DNN

2014 CONNECT

Amazing 1hour-Apps

Create an amazing App

in less than 1 hour

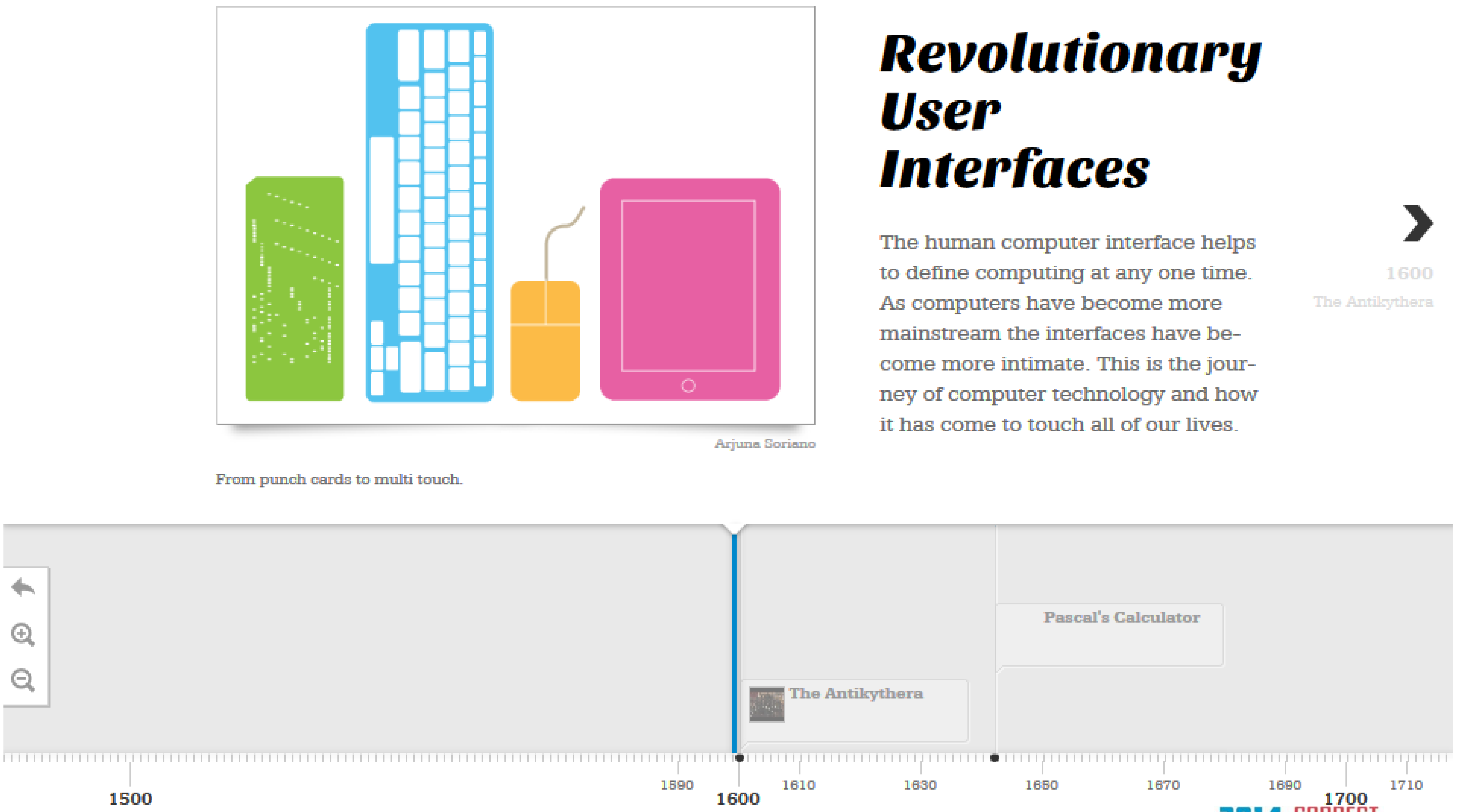
We believe in low hanging fruits

Thousands of amazing visuals & scripts exist…

…to keep up, we don't need to re-invent the wheel…

…we only need to integrate them into DNN

So let's create a DNN TimelineJS



Revolutionary User Interfaces

The human computer interface helps to define computing at any one time. As computers have become more mainstream the interfaces have become more intimate. This is the journey of computer technology and how it has come to touch all of our lives.

Arjuna Soriano

From punch cards to multi touch.

…but better

1. Timeline scripts exist – "free lunch"
    1. Responsive
    2. Image, Text, Video, Google Map, etc.
2. Data functionality exists – "free lunch"
    1. Data input, storage, versioning
    2. Multi-language
    3. Data output as JSON
    4. DNN-Search integrated
3. All we have to do is combine this

Presented by 2sic (creators of 2sxc)

18 DNN Pros
loving web technologies
since 1999

2SIC

Let’s work!

DNN
2014 CONNECT

Materials and Technology

All Open Source Software

- TimelineJS 2.3+ (MPL 2.0)
  - http://timeline.knightlab.com/
  - https://github.com/NUKnightLab/TimelineJS
- DNN 7.2+ (MIT)
  - http://dnnsoftware.com
  - https://dotnetnuke.codeplex.com/
- 2sxc / 2SexyContent 6+ (MIT)
  - http://2sexycontent.org/
  - https://github.com/2sic/2SexyContent

Technologies

1. DNN & 2sxc Out-Of-The-Box
2. A bit of HTML
3. A tiny bit of ASP.net Razor
4. JSON for data-transport
5. Some JavaScript for attaching the default 2sxc-JSON-format to the Timeline JS

The few steps

Easy and amazingly quick

In DNN / 2SexyContent

1. Create App
2. Configure Content-Types based on what the script needs
3. Enable JSON data-publishing in code (so it will always work)

In HTML / JavaScript

1. Integrate TimelineJS
2. Setup JSON-call to get data
3. Tie up data to the TimelineJS
4. Add some edit-buttons for the full user experience

We need the following data

Timeline Entries

1. Headline (text)
2. Start Date & End Date (dates)
3. Body (text, formatted)
4. Media (image, video, tweet, address, …)
5. Credit, Caption (texts)

Minimal Configuration (optional)

1. Language (many exist)
2. Where to start (beginning/end)
3. Start Zoom

Live work

Continue with the View

1. Create the Razor view-file
    1. set types for content, header, configuration
2. Add records of data in list-management
3. Integrate the TimelineJS with Client-Dependency
4. Integrate the 2sxc-controller with Client-Dependency
5. Activate data-publishing

Live work

Connect JSON to TimelineJS

1. Write the bit of JS code to retrieve data
2. Write the callback to re-map the data
3. Attach remapped data to Timeline JS

Live work

2014 CONNECT

Temporary Summary

Status Quo

- Data editing works
- Output works

- Could be even better
  → editing from the inside the timeline would be sexy
  - Requires Edit-Button
  - Requires Add-After-Button on every record

Add inline-editing

1. Use the 2sxc-controller to provide the buttons
    1. that’s it ☺

Live work

2SIC

Let’s Package & Distribute

DNN 2014 CONNECT

Simple steps

1. Create Package
2. Install in another portal to test
3. done ☺

Live work

Questions

SEO and JavaScript Apps

1. Either output the relevant data as HTML, then hide it when the JS loads
2. Or output it as HTML, then convert it to a JS-object when it loads (avoids re-loading in JSON but more work)
3. Or use the Hashbang-Notation #!
    1. You can even use this *without* sub-pages, because the root-Hashbang is also allowed <meta name="fragment" content="!">
4. Pushstate may work one day…

AngularJS vs. knockoutJS

1. Both are great
2. AngularJS is more powerfull, more feature-rich, better developed, has more support
3. knockoutJS is easier, simpler, less pattern-development-style

Start with either, when things get complex (many views…) you'll probably migrate to AngularJS (we're currently using KO but moving to Angular)

App-Catalog

- Goal is to provide many free Apps
- The ecosystem needs free-stuff
- maybe we’ll add a payment option, but that’s low priority, especially because most work is meant for re-use (MIT-style)

Content vs. App

- Content is the bread-and-butter, the ersatz-wysiwyg
  - Simple texts, images, headlines, various arrangements, etc.
- Apps are “sealed” functionalities
  - add/remove without side-effects
  - own views, own routes, own data
  - own configuration, ml-resources, etc.
  - by default, they cannot access data from other apps

Ideas for Apps

1. Image galleries
2. Books like http://www.turnjs.com
3. Interactive FAQs with searches and more
4. Floor-Plans
5. Tab-Style presentations
6. Interactive Animations
7. Mapping-Tools with various configurations
8. Configure JSs like Feed-readers…
9. GTM-Integration…

Enable JSON by code

```
@functions
{
public override void CustomizeData()
{
      // enable publishing
      Data.Publish.Enabled = true;
      Data.Publish.Streams =
      "Default,UIResources";
}
}
```