {
   "complete_device_name":"generic web browser",
   "is_mobile":false,
   "form_factor":"Smartphone"};
```

WURFL.js

- **Form factors**
  - Desktop | App | Tablet | Smartphone | Feature Phone | Smart-TV | Robot | Other non-Mobile | Other Mobile
- **Property is_mobile**
  - Quick check whether it is desktop or not
- **debug=true URL parameter**
  - Useful while developing, switches off any cache

Foundation of Server-side Detection

- **Based on UA sniffing**
  - Sophisticated analysis of user agent strings
  - Map strings to known device profiles
- **DDRs are the key to effective detection**
  - Must be continuously updated
  - Multi-platform/multi-language API
- **Matching "known" capabilities**
  - Tell you what perceived capabilities
  - Human eye working around possible lies/inaccuracies of browsers

Available DDRs

- **WURFL** (multi-platform) www.scientiamobile.com
- **DeviceAtlas** (multi-platform) www.deviceatlas.com
- **51 degrees** (ASP.NET) www.51degrees.com

Some Web giants
(Google, Facebook, PayPal)
use WURFL.

F12 multi.html - F12 Developer Tools

Network

SUMMARY DETAILS

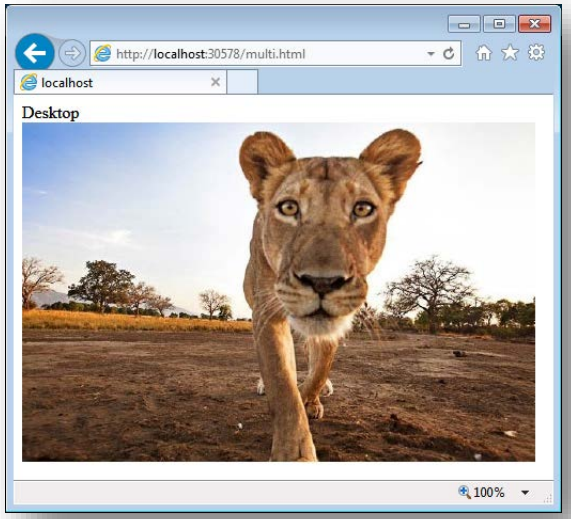
URL	Protocol	Method	Result	Type	Received	Taken
http://localhost:30578/multi.html	HTTP	GET	200	text/html	1.47 KB	62 ms
/Content/Scripts/jquery-1.8.3.min.js	HTTP	GET	200	application/javascript	91.87 KB	94 ms
http://wurfl.io/wurfl.js?debug=true	HTTP	GET	200	text/javascript	1.17 KB	141 ms
http://wit.wurfl.io/http://www.expoware.org/images/cat.jpg	HTTP	GET	200	image/jpeg	86.80 KB	0.71 s

F12 multi.html - F12 Developer Tools

Network

SUMMARY DETAILS

URL	Protocol	Method	Result	Type	Received	Taken
http://localhost:30578/multi.html	HTTP	GET	200	text/html	1.47 KB	15 ms
/Content/Scripts/jquery-1.8.3.min.js	HTTP	GET	200	application/javascript	91.87 KB	94 ms
http://wurfl.io/wurfl.js?debug=true	HTTP	GET	200	text/javascript	1.22 KB	156 ms
http://wit.wurfl.io/http://www.expoware.org/images/cat.jpg	HTTP	GET	200	image/jpeg	52.99 KB	0.59 s



619 x 412 (87 KB) original
480 x 319 (53 KB) resized for smartphones

Same URL
wit.wurfl.io/**http://www.expoware.org/images/cat.jpg**

Summary

- **You can't plan a device-friendly site and …**
  - Just serve markup regardless of the actual device
  - Just focus on features and ignore the actual device
- **Feature vs. Browser**
  - Ill-posed question
  - It's all about knowing form factor and platform details
  - No official standards
- **Real-world Form factor detection**
  - WURFL.JS on the client **free**
  - WURFL and other DDRs on the server **freemium**