college CDI

Interactive contents 2 (C12)

Lesson plan

Class 01

Revision : transition + transform

(See Course notes : C11 /Class 03)

- CSS : Transitions
- CSS : Transform
- CSS : Transition/transform on nested containers

- Triggering transition on :hover (CSS)
- Triggering transition on :target (CSS)

- Triggering transitions on events (JavaScript)
    - onclick / onDoubleClick
    - onMouseOver / onMouseOut
    - onScroll...

- Media queries (width/orientation)
- Media query showing burger-icon/text navigation (on page width)

- Drop-down menus (CSS)
- Burger showing menu on :target
- Complex drop-down menus (CSS + JavaScript)

Assignment 01

Class 02

Revision

(see Course notes : C11 /Class 04-05-09-10-11-12-13)

- CSS : basic animation (keyframe from:/to: + %)
- CSS : Trigerring animations (:hover / : target)

- Creating animating masks + clippings + Paths + SVG
- <canvas> drawing + animations
- SMIL animation <animate>

- Triggering animations on events (JavaScript)

Online resources

- CSS animations :
  http://animista.net/

- HTML5 animated banners
  https://www.bannersnack.com/animated-banners.html (7$/mois for pro actions)

- Bounce.js
  http://bouncejs.com/

- Animate.css
  https://daneden.github.io/animate.css/

- Animejs
  https://animejs.com/

- Magic Animation
  https://www.minimamente.com/project/magic/

- CSSshake
  http://elrumordelaluz.github.io/csshake/#1

- Hover.css
  http://ianlunn.github.io/Hover/

- AnisJS
  http://anijs.github.io/

Assignment 02

Class 03

Revision

(see Course notes : C11 /Class 08)

- <picture> tag
- <audio> tag
- <video> tag
- Preloading elements

- CSS progression bars with JavaScript

- Triggering actions onscroll

    - ScrollMagic
      https://scrollmagic.io/

    - AOS : Animate on scroll library
      https://michalsnik.github.io/aos/

Assignment 03

Class 04
Lightbox and pop-up elements

When should lightbox be used

- Makes it possible to grab visitors' attention + focus on the marketing message
- Allow visitors only two options : taking action or closing the popup

- Perfect to share important announcements, onboard new users, promote big discounts, share a message on visitors' exit, to confirm important actions, etc.

Ask yourself the following questions :

- Does message worth interrupting my visitor's navigation?
- Does the benefit for the user justifies that interruption?
- As a visitor, would find that popup intrusive?

Important
Ttoo many Pop-ups can easily have a negative impact on user experience.

Good reasons to use pop-ups

- Pop-ups are attention-grabbing
  + offer short punchy messages

- Draw the eyes to the important content
  + offer extra value to users and they know it
  = When a pop-up appears, the users' attention is immediately drawn to the offer

- Pop-ups are versatile
  = no longer send users out of the browser window + don't fill-up the window or with unwanted ads.
  also possible to trigger the pop-ups in many different ways :
  - Upon loading or unloading a page
  - After a certain point of scrolling
  - Upon different events and/or actions, delay

- Pop-ups keep the site clean
  = minimalism is important = pop-ups keep from having to integrate the offers into the actual page's content

- They Increase Conversions have the potential to convert at a rate of about 3% to 9% ->for those very well designed

- They increase engagement = convince users to fill out a survey, share something on social media, watch a video on a landing page...

Designing the popup

Content

- Pop-ups interrupt the users => keep the content short, relevant and correctly laid out

Overlay color and opacity

- Pop-ups darken the website content (semi transparent dark gray or black overlay)
- Advanced popup editors = customize the overlay's color and opacity level
- Regarding the opacity, *UX for the Masses* summed it up perfectly:

  "Too dark and users will no longer be able to see which page they're on. Too light and users might think that the page is still active and might not even notice the overlay in the first place."

Closing options

- Can be a closing X (highly visible) or a dismiss link
- If the closing option is too visible = visitors may click on them immediately without reading the message
- If it's too discreet and difficult to find, it might create frustrations.

Terminology

- Alert / pop-up
- Flash notice / growl communication
- Modal
- Lightbox theater
- ¨Popover/Tooltip/Hovercard

Fancybox library

- Tool to create lightbox elements
- Open source but licence required for commercial use
- Can be linked statically or with CDN
- What is a content delivery system (CDN)

Hands on project

Assignment 04 : Create a fancybox

Class 05
JavaScript libraries

Carousels and sliders element

- Make it possible to show high volume of content within one page
- Can be difficult to program = why reinvent the wheel = use libraries
- Sliders = to make it possible to navigate from one content to another, slide by slide

When to use sliders & carousels

- To segment important volume of content
- Step by step demonstrations
- Products tours
- Photo gallery, portfolio, etc.
- They can be more than a slide show = can be powerful marketing tools

To use or not to use sliders & carousels

- Nor everyone agrees on its efficiency
- 50% think they should be abandoned
- For sure : they shouldn't be overused
- Studies show 50% of users don't check them and find them annoying
- Sliders = prone to banner blindness (explain banner blindness)
  (show and explain heat maps)
- Accessibility issues (poor contrast of dots, etc.)
- Loading time and mobile optimization
- Impact of the slide show images on the perception of the website / brand
  Sorting images is important = connotation can impact the way the site is considered

What's the difference between a framework and a library?

- Framework : inverts the program flow
  It leads and leave space for you personalization

- Library : You lead the work flow
  You call the library where and when needed

- Popular libraries to discover (see course notes #4)

Implementing a lightbox

- FANCY BOX
  http://fancyapps.com/fancybox/3/

- Setting + implementing
    - Photos
    - Videos
    - Hidden HMTL
    - Open/close overlay animations
    - Modal windows
    - iframe based content
    - Google maps

Assignment 04

Create a basic HTML page and implement a lightbox carousel.

Class 06
Carousels and sliders (suite)

- Carousels and sliders elements = very popular features on the Internet

- Don't always (or rarely according to some people) constitute the best solution
  BUT every front-end developer needs to be able to build sliders and to use elements from libraries
  TO satisfy the clients AND to save a lot of time

Owl carousel

- One of the most popular choices
- Can be called a library (it is one), or just a jQuery plugin since it depends on jQuery to work
- On its second version, received many uploads and a brand new documentation website

Main characteristics:

- Fully customizable
- It contains touch and drag support (designed specially to boost mobile browsing experience)
- It is fully responsive
- It has fall-backs for many old browser limitations

Owl carousel's full documentation :
https://owlcarousel2.github.io/OwlCarousel2/docs/started-welcome.html

Owl carousel in action!

- Basic installation and application of the carousel

- OWL comes with a big pack of files including the *Grunt task runner*, *json* elements
  and even *SASS* multiple and separated blocs

- Depending on the work environment, these files may be very useful
  BUT we will keep it simple for this demonstrations.

STEP 01 : CSS Installation

- Start with a HTML structure (include the usual basic and custom CSS file in the <head> section)

- Then, include the OWL carousel's CSS in the <head> section (*theme* file is optional)

- **REMEMBER** = ***owl.carousel.css*** is required and should be included before any ****.js*** files.
  CND ***links*** can be used but you can also download the files and keep them within the project's folder

STEP 02 : Installation of JS, JQUERY and OWL.JS

- jQuery, Owl.js and Custom Javascript files =
  must be included in the file, in that exact order, before the closing body tag </body>.

- JavaScript custom file is optional (and the best practice)
  for this lesson, we keep the script parameters at the end of the html codes, right after the ***owl.js***
  and the custom javascript links (CDN links will here be used again).

- Now a small test = add some text in the HTML file
  + check with a browser's inspector if any error message shows in the console

- At this point of the installation, if any error
  = probably a mistake with your links, misspelled word or missing quotes

- Google Chrome recommended

STEP 03 : Setting HTML base

- Start with a very basic skeleton of our slide without specific positioning styles

- Avoid at this point =
  adding positions attribute (absolute, flex or overflows)... could cause conflicts

- In the example (page 4), a * * work as a target for the library

- The container may be targeted using a class or an id (which is a better practice)
  Every direct children of this container is considered as a *slide* by the library

- An extra class can also be used to point at the slider for eventual custom styles.

- Very basic CSS style was applied to this example (view style.css file)

- Every inside the owl-slider container div received a different background color
  (to help visualize them better during the development process)

STEP 04 :Initializing OWL.JS

- To put the library into action = must be initialized.

- First line of the code = jQuery function that will run the inner code once the document is ready

- Second line = container (div) of the slider is selected, and the *owlCarousel()* function is initialized
  (No parameters applied to this example yet, but the carousel is ready to be customized)
  nav buttons are showing

STEP 05 : Mandatory and optional CSS classes

- A specific class (*owl-carousel*) is required in the container tag
  Without this class, the slider won't behave as expected.

- For the example = *owl-theme* CSS file (optional) is used
  = added to the container tag

STEP 06 : Applying some basic parameters via script

- Basic carousel has been created = basic parameters can be applied to it

- Most of the slider's characteristics can (or must) be applied on its script
  using simple and very descriptive parameters

- Every parameter comes with a *default value* (can be changed with JavaScript)

- For instance, we can :

    - Define margin between slides
    - decide how many slides will appear at the page at the same time
    - make it possible to loop infinitely

- To apply an image inside each slide and changing its text : very easy.

Sources & references :
https://owlcarousel2.github.io/OwlCarousel2/

Assignment 3 : Create a slider

Parameters to be defined.

Class 07

Revision

Workshop

Class 08

Mid-term exam

Class 09

On scroll animations

Important concept: device pixels vs CSS pixels.

- Device pixels correspond to the formal resolution of the device used (screen.width/height)
- An element of 128px width on a 1024px wide monitor (maximize the browser screen)
  = element would roughly fit eight times in the monitor
- HOWEVER, screen is zoomed at 200%,the element would now only fit four times
  BECAUSE zooming = stretching up pixels (simply doubles the pixels size)
- Zooming to 200% makes one CSS pixel grow to four times the size of one device pixels
  (two times the width and two times the height, so four times in total)
- Important to keep this in mind when thinking about scrolling
  scrolling = calculated by the vertical distance between the target element
  and the desired distance point from the top or the bottom of the visible part of the screen

Important concept: the viewport and its importance

- Viewport = normally represents a rectangular computer graphics viewing area
- For a web browser = the window in which is viewed a web document
  (or the entire screen when in full screen mode)
- Content outside the viewport is not visible on screen until scrolled into view
  Currently visible portion of a viewport =*visual viewport*

Important concept: scrolling offset

- Building an animations which plays when visible to the user
  = based the distance between the top of the target element and the top of the current visible portion of the viewport
- In javascript = *window.pageXOffset* and *window.pageYOffset*,
  contain the horizontal and vertical scrolling offsets of the document
  Makes it possible to know how much the user has scrolled (in CSS pixels)
  BUT you want to know how much of the document has already been scrolled up, whatever zoom state it's in
- In theory, if a page is scrolled up and then zoomed in, *window.pageX/YOffset* will change
  BUT browsers tend to keep the same element at the top of the visible page when a page is zoomed.
  SO in practice, the number of CSS pixels scrolled out remains roughly the same

To use or not to use on scroll animations?

- Web animation = not only nice gadgets
  = powerful tools increasing users engagement and participate to user experience

- Important not to abuse.

Overuse of animation

- Animation needs to have a purpose
- Should not get in the users way + take precious time away
- The interface is not entertainment

Finding the Sweet Spot

- Tricky aspect = to determine the duration of the action

- Too fast and the animation will be abrupt and shocking
  Too slow and you risk to bore or frustrate the users

- Jakob Nielson's (seminal usability study on the interaction between humans and computers):
  - Anything shorter than 0.1 second (100 milliseconds) won't be perceptible to most people
  - At 1 second, users' attention is maintained
  - After 10 seconds, most users become disconnected or bored

AOS.JS step-by-step

- Animate On Scroll (AOS.JS) = JavaScript library containing many customizable effects
  (fade transitions and flip, slide and zoom animations)
  Documentation: https://css-tricks.com/aos-css-driven-scroll-animation-library/

STEP 1: Creating the HTML structure

- Create a very simple page containing multiple small containers on each side of a main container
  (They will be placeholders for animation elements)

STEP 2: Installing CSS and JS library

- CSS and the JS code of the library are now added (CDN link in the example)
  But downloading the files into your project folder might be a better solution.

- **CSS:** In the <head> section, BEFORE the custom css file.
  JavaScript: Before the closing </body>

- No jQuery required.

STEP 3: Initializing the library and using its parameters via HTML data-aos attributes

- After adding the JS file, apply the following code :

```
<script>
AOS.init();
</script>
```

- To apply animations from this library = add default values them to the attribute ***data-aos***

  Example : <div ***data-aos="fade-down***">My Animation

STEP 4 (Optional): Pass / check default settings object

- Parameters can also be passed to the function as an object
  Useful to pass the same parameters to multiple animations
  without needing to repeat multiple attributes directly into the HTML elements.

Class 10
Responsive tables

The importance of tables

- Early ages of internet = useful to help layout contents at the early age of Internet
- BUT Data tables have been used since the beginning of web development and the structure hasn't changed
- Even though the structure stayed unchanged, there has been a great evolution in styling
- Data is becoming the raw material of the global economy
- Data is meaningless if it cannot be visualize or if it is impossible to interact with it
- Enterprises that will survive the next decade won't only have superior data BUT ALSO superior user experience.
- Good user interface design is based on human goals and behaviour
  On its turn, user interface affects the users behaviour
  which influences the interfaces design decisions
- In a very subtle and unconscious ways =
  user experience modifies the way humans make decisions and influence their actions

The pertinence of using tables

- Standard table markup makes semantic sense
- Accessibility = native table markup helps users browsers understand the data structure
- In an interesting article which can be found at the following URL, some of the author's various tests are explained :
  https://css-tricks.com/accessible-simple-responsive-tables

See course notes class 10

To use or not to use tables?

- No definitive answer to this question
  But for sure, the table markup offers specific tags which will be useful

The complexity of responsive tables

- Modern web development and the many devices in use
  = viewports and screens of many different sizes

- Responsive design is a must
  Front-end developer needs to develop this skill.

- One of the most complex aspect of responsive web design
  = how format complex tabular data for display on smaller screens

- Many of different ways to handle tables in responsive designs
  BUT content should always dictate the best solution

- Short list of things to be considered :

    - What is the meaning of the table?
    - What is the essential information?
    - What information do people care about the most?
    - Which columns are most critical to understanding?
    - Does all the content needs to be displayed in one screen?
    -

Building a simple responsive table

- Many resources are available on the Internet (including free copyright plugins)

- *Basictable* from *Jerry Low* (https://github.com/jerrylow) is used in the example

- Some others to consider would be :

    - *Stacktable.js*
    - *responsive-table* (by *Scott Vinkle*)
    - *Bootstrap* has also a solution for responsive table, only applying to horizontal scroll.

STEP 1 : HTML and CSS

- Build a simple table structure consisting in a <thead> with the column information
  (each row representing a person).

STEP 2 : Adding the library's CSS and JavaScript

- CSS is linked in the <head> section of the HTML document before the custom CSS
- JavaScript is placed before the ending </body> tag
  (after the jQuery and before the custom JavaScript)

STEP 3 : Initializing the plugin or library

- An ***id="table"*** added to the <table> tag
  (will be used to target table element which will be manipulated by the library's javascript)
- This code will apply all the default parameters, and some of them can be personalized.

```
<script>
$('table').basictable();
</script>
```

- See example desktop/mobile

STEP 4 (Optional) : Additional parameters or manipulation

- Basictable plugin is not a very large library
  SO number of parameters it is possible to change is limited

- Here are two of the most useful parameters to customize
  (Remember to always include a coma between each parameter line) :

```
<script>
$('table').basictable({
        breakpoint: 1024,
        showEmptyCells: true
 );
</script>
```

breakpoint :
Defines the point when the table engages in responsive mode
Default : 568px.

showEmptyCells :
Boolean value. When true, empty cells will be shown.

Assignment 5 : Responsive table

Parameters to be defined.

Class 11

Social media share button

- A good user experience includes = sharing content to social media
- A share = your content is worthy (cost effective way to bring in good targeted traffic)
- Research shows that direct messaging = great opportunity to gain new customers
- Brings relevant traffic
  Shares are re-shares on social medias make the ripple effect widens sometimes very rapidly (viral)
- Place social share buttons at the top and bottom of the content:
    - Careful readers only share content after reading it thoroughly
      (natural reaction is to click share buttons at the bottom)
    - Scanners and skimmers types of people (more impulsive) may be ready to share content based on the title, a few sentences, and the reputation of the writer and/or website
      (Top-positioned share buttons are ideal then0
- Don't display social share buttons on every website page.
- Display social share buttons on sharable website pages.
- Display the correct share buttons.
- The importance of including the "EMAIL SHARE BUTTON":
    - Can be efficiently personalized
    - Generates more referral traffic
    - Encourages more email shares

JS-SOCIALS.js

Step 1 : installing necessary file in <head> section

```
<!-- FONT-AWESOME -->
<link rel="stylesheet" type="text/css" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.css">

<!-- JS-SOCIALS -->
<link type="text/css" rel="stylesheet" href="https://cdn.jsdelivr.net/jquery.jssocials/1.4.0/jssocials.css" />

<!-- JS-SOCIALS : FLAT THEME -->
<link type="text/css" rel="stylesheet" href="https://cdn.jsdelivr.net/jquery.jssocials/1.4.0/jssocials-theme-flat.css" />

<!-- CUSTOM CSS -->
<link rel="stylesheet" type="text/css" href="css/style.css">
```

Available themes :

jssocials-theme-flat.css
(flat theme)

jssocials-theme-classic.css
(classical theme with raised buttons)

jssocials-theme-minima.css
(minimalistic theme with logos instead of buttons)

jssocials-theme-plain.css
(Monochromatic)

Step 2 : Installing the necessary Javascript Files

```
<script>
$("#share").jsSocials({

});
</script>
```

Step 3 : Initialize the library and apply a simple parameter (argument)

```
<div id="share"></div>

        <!-- jQUERY -->
        <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.4.1/jquery.js"></script>

        <!-- JS-Socials JS -->
        <script type="text/javascript" src="https://cdn.jsdelivr.net/jquery.jssocials/1.4.0/jssocials.min.js"></script>

        <!-- Our Custom JS -->
        <script>
        $("#share").jsSocials({
                shareIn: "popup",
                showLabel: false,
                showCount: false,
                shares: [
                        "email",
                        "twitter",
                        "facebook",
                        "googleplus",
                        "linkedin",
                        "pinterest",
                        "stumbleupon",
                        "whatsapp"
                ]
        });
        </script>
```

Other useful share buttons libraries

https://sharethis.com
https://www.addthis.com/get/share/
http://social-likes.js.org/
https://github.com/enjoyiacm/goodshare.js
https://simplesharingbuttons.com/#intro
http://sharrre.com/
http://socialitejs.com/
http://share42.com/
https://simplesharebuttons.com/
http://sharebuttons.com/
https://warfareplugins.com/

Class 12

The importance of interactive contents

- Interactive content isn't only a technical or programming thing
  it also include all the marketing aspects, for instance

- *State of Inbound* report listed « interactive content creation »
  as a top-five priority for marketing teams in North America
  (38% of respondents considering it as a priority for the coming year)

- Interactive content ranked even higher than marketing automation (37%),
  webinars (21%), and product how-to videos (16%).

- Many people (especially millennials) prefer to engage with interactive content
  rather than simply read a post or watch a webinar

- 45% of people said that interactive content was a top-three preferred content type

- Can bring is lead generation = Users more likely to share their information in order to use the content,
  or see the results of the actions they took to use it

- Is a strategic way to engage with users = can improve the experience in an industry or with products that may
  usually feel boring + This can result in reactivated or new leads

- Remember it's more important to provide visitors with quality content that encourages them to stay on your page.

Types of interactive contents

Calculators

Interactive calculators take numeric information from the users and convert it into data it can be used to guide or support the users with a particular initiative.

Examples :

ROI Calculators :
Help users see the value of your product or service.

Cost Calculators :
Outline how much a product or service will cost, (tuition, salary, mortgage...)

Revenue Generation Calculators :
Keep track of how much revenue can be gained by comparing costs, profit, and time.

Time Saving Calculators :
Help your users see how much time they can save by using your product or service with this type of calculator.

Assessments

Assessments are usually in the form of a quiz (series of questions ending with a result).

Personality Tests :

- Tell the users what type of personality they have
  (after they have answered a series of personal preference questions)

Quizzes :

- Quizzes congratulate users for the high numbers of good answers provided
- Can help your users identify gaps in their knowledge
- You can educate users through the process.

Polls and Surveys :

- Best way to get to know users preferences about a specific subject
- Services like *Hotjar* enable you to add polls to website pages and emails

Interactive white papers, infographics, and eBooks :

- Interactive white papers = excellent investment
  (especially for older pieces of content that's already performing well)
- Incorporating a quiz, poll, survey, or calculator within the downloadable content
  (take the content to the next level)
- Static content can become tedious to read
  interactive content keeps the reader engaged for much longer

Brackets and Contests :

- All sorts of contests are great ways to ignite fresh engagement
- Brackets place users against each other and track their progress in real-time
  (The two oponnents are shown side by side, and each participant goes head to head)
  The winner advances to the next round, while the other is eliminated
  Each match winner advances again and again until there is only one winner left

Interactive Video :

- Uses the power of visual footage to enhance the user-experience
- Traditional video = unidirectional transmission of a message
- Interactive video invites the user to participate

Webinars :

- Make webminars more dynamic by using interactive content (surveys, questions, quizzes, etc).
- The answers to the questions may be answered during the webinar

Tutorials and Product Demos :

- Product demo / interactive tutorial = powerful tool for boosting the sales
- You can enable visitors to choose specific aspect of the product or service they're interested in, and then be taken directly there

Lookbooks :

- Interactive lookbooks are a major trend nowadays
  allows a user to select and scroll though combinations of clothing, accessories...
  to find the perfect look
- Any product that can be customized can benefit from a lookbook-style interactive content

Interactive press releases :

- Boring static content aren't very inviting
- Videos, resource centers, and survey polls, on the contrary, invite engagement
- The press release becomes more memorable and more likely to be shared.

Useful interactive content tools

Hotjar
Visitor heat mapping + makes it easy to incorporate polls, surveys,
and incoming feedback from website users.

SnapApp
Thought to be a leader and innovator in interactive content creation and management.

Ceros
Interactive content platform allowing to create visually stunning interactive content
without the knowledge of a developer.

Engageform
Allow to easily create interactive quizzes and assessments.

WebyClip
Allows user watching your video to engage directly with products shown in the video content.

Ion Interactive
Allows you to create an interactive white paper, eBook, infographic, etc.

Repurposing content

- Creating content = time + costs

- Possible to use existing contents and make it interactive

  Examples :

  You've written an e-book = insert links, images, and quizzes in certain places to make it interactive
  Users could be sent to different pages depending on their answers

  Travel agency could ask something like this : « You have five days of vacation in front of you.
  Money isn't an object. Do you go urban or rural for your vacation? ».
  Each answer from a dropdown select box would then take the user to another question related to it
  or to a specific page.

  Possible to turn your e-book into a quiz, simply brainstorming a few questions directly from the e-book,
  then offering the custom plan to those who take the quiz

  The same goes with email, keeping the email copy the same, but providing a link to an interactive experience.

The 4 steps of creating interactive content

Not a method = a basic way to organize your work.

Step 1. Brainstorming Stage

- Usually involving several participants
  (gather as many ideas as possible + Write down anything that comes out without censoring participants)

  Keep in mind :

  - What would help you make the decision to buy, as a customer?
  - In what ways does the website lack in visual content?
  - How can the user experience be made more immersive?
  - What types of information needs to be gathered from your audience?

- Select the ideas based on feasibility from different points of view
  (economical, production calendar, and other possible limitations).

- Most importantly, select the ideas that will make the users say « wow! »

Step 2. Creation Stage

- From the idea chosen from the brainstorming session, create the interactive content.
- Focus on making it as entertaining and useful as possible
- Make sure to get the benefit of improved/increased data or better brand recognition
- There needs to be a balance = If the experience isn't fun, people will turn away
- For a quiz, add humor and personality to refined questions
- For a webinar, choose the most extroverted person on your team to host it.
- Manage to make interactive contents unusual as well as aesthetically impressive
- Do not hesitate to survey the competition and make sure your interactive content is different from theirs.

Step 3. Marketing Stage

- Promote your interactive content widely
- Encourage your followers to share it on social medias
- send a link to your email list
- optimize a blog post for SEO so you can link to it from there
- Add a highly clickable headline and call to action so user feel compelled to pass on your good work

Step 4. Testing Stage

- Set a time period (such as 60 days) and pay attention to your metrics
- How much engagement are you getting?
- Have you gotten more leads as a result of your content?
- What can you change to make it better?
- Focus on user experience and conversion rates
- Those metrics will tell you how well you're meeting your original goal.

Displace.js library

- Gathers visual elements for a website's page to become moveable, with a drag & drop movement
- ALSO offers the possibility to build a magnifier element

Step 1 : Basic structure of the page

- A very simple section (grey background)
- PLUS a yellow-box (our draggable element)
- *class="element"* = movable element
  (Make it position: *absolute*)

HTML :

```
<section>
        <div class="element"> </div>
</section>
```

Step 2 : Installing and initialize displace.js file

- CDN is not easily available
  Installation possible from NPM (or just download the code and have it local in your project folder)
- Click the download link from the web site https://catc.github.io/displace/.
- Copy the code supplied in a new tab and paste it in a new file inside your project folder.
- Include the <script> tag at the end of your </body> and link it to the library.
- Have another <script> tag for the custom code
  (not the best practice, but this is just for learning purpose, keeping all the code together).

  Script :

```
<script src="displace.js"></script>

<script>
// constrained to parent container
const el = document.querySelector('.element');
const options = {
        constrain: true;
}
displacejs(el, options);
</script>
```

NOTA :

- The folder architecture of the project is optional
- Either a folder for every javascript file, or simply keeping them the same folder as index.html
- ***constrain: true*** = making the element only capable of moving within its parent element (<section>)
- ***constrain: false*** (or keep it default) = make the element moving everywhere around the page

Displace.Js library: Magnifier

Step 1 : Basic structure of the page

For this sample, we are using the demo code provided by at the library's website.

HTML :

```
<div class="magnifier-demo">
	<div class="magnifier-demo__image-wrapper">
		<img id="magnifier-img" src="https://d3hfxk7rwdcpol.cloudfront.net/CSN/1aff0b69-5ccd-4609-
8817-10713159a606/images/fff6f853902b498984cfa0a0b9e45e13_LARGE.jpg">
		<span class="magnifier-demo__zoomer"> </span>
	</div>
	<div class="magnifier-demo__zoom-preview"> </div>
</div>
```

CSS :

```
.magnifier-demo {
	width: 600px;
}

.magnifier-demo__image-wrapper {
	width: 400px;
	position: relative;
	display: inline-block;
}

.magnifier-demo img {
	height: auto;
	max-width: 100%;
	vertical-align: middle;
	display: block;
}

.magnifier-demo__zoomer {
	display: inline-block;
```

```
        position: absolute;
        left: 245px;
        top: 102px;
        background: rgba(255, 255, 255, 0.4);
        cursor: move;
        border: 1px solid rgba(255, 255, 255, 0.7);
}

.magnifier-demo__zoom-preview {
        display: inline-block;
        background: yellow;
        width: 200px;
        float: right;
        position: relative;
         height: 100%;
        border-left: 3px solid white;
        box-sizing: border-box;
}
```

Javascript installation and entire code for functionality

It collects the original image finding the position of the mouse,
then it transcribes a zoomed (scaled) version of it into another container.

```
<script>
const img = document.querySelector('#magnifier-img');
const preview = document.querySelector('.magnifier-demo__zoom-preview');
const zoomer = document.querySelector('.magnifier-demo__zoomer');
const imgDim = {};

if (img.complete){
    setupMagnifier();
} else {
    img.addEventListener('load', setupMagnifier);
}

function setupMagnifier(){
// set preview bg
    preview.style.background = `url('${img.src}') no-repeat center center`;
// set preview height
    preview.style.height = img.offsetHeight + 'px';
    const previewR = preview.offsetWidth / preview.offsetHeight;
// set zoomer dimensions
    const zoomerW = 50;
    zoomer.style.width = zoomerW + 'px';
    zoomer.style.height = zoomerW / previewR + 'px';
// set bg size for preview div
```

```
        const sizeRatio = img.offsetWidth / preview.offsetWidth;
        preview.style.backgroundSize = `${img.naturalWidth / sizeRatio}px ${img.naturalHeight / sizeRatio}px`;
// cache dimensions
        imgDim.w = img.offsetWidth;
        imgDim.h = img.offsetHeight;
// init displace
        displacejs(zoomer, {
            constrain: true,
            onMouseMove: updatePreview
        });
// update preview
        updatePreview(zoomer);
}

function updatePreview(el){
        const x = el.offsetLeft / (imgDim.w - el.offsetWidth) * 100;
        const y = el.offsetTop / (imgDim.h - el.offsetHeight) * 100;
        preview.style.backgroundPosition = `${x}% ${y}%`;
}
</script>
```

JavaScript frameworks

- JavaScript= multi-paradigm language supporting event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles
- Initially client-side only = now used as a server-side programming language as well
- JavaScript is definitely the language of the web.

The difference between a library and a framework

- Developers often use the terms « library » and « framework » interchangeably.
  But there is a difference.
- Both frameworks and libraries are code written by someone else used to help solve common problems.
- For example, you have a program where you plan on working with strings.
  You decide to keep your code DRY (don't repeat yourself) and write some reusable functions
  By doing so, you've just created a library.

Library

- Just like going to Ikea = You already have a home, but you need a bit of help with furniture
- You don't feel like making your own table from scratch
- Ikea allows you to pick and choose different things to go in your home. You are in control.
- The developers call the library where and when they need it.
- You are in control

Framework

- Framework = building a model home
- You have a set of blueprints and a few limited choices when it comes to architecture and design
- Ultimately, the contractor and blueprint are in control
- They will let you know when and where you can provide your input.
- A framework inverts the control of the program, telling the developers what they need.

JavaScript framework

- A software framework is an abstraction in which software providing generic functionality can be selectively changed by additional user-written code.
- JavaScript framework is an application written in JavaScript
  where the programmers can manipulate the functions and use them for their convenience
- Frameworks are more adaptable for the designing of websites and hence

- They are preferred by most of the website developers.
- JavaScript frameworks = type of tool that makes working with JavaScript easier and smoother.
- Also make it possible for the programmer to code the application as a device responsive.
  (another reason why the JavaScript frameworks are quite popular when it comes to the question of using a high-level machine language.)

Top JavaScript frameworks (2019)

Angular

Angular (along with React) = at the top of the charts for JavaScript frameworks.

Although pretty heavy, Angular 2 is a large, feature-filled platform with a ton of useful content powered by Google.

Points worth considering:

- Uses TypeScript = superset of JavaScript that can compile down to vanilla JS
  (Many find this to be one of its biggest selling points)
- Pre-rendering of content on the server can allow for better SEO + faster browsing.
- Angular project is built on years of experience, from Angular to Angular 2 + beyond.
- Angular 2 = full of useful features (templates, forms...)
- Maintained by Google = users feel confident it will be around and used for a long while.

React

Originally created, and is maintained, by Facebook
(huge weight on the scales when choosing it for a project)

For any large, comprehensive project, React is almost a shoe-in for usage.

Biggest worry = Facebook / about the terms of use for React, as well as Facebook's ability to stop updates or change the licensing.

Points to consider:

- It's maintained by Facebook.
- The high volume of adoption and continued growth mean it will be easier to find competent React developers as well as help and outside components.

- The adoption of JSX allows for the structuring of components that will then compile into JS React, when running on the server, can make for more SEO friendly web pages than other JS frameworks, as the only thing seen by the client at that point is the generated HTML page(s).

- React Native can provide a nearly seamless mobile experience to pair alongside your React application for the web.

Vue.js

Several years old and mature
(still looked at as a new player in the JavaScript framework scene)
although it's growth of the Vue community is enormous.

Especially recommended for people just getting started with JavaScript frameworks
(or who need something lightweight for a project)

It doesn't have all the options of the more prominent contenders
BUT these aren't always needed

It's less fully featured, but that also makes it less restrictive.

Points to consider :

- Incredibly small file size, making it easy to include in projects without slowing it down

- Vue has a relatively easier learning curve than some of the bigger frameworks, and fantastic documentation to assist those new to JS frameworks and veterans alike.

- It is easy to integrate into other applications and languages. For example, Vue.js bundles with Laravel and couples nicely with it to create frontends for Laravel applications

- Vue uses an HTML-based templating syntax, allowing developers to write components quickly and easily.

Ember.js

Focused on getting things done quickly

Mmore opinionated framework, funneling developers into best practices for the platform
(much easier for projects to get started, but some may chafe at limitations)

Also provides several external tools to help developers along.

Great tool for rapidly getting your application into production
(and for dealing with problems and troubleshooting issues)

Only framework with an inspector plugin or a CLI,
but their inclusion along with its rigid best practices allow for a great quick start experience.

Points to consider :

- Opinionated best practices make Ember very easy to build with,
  but it is needed to follow Ember's structure.
  Otherwise, Ember may not be the right tool for your project.

- The command line tools package « ember-cli » opens up a whole new world of useful
  tools to add to an Ember app developer's toolkit.

- The Ember Inspector makes it easy to inspect Ember objects in your browser's developer
  tools. This can be used both while developing and while debugging issues.

Backbone.js

Been around for a long time (but it is still incredibly prevalent)
in legacy applications and in newer projects that need a less prescriptive framework or library to build on top of.

Declining in popularity (perhaps due to its age and minimalism)
still a relevant and powerful tool for the right needs

Reasonably easy to get a handle on, and if you need a framework that will allow you to build your application the way you see fit and provide you with a core to build off of, as the name implies, Backbone is a solid choice.

Points to consider :

- The maturity of the platform means that many issues have already been discovered and
  solved, and its community resources and support are solid.

- Backbone's approach to application structure is minimalist.
  It is incredibly lightweight, at a smaller file size than most frameworks on this list.
  It also is providing the bare bones of a framework, allowing the developer to build what
  they need off of it.

- It forces no templating engine, leaving that up to the individual developer. However, it
  does have Underscore as a dependency, allowing that to be the easiest default choice for
  templating.

SASS / SCSS

- (Syntactically Awesome Style Sheets) = extension of CSS
- Enables you to use things like variables, nested rules, inline imports and more
- Also helps to keep things organized and allows you to create style sheets faster.
  CSS preprocessor = scripting language that extends CSS
  by allowing developers to write code in one language and then compile it into CSS
- Sass is perhaps the most popular preprocessor around,
  but other common examples include *Less* and *Stylus*.
  You can't send Sass code directly to the browser; it won't know what to do with it
  Instead, you need to use the Sass pre-processor to translate the Sass code into standard CSS
  (a process known as *transpiling*)
- Transpilation = very much like compilation
  (but instead of translating from human-readable source code to machine-readable object code,
  it translates from one human-readable language to another, in this case from Sass to CSS)

Syntax

Sass includes two syntax options:

- **SCSS (Sassy CSS):**
  Uses the .scss file extension and is fully compliant with CSS syntax

- **Indented (simply called Sass):**
  Uses .sass file extension and indentation rather than brackets; it is not fully compliant with CSS syntax, but it's quicker to write

Variables

- Just like other programming languages
  Allows the use of variables that can store information you can use throughout your style sheet.

- For instance, you can store a colour value in a variable at the top of the file, and then use this variable when setting the colour of your elements. This enables you to quickly change your colours without having to modify each line separately.

Nesting

- Provides an excellent method for reducing the amount of code you need to write
- Can also lead to over-qualified CSS if not executed carefully
- The idea = to nest your CSS selectors in such a way as to mimic your HTML hierarchy

```
1  nav {
2      ul {
3          margin: 0;
4          padding: 0;
5          list-style: none;
6        }
7
8      li {
9          display: inline-block;
```

Class 13

Workshop

Class 14

Revision

Workshop

Class 15

Final exam