EUROPE'S LEADING AEM DEVELOPER CONFERENCE

28th – 30th SEPTEMBER 2020

A programmatic approach to Adobe Target

Albert Wognar, diva-e Platforms

adaptTo()

Albert Wognar

Solution Architect @ diva-e
Munich, Germany

albert.wognar@diva-e.com

adaptTo()

Agenda

- Adobe Target in a Nutshell
- VEC vs. “Formbased Activity”
- ttXp Library
- e·on NBA (Next Best Action) Campaign

Adobe Target in a Nutshell

- Target is Adobe‘s Testing & Personalization solution.
- A framework that allows manipulation of digital content (HTML, Mobile, API)

  - Decisioning (Audiences)
  - Definition of Manipulation (Offers)
  - Measurement & Reporting of Goals (Metrics)

Activity
(a.k.a. Experiment)

Adobe Target in a Nutshell

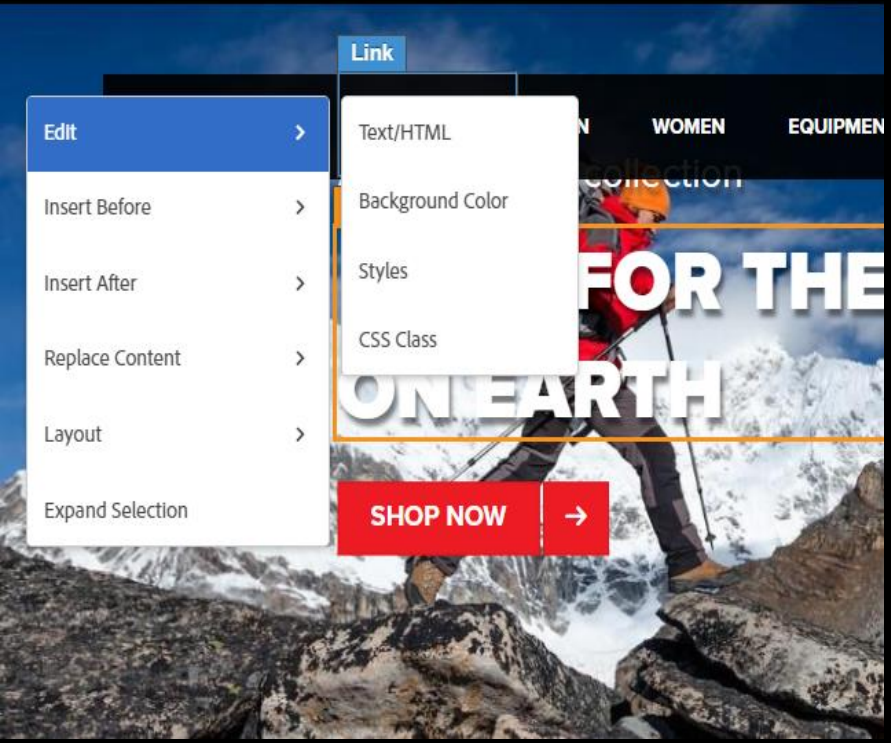
In the HTML Context, the standard approach is client side

- After page load, request to Target server is initiated by Target JS library at.js
- Qualified Offers are delivered to at.js as Response Payload
- Offer is evaluated by at.js.
- DOM manipulation handled by JS
- Metrics monitoring & reporting back to Target handled by JS

Two other modes are

- Server side: Communication with Target and Manipulation ss
- Hybrid (the new hot sh..): Communication with Target ss, Manipulation cs
  https://docs.adobe.com/content/help/en/target/using/implement-target/client-side/functions-overview/targetgobalsettings.html#server-state

Visual Experience Composer (VEC)



- Activity offer is created via Target's WYSIWIG Editor
- Blackboxed JS & CSS instructions
- Activity communication with Target methods handled by VEC / at.js
- Offer code is interpreted & executed directly by at.js

Formbased Experiments

```
if (typeof ttXp === 'object') {
    ttXp.initXp({
        values: {
            id: 'P04',
            name: 'B2C-Heating-DasHaus',
            category: 'Personalisierung',
            variant: 'B',
            selectors: {
                stage: '#ttXp-stage'
            },
            mboxes: {
                remoteOffer: 'P04 - RO',
            }
        },
        init: function () {
            var _this = this,
                values = _this.values,
                remoteOfferMbox = values.mboxes.remoteOffer,
                stageSelector = values.selectors.stage;

            ttXp.getOffer({
                experiment: _this,
                selector: stageSelector,
                mbox: remoteOfferMbox,
                method: 'replaceParent',
                parseOffer: true,
                callbackSuccess: function() { ttXp.flickerCSS.reset(values.id) },
                callbackError: function() { ttXp.abort(_this, 'Could not get offer ' + remoteOfferMbox) }
            });
        },
    });
} else {
    console.log('ttXp not defined, aborting experiment P04!');
}
```

- Activity offer consists only of custom JS & CSS
- All Target method calls, conversion event reporting, selectors, flicker prevention etc. need to be taken care of by DEV.
- at.js will only inject the code in <head>, no further evaluation or execution of offer code by at.js

adaptTo()

versus

Visual Experience Composer (VEC)

Pros

- No / little coding requirements
- Fast learning curve

Cons

- Not-so-robust selectors
- WYSIWIG has its natural limits
- Complex activities out of scope

Formbased Activities

Pros

- Enables vigh compexity of activities (e.g API calls)
- It‘s code: versioning, reviews, best practices, …

Cons

- (initially) bigger effort to create an activity

diva-e ttXp Library

- JS Library to standardize & optimize FB activities
- Fast & lean creation of FB activities
- Robust, reviewable, reusable & refactorable activity code
- Flexible integration of external HTML fragments, SPAs, APIs, etc
- Automatic metrics reporting & flicker prevention
- Automatic Logging & Error Handling
- Multitenant Architecture allowing Client-specific Customizations

adaptTo()

diva-e ttXp Extension

- Each activity is registered as an object within the ttXp namespace
- ttXp.initXp performs the following standard steps
  - Health check : ttXp.selfCheck
  - Flicker Prevention : ttXp.flickerCSS()
  - Provisioning of success & error handling : ttXp.success(), ttXp.error()
  - Other standard operations, e.g. Console Reporting, Google Analytics Calls : ttXp.console()
- Adobe's own target methods getOffer(), trackEvent and applyOffer() are capsuled and extended in matching ttXp methods (e.g. ttXp.getOffer()).

diva-e ttXp Library

```
if (typeof ttXp === 'object') {
    ttXp.initXp({
        values: {
            id: 'P04',
            name: 'B2C-Heating-DasHaus',
            category: 'Personalisierung',
            variant: 'B',
            selectors: {
                stage: '#ttXp-stage'
            },
            mboxes: {
                remoteOffer: 'P04 - RO',
            }
        },
        init: function () {
            var _this = this,
                values = _this.values,
                remoteOfferMbox = values.mboxes.remoteOffer,
                stageSelector = values.selectors.stage;

            ttXp.getOffer({
                experiment: _this,
                selector: stageSelector,
                mbox: remoteOfferMbox,
                method: 'replaceParent',
                parseOffer: true,
                callbackSuccess: function() { ttXp.flickerCSS.reset(values.id) },
                callbackError: function() { ttXp.abort(_this, 'Could not get offer ' + remoteOfferMbox) }
            });
        },
    });
} else {
    console.log('ttXp not defined, aborting experiment P04!');
}
```

e·on NBA Campaign

2nd Place Adobe Summit EMEA 2019 Experience Awards

Next Best Actions

- EON‘s NBA Engine computes the most relevant actions for each of it‘s existing clients. Examples:
  X-Sell, Upsell, SEPA mandate, churn prevention, SmartHome Sales
- Decision algorithm is based on multiple sources & recalculated daily
- EON also wanted to use these decisions in the online web context
- Solution should be highly flexible & as automated as possible.

Next Best Actions – Specsheet (exc)

1. Make E.ON NBA Engine's calculated accessible for Target's audiencing engine
2. Enable real-time configuration of activity outside of Target
   - Activation , Pausing, Deletion of NBA Campaigns
   - Definition of multiple output formats (e.g. Hero, Section, Pop-up)
   - Flexible NBA allocation : Not every NBA / output format fits every target page
   - Flexible content allocation: which HTML fragment belongs to which NBA / output format
   - Freely scalable number of NBAs, Output Formats & target pages
3. Creation & provisioning of content (HTML Fragments) by AEM (6.3)

Next Best Actions – Requirement 1

Make pseudonymized customer data from E.ON NBA Engine accessible for Target's audiencing engine

- Daily Ingestion of calculated NBAs as Customer Attributes (Adobe People Core Service)
  https://docs.adobe.com/content/help/en/target/using/audiences/visitor-profiles/working-with-customer-attributes.html
- During Audiencing, Customer attributes are evaluated and matched to hashed Onsite Customer ID cookie, which is passed to Target as mbox3rdPartyID parameter.
  https://docs.adobe.com/content/help/en/target/using/audiences/visitor-profiles/3rd-party-id.html
- CA information is preprocessed by Target User Profile Script and saved to user profile.
  https://docs.adobe.com/content/help/en/target/using/audiences/visitor-profiles/profile-parameters.html
- NBA User Profile value is evaluated programmatically by ttXp offer code through Target's dynamic attributes syntax ${user.nba}
  https://docs.adobe.com/content/help/en/target/using/experiences/offers/passing-profile-attributes-to-the-html-offer.html

Next Best Actions – Requirement 2

Enable real-time configuration of activity outside of Target

- Target ttXp offer does not contain any configuration or content information, only execution code (Model)
- Content (HTML Fragments) is created & provisioned by AEM (View)
- Activity configuration is defined in separate JSON (Control)
  The document is maintained & stored by client.

1. On activity execution, NBA Config JSON is loaded via XHR by ttXP Offer.
2. Offer evaluates NBA Config to find most relevant NBA / output format
3. Offer retrieves matching HTML Fragment URI from NBA Config
4. Offer fetches HTML Fragment via 2nd XHR call and injects it into DOM.

Next Best Actions – Requirement 3

Content creation & provisioning by AEM

First thought: Experience Fragments. But:

- In AEM 6.3, XF were not really usable for Target if you had a high numbers of fragments due to being unsortable (Fixed with AEM 6.5.2. Thank you, p!v)

Alternative solution:

- Direct ttXp XHR call on Fragment JCR Node, e.g. /content/de/.../_jcr_content/par/targetwrapper.html
- Small custom AEM parsys component to wrap multiple components, expose JCR node, & add some valuable data attributes.

JCR Wrapper Component

```
<sly data-sly-test="${(wcmmode.edit|| wcmmode.design)}" data-sly-unwrap>
    <h4>${properties.id}</h4>
    JCR Content Path: ${currentNode.path}
</sly>
<ttxpremoteoffer
    data-ttxp-remoteoffer-id='${properties.id}'
    data-ttxp-jcrpath="${currentNode.path}"
    data-ttxp-remoteoffer-variant='${properties.variant}'
    data-ttxp-remoteoffer-misc='${properties.misc}'
    class='${properties.class}'>
    <sly data-sly-resource="${ @path=' par',  resourceType='foundation/components/parsys'}" data-sly-unwrap>
    </sly>
</ttxpremoteoffer>
```

Next Best Actions – a few numbers

32

different NBA Campaigns:
XSell, SmartHome, SEPA Direct Debit Mandate, …

3

Layout-Types:
Hero Stage, Grid Section, Modal (PopUp)

1

Target Activity

= 94

live NBA Experiences

1-3

different ContentVariants

9

qualified Webpages

adaptTo()

… and some more numbers

20%

Smart Home Sales were generated by NBA Campaign

400%

Higher Clickthrough Rates compared to Generic Teasers.

2.5

Individual Messages per Session

>5 Mio

NBA Datasets calculated on a daily basis

that’s’s it

Thank you for your time & interest!