The Starry Messenger – Site Documentation

Scott Steele

FHSU INF_678_VC

Overview:

The Starry Messenger (Sidereus Nuncius) is a treatise by Galileo detailing his first observations through a telescope. The Starry Messenger is a website that, in the spirit of Galileo's work, gathers observations of our universe and makes them available to the public via NASA's public API's. Visitors of the site will be able to discover and peruse a images. These images contain interesting metadata. As they browse, users can collect their favorite images into a cart and then download them.

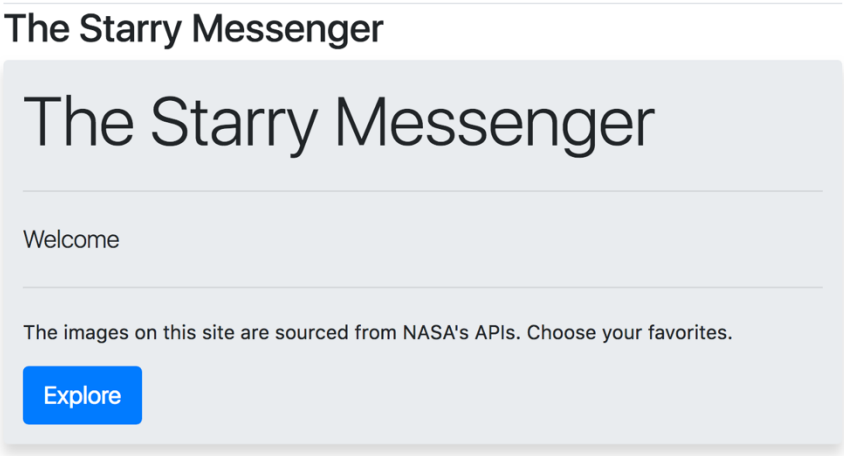
The source code for the site can be found at https://github.com/scott9s/starry-messenger

Features:

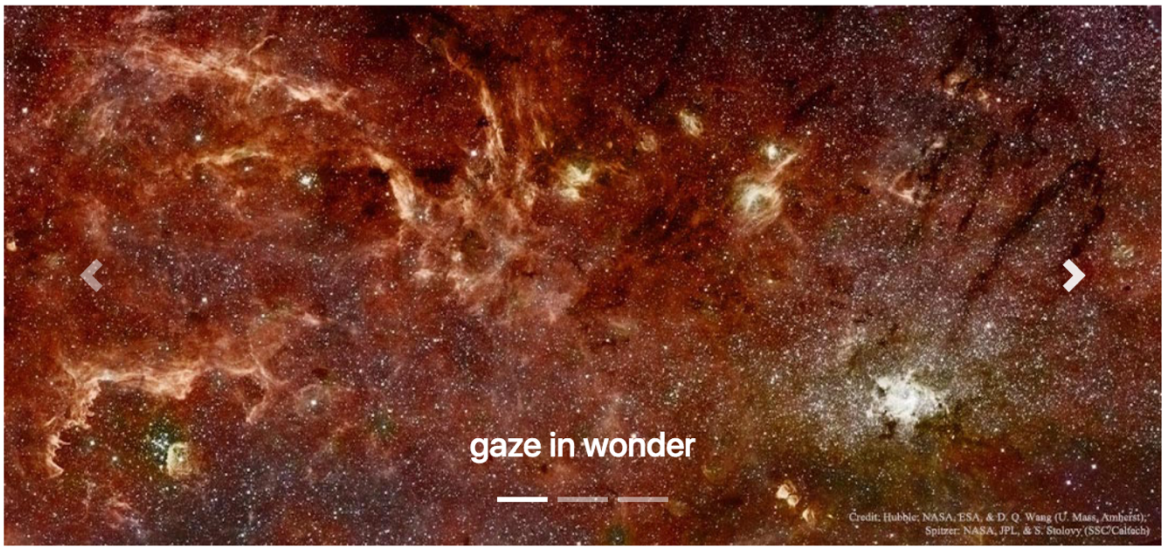
The features of The Starry Messenger are listed by page.

Home/Index:

The home page features a jumbotron introducing the site and inviting users to explore. The explore button navigates to the favorites page.

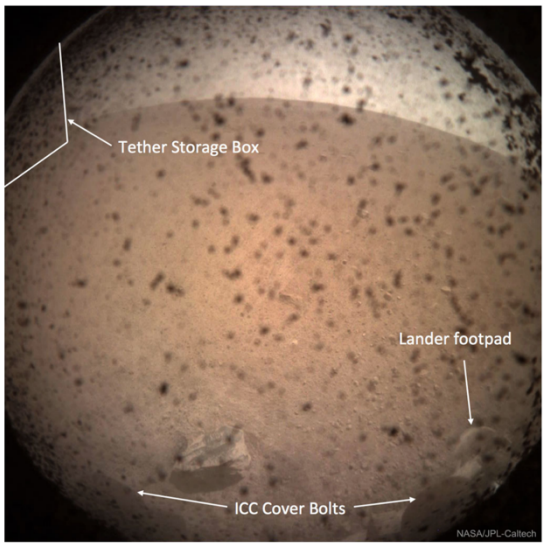


An attractive carousel loops between a set of preselected, annotated images from the NASA API's. Users can navigate the carousel via the arrows on the sides or the bars at the bottom.



Finally, a picture of the day section highlights an image selected by NASA. Most images, it is possible to save this image to the favorites cart. (The image below happens to be the feature from the recent touchdown of the Insight rover!)

Picture of the day



InSight's First Image from Mars

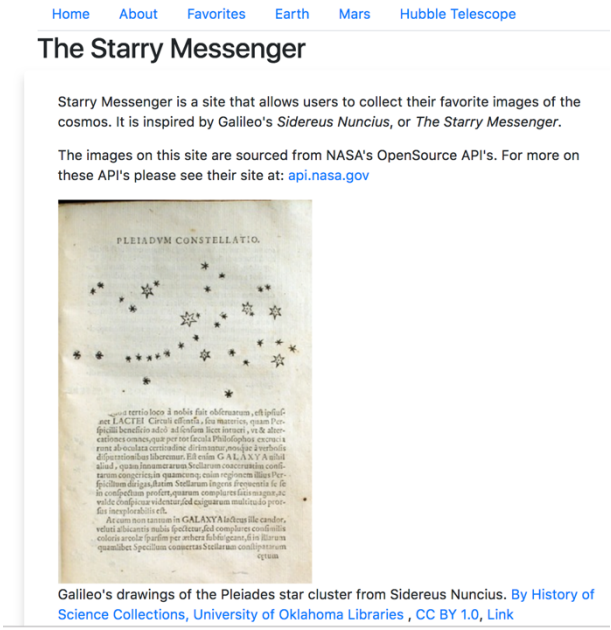
add to favorites

Welcome to Mars, NASA Insight. Yesterday NASA's robotic spacecraft InSight made a dramatic landing on Mars after a six-month trek across the inner Solar System. Needing to brake from 20,000 km per hour to zero in about seven minutes, Insight decelerated by as much as 8 g's and heated up to 1500 degrees Celsius as it deployed a heat shield, a parachute, and at the end, rockets. The featured image was the first taken by InSight on Mars, and welcome proof that the spacecraft had shed enough speed to land softly and function on the red planet. During its final descent, InSight's rockets kicked up dust which can be seen stuck to the lens cap of the Instrument Context Camera. Past the spotty dirt, parts of the lander that are visible include cover bolts at the bottom and a lander footpad on the lower right. Small rocks are visible across the rusty red soil, while the arc across the top of the image is the Martian horizon dividing land and sky. Over the next few weeks InSight will deploy several scientific instruments, including a rumble-detecting seismometer. These instruments are expected to give humanity unprecedented data involving the interior of Mars, a region thought to harbor formation clues not only about Mars, but Earth.

Image Credits: Public Domain

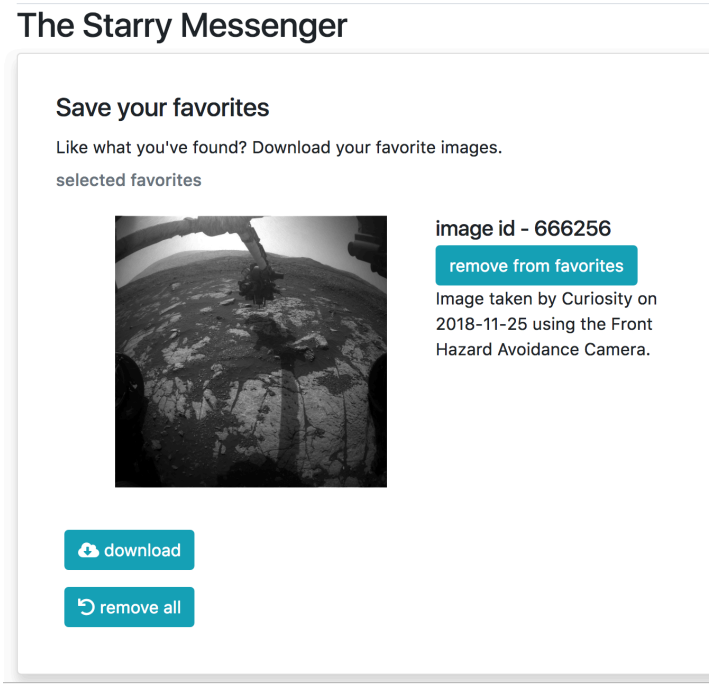
About:

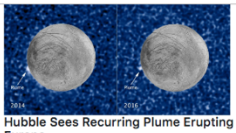
The About page provides a description of the site. It also documents the source of the images and provides a link to the NASA API documentation. An image from *Sidereus Nuncius* is provided as a special treat.



Favorites:

The Favorites page has two main features. First is a cart displaying all favorites selected by the user. The cart provides download functionality via a download button. Additionally, the user can completely clear the cart with another button.

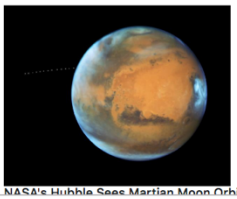




Hubble Sees Recurring Plume Erupting From Europa

add to favorites

These composite images show a suspected plume of material erupting two years apart from the same location on Jupiter's icy moon Europa. The images bolster evidence that the plumes are a real phenomenon, flaring up intermittently in the same region on the satellite. Both plumes, photographed in ultraviolet light by NASA's Hubble's Space Telescope Imaging Spectrograph, were seen in silhouette as the moon passed in front of Jupiter. The newly imaged plume, shown at right, rises about 62 miles (100 kilometers) above Europa's frozen surface. The image was taken Feb. 22, 2016. The plume in the image at left, observed by Hubble on March 17, 2014, originates from the same location. It is estimated to be about 30 miles (50 kilometers) high. The snapshot of Europa, superimposed on the Hubble image, was assembled from data from NASA's Galileo mission to Jupiter. The plumes correspond to the location of an unusually warm spot on the moon's icy crust, seen in the late 1990s by the Galileo spacecraft (see PIA21444). Researchers speculate that this might be circumstantial evidence for water venting from the moon's subsurface. The material could be associated with the global ocean that is believed to be present beneath the frozen crust.
https://photojournal.jpl.nasa.gov/catalog/PIA21443



NASA's Hubble Sees Martian Moon Orbiting the

The second feature is a feed consisting of images from the focus pages all in one place. Images in the feed can be added and removed from the cart.

Earth:

The Earth page consists of NASA images of the Earth. It is a feed of images with functionality similar to all other feed on the site.

The Starry Messenger

Earth - the Blue Marble

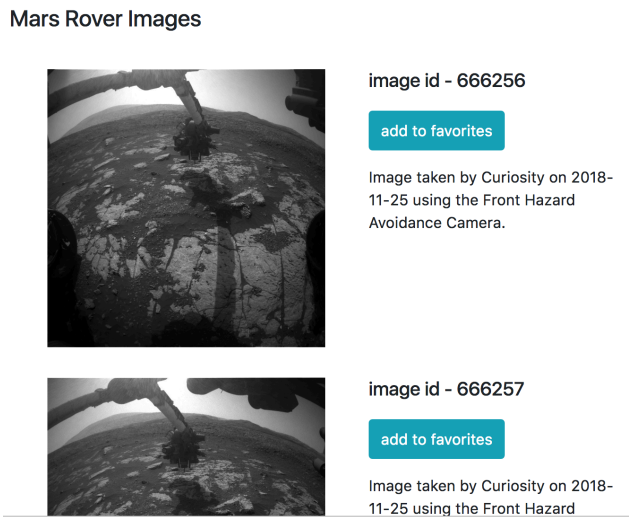
Eigg, Scotland

add to favorites

The island of Eigg is one of the small isles in the Scottish Inner Hebrides, south of the Skye peninsula. The main settlement of the 31 km2 island is Cleadale. In 2008, Eigg began a project to become completely energy self-sufficient. Using a

Mars:

The Mars page consists of NASA images of the red planet. It is a feed of images with functionality similar to all other feed on the site.



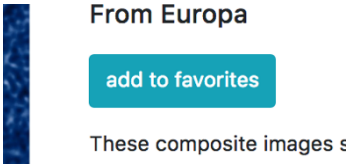
Hubble Telescope:

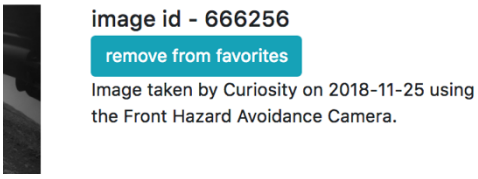
The Hubble Telescope page consists of NASA images taken by the Hubble Telescope. It is a feed of images with functionality similar to all other feed on the site.

Additionally, the site features a navigation bar and footer on each page.

Instructions:

To use the site, the main workflow is to browse the pages and select favorites. Favorites can be selected by clicking the “add to favorites button.” Doing so will ensure that the image appears in the favorites cart on the favorites page. Favorites added are globally added across the site.

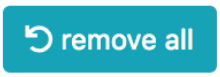




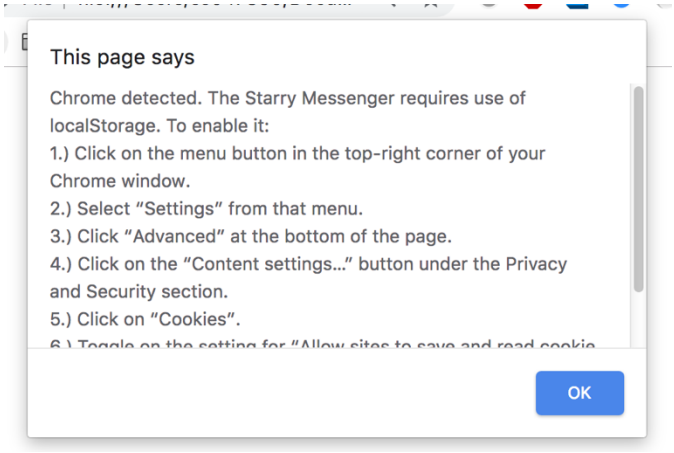
Note that selected favorites will display a “remove from favorites” button. Clicking this button will remove the selected item from the favorites cart. Favorites removed are globally removed across the site.

Once the user has collected all their favorites, they can download them using the download button located on the favorites cart.

Should the user want to remove all favorites from the cart, they can click the “remove all” button located inside the favorites cart.



It is important to note that the site relies on HTML5 local storage for tracking the state of the favorites cart across pages. The user needs to ensure their browser has local storage enabled. Regardless, the site will make an best effort attempt to detect what type of browser the user is using and display an alert customized to their browser instructing them how to turn enable local storage. Once displayed and enabled, the site stores having displayed the message, ironically, in local storage so that it is only displayed once.



Data Sources:

All data is sourced dynamically from NASA's API's. There are no static images used for the site. Data is fetched using AJAX calls. All calls here are using http GET verbs.

Below is a list of all endpoints used for the site along with their query parameters:

Resource	URL
Earths	https://images-api.nasa.gov/search?year_start=2018&keywords=earth&media_type=image
Hubble Telescopes	https://images-api.nasa.gov/search?year_start=2017&keywords=Hubble%20Space%20Telescope&media_type=image
Mars Rovers	https://api.nasa.gov/mars-photos/api/v1/rovers/curiosity/latest_photos
Pictures of the day	https://api.nasa.gov/planetary/apod

Authentication to the API's is accomplished by appending an API key as a query parameter at the end of a URL. Registering for an API key is free and can be done at https://api.nasa.gov/index.html#apply-for-an-api-key .

Below is an example taken from extractAndAppend.js. It illustrates how data is fetched from the API's and then appended to the feed of the calling page.

```
A convenience method for constructing <li> elements.
// Given resourceName, the mediaSrcUrl, title, and description of the element,
// it returns a string representation of the element.
function createLiElement(resourceName, mediaSrcUrl, title, description) {
    let elementToAppend = "<li class=\"media p-3 row\"><img class=\"mr-3 col-lg-4\"
src=\""
        // set src image
        + mediaSrcUrl
        // set resourceName
        + "\" alt=\"" + resourceName + "\" style=\"max-width: 50%; height: auto;\"
download>"
        + "<div class=\"media-body col-lg-8\">"
        + " <h5 class=\"mt-0 mb-1\">"
        // set title
        + title
        + "</h5><button type=\"submit\" class=\"btn btn-info\"
onclick=\"handleClick(this)\">"
        + "add to favorites</button><p>"
        // set explaination
        + description + "</p></div ></li> ";

    return elementToAppend;
```

```
}

// Given a sourceUrl and the name of the resource to extract,
// this method fetches the resources from the source and appends
// them as <li> elements to the target class.
function extractAndAppend(sourceUrl, resourceName) {
    $.ajax({
        url: sourceUrl,
        dataType: "json",
        success: (result) => {
            for (const record of result.collection.items) {
                // construct li element to append to <ul class="feed">
                let elementToAppend = createLiElement(resourceName,
record.links[0].href,
                    record.data[0].title, record.data[0].description);

                console.log(elementToAppend);
                // apend element to targetId
                $(targetClass).append(elementToAppend);
            }
        }
    });
}
```

As shown in extractAndAppend, an API call is made using the sourceUrl. On success, a lambda expression iterates through the collection of items returned, constructs a new <li> tag, and appends it to the feed.

As a side note, this approach of using JQuery to append elements to pages is also used for inserting the navigation bar and the footer. This was done in an effort be DRY and only implement elements once. Below, footer.js provides an example of this.

```
const footerClass = ".footer";

const footerContent =
    "<div class=\"col footer-copyright rounded shadow bg-info text-center py-3\">" +
        "<p>" +
            "© 2018 Scott Steele All Rights Reserved" +
        "</p>" +
    "</div>";

$(footerClass).append(footerContent);
```

State Across Pages:

With the use of a cart, state must be tracked across pages. This is accomplished using local storage. Cart.js implements all CRUD functionality for adding and removing favorites from local storage.

```
function addToCart(elem) {
    let favorites = JSON.parse(localStorage.getItem(favoritesKey));
    // if favorites is empty, initialize
    if (!favorites) {
        favorites = [];
    }
    // elem is the button that was clicked
    // get parents to capture media element
    favorites.push($(elem).parent().parent().html());

    // save favorites to local storage
    localStorage.setItem(favoritesKey, JSON.stringify(favorites));
    console.log("localStorage.favorites:\n" + localStorage.getItem(favoritesKey));
}
```

It also provides event handlers for interacting the with cart and images on the pages, such as toggling buttons between “add to favorites” and “remove from favorites.”

```
// toggles add/remove button label
// directly appends/removes to cart or feed without reloading.
// saves/removes from localStorage
function handleClick(elem) {

    // if the current page == favoritesPage
    // use favorites workflow
    if (getCurrentPage() === favoritesPage) {
        handleClickFavoritesPage(elem);
    }
    // else use the page external workflow
    else {
        handleClickPageExternal(elem);
    }
}
```

Since local storage is required, browserWarning.js is used to provide the user with a hint in case they are experiencing difficulty with the site. It detects the type of browser in use and displays a custom alert. Local storage is then used to record the alert having been shown so that it is only displayed once.

```
function displayWarning() {
    // check if warning has already been displayed.
    // It only needs to be displayed once.
    let warningDisplayed = localStorage.getItem(warningDisplayedKey);
    if (warningDisplayed !== warningDisplayedValue) {
        if (isSafari) {
            alert(new Date() +
                "Safari detected. The Starry Messenger requires use of localStorage. "
                + "Be sure to have \"Disable local file restrictions\" set to true in
your menu bar.\n\n"
                + "For more on working around this issue, please see: "
                + "https://stackoverflow.com/questions/46374291/safari-11-gets-
securityerror-dom-exception-18-when-accessing-localstorage");
        }

// more below
```

Downloading:

Users are able to download their favorites. This is accomplished using download.js. It finds each <img> tag in the favorites cart, extracts the src attribute, calls the source, and returns the result as a blob. To perform the download, a URL is created for the blob, an invisible <a> tag is created with the download attribute set, it is clicked, and then removed. This downloads the file. This approach was taken in order to support running the site from a local file system without having to worry about cross site issues and redirects.

```
function download(blob, imageSrcUrl) {

    console.log("downloading " + imageSrcUrl);

    // create the <a> tag and set its attributes
    let a = document.createElement("a");
    document.body.appendChild(a);
    a.style = "display: none";
    // create a URL for the blob
    let downloadUrl = window.URL.createObjectURL(blob);
    a.href = downloadUrl;
    // set the download attribute to the imageSrcUrl
    a.download = imageSrcUrl;
    // click to download
    a.click();
    // cleanup
    window.URL.revokeObjectURL(downloadUrl);
    document.body.removeChild(a);
}
```