Adaptive Progressive Web Apps
Progressive Web Apps are just great websites that can behave like native apps.

Progressive Web Apps are just great apps, powered by Web technologies and delivered with Web infrastructure.

Mobile PWA’s
Mobile Web Engagement

Top 1000 mobile apps vs top 1000 mobile websites

- Mobile Web
- Apps

Average minutes per visitor

- Mobile Web: 188.6
- Apps: 9.3

Source: comScore Mobile Metrix, U.S., Age 18+, June 2016
Desktop PWA’s

Source: https://developers.google.com/web/updates/2018/05/dpwa
User Expectations
Design guidelines

Design guidelines for building high quality PWA’s

Start by forgetting everything you know about conventional web design, and instead imagine designing a native app.

Pay attention to detail because native apps have set a precedent for users expectations.
Navigation Patterns

**Web**

- **Twitter**
  - Página inicial
  - pia_mancini @pia_mancini · 42 s
  - Hi @cnk1988 thanks for your donation to @DemocracyEarth opencollective/democracyearth 😊<3

- Kyle Mathews @kylemathews · 7 h
  - Working in the morning in Europe is so nice. So quiet online with everyone in the US asleep.
  - Mostrar mais respostas

- Sunil Pal @threepointone · 39 s
  - Where you peeps staying? I'm heading to the

**Native**

- **Twitter**
  - Página inicial
  - Público @Publico · 35 s
    - Meteorologia. Quatro regiões do continente, Açores e Madeira com risco muito elevado de radiação UV

- Anthony Gore @anthonygore · 4 h
  - if you want to extract critical CSS from your webpack powered app, check out the HTML Critical Webpack plugin. We've just released v2.0.0 which supports Webpack 4
  - github.com/anthonygore/html-critical
  - webpack-plugin

- Francisco J. Marques @FraJ · 30 min
  - Sempre na moda. Estação 18/19
The most important UX feature on mobile is performance

<table>
<thead>
<tr>
<th>Feature</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>The speed it takes to load a page</td>
<td>75%</td>
</tr>
<tr>
<td>How easy it is to find what i’m looking for</td>
<td>66%</td>
</tr>
<tr>
<td>How well the size fits my screen</td>
<td>61%</td>
</tr>
<tr>
<td>How simple it is to use</td>
<td>58%</td>
</tr>
<tr>
<td>How attractive the site looks</td>
<td>24%</td>
</tr>
</tbody>
</table>

Device & Context

On the move things feel slower

% of user that perceived a website to have loaded relatively fast

75% 52%

Sat Down On the move

When you need the information
ASAP things feel slower

Device & Context

Users expectations differ depending on their current context or device

**Mobile**
- Lightweight
- Performant
- Fast

**Desktop**
- Immersive
- Complementary
- Convenient

Source: https://developer.microsoft.com/en-us/events/build/content/modernizing-twitter-for-windows-as-a-pwa
Adaptive
Adaptive

Adapting to user device and context

One codebase, One URL

Adaptive & Responsive

Differentiated & optimal experience based on available information
Adaptation Properties

Static
- Operative System
- Device Memory
- Feature Support

Dynamic
- Quality of connection
- Location
- Available storage
- User preferences
We’ll build an application on top of three concepts:

1. Targeted builds
2. Smart-start
3. Capability reporting
Targeted Builds

- android.js
- ios.js
- desktop.js
- es6.js
- lite.js
const createVariants = require('parallel-webpack').createVariants;
const webpackConfig = require('./webpack.client.config');

const buildVariants = {
  capability: ['modern', 'legacy'],
  platform: ['default', 'ios', 'android'],
};

const createConfig = options => {
  return webpackConfig(options);
};

module.exports = createVariants({}, buildVariants, createConfig);
Targeted Builds

Use cases

Bundle only necessary polyfills based on client browser support

Adjust javascript transpilation

Bundle different components based on build parameters
Smart-Start

Client → Request → Server → Variant 1 → Variant 2 → Variant 3 → Variant 4

Response
app.get('/', async (req, res) => {
  const agent = userAgent.parse(req.headers['user-agent']);
  const capabilities = {
    'browser': agent.family,
    'browserVersion': agent.major,
    'OS': agent.os.family,
    'OSVersion': agent.os.major,
  };
  const capability = capabilityFilter(capabilities);
  const platform = platformFilter(capabilities);

  // Create file path based on negotiation results
  const variant = `${platform}.${capability}.js`;

  res.send(variant);
});
Serve different targeted builds

Adjust API responses
Capability Reporting

Service-worker

Client -> Service Worker

Server

Fallback
"use strict";

// Listen to fetch events
self.addEventListener('fetch', function(event) {
    // Check for network connection speed
    const networkSpeed = navigator.connection.downlinkMax
    // Add extra header to request
    event.respondWith(
        fetch(`${req.url}`),{
            headers: {
                'Content-Type': 'image/svg+xml',
                'downlinkMax': networkSpeed,
            },
        }
    );
});
Capability Reporting

Use cases

Enhance requests with information available on the client

Selectively load images

Provide request fallbacks

Define Request SLA´s
Demo
Adaptive

Further Use cases

Cache different sets of assets depending on the user device storage capacity

Reduce or completely halt 3rd party loading

Disable high dpi image loading if the connection is slow let the user selectively choose which images he wants to see in high resolution
## Browser support

**Service Worker**

Source: [https://caniuse.com/#feat=serviceworkers](https://caniuse.com/#feat=serviceworkers)

<table>
<thead>
<tr>
<th></th>
<th>IE</th>
<th>Edge</th>
<th>Firefox</th>
<th>Chrome</th>
<th>Safari</th>
<th>iOS Safari</th>
<th>Opera Mini</th>
<th>Chrome for Android</th>
<th>UC Browser for Android</th>
<th>Samsung Internet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>11</td>
<td>17</td>
<td>60</td>
<td>66</td>
<td>11.1</td>
<td>11.3</td>
<td>66</td>
<td>all</td>
<td>66</td>
<td>11.8</td>
</tr>
<tr>
<td>Version</td>
<td>18</td>
<td>61</td>
<td>67</td>
<td>68</td>
<td>69</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6.2</td>
</tr>
</tbody>
</table>
Browser support

Network Information

Source: https://caniuse.com/#feat=netinfo
Conclusion
Conclusion

Adaptive PWA’s

Start with a great base PWA and then account for adaptation parameters

Think about multiple devices, platforms, contexts and adaptation strategies

Ultimately it’s all about providing a great user experience.
Thank you

@luisvieira_gmr
luis.vieira@farfetch.com