Consumer Interaction in Store

Purpose of experiments

“Quantification of perceived web AR application performance”

Research design

Fundamental question “Is the web a viable platform for AR experiences?”

Hypothesis “An AR experience based on web technologies can provide a similar user experience compared to a native solution”

Experiments & metrics
- Time to load, render and provide experience
- Ability and speed to recognize pattern markers

Performance on the web (1)

Components of performance

- Users rapidly leave website when load exceeds ~1 second
- Webpage should not just *load* quickly, but also be functional

(source: https://blog.mozilla.org/metrics/)

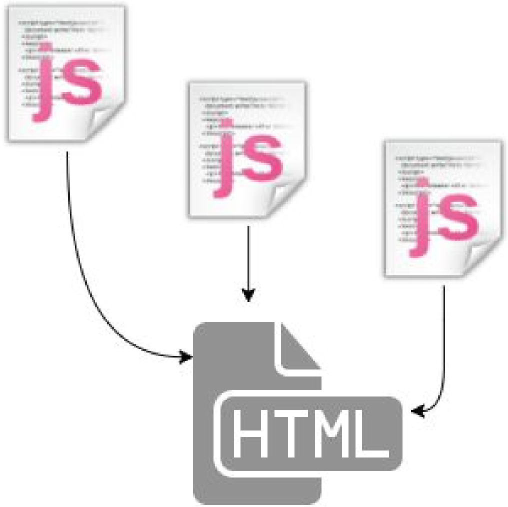
Performance on the web *(2)*

Components of AR application:

- Aframe library (1100 KB)
- AR.js library (932 KB)
- Business logic (~50 KB)
- **Total bundle** (~2000 KB) ... *it's big!*

Comparisons:

- jQuery library (32.5 KB)
- React library (45 KB)



"1 MB downloads in ~5 seconds on fast 3G mobile network"

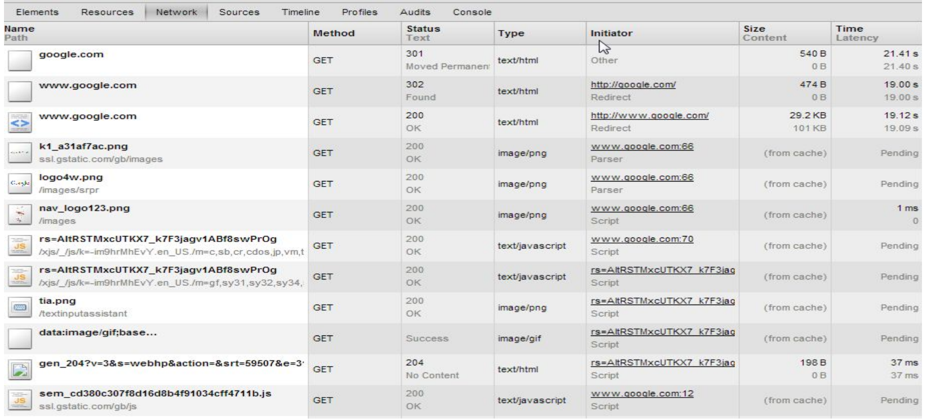
(source: Google Developer Tools)

*All bundles are minified

Performance on the web *(3)*

Implicit caching

Developer tools



Elements | Resources | Network | Sources | Timeline | Profiles | Audits | Console

Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency
google.com	GET	301 Moved Permanen	text/html	Other	540 B 0 B	21.41 s 21.40 s
www.google.com	GET	302 Found	text/html	http://google.com/ Redirect	474 B 0 B	19.00 s 19.00 s
www.google.com	GET	200 OK	text/html	http://www.google.com/ Redirect	29.2 KB 101 KB	19.12 s 19.09 s
k1_a31af7ac.png ssl.gstatic.com/gb/images	GET	200 OK	image/png	www.google.com:66 Parser	(from cache)	Pending
logo4w.png /images/srpr	GET	200 OK	image/png	www.google.com:66 Parser	(from cache)	Pending
nav_logo123.png /images	GET	200 OK	image/png	www.google.com:66 Script	(from cache)	1 ms 0
rs=AItRSTMxcUTKX7_k7F3jagv1ABf8swPrOg /xjs/_/js/k=-im9hrMhEvY.en_US./m=c,sb,cr,cdos,jp,vm,t	GET	200 OK	text/javascript	www.google.com:70 Script	(from cache)	Pending
rs=AItRSTMxcUTKX7_k7F3jagv1ABf8swPrOg /xjs/_/js/k=-im9hrMhEvY.en_US./m=gf,sy31,sy32,sy34,	GET	200 OK	text/javascript	rs=AItRSTMxcUTKX7_k7F3jag Script	(from cache)	Pending
tia.png /textinputassistant	GET	200 OK	image/png	rs=AItRSTMxcUTKX7_k7F3jag Script	(from cache)	Pending
data:image/gif;base...	GET	Success	image/gif	rs=AItRSTMxcUTKX7_k7F3jag Script	(from cache)	Pending
gen_204?v=3&s=webhp&action=&srt=59507&e=3	GET	204 No Content	text/html	rs=AItRSTMxcUTKX7_k7F3jag Script	198 B 0 B	37 ms 37 ms
sem_cd380c307f8d16d8b4f91034cff4711b.js ssl.gstatic.com/gb/js	GET	200 OK	text/javascript	www.google.com:12 Script	(from cache)	Pending

Explicit caching

manifest.json

```
CACHE MANIFEST
index.html
stylesheet.css
images/logo.png
scripts/main.js
http://cdn.example.com/scripts/main.js
```

Performance on the web (4)

ANDROID

iOS

VS

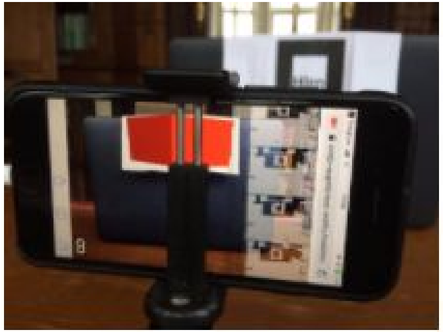
HTML CSS JS

Due to the AR.js bundle size, the initial download is painfully slow (8.3 seconds). Thus still posing a substantial barrier to entry for the end user.

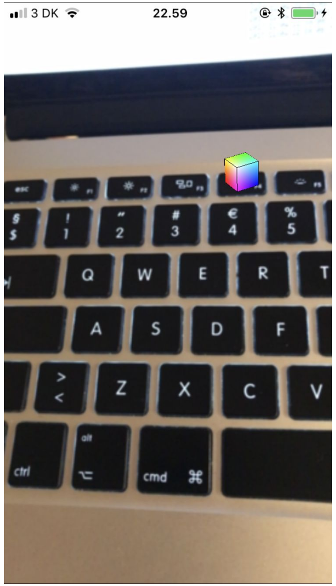
However, caching techniques are available and initial load time is not considered in subsequent experiments.

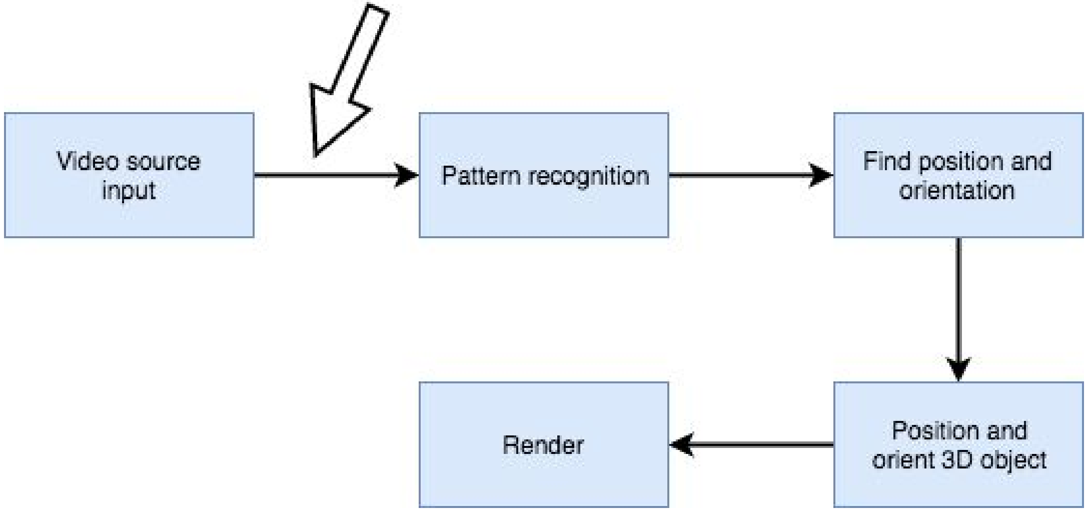
ARToolkit

- Released in 2004
- Open source

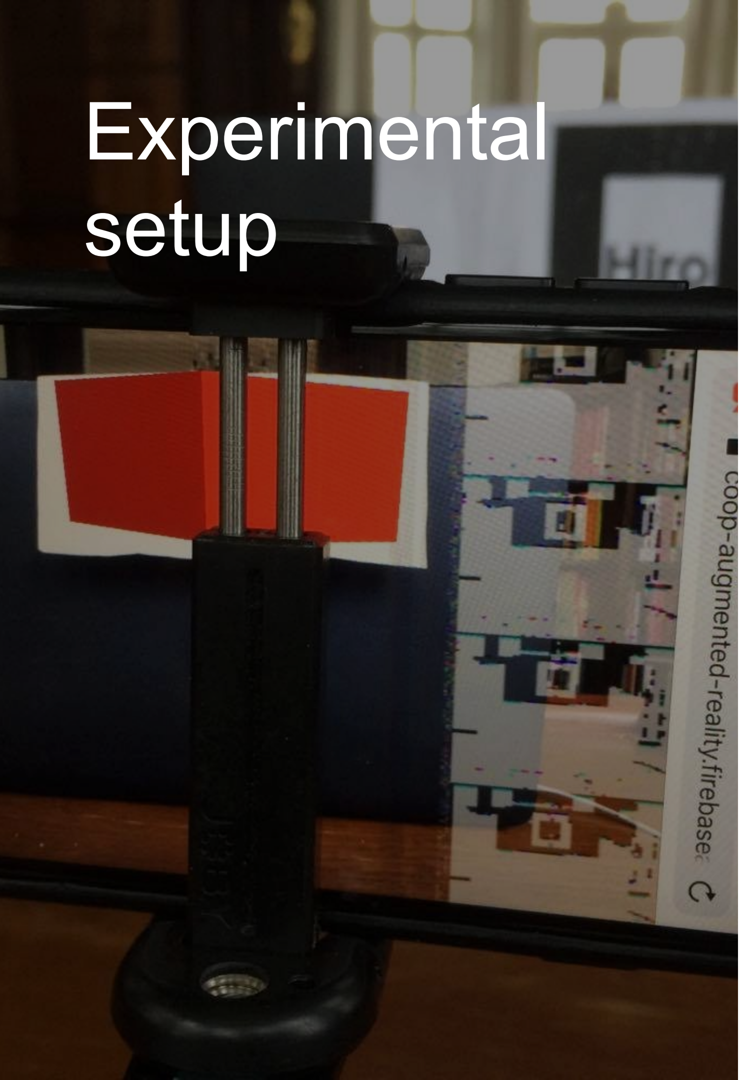


ARToolkit in debug mode rendering the binary black and white image for processing





Experimental setup

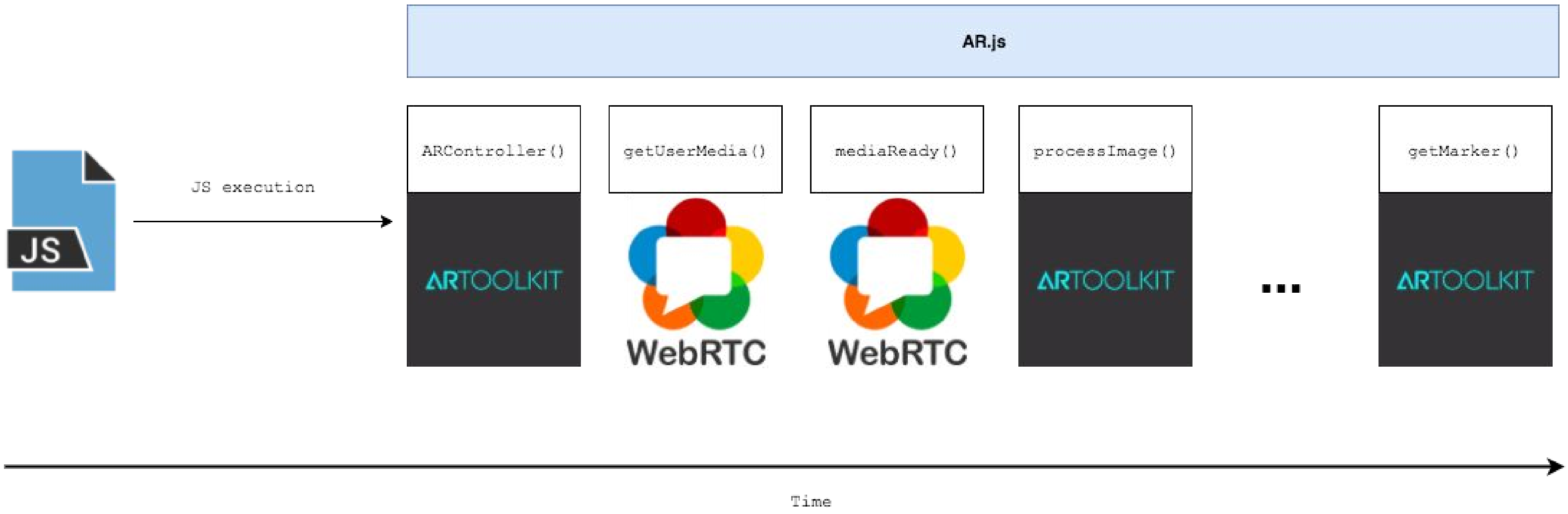


Physical constants:
- Hardware (iPhone 7)
- Daylight lighting conditions
- Distance to marker (25 cm)
- Camera angle (0 degrees tilt)

Software constants:
- Simple 3D rendered cube
- No business logic

Experiment *(1)*

- Sequence of events from load to marker rendering

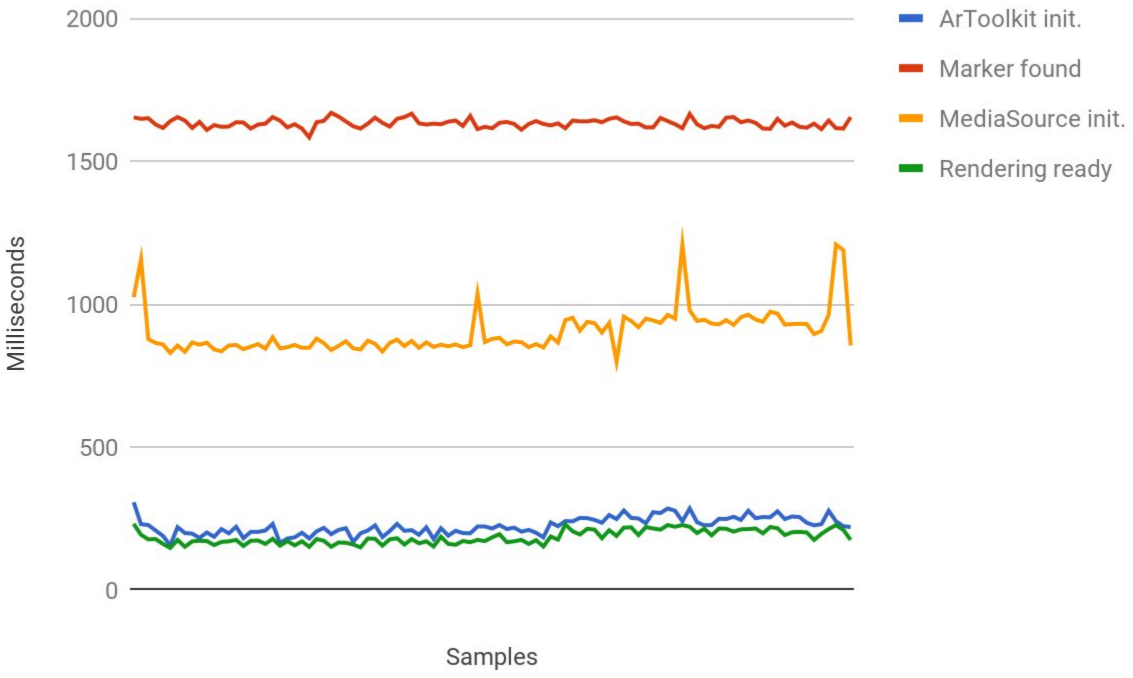


Experiment (2)

Sample size (N) is 100

Relatively smooth data

No substantial outliers

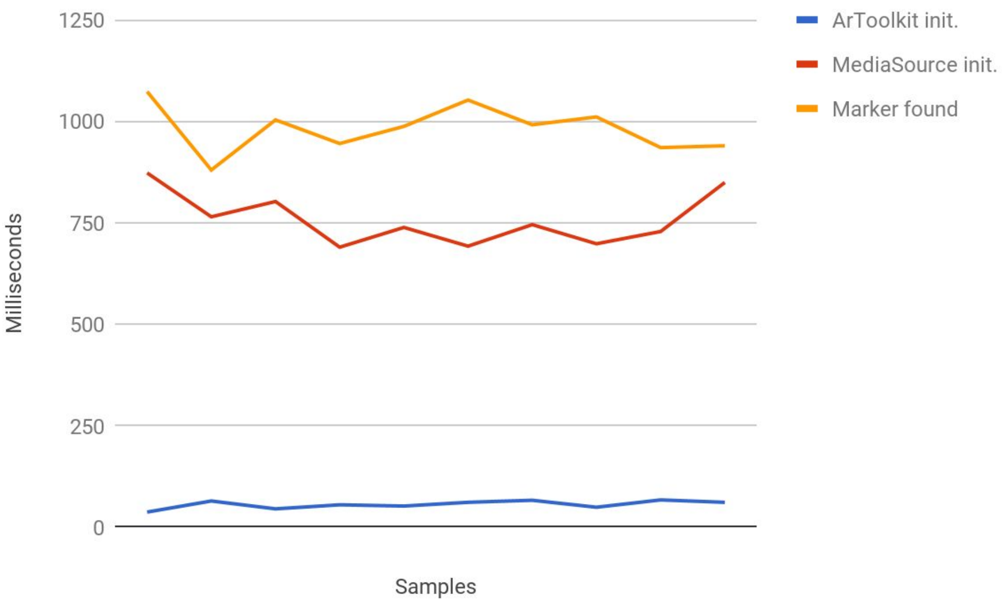


Experiment (3)

- Native implementation on IOS (iPhone 7)

- Sample size (N) is 10 (WIP!)

- Faster on all metrics



Experiment *(4)*

- Video stream to pattern recognized (web) = **831 ms**
- Video stream to pattern recognized (IOS) = **224 ms**

~ *factor 4 difference*

- Cold start to pattern recognized (web) = **1632 ms**
- Cold start to pattern recognized (IOS) = **982 ms**

~ *factor 1.5 difference*

	Web			**IOS**		
	ARToolkit init.	Media init.	Marker found	ARToolkit init.	Media init.	Marker found
Mean	224	801	1632	56	758	982
Standard dev.	30	76	15	10	65	58

Preliminary takeaways

- Web based implementation of ARToolkit is remarkably slower than its native counterpart.

- Media access makes up for a substantial part of time spent (equal for both platforms)

- AR.js makes cross platform development easy by running on the web platform

- AR.js is a young open source project in current development, thus not optimized for performance. A POC.

- It was observed how the native implementation was much more resilient to movement

Further research opportunities

- ARToolkit for JS is currently compiled with Emscripten - maybe WASM would be faster?

- Experiments with other ARToolkit options - fx tweaking the threshold for the black/white image conversion.

- Varying lighting conditions, distances and tilts

- Place more hooks throughout source to get more fine grained data