BOSUN TIN SHIN DESIGN DOCUMENT by Mikhail Leon

1 - INTRODUCTION

Creating the Bosun Tin Shin website posed a few challenges. Matt requested something 'simple & black', but there was a simultaneous necessity to show the best aspects of him as a person and as a professional - the sexyness of adventuring & sailing presented in a dark & rich nature within a nautical theme and a touch of class to go along.
The technical requirements demanded the implementation of many complex, co-dependent features all working in tandem. Every feature and function of the project is entirely handcrafted and handwritten.

2 - COLORS

Primary color	Off-Black	#0A0A0A
Secondary color	Off-White	#F8F8F8
Accent color	Gold	#8D671B
Interaction color	Blue	#0050BE

3 - FONTS

The 'Josefin' Family font offers a lot of variability in its look, with constant legibility and a slight edge, fitting to the nautical theme & overall style for the project.

Bosun Tin Shin **Bosun Tin Shin BOSUN TIN SHIN**

'Bodoni' adds class to the 2 main title sections of the website's content - the Collections, and the Gallery. The words 'Collections' and 'Gallery' are intentionally evocative of high art & fashion.

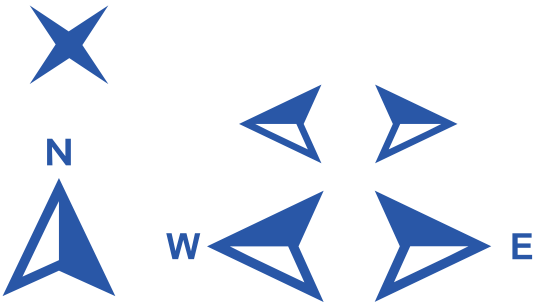
COLLECTIONS & GALLERY

4 - GRAPHICS

The main logo is designed as a round nautical patch with anchor & sailing rope elements. It fits under most backgrounds with its own solid extended background and circular shape. The rope wrapped in front & behind the anchor is shaped like an 'M' for Matt. The logo is reused to form a monogram.

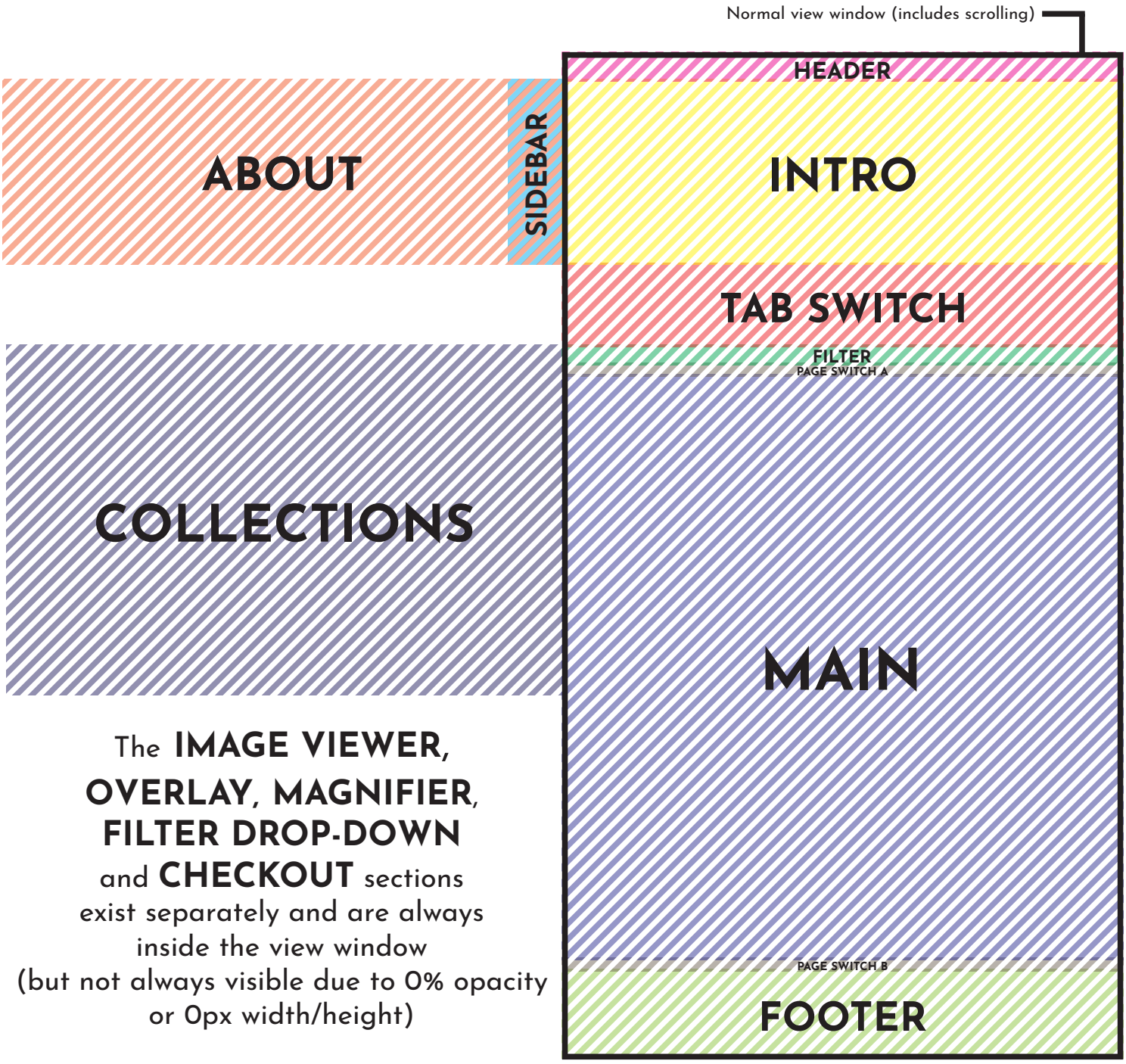
The golden ropes graphics tug & move are simple in shape, but through repetition and animation come to life as genuine ropes being pulled. With 'Bodoni', they stand more proudly, less as sailing rope & more as velvet.

The interactive content is a blue that is rich, not deep and not light - more like the color of the beautiful Mediterranean Summer sky more than the sea, one that works well with the off-black that occupies the background. The nautical arrows are not rotated but symmetrical when appearing in pairs, which gives them a more cohesive look. In larger variants, they come with letters that indicate cardinal directions. For example, the button that brings the user "back to the top" bears the North indication. The 'X' for closing windows is present here as a rotated Wind Rose that goes back to its original orientation when clicked.



The 'menu burger' icon has thin lines; the bottom one is longer & blue to evoke the sea. The 'checkmark' and 'shopping cart' icons are also thin, matching the font weight they are paired with.

5 - LAYOUT

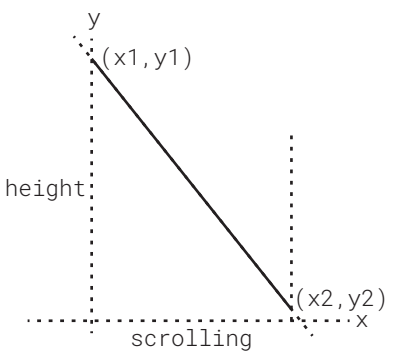


The **IMAGE VIEWER**, **OVERLAY**, **MAGNIFIER**, **FILTER DROP-DOWN** and **CHECKOUT** sections exist separately and are always inside the view window (but not always visible due to 0% opacity or 0px width/height)

6 - SCROLL PARALLAX

To create a parallax scrolling effect with 2 different vertically adjacent sections, the top section's height can be decreased during scrolling of the page. The bottom section will then seemingly scroll faster, completing the effect. The formula that ties the height of the top section (y) to vertical scrolling (x) is $y = y1 - ((x - x1) * ((y1 - y2) / (x2 - x1)))$.

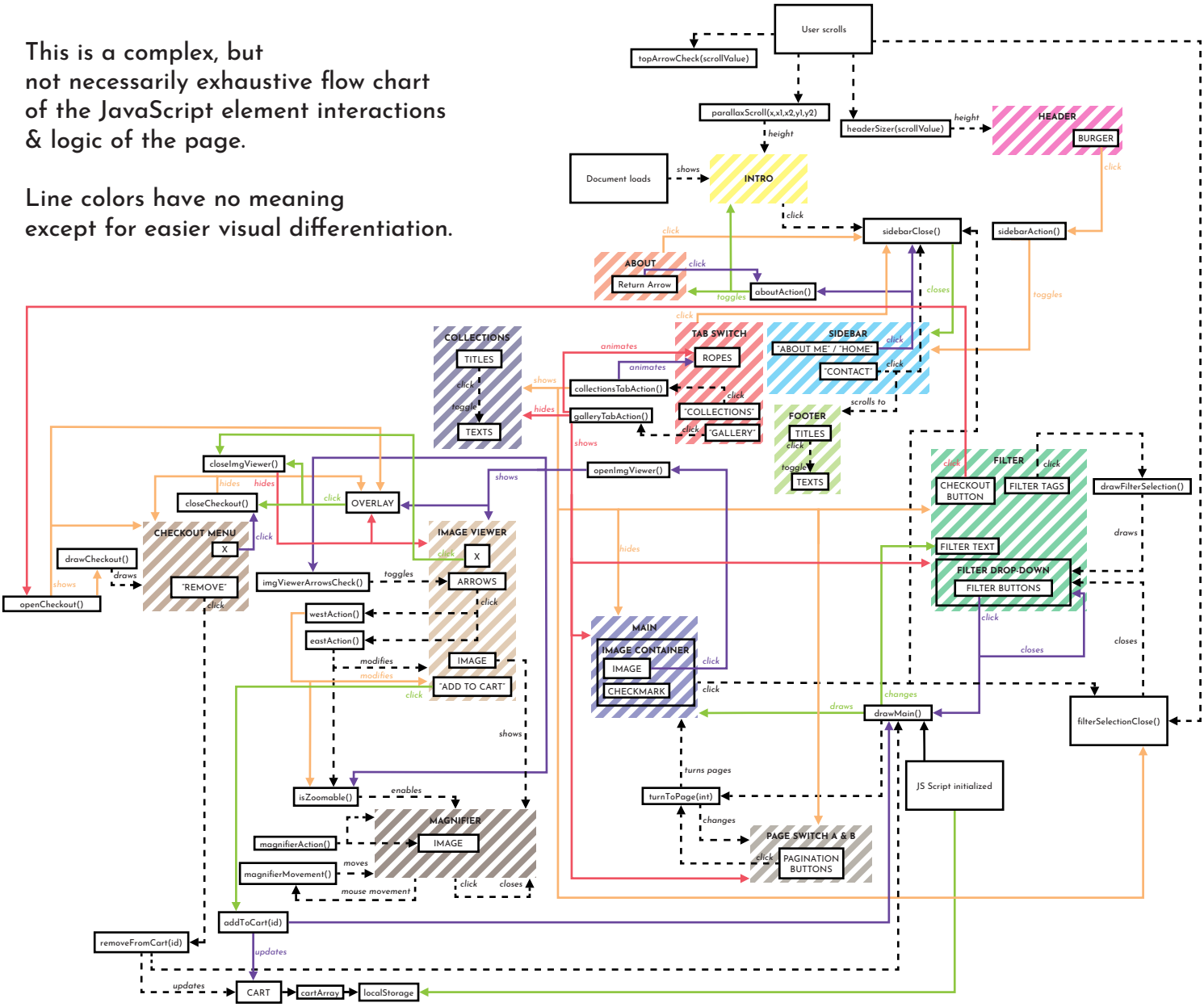
This is used with the **INTRO** section as the top and **TAB SWITCH** as the bottom, with values (in pixels) y1 = 500, y2 = 250, x1 = 40, x2 = 240 .



7 - FLOW CHART

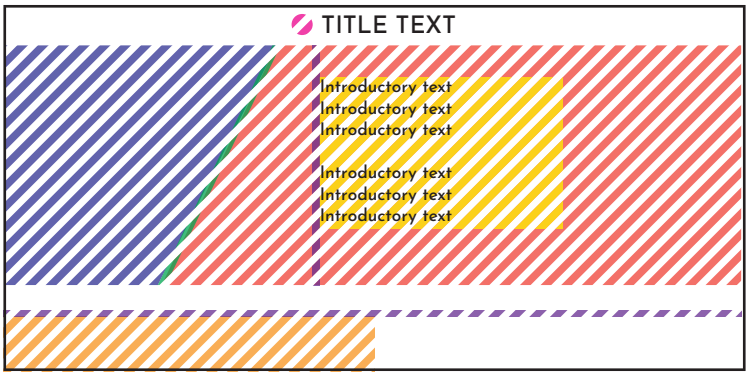
This is a complex, but
not necessarily exhaustive flow chart
of the JavaScript element interactions
& logic of the page.

Line colors have no meaning
except for easier visual differentiation.



8 - INTRO

To maintain simplicity & encourage user action, **INTRO** avoids a full-window banner and lets the user immediately see the **TAB SWITCH** section below, with a further parallax scroll effect previously mentioned. It still maintains a sense of dynamism, sexyness and thematic display through its visuals & animation.



The diagonal line is a leading-line to the logo & the button to switch tabs, its angle being equal to the angle of Matt's torso in image A. There are other leading-lines to give the layout balance & structure. The parts that constitute **INTRO** slide into view from the left upon loading the page, all at different z-indexes.

The elements all move at different velocities. **INTRO** is also surrounded by black bars, giving it a more cinematic, widescreen look.

9 - ABOUT & SIDEBAR

The **HEADER** section contains a "burger"-type icon which accesses the **SIDEBAR**.

The **SIDEBAR** allows the user to press the "ABOUT ME" button, moving the **ABOUT** section into view. Once **ABOUT** is visible, the user can go back to the **INTRO** section through an arrow, or by using the same button in the **SIDEBAR** like previously, which now reads "HOME".

The user can also choose to ignore this & continue using the rest of the website as normal.

The **SIDEBAR** also has a "CONTACT" button - upon clicking it, the user will be scrolled to the bottom of the page containing all of the contact info.

10 - SYSTEMIC DESIGN

The entire experience of the website was built as a Single Page Application with a systemic, modular aspect in mind. This means that many aspects of the site work independently of others while still interacting with each other. Some other sections are more ephemeral, such as **SIDEBAR** & **FILTER DROP-DOWN**, which will usually close should the user disengage from them.

This is exemplified at the very start here, where variables are kept in memory that track the status of the **SIDEBAR** (eg: `sidebarActiveStatus`) & helper functions exist (eg: `sidebarClose()`) that will close it on certain triggers (eg: scrolling at any point while the **SIDEBAR** is open will close it, as will clicking outside of it).

On the more permanent side of things, the user can keep the **ABOUT** section open, scroll to the Gallery (**MAIN**), use a filter on it, flip to page 3, open the "Terms and Conditions" text in the **FOOTER**, then switch tabs to **COLLECTIONS** and select "Morocco 2020"; all of the other sections & logic used will remain in the state they were left in, ready to come back to.

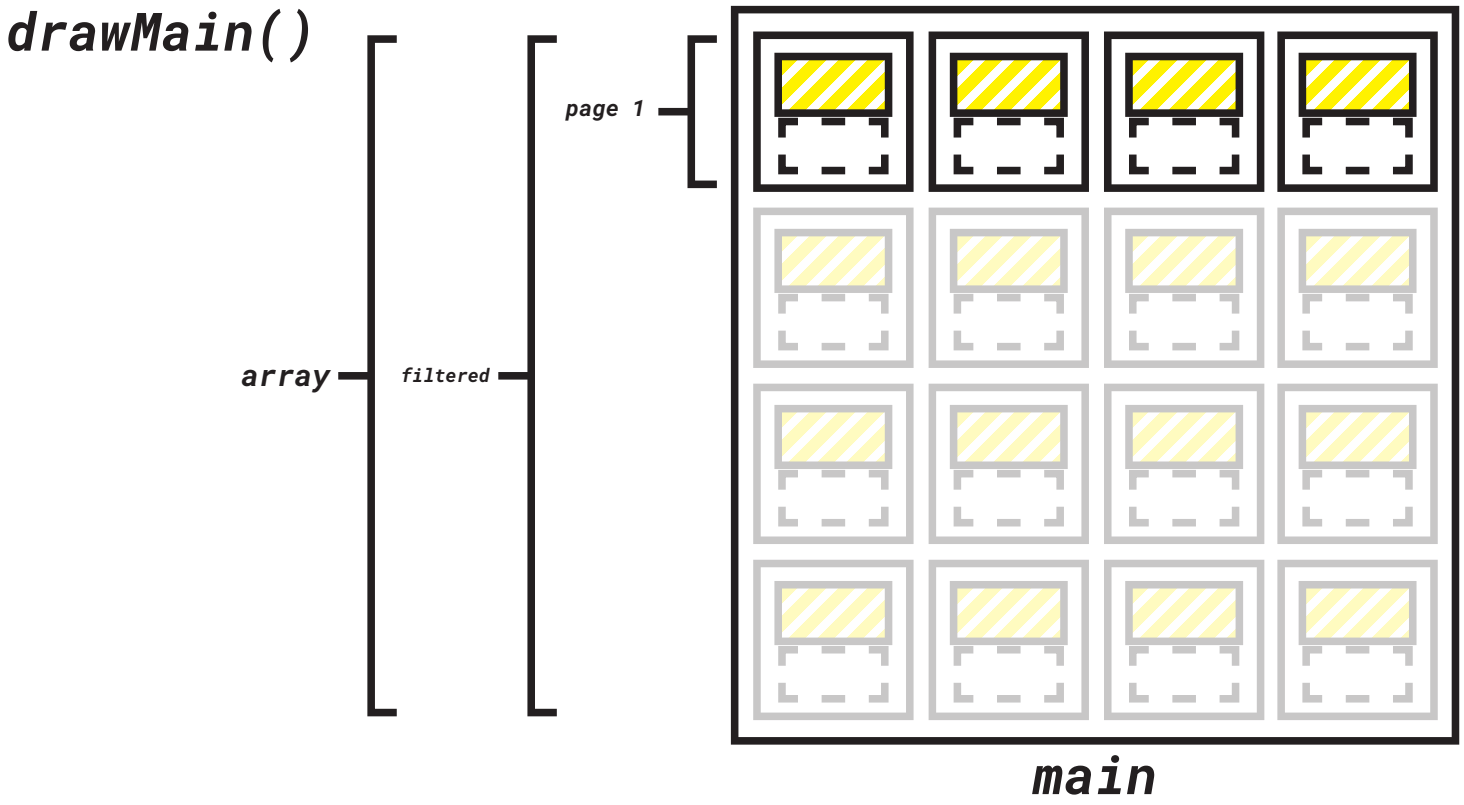
11 - MAIN GALLERY

This is the most extensive section of Bosun Tin Shin and features the most complexity. It was designed to look & feel like an art gallery, hence why the images are taller than they are wide and the backgrounds are extensively patterned. We will start with the drawing of the gallery itself, and then move onto peripherals.

The entire gallery is loaded by the function `drawMain()`, which is called upon at page load, and subsequently every time the gallery needs to be redrawn (except on pagination).

`drawMain()` requires a structured ***array*** of objects representing possible photographs. Each photograph contains: ID, Location, Year, Path. This way, photographs can be easily added or removed from the list, kept in a remote database or in the front-end, etc.

```
{id:1,location:"UK",year:2020,path:"/img/abc.jpg"}, {id:2,location:"UK", year:2022,path:"/img/def.jpg"}, {id:3, location:"UK",year:2021,path:"/img/ghi.jpg"},
{id:5,location:"FR",year:2020,path:"/img/jkl.jpg"}, {id:6,location:"GIB",year:2022,path:"/img/mno.jpg"}, {id:7, location:"PT",year:2020,path:"/img/pqr.jpg"},
{id:8,location:"US",year:2020,path:"/img/stu.jpg"}, {id:9,location:"UK", year:2018,path:"/img/vwx.jpg"}, {id:10,location:"SP",year:2020,path: "/img/yz.jpg"},
...
```



Each image is wrapped inside a container - `img-case` - which can have 2 children. One is `gallery-image`, containing the image itself; the other is optional - `img-case-checkmark-div` - a checkmark designed to inform the user whether the image in question is already in the cart or not.
It is drawn at the `drawMain()` stage and reads from the cart set, which will be explained later.

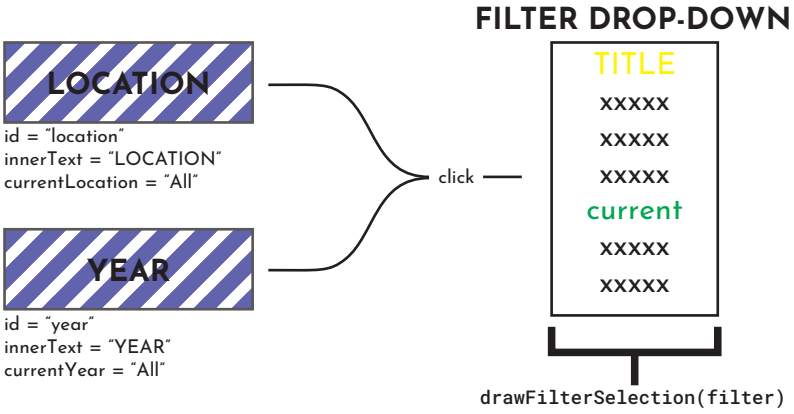
Next, `drawMain()` will check what the filters are set to by reading the values of 2 variables - `currentLocation` & `currentYear` - and filter the `array` appropriately with the helper JS set, `validIDset`. The values of the variables are initially set to the string "All". These values can be changed by clicking on the filter buttons in the **FILTER DROP-DOWN**, which incurs another `drawMain()` call.

By default, all `img-case` elements are set to '`display:hidden`' and it is the pagination function - `turnToPage(int)`- that decides which elements will be shown through '`display:block`'.
Every `drawMain()` has a `turnToPage(1)` call inside it.

The max amount of images per page is a variable - `imgPerPage` - kept at 4 here, and used to perform pagination logic. The current page value is stored directly in the text, and the arrows are used to call `turnToPage(current-1)` and `turnToPage(current+1)`.
This does not incur another `drawMain()` call.

12 - FILTER

Filtering is a very dynamic process that involves multiple functions and many co-dependencies.
Upon initialization, none of the filter elements exist, so some declarations & hydration must be nested inside functions that are only called once the process is underway.
It all starts with one of the Filter Tag Buttons - `filter-tag` - being clicked.

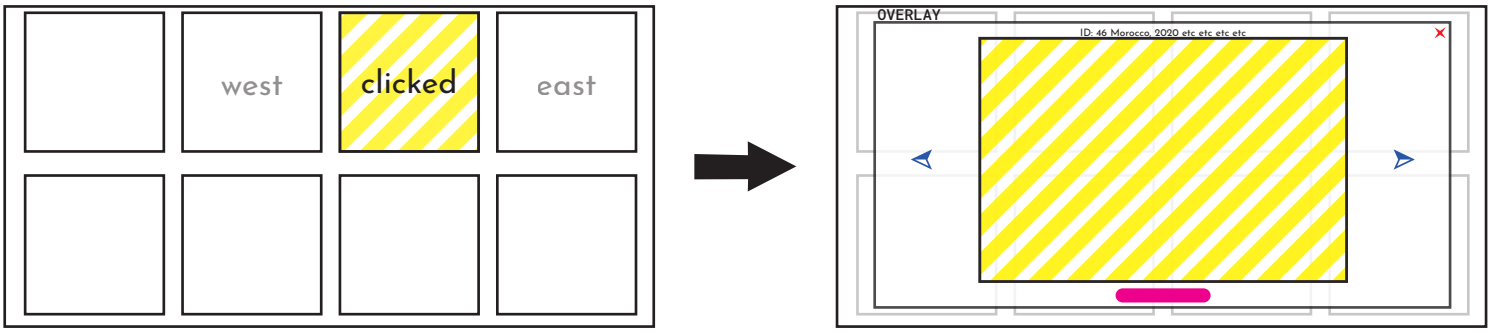


The drop-down menu is singular, thus its content is dynamic. It reads from `allLocations` & `allYears`, which store the respective values from the `array` of photograph objects upon initialization, and are used to populate the menu.

The TITLE value is filled from the particular `filter-tag` clicked, while current is highlighted based on an intersection of `currentLocation/currentYear` and whichever corresponding tag is being selected.

13 - IMAGE VIEWER

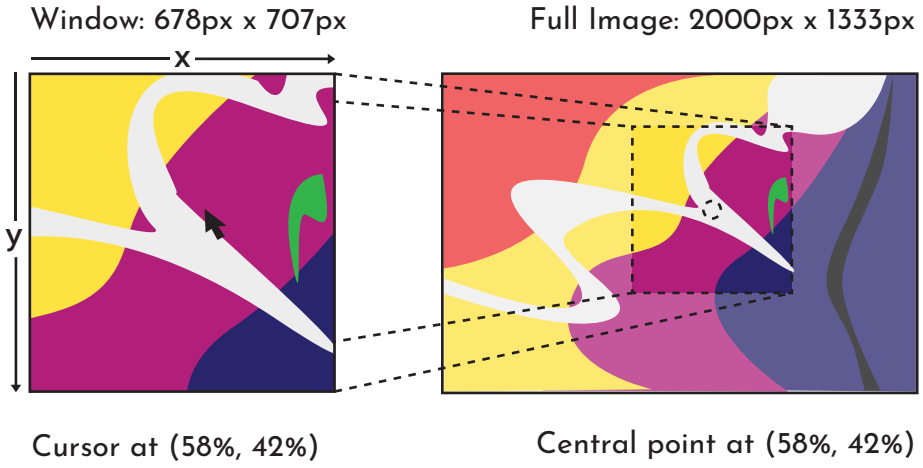
The **IMAGE VIEWER** becomes accessible once an image in the gallery has been clicked. It reads the values from the image - ID, Location, Year, Path - and uses them to populate the viewer.
It also enables the **OVERLAY** and performs multiple logic checks:
`imgViewerArrowsCheck(index)` - to determine which, if any, arrows should be available to click;
if the image is in the cart or not - reflected in the appearance & functionality of the button below;
`isZoomable()` - to determine if zooming is enabled through `magnifierAction()`.



Clicking the X, the **OVERLAY**, or pressing 'Escape' will close the viewer and the overlay itself..
If available, clicking the cart button will add the image to the cart.
If available, clicking the arrows (or pressing 'left' or 'right' on the keyboard) will make the viewer display the information from the image to the left (west) or right (east) in the gallery through `westAction()` or `eastAction()`, which read that information, perform animations and employ async functions, making it visually seem like it is swiping through multiple elements;
they also perform some of the same checks as the **IMAGE VIEWER** itself.
There is an additional check to prevent keyboard spam.

14 - MAGNIFIER

If `isZoomable()` returns `true` - this happens if the image's `naturalWidth` & `naturalHeight` are both bigger than the user's respective inner browser width & height - the cursor will turn into the "zoom-in" variant, indicating that the user can click on the image in the **IMAGE VIEWER** to open the **MAGNIFIER** and perform zooming.



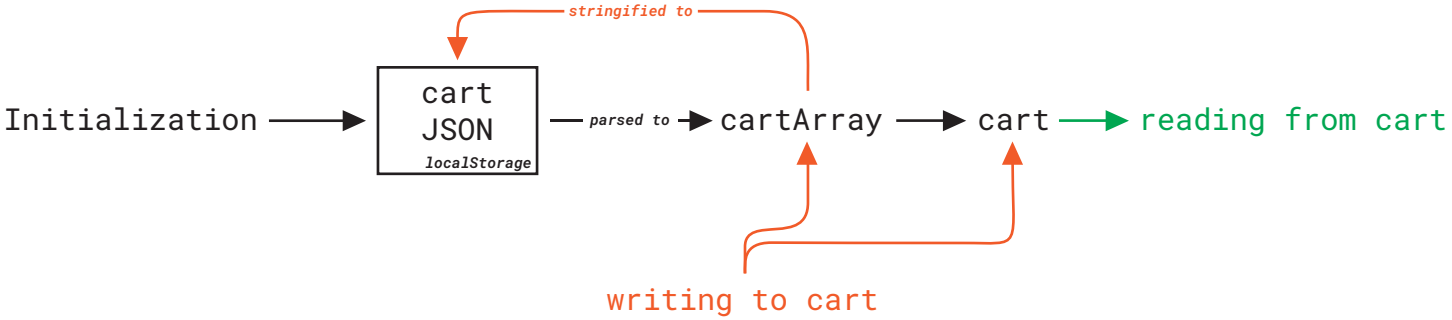
Moving the mouse will inspect different parts of the image at 100% zoom according to mouse position - this is handled by `magnifierMovement(x,y)`.

Clicking again will close the **MAGNIFIER**.

There is a struck balance between optimizing the image file-size through compression while only using 1 version of 1 photograph - in other words, no thumbnails (to minimize network requests, retain design simplicity and naturally safeguard copyrighted content) - and allowing the user to see the beauty of the photography on display.

15 - CART

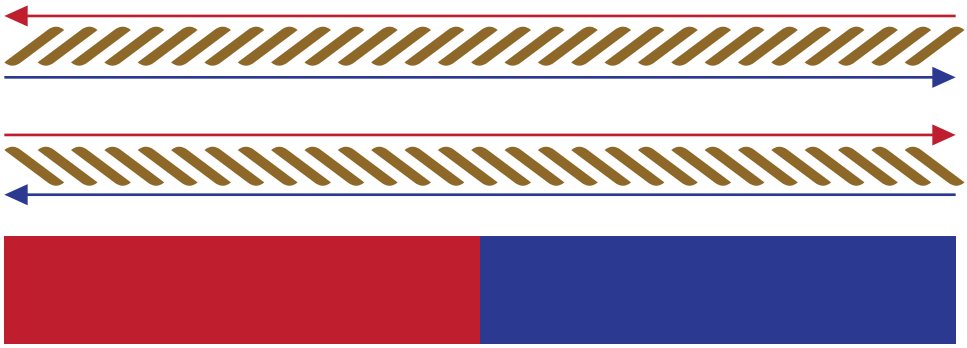
The cart uses multiple structures to function. At its base, it is a simple JS set that contains the unique ID's of the photographs. In order to be saved in the user's browser localStorage, there is a form of the cart that is also a JS array, which is stored in JSON.



16 - COLLECTIONS

If the gallery is a larger body of work, the **COLLECTIONS** are
encapsulations of a particular time & place, and more importantly - a particular story.
A great deal of the meaning of art is held in its context - provided here in the form of highlights & writing.

Switching tabs is a simple endeavor of bringing one tab into view & the other out of view,
but is accompanied by animation of ropes resembling a pulley system on a stage.



Each collection has a curated image cover and a variable height. On page load, each collection's height is individually recorded & then set to 0. Clicking on one will then restore the respective height.
This recording process - done be `recordCollectionsHeights()` - would seem superfluous, but due to idiosynchracies of CSS, is important to the smooth animation transitions between showing & hiding each collection.

16 - LEGAL

The legal text at the bottom of the page is not dissimilar from the collections - click to show/hide individual texts - but works differently in that it only modifies font sizes & requires no special procedure.
It records boolean values for the status of each text & uses a more generalized function - footerAction(description,status) - which serves all 3 available texts -
Terms and Conditions, Privacy Policy and Copyright.