Predix Design System

Contents

Predix Design System Overview

Create Modern Web Applications

Web applications have evolved to implement many coordinated user functions and tasks traditionally associated with desktop software (for example, Google Docs and Microsoft Office).

In addition, modern browsers have built-in support for open-source standards such as JavaScript, HTML5, and Cascading Style Sheets (CSS) that reduce the dependency on proprietary browser plugins such as Adobe Flash or Microsoft Silverlight to run rich interactive content (such as streaming audio and video) .

As a developer, you can create modern web applications that run across multiple browsers following the Model View Controller (MVC) architecture, which separates functions that are built and maintained as independent modules:

- **Model** – The Model function stores and retrieves information from a database or data service. It defines the data structure for the information that is passed to the controller for processing and to the view for displaying to the user.
- **View** – The View function is a visual summary of the information available in a web application. It is the user interface (implemented using HTML, CSS and JavaScript) that translates tasks and results to information that the user can understand and act on.
- **Controller** – The Controller function (implemented using JavaScript or other web technologies such as Java EE, Node.js, or Python) defines the business logic used to process commands, make logical decisions, and perform calculations. It also moves and processes data between the model and view modules.

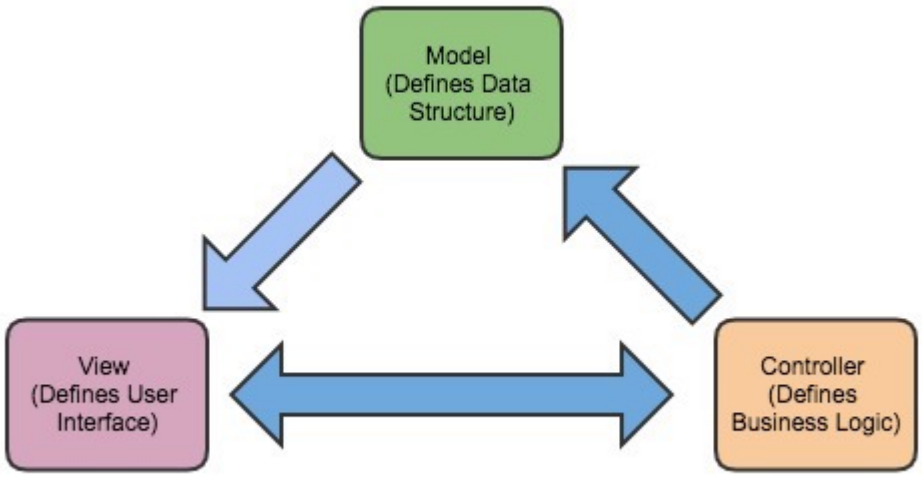

Figure 1: Model View Controller Architecture

See the following resources for more information:

- Modern Web Application Architecture
- MVC Architecture

Single-Page Application Approach

Modern web applications support rich interactions with multiple web components on a page. Each web component has many intermediate states (such as menu open, menu item X selected, menu item Y selected, menu item clicked). Server-side rendering is hard to implement for all the intermediate states because small view states do not map well to URLs.

An alternative to server-side rendering is the single-page application approach that works inside a browser. A single-page application can update any part of the View function without requiring a server round trip to reload a page during use. This is achieved by separating the data from the presentation of data by having the Model function handle data and the View function read from the Model function (see Create Modern Web Applications on page 1 for more information).

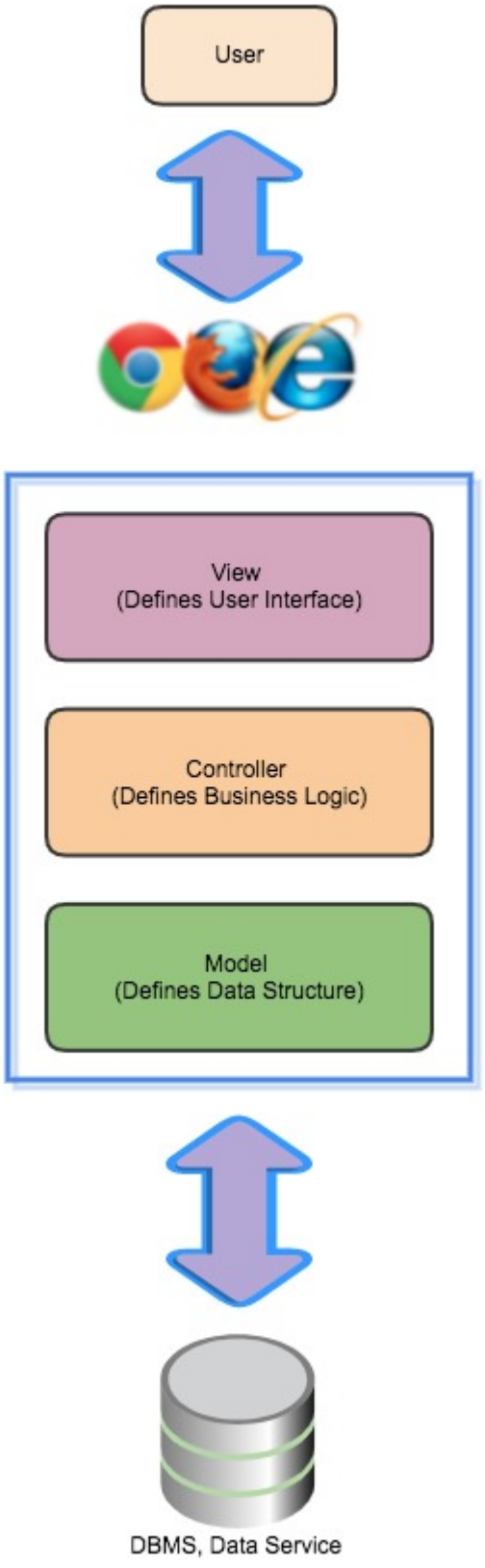

Figure 2: Single-Page Application Approach

A single-page application still needs to make calls to the server back end to get its data, but that server can be built with a completely different technology: (like Node, Java or PHP).

See the following resources for more information:

- Build Modern, Responsive Web Applications
- Scalable JavaScript Application Architecture

Microapp Approach

A single-page application uses Asynchronous JavaScript (AJAX) requests and HTML5 to create fluid and responsive user interactions without constant page reloads. However, much of this work running in a browser occurs as complex state transitions following the Model View Controller architecture (see Modern Web Application for more information):

- There are global state changes (such as going offline in a real time application) .
- There are delayed results from AJAX requests that get returned at some point from back end operations
- There are changes to the Model function as values change.
- There are Domain Object Model (DOM) events that cause small state changes in the View function.
- There are changes to the Controller function that cause state changes in the View function..

The microapp approach implements a conventional single-page application as a suite of microservices. Microservices are atomic, self-contained services that typically perform a single operation on a back-end system, such as a retrieving a customer record.

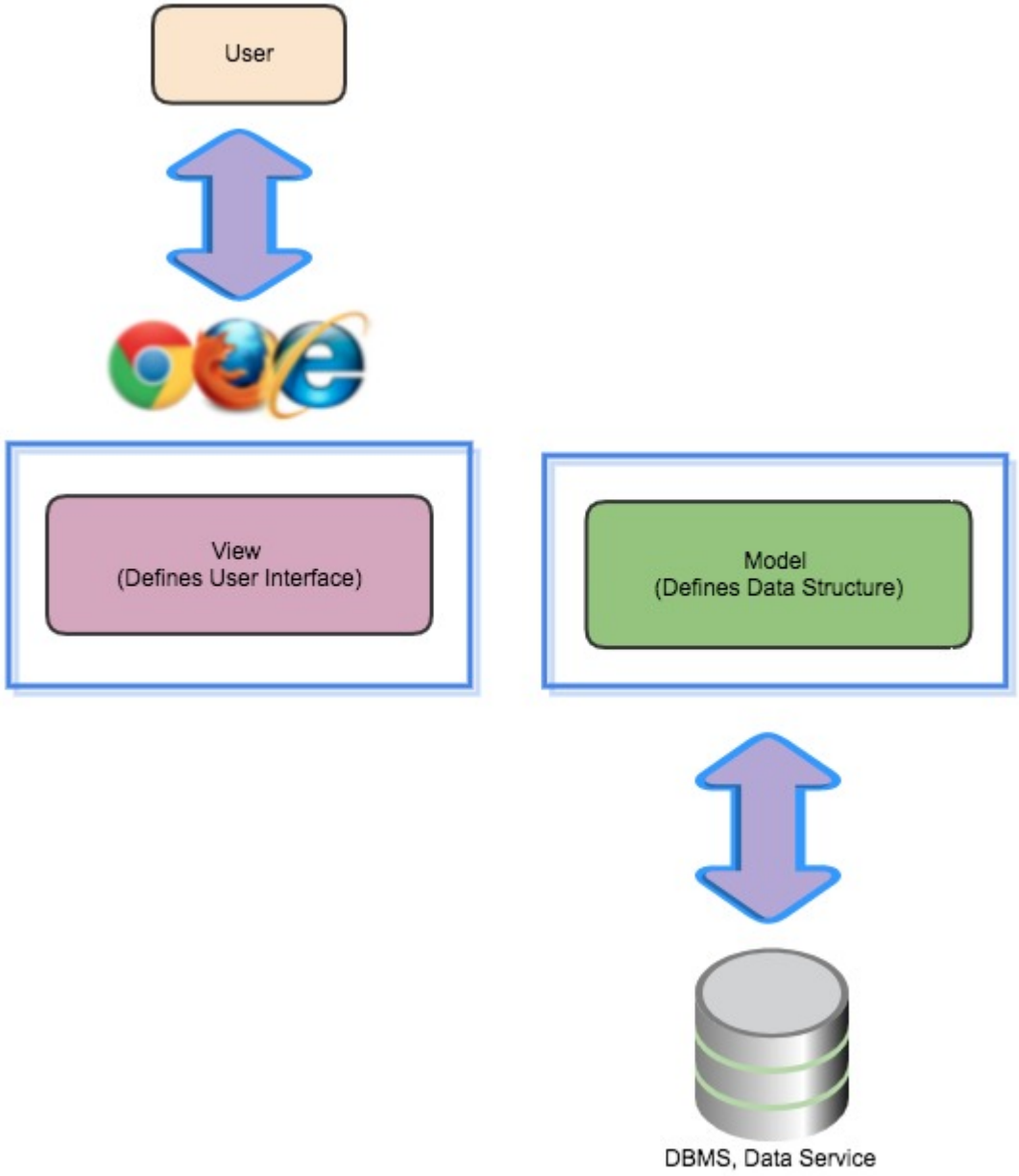

Figure 3: Microapp Approach

This means that microapps provide a range of capabilities that can be focused on specific tasks. For example, end users benefit from a View function that is tailored to their specific requirements.

See the following resources for more information:

- Why Application Development is Going Micro
- Understanding the Micro Apps Trend in Application Development
- About Microapps and Predix Development

About the Predix Design System

The Predix Design System gives developers and designers the ability to create and customize the user interface of a modern web application that runs on top of Predix data and services..

Predix Design System Features

The Predix Design System (https://www.predix-ui.com) includes these features to build and customize the user interface of an industrial web application:

- A set of web components based on the Polymer library that simplify the effort to build web applications that provide many of the characteristics of desktop software across supported web browsers (see Supported Browsers for Web Applications on page 9 for more information).
- A set of CSS modules to customize the base typography, layout, and content design in a web application..
- A gallery of available web components and CSS modules.
- A set of layout examples that combine branding, navigation, routing, asset selection, data visualization, and other Predix UI components for common views in a Predix application.
- A design starter kit that supports quick assembly of components in supported configurations of a web application.

See the video Introduction to Predix UI for more information.

Predix Design System Benefits

As a developer or designer, the Predix Design System provides these benefits:

Fundamental Capabilities

The Predix Design System provides the following fundamental capabilities:

- Data-first visual design language that focuses on making information meaningful and useful for industrial users.
- Responsive web design that provides a consistent viewing and interaction experience across laptop, tablet, phone, or large-screen devices (see Supported Browsers for Web Applications on page 9 for more information).

Charts

The Predix Design System provides the data-driven Predix UI Viz charting framework that uses the D3.js JavaScript library and Polymer web components. You do not need to know D3.js to use the Predix UI Viz framework. If you have a knowledge of Polymer, you can build a new chart using the building blocks and charts that are provided with the Predix UI Viz framework. See Predix UI Vis Framework for more information.

Maps

The Predix Design System provides basic mapping functionality (such as markers, clusters, popups, locator, panning/zooming, and the ability to apply layers). The `px-map` component provides the following benefits:

- Developers can provide users the ability to visualize the geographic location and other information about assets or asset groups on a map.
- Developers can integrate applications with a preferred mapping/GIS tile service.

Modular Architecture

The Predix Design System provides a set of repositories in the `PredixDev` public GitHub organization that allows you to add only the features and functions that you need. The modular architecture takes advantage of defined variables and default variants that provide the flexibility to change and configure existing applications, rather than use a large, complex, single-component repository.

Framework Agnostic

Several JavaScript MVC frameworks exist (such as AngularJS and Vue.js) to encourage developers to write more structured code for client-side rendering of HTML pages. While each has its unique advantages, they all can use the Predix Design System repositories to quickly create web applications.

See the following resources for more information:

- Predix Design System: Components that Workk in Your Framework
- Use Predix Design System with AngularJS
- Use Predix Design System with Vue.js

Application Development with the Predix Design System

As a designer or developer, you should review these technologies before using the Predix Design System to build and customize the user interface of a web application that runs on top of Predix data and services.

Open-Source Standards

As a designer or developer, you should review the conceptual, task, and reference information for the following open-source standards before using the Predix Design System (https://www.predix-ui.com/#/home):

- HTML5 – See the following resources for more information:
  - HTML5 Reference
  - HTML5 Developer Guide
  - HTML Code Guidelines
- CSS (Cascading Style Sheets) – See the following resources for more information:
  - Cascading Style Sheets Reference
  - CSS3 Developer Guide
  - CSS Code Guidelines
- JavaScript – See the following resources for more information about commonly supported features:
  - JavaScript Reference
  - JavaScript Developer Guide
  - JavaScript Code Guidelines

Design Tools

As a designer, you should review the conceptual, task, and reference information for the following design tools before using the Predix Design System:

- Design guidelines that describe best practices to assist designers make key decisions about building and extending the user interface of a web application.
  - Color
  - Materiality

    - Elevation and Layering
    - Data Visualization
    - Forms
    - Navigation
    - Mobile
    - Page Layout
    - Layout Examples
    - Typography
    - Iconography
    - Writing Style for Components
- A set of example layouts that combine branding, navigation, routing, asset selection, data visualization, and other Predix UI components for common views in a Predix application.
- A design starter kit that supports quick assembly of Predix UI components in supported configurations of a web application.

Developer Tools

As a developer, you should review the conceptual, task, and reference information for the following open-source tools before using the Predix Design System :

- GitHub code repository
- Node.js runtime environment
- Bower package manager
- Gulp build system

GitHub Code Repository

As a developer, you use GitHub as a source code repository and version control system. The Predix Design System is modular by design, allowing you to add only the features and functions that you need from the individual repositories available in the `PredixDev` public GitHub organization..

See the following resources for more information about the GitHub code repository:

- GitHub Overview
- GitHub Documentation

Node.js Runtime Environment

As a developer, you use Node.js as a cross- runtime environment to build web applications. Node.js includes the `npm` utility to manage node modules such as the Bower package manager and the Gulp build system.

See the following resources for more information about the Node.js runtime environment:

- Node.js Overview
- Node.js Documentation

Bower Package Manager

As a developer, you use the Bower package manager to install the correct package versions needed to configure a web application (such as HTML, CSS, or JavaScript files) and their dependencies. Bower fetches and installs the required packages, then records the package dependencies in a `bower.json` manifest file.

See the following resources for more information about the Bower package manager:

- Bower Overview
- Bower Documentation

- What You Nneed to Know about Predix UI, Bower, and SemVer

Gulp Build System

As a developer, you use the Gulp build system to start a web server that runs tasks and serves files to build a web application. This web server can be accessed by entering `http://localhost:9000/` in a web browser .

See the following resources for more information about the Gulp build tool:

- Gulp Overview
- Gulp Documentation

Related Resources

- Create Modern Web Applications on page 1
- About the Predix Design System on page 6
- Using the Predix Design System on page 12
- Developing with the Predix Design System

Supported Browsers for Web Applications

The Predix Design System has been tested to support these browsers.

The following minimum versions of browser software have been fully tested to support all Predix Design System repositories that are available in the `PredixDev` public GitHub organization:

- MacOS: Last two versions of Safari, Chrome, and Firefox
- Windows:
  - Last two versions of Edge, Firefox, Chrome (Wndows)
  - Internet Explorer 11
- Android: Last two versions of Chrome (Android)
- iOS: Last two versions of Safari

Note: File a GitHub issue if you encounter a problem using a Predix Design System repository with a supported web browser. For more information on filing a GitHub issue, see https://guides.github.com/features/issues/.

Some Predix UI components use EcmaScript 2015 (aka ES6) JavaScript features supported by the following browsers:

- Chrome or Chromium version 49 or later
- Safari or Mobile Safari 10 or later
- Edge 15.15063 or later
- Firefox 51 or later

See Transpile ES6 to ES5 for more information on browsers that require code transpilation to parse ES6 components.

Predix Design System Glossary

As a developer, you should understand these Predix Design System terms and definitions.

Component

A component is a user interface element that enables a user to perform a function or access a service. It defines the fundamental interactions between presentation, interaction and behavior with context, data-binding and services.

Components adhere to the following principles of workflow and data aggregation at a granular level:

- Definition of dependencies, behavior, styling and template using simple HTML syntax (for example, `<px-datepicker/>`.
- Rendering of "sketch" (minimally-styled) and "Predix" (fully-styled) themes.
- Independent releases and versions.
- Ability to inherit functionality from other components by either extension or composition.
- Nesting as part of another component that can exercise its public API.
- Ability to expose an API used by an application framework such as AngularJS. This includes scope and bidirectional binding.

Card

A card is a component that can be shared across different display environments. A card acts as a container that resides within the content area of a web browser. It can contain embedded data for use when connectivity is limited or cut off. A developer can show and hide cards as needed.

A card can receive and transmit events, and is both aware of and reactive to the context in which it resides. Components can interact within an individual card, and a card can interact with other cards. You can also create a deck from a collection of cards to create more intricate user interactions. See Getting Started with the Views Service Database for information about how cards and decks are used.

Deck

A deck is a component that uses a specified layout to determine the arrangement of cards. See Getting Started with the Views Service Database for information on how cards and decks are used.

Module

A CSS *module* is a CSS file in chich all class names and animations names are scoped locally by defaul.

View

A view is the visual summary of the information that is displayed in a web application. A view can have different decks (cards and other components arranged in a specified order) that a user can select at run time.

Basics

A basic component is a building block of user interface code. Simple, yet powerful, a basic component can be used in isolation or combined to create more complex components.

Template

A template component is a layout of an entire screen or region of a screen. A template component focuses on the content structure or layout, not the actual content itself.

Context

Context is the environment and manner in which the object or system is being used by the user or by other parts of the system. It is represented by its own current state and the state of the system around it. It can take into account intent and history. An object or entity can have context, be part of context, or be context itself by meeting these requirements.

A context-aware dashboard (contextual dashboard) can behave and look differently when you combine any of the following factors:

- User or persona.
- Base object which can be represented by the dashboard, or on which the dashboard may be acting.
- Time (such as weekday, AM/PM, year).
- Device.
- Roles and privileges.
- User group.
- State of the system, device, and user.
- Personal preferences.
- Sensory factors.
- Environment.
- Language.
- Locale.
- Writing system (such as phonetic, alphabetic, pictographic, or ideographic).

Additional Information

- Front End Microservice Template
- Predix Web App Starter

Use the Predix Design System

Using the Predix Design System

As a developer or designer, use the Predix Design System to build and customize the user interface of an industrial web application that runs on top of Predix services and data.

About This Task

The following tasks show how to get started with the Predix Design System (https://www.predix-ui.com).

Table 1: Getting Started with the Predix Design System

#	Task	Information
1	As a designer or developer, explore the Predix Design System as a data visualization framework for industrial web applications.	See Introducing GE's data visualization framework for IoT.
2	As a designer or developer, explore Predix UI components and CSS modules that are available in the Predix Design System.	See Predix Design System Gallery.
3	As a designer or developer, explore example layouts that combine branding, navigation, routing, asset selection, and data visualization for common views in a Predix application.	See Predix Design System Layout Examples.
4	As a designer or developer, explore the features and enhancements available in Predix Design System Cirrus.	See Migrating to Predix Design System Cirrus on page 15.
5	As a developer, set up the Predix Design System developer environment.	See Setting Up the Predix Design System Developer Environment on page 13.
6	As a developer, use the Predix UI components in the `PredixDev` public GitHub organization for common user interactions (such as application navigation and context browsing).	See Getting Started with Predix UI Components on page 27.
7	As a designer, use the design guidelines and starter kit that support quick assembly of Predix UI components in supported configurations of a web application.	See Design Tools on page 7

After completing the getting started tasks, you can follow these optional developer tasks that describe how to use the Predix Design System in a web application:

Table 2: Using the Predix Design System

#	Description	Information
1	Download the code of an application example to run in a local developer environment or Cloud Foundry.	See Sample Application.
2	Localize a Predix UI component for a specific country or region	See Localizing Predix UI Components on page 39.
3	Change the values of CSS custom properties defined for Predix UI components to apply a theme to a web application.	See Theming Web Applications on page 51.

#	Description	Information
4	Use the Predix UI CSS modules in the `PredixDev` public GitHub organization to satisfy custom styling requirements.	See Getting Started with Predix UI CSS Modules on page 56
5	Use the Predix UI Vis framework to create a custom chart.	See Predix UI Vis Framework on page 35.
6	Build a custom Predix UI component to satisfy specific application requirements. .	See Px Component Generator.

Setting Up the Predix Design System Developer Environment

The Predix Design System developer environment requires Node.js and Bower.

Before You begin

You should understand the conceptual, task, and reference information needed to use the Predix Design System to create web applications with the following open-source tools:

- GitHub code repository
- Node.js runtime environment
- Bower package manager
- Gulp build system

For more information, see Developer Tools on page 8.

Accounts

You need to register for the following accounts to use the Predix Design System in a web application:

- A Predix.io account. Go to https://www.predix.io/registration/.
  When you register for a Predix.io account, an org and space is created for you in Cloud Foundry.
- A Github account to download the Predix Design System repositories in the `PredixDev` public GitHub organization.. Go to https://github.com/join.

Software

You need to set up these common tools to use the Predix Design System in a web application:

Table 3: Common Tools for Web Application Development

Software	Version	Description
Cloud Foundry CLI	Latest stable binary version	Use the Cloud Foundry CLI to deploy and manage applications and services. Download the latest stable binary from https://github.com/cloudfoundry/cli#downloads.
Git CLI	Latest	Use the Git CLI to download Predix Design System repositories in the `PredixDev` public GitHub organization. Download the latest stable binary from https://git-scm.com/downloads.

Procedure

1. Install the latest version of the Node.js runtime environment for your operating system. See https://nodejs.org/en/download/.
2. Modify the file permissions for the Node.js `npm` package-management utility. See https://docs.npmjs.com/getting-started/fixing-npm-permissions.
3. Install the `gulp` build system and `bower` package manager.

```
npm install gulp gulp-cli bower -g
```

Migrate to Predix Design System Cirrus

Migrating to Predix Design System Cirrus

Predix Design System Cirrus updates the existing Predix Design System Classic GitHub repositories in the `predixdesignsystem` public organization to achieve the following goals:

- Establish a more visually sophisticated design language.
- Increase the diversity of design patterns.
- Involve the design community to improve quality and adoption..
- Provide better guidance about how to use the Predix Design System.

As a developer, migrating from the Predix Design System Classic GitHub repositories to the Predix Design System Cirrus GitHub repositories involves the following global changes:

- Default Predix color values have been removed from all of the Predix UI component CSS variables. Without a theme, components will appear with a mostly monochromatic "generic" theme.

  Note: Default Predix color values have not been removed from the Predix Design System Cirrus CSS module repositories.
- You should include either the `px-theme` or `px-dark-theme` style modules in your application to style Predix UI components (or create and include your own theme). See Theming Web Applications on page 51 for more information.
- All of the Font Awesome icons have been replaced with custom Predix Design System icons in the `px-icon-set` repository. You should include both `px-icon-set.html` and `px-icon.html` in your web application. See Migrate to New Icons for more information.
- You should review the Vis Migration Guide to determine if there are any changes needed to upgrade any data visualization components.

Use the Version Finder tool to determine which GitHub repository (Predix Design System Classic or Predix Design System Cirrus) applies to your web application depending on your needs:

- Predix developers maintaining an existing web application will need to research which Predix Design System Classic GitHub repositories are used before upgrading to the Predix Design System Cirrus GitHub repositories.
- Predix developers building a new web application should only use Predix Design System Cirrus GitHub repositories.

Related reference

New Predix UI Components for Predix Design System Cirrus on page 17

Deprecated Predix UI Components for Predix Design System Cirrus on page 18

Predix Design System Cirrus Design Changes on page 18

Predix Design System Cirrus API Changes on page 19

Using the Predix Design System Classic Repositories

Before You Begin

As a developer, you use the Bower package manager to install the correct package versions needed to configure a web application (such as HTML, CSS, or JavaScript files) and their dependencies. Bower

fetches and installs the required packages declared in the `bower.json` file. See Application Development with the Predix Design System on page 7 for more information.

Using a Predix Design System Classic repository in an existing web application

1. Use the Version Finder tool to determine which Predix Design System Classic repository in the `predixdesignsystem` public organization is used in an existing web application.
2. List GitHub repositories as dependencies in the `bower.json` file of your web application.
3. Verify that `bower.json` file includes either Predix Design System Classic GitHub repositories or Predix Design System Cirrus GitHub repositories.

   Note: Predix Design System Classic GitHub repositories and Predix Design System Cirrus GitHub repositories should not be mixed.
4. Copy the version ranges of the GitHub repositories that are used from the Version Finder tool into your `bower.json` file to ensure your builds are consistent.

Ugrading a Predix Design System Classic repository in an existing web application

Predix developers who use the Predix Design System Classic repositories in the `predixdesignsystem` public GitHub organization in an existing web application will need to research the necessary changes to upgrade to the Predix Design System Cirrus GitHub repositories.

- New Predix UI Components for Predix Design System Cirrus on page 17
- Deprecated Predix UI Components for Predix Design System Cirrus on page 18
- Predix Design System Cirrus Design Changes on page 18
- Predix Design System Cirrus API Changes on page 19

The necessary changes to upgrade all the Predix Design System Classic repositories in an existing web application should be made as soon as possible (see Using Predix Design System Cirrus Repositories on page 16 for more information).

Using Predix Design System Cirrus Repositories

As a developer, you should build a new web application with the Predix Design System Cirrus GitHub repositories in the `predixdesignsystem` public organization:

- To use the Predix Design System Cirrus GitHub repository for a Predix UI component in a new web application, see Using a Predix UI Component in a Web Application on page 27.
- To use the Predix Design System Cirrus repository for a Predix UI CSS module , see Using a Predix UI CSS Module in a Web Application on page 58.

Note: Use the Version Finder tool to determine which Predix Design System Cirrus GitHub repository to use (depending on your needs).

Predix Design System Cirrus FAQ

Q: Can I take one of the Predix Design System Cirrus GitHub repository without upgrading all of my Predix Design System Classic GitHub repositories?

A: No. There are dependencies between Predix Design System Cirrus GitHub repositories, so this is not recommended. You should upgrade your web application to use the Predix Design System Cirrus GitHub repositories in order to ensure a harmonious user experience.

Q: I need the Predix UI team to make an enhancement to a Predix Design System Classic GitHub repository. Can you do this on the Predix Design System Classic GitHub repository?

A: No. the Predix UI team is only maintaining and enhancing Predix Design System Cirrus GitHub repositories going forward. Please contact the Predix UI team if there is an urgent business case for your requirements.

Q: I'm using a Predix Design System Classic GitHub repository. Can I still use this version and make a fix or enhancement to it myself?

A: Yes you can, but you should also make the fix or enhancement in the Predix Design System Cirrus GitHub repository.

New Predix UI Components for Predix Design System Cirrus

Predix Design System Cirrus includes new Predix UI component repositories in the `predixdesignsystem` public organization. To use a new Predix UI component, see Using a Predix UI Component in a Web Application on page 27.

px-accordion

- A user interface component that provides expandable and collapsible subsection headers for a page. An optional "action" icon on the right side will fire an event when pressed.

See Predix UI Components on page 31 for the component API reference that includes more details and code examples.

px-app-header

- A user interface component that allows the `px-branding-bar` and `px-app-nav` components to work together.

See Predix UI Components on page 31 for the specific component API reference that includes more details and code examples.

px-app-route

- A user interface component that supports easy binding between various Predix UI components and a web application's URL. It helps keep the user interface state in sync with the URL by allowing users to bookmark or share views by copying the URL

See Predix UI Components on page 31 for the specific component API reference that includes more details and code examples.

px-branding-bar

- A user interface component that provides a header area to contain title, logo and branding content.

See Predix UI Components on page 31 for the specific component API reference that includes more details and code examples.

px-breadcrumb

- A user interface component that works with the `px-context-browser` and `px-tree` components to visually represent the asset model hierarchy and jump to levels of the hierarchy.

See Predix UI Components on page 31 for the specific component API reference that includes more details and code examples.

px-heatmap-grid

- A flexible and interactive data visualization component for a data set based on a heat map scale of colors. See the API Documentation for details.

See Predix UI Components on page 31 for the specific component API reference that includes more details and code examples.

px-icon-set

- A user interface component that displays Predix icons. The icons are optimized for different sizes based on their category or intended usage.

See Predix UI Components on page 31 for the specific component API reference that includes more details and code examples.

px-key-value-pair

- A user interface component that allows you to prominently display information in a dashboard.

See Predix UI Components on page 31 for the specific component API reference that includes more details and code examples.

px-tree

- A user interface component that provides an expandable and selectable tree. The contents are controlled primarily through a structured object of branch and leaf nodes.

See Predix UI Components on page 31 for the specific component API reference that includes more details and code examples.

Deprecated Predix UI Components for Predix Design System Cirrus

Predix Design System Cirrus has deprecated certain Predix UI component repositories in the `predixdesignsystem` public GitHub organization.

px-drawer

- Use the `app-drawer` element instead: https://github.com/PolymerElements/app-layout/tree/master/app-drawer

px-input-group-design

- No longer maintained going forward. Superseded by the Predix Design System Cirrus `px-forms-design` GitHub repository.

px-localize-behavior

- The `app-localize-behavior` element is now used internally instead (https://github.com/PolymerElements/app-localize-behavior).

px-sample-cards

- Use the Sample Application instead (https://www.predix-ui.com/#/about/sample-app).

Predix Design System Cirrus Design Changes

Predix Design System Cirrus includes updates to the existing Predix Design System Classic GitHub repositories in the `predixdesignsystem` public organization that achieve the following goals:

- Establish a more visually sophisticated design language.
- Increase the diversity of design patterns.
- Involve the design community to improve quality and adoption.
- Provide better guidance about how to use the design system

Predix Design System Cirrus includes these design changes to extend and customize the user interface of a web application:

- A set of design guidelines that describe best practices and assist designers and developers in making key decisions about extending and customizing the user interface of web applications.
  - Navigation
  - Page Layout
  - Materiality
  - Typography
  - Iconography
  - Elevation and Layering
  - Data Visualization
  - Mobile
- A design starter kit that supports quick assembly of components in supported configurations of a web application.
- An application example that combines branding, navigation, routing, asset selection, data visualization, and other components to illustrate several layouts that are possible with the Predix components.

Predix Design System Cirrus API Changes

Some of the repositories in the `predixdesignsystem` public GitHub organization include API changes for Predix Design System Cirrus.

px-alert-label (v2)

- A new boolean property `badge` has been added for enabling the alert/severity badges currently used in the `px-alert-message` and `px-inbox` components - the `type` property determines what color and shape will appear, whereas the `label` property can be repurposed for the (single-digit) numeric indicator.
- The `type` value "info" was expanded to "information" in order to match the API of the `px-alert-message` component. However, either value will work as expected.

See Predix UI Components on page 31 for the specific component API reference that includes more details and code examples.

px-app-nav (v2)

- The `px-app-nav` API, design and usage was completely overhauled for the post-Cirrus version, see the API documentation for details..
- Clicking on an item in `px-app-nav` no longer automatically navigates to the item's URL by setting the `window.location`. Instead, listen to the event `selected-changed` for updates when an item is selected and sync the state to the URL if you choose (e.g. by setting `window.location`, using a framework router, or using the new `px-app-route` helper element).
- The `navItems` property has been renamed to `items`. The format of the navigation items object has also changed.
- In `navItems`, child items were added to a parent item with the `subitems` key. In `items`, child items are added to a parent item with the `children` key.
- In `navItems`, item paths were set with the `path` key. In `items`, item paths are set with the `id` key.
- In `navItems`, an `eventName` used to be passed so that it would be fired when the item was tapped. That option has been removed from `items`. Instead, watch `selected-changed` for updates when a new item is selected (when tapped or through data binding).

- All icon names used in `navItems` should be updated to use the new `px-icon-set` names. If you do not update to use the `px-icon-set` names, you will need to load the Font Awesome icons in your application. The Font Awesome icons will not be loaded with the `px-app-nav` component by default.
- The `navContracting` and `navExpanding` properties have been removed. See the `collapseOpened` property for an approximate replacement.
- The `pathPrefix` property has been removed as `px-app-nav` no longer navigates the page directly when an item is clicked. You can implement this yourself by listening to `selected-changed` and setting the URL
- The `pathKey` property has been removed.
- The `markSelected` method has been removed. Use the `select()` method to select an item by reference.
- The navigation is no longer configured to be vertical and toggle to open/close by default. There are now a variety of modes available (horizontal, vertical, collapsed) for different use cases. To use the vertical mode, set the `vertical` property to `true`. The vertical mode is no longer meant to be used on mobile. The `collapsed` mode should be used on mobile instead. See the API documentation for details.

See Predix UI Components on page 31 for the specific component API reference that includes more details and code examples.

px-calendar-picker (v1)

- Updated the style for `today` resulting in `--px-calendar-today-border` being added.
- The CSS variable `--px-calendar-labels-text-color` was updated to `--px-calendar-title-color`. The CSS variables `--px-calendar-background-color`, `--px-calendar-background-color-selected-hover`, and `--px-calendar-background-color-selected-pressed` have been removed due to design changes.

px-card (v1)

- The `chevron` property has been removed,. Instead, a content slot has been added so that anything included in a container of class "actions" will be displayed in the top right corner of the card in the header area. See the API referencr documentation for an example implementation.
- A new property called `fullBleed` has been added, which causes the content area of the card to stretch the full width of the card, removing the default horizontal padding. This is useful for cards containing maps, images, and other content that requires more horizontal space.

See Predix UI Templates on page 30 for the specific component API reference that includes more details and code examples.

Note: The `px-card` component no longer supports Views service behaviors:

- `window.px.card` behavior removed
- `window.px.deck` behavior removed
- `window.px.dashboard` behavior removed
- `window.px.dealer` behavior removed
- `px-deck` component removed
- `px-dashboard` component removed
- `px-sample-card` component removed
- `px-card/px-dashboard.html` file deleted
- `px-card/px-deck.html` file deleted
- `px-card/px-sample-card.html` file deleted

Use the Version Finder tool to determine that you do not import any deleted files or rely on any removed behaviors/components in an existing application.

Note: As a developer, you can upgrade to the Predix Design System Cirrus repository of the `px-card` component and use the preprecated Views service behaviors by following these steps:

1. Copy the `px-card-view-service-behaviors.html` file from Github (see commit with relevant file: https://github.com/predixdesignsystem/px-card/commit/ebb3b0a785d675e874917b5d57881c630c2daca3)
2. Place the file in their application repository and check it in to source control.
3. Load the local `px-card-view-service-behaviors.html` behavior before loading any code that relies on it.

See Using the Predix Design System Classic Repositories on page 15 for more information about migrating an existing web web application to use a Predix Design System Cirrus repository.

px-colors-design (v 1.0.1)

- All of the SASS variable names have changed to support the completely redesigned Predix Design System color palette..

See Predix UI CSS Visual Library on page 59 for the specific module API reference that includes more details and code examples.

px-context-browser (v 2.0)

- The `px-context-browser` API, design, and usage was completely overhauled for the post-Cirrus version. See the API documentation for details.
- The `<px-context-browser></px-context-browser>` tag no longer places a button on the page. Use the `px-context-browser` tag together with a `<px-context-browser-trigger></px-context-browser-trigger>` tag show a button that opens the browser. See the API documentation for details on binding these two elements together.
- The `browserContext` property has been renamed to `items`. The structure of the context browser items has been completely overhauled.
- The `browserContext` property `name` has been replaced with `label`.
- The `` `browserContext `` property `identifier` has been replaced with `id`.
- The `browserContext` property `isOpenable` has been removed.
- The initial context and direct context modes have been removed. Set the `selected` property instead to select an item after loading the data.
- The `disabled` property has been removed.
- The `disableInfiniteScroll` property has been removed.
- The `handlers` property has been removed. Listen to the event `px-app-asset-children-requested` to be notified when new children should be loaded. Listen to the event `selected-changed` to be notified when an item is selected (tapped or selected through data binding). There is no event fired when an item is tapped but not selected as that interaction pattern has been removed. See the API documentation for details.
- The `idField` and `labelField` properties have been removed. Use the `` `keys` `` property to set both instead.
- The `openedItemName` property has been removed. Listen to the event `selected-changed` to be notified when a new item is selected
- The `resize` property has been removed.
- The `selectedItem` property has been renamed to `selected`. Its behaviors are also slightly different. See the API documentation for details.

- The `showChevron` property has been renamed to `showArrow`.
- The `showColumnBrowser` property has been removed.
- The `showColumnTypeahead` property has been removed. Use the `showFilter` property to show a filter field at the top of the column.
- The `addParents` method has been removed. Instead, add items from the root of the graph using `addChildren`.
- The `toggleColumnBrowser` method has been removed. Toggle the `collapsed` property to show or hide the context browser.
- The events have changed completely. See the API documentation for details.

See Predix UI Components on page 31 for the specific component API reference that includes more details and code examples.

px-data-table (v 4)

- The "rows per page" dropdown has been updated to use the `px-dropdown` component for cross-browser consistency.
- When a custom `pageSize` value is passed into the table, it will be displayed as the selected choice in the "rows per page" dropdown.
- A new property for `pageSizeOptions` has been added, which allows developers to specify their own array of options for the "rows per page" dropdown. Add the same option to both `pageSize` and `pageSizeOptions` to make it both the default choice as well as a persistent option.
- The `tableRows` property has been removed, as horizontal borders are now the default style for Predix Design System tables. It has been replaced with a `tableCells` property, which can be used to create the previous default style, a table with both horizontal and vertical borders.

See Predix UI Components on page 31 for the specific component API reference that includes more details and code examples.

px-datetime-field (v1)

- The `allowWrap` property has been added to allow the buttons, if showing, to wrap under the field.
- The CSS variables for the height `--px-datetime-field-height` and background color `--px-datetime-field-background-color` have been added. The validation design has been updated; therefore, `--px-datetime-border-color-validation-failed` has been removed.

px-datetime-range-field (v1)

- The `allowWrap` property has been added to allow the second field and buttons, if showing, to wrap under each other. The `hideIcon` has been exposed to hide the calendar and clock icons if desired.
- The validation design has been updated; therefore, `--px-datetime-range-field-text-color-validation-failed` has been removed.

datetime-range-panel (v1)

- The z-index value has been exposed in the `--px-range-panel-z-index` CSS variable.
- The validation design has been updated; therefore, `--px-datetime-range-field-text-color-validation-failed` has been removed.
- Presets can now be functions. See the API documentation for an example (https://www.predix-ui.com/#/elements/datetime/px-datetime-range-panel:property-presetRanges) for an example.

px-dropdown (v1)

- The `px-dropdown-content` subcomponent no longer exists. Several properties have been moved to the main `px-dropdown` component.
- Most of the CSS variables for `px-dropdown` have gone away. Styling for the invoking element can still be achieved using the `buttonStyle` property and the CSS variables for buttons.
- The `boundTarget` property now expects a CSS selector string instead of a reference to the actual HTML element that will be used for the outer bounds of the dropdown.
- The `checkboxMode` property is now called `multi`.
- By default, the dropdown button will display the selected item or number of selected items. Use the `hideSelected` property to disable this behavior in use cases such as the `px-data-table` component, where the `displayValue` of "Show/Hide Columns" should always be displayed.
- The `displayValue` property is no longer overwritten when the user selects an item. In `multi` mode, the `displayValue` is appropriately restored when all selections are cleared.
- A new clear button has been added for quickly de-selecting all values and restoring the dropdown to its default state and `displayValue`. This also allows users to clear the dropdown when not in multi- select mode, which was not previously possible
- When `multi` is false (i.e. single-select mode), use `selectBy` to determine whether a "key" or "val" will be passed to the `selected` property. When `multi` is true, use `selectBy` to determine whether an array of "key" or "val" items will be passed to the `selectedValuee` property. In either single or multi mode, `selectedItems` property will contain a reference to the actual `<div>` elements in the dropdown that have been selected.
- Set `sortMode` to "key" or "val" to sort the dropdown options. By default, they will be displayed in the order they are passed in to the `items` array. The `sortCheckedItems` and `sortDisabledItems` properties have been removed for simplicity.
- The `px-dropdown-value-changed` and `px-dropdown-checkbox-changed` events have been consolidated into a single `px-dropdown-selection-changed event`, which will fire in either single or multi-select mode.
- All of the orientation and sizing logic has been simplified by the use of the `iron-dropdown` element instead of the custom code in previous versions of the `px-dropdown` component:.
- The styling for the exterior borders and interior borders have been separated to enable the new design (similar to the `px-tables-design` repository).The properties for `extendDropdown`, `extendDropdownBy`, `maxContentCharacterWidth`, and the use of the px-tooltip component have all been removed.
- The styling for the exterior borders and interior borders have been separated to enable the new design (similar to the `px-tables-design` repository). The CSS variables have been separated accordingly to distinguish between `--px-data-table-border-color` (exterior) and `--px-data-table-separator-color` (interior).
- Keyboard support has been added for accessibility.

See Predix UI Components on page 31 for the specific component API reference that includes more details and code examples.

px-forms-design (v2)

- Many of the CSS variables for the form inputs have been removed, changed, or renamed based on the redesigned components. See the API reference documentation for the full list of updated CSS variables.

See Predix UI CSS Visual Library on page 59 for the specific component API reference that includes more details and code examples.

px-headings-design (v1)

- In addition to the six heading sizes previously available, new classes have been added to enable aspects of the Predix Design System Cirrus refresh. New heading styles include `heading-page` (large), `heading-section` (all caps), and `heading-subsection` (smaller, with a background). Additional styles have also been added for the `label/value` pairs often seen in Predix applications with the classes label and value.

See Predix UI CSS Visual Library on page 59 for the specific module API reference that includes more details and code examples.

px-icon-set (v2)

- The `px-polymer-font-awesome` repository has been renamed to `px-icon-set`, and all of the Font Awesome icons have been replaced with custom Predix UI icons created during the Cirrus refresh. The API has also changed (`<px-icon icon="">` is now used), and all of the new icons fall into three different namespaces (`pxs:icon`, `pxm:icon`, and `pxl:icon` as opposed to the previous `fa:fa-icon`) and new icon names. Both `px-icon-set.html` and `px-icon.html` should be imported in your application. See the API reference documentation for the full icon list and cheat sheet.

See Predix UI Components on page 31 for the specific component API reference that includes more details and code examples.

px-inbox (v2)

- The `height` property has been removed in favor of the `--px-inbox-height` CSS variable, which defaults to 100vh. Also, a `--px-inbox-list-width` variable has been added, which defaults to 320px (previously the list was set to take up 1/3 of the total component).
- List items can now be passed without a severity, in which case an icon will not appear next to the item.
- A new boolean property called `disableAutoSelect` has been added which will prevent the `px-inbox` component from automatically selecting the first item in the list. A check was also added so that when a `selectedItem` is passed into the `px-inbox` component, it will not be overridden by this behavior
- Both ascending and descending sort are now possible using either the `sortOrder` property or the arrow icon next to the sort dropdown.

See Predix UI Components on page 31 for the specific component API reference that includes more details and code examples.

px-list-ui-design (v2)

- Similar to the `px-tables-design` repository, the styling for the top/bottom borders and interior borders have been separated to enable the new design. The CSS variables have been separated accordingly to distinguish between `---pxlist-ui-border-color` (top/bottom borders) and `--px-list-ui-separator-color` (interior borders).

See Predix UI CSS Visual Library on page 59 for the specific module API reference that includes more details and code examples.

px-overlay (v1)

- The `type` property and the two corresponding CSS variables have been removed. Use the new `--px-overlay-color` CSS variable with an rgba value to adjust the color and opacity of the overlay.

See Predix UI Components on page 31 for the specific component API reference that includes more details and code examples.

px-progress-bar (v1)

- The `striped` and `animated` properties have been removed as part of the Cirrus refresh. An `infinite` property takes the place of both properties for distinguishing between a determinate/measurable progress bar that fills from left to right (when "false" or omitted) and an indeterminate/infinite progress bar (when "true") that continually animates when a discrete percentage cannot be calculated.

See Predix UI Components on page 31 Predix UI Components for the specific component API reference that includes more details and code examples.

px-rangepicker (v2)

- The overlay and `--px-datetime-picker-overlay-color` have been removed.
- The `allowWrap` property has been added to allow the second field to wrap.

px-tables-design (v2)

- The `table-rows` flag and class have been removed, as horizontal borders are now the default style for Predix Design System tables. It has been replaced with a `table-cells` flag and class, which can be used to create the previous default style, a table with both horizontal and vertical borders.
- Styling for the exterior borders and interior borders have been separated to enable the new Predix Design System Cirrus refresh for tables, which features a darker top and bottom border and lighter interior borders. The CSS variables have been separated accordingly to distinguish between –-px-table-border-color (top/bottom) and `--pxtable-separator-color` (interior borders).

See Predix UI CSS Visual Library on page 59 for the specific module API reference that includes more details and code examples.

px-tabs (v2)

- The `px-tabs-set` repository and parent component has been renamed to `px-tabs` (the subcomponent name is still `px-tab`).
- The `bare` and `noBottomBorder` properties have been removed. The bare style is now the default/only appearance for Predix Design System tabs. Several of the related CSS variables have also been removed.

See Predix UI Components on page 31 Predix UI Components for the specific component API reference that includes more details and code examples.

px-tooltip (v1)

- The e`tooltip-style` property has been removed. See Theming Web Applications on page 51 for more information on using the px-theme and px-dark-theme style modules, as well as use the provided CSS variables to override the style of the component

See Predix UI Components on page 31 for the specific component API reference that includes more details and code examples.

Get Started with Predix UI Components

About Predix UI Components

The Predix Design System includes a set of reusable web components in the `predixdesignsystem` public GitHub organization to add features or access services following consistent UX practices, standards, and specifications.

The Predix UI components simplify the effort to build web applications that provide many of the characteristics of desktop software across supported web browsers (see Supported Browsers for Web Applications on page 9 for more information).

The Predix UI components (https//www.predix-ui.com/) are modular by design. You can consume as many Predix UI components in the `predixdesignsystem` public GitHub organization as necessary to create a web application that runs on top of Predix services and data.

The modular architecture also provides you the flexibility to change and configure existing web applications without manipulating a large, complex single-component repository. You also can use available services in the Predix marketplace to extend a web application.

The Predix UI components are grouped into the following categories:

- Predix UI Basics on page 29
- Predix UI Templates on page 30
- Predix UI Components on page 31
- Predix UI Datetime Components on page 33
- Predix UI Mobile Components on page 33
- Predix UI Data Visualization Components on page 34
- Predix UI Vis Framework on page 35

Related concepts

About the Predix Design System on page 6

The Predix Design System gives developers and designers the ability to create and customize the user interface of a modern web application that runs on top of Predix data and services..

Polymer Library on page 26

The Predix UI components are implemented using the Polymer library.

Polymer Library

The Predix UI components are implemented using the Polymer library.

The Polymer library extends HTML to facilitate sophisticated user-interface presentation in your web applications. to support such features as:

- Custom elements - Define new elements in HTML
- Shadow DOM - Self-contained web components.
- Polyfills - Provide support for older browsers.

The Polymer library also provides a set of features for creating custom, reusable web componnents for web applications. It supports registration, callbacks, DOM templates, data binding, and property observation. You can use these features to efficiently build complex, interactive applications across web browsers (see Supported Browsers for Web Applications on page 9 for more information).

See the following resources for more information about the Polymer library:

- Polymer Library Overview
- Polycasts
- Build Your First Polymer Element
- Polymer App Toolbox

Getting Started with Predix UI Components

Use the Predix UI components in the `predixdesignsystem` public GitHub organization for common user interactions in a web application.

Purpose

The Predix UI components use the Polymer library to simplify the effort to build web applications that provide many of the characteristics of desktop software. For more information, see the video Using Predix UI and Polymer.

Before You Begin

As a developer, you should review the following resources to understand the conceptual and reference information needed to create a web application with Predix UI components:

- About the Predix Design System on page 6
- Application Development with the Predix Design System on page 7
- Supported Browsers for Web Applications on page 9

Note: For more information on the Predix UI components, see About Predix UI Components on page 26

Task Roadmap

The following tasks show how to use a Predix UI component in a web application with `px-spinner` as a reference.

#	Task	Information
1	Set up the Predix Design System developer environment.	See Setting Up the Predix Design System Developer Environment on page 13.
2	Configure the `px-spinner` component to use in a web application.	See Using a Predix UI Component in a Web Application on page 27.

Using a Predix UI Component in a Web Application

Set up the `bower` task nunner utility to import the correct version of each Predix UI component required for your web application with the `px-spinner` component as a reference.

Before You begin

- Review https://www.predix-ui.com/#/guides/getting-started/ for information on using a Predix UI component from the `predixdesignsystem` public GitHub organization..
- Review the Predix UI component API documentation for the properties and methods that are available.
- Set up the required services for Predix Design System. See Setting Up the Predix Design System Developer Environment on page 13 for more information.

- Navigate to the project directory of your web application <web_app_project>, where the following files will be added:
  - webcomponents-lite polyfill
  - Predix UI component

About This Task

Each Predix UI component includes a bower.json file that lists other required components. This is displayed as a series of @import declarations in the respective file. For more information, see Application Development with the Predix Design System on page 7 for more information.

Procedure

1. Add the webcomponents-lite polyfill to your web application as the first line in the <head> section of the HTML page to provide polyfill support for older browsers (see.Polymer Library on page 26). For example:

```
<!DOCTYPE html>
 <html>
  <head>
    <script src="bower_components/webcomponentsjs/
webcomponents.js"></script>
   </head>
 </html>
```

2. To clone your copy of the px-spinner component, at a command prompt, enter.

```
git clone https://github.com/predixdesignsystem/px-spinner.git
```

   The command also creates a px-spinner directory containing the component files.

3. Install the px-spinner component as a web component in the project directory by using the bower package-management utility (the px-spinner component is used as an example):

```
bower install px-spinner --save
```

4. Add the px-spinner component to your web application by adding it to the <head> section of the HTML page. For example:

```
<!DOCTYPE html>
 <html>
  <head>
   <script src="bower_components/webcomponentsjs/
webcomponents.min.js"></script>
   <link rel="import" href="bower_components/px-spinner/px-
spinner.html"/>
  </head>
 </html>
```

   The px-spinner component files are installed in the bower_components directory.

5. Call the Predix UI component in your web application by adding it to the <body> section of the HTML page. For example:

```
<!DOCTYPE html>
 <html>
  <head>
   <script src="bower_components/webcomponentsjs/webcomponents.js"></
```

```
script>
   <link rel="import" href="bower_components/px-spinner/px-
spinner.html"/>
  </head>
  <body>
   <px-spinner></px-spinner>
  </body>
 </html>
```

Related concepts

About the Predix Design System on page 6

The Predix Design System gives developers and designers the ability to create and customize the user interface of a modern web application that runs on top of Predix data and services..

About Predix UI Components on page 26

The Predix Design System includes a set of reusable web components in the `predixdesignsystem` public GitHub organization to add features or access services following consistent UX practices, standards, and specifications.

Predix UI Basics

The `predixdesignsystem` public GitHub organization stores Predix UI basics for Predix Design System that can serve as building blocks to create the user interface of a web application.

The Predix UI basics are used in isolation or combined to create more complex Predix UI components.

As a developer, you can use the code in the following repositories to produce these interactions.

Table 4: Predix UI Basics Repositories

GitHub Repository	Description	Documentation
px-alert-label	User interface element that creates default alert messages.	API Reference
px-file-upload	User interface element that provides users the ability to upload one or more files, whether as part of a larger form, or as a standalone component.	API Reference
px-inbox	User interface element that displays a list of items with detailed information or actions associated with each item.	API Reference
px-number-formatter	User interface element that displays a formatted string of a numeric value, or a numeric value of a formatted string.	API Reference
px-popover	User interface element that displays additional information to a user.	API Reference
px-progress-bar	User interface element that displays a progress bar that indicates the progress of an activity with explicit start and end times, or known duration.	API Reference

GitHub Repository	Description	Documentation
px-spinner	User interface element that indicates when data is being loaded, a view isi changing, or any place a visual indication is needed to show that work is being performed for an indefinite period of time.	API Reference
px-tabs	User interface element that c tabs to insert content.	API Reference
px-tooltip	User interface element that displays additional information to users once hovered over the target element for a specified amount of time.	API Reference
px-typeahead	User interface element that searches user input and lists all the suggestions which has the input	API Reference
px-validation	User interface element that provides built-in as well as developer-supplied validation solutions.	API Reference

Note:

The API reference documentation for individual Predix UI basics (https://www.predix-ui.com/) also includes usage instructions and examples.

File a GitHub issue if you encounter a problem using a Predix UI repository. For more information on filing a GitHub issue, see https://guides.github.com/features/issues/.

Predix UI Templates

The `predixdesignsystem` public GitHub organization stores Predix UI template components for the Predix Design System that can lay out the entire user interface (or region of the user interface) of a web application.

The Predix UI template components focus on the content structure or layout, not the actual content. when creating the user interface of a web application.

As a developer, you can use the code in the following GitHub repositories to produce these interactions.

Table 5: Predix UI Templates

GitHub Repository	Description	Documentation
px-card	User interface element that can layout a region of a web application.	API Reference
px-chip	A user interface component that is a representation of an object or a series of information.	API Reference
px-deck-selector	A user interface element to draw a drop-down display of available decks that returns the URL for a selected deck.	API Reference
px-kpi	A data visualization element that displays key performance indicators (KPI) in a dashboard layout.	API Reference
px-login	A user interface element that displays login/logout button.	API Reference

GitHub Repository	Description	Documentation
px-panel	A user interface component that provides a container for placing controls or content.	API Reference
px-steps	A user interface component that visually represents a sequence of steps.	API Reference
px-tile	A user interface component that displays an image, a title, and a description.	API Reference
px-widget-cards	Card objects consisting of a header area and content area.	API Reference

Note:

The API reference documentation for individual Predix UI templates (https://www.predix-ui.com/) also includes usage instructions and examples.

File a GitHub issue if you encounter a problem using a Predix UI repository. For more information on filing a GitHub issue, see https://guides.github.com/features/issues/.

Predix UI Components

The `predixdesignsystem` public GitHub organization stores Predix UI components for Predix Design System that are designed to achieve more complex user interactions when creating the user interface of a web application.

As a developer, you can use the code in the following GitHub repositories to produce these interactions.

Table 6: Predix UI Components Repositories

GitHub Repository	Description	Documentation
px-accordian	A user interface element that provides expandable and collapsible subsection headers for a page.	API Reference
px-alert-message	A user interface element that displays alert information to a user.	API Reference
px-app-header	A user interface element that allows the `px=branding-bar` and `px-app-nav` user interface elements to work together.	API Reference
px-app-nav	A user interface element that displays a navigation bar or drawer for applications.	API Reference
px-app-route	A user interface element that supports easy binding between various Predix UI components and a web application's URL.	API Reference
px-branding-bar	A user interface component that provides a header area to contain title, logo and branding content.	API Reference
px-breadcrumbs	A user interface element that works with the `px-context-browser` and `px-tree` user interface elements to visually represent the asset model hierarchy and jump to levels of the hierarchy.	API Reference

GitHub Repository	Description	Documentation
px-clipboard	A user interface element that copies a value into the user's computer	API Reference
px-context-browser	A user interface content browser displaying multiple levels of hierarchy for the current item.	API Reference
px-data-table	A user interface element that displays a data table with the expected data format. The data is passed as a JSON array.	API Reference
px-dropdown	A user interface element that displays a customizable dropdown list.	API Reference
px-heatmap-grid	A data visualization component based on a heat map scale of colors	API Reference
px-icon-set	A wrapper for the Predix icon set	API Reference
px-key-value-pair	A user interface element that allows you to prominently display information in a dashboard.	API Reference
px-map	A framework to create maps for an application.	API Reference
px-modal	A user interface element that displays user interaction workflow using a modal.	API Reference
px-notification	A user interface componene to notify the user of a state change or actions taken on a specific context.	API Reference
px-overlay	A user interface element that displays an overlay with other user interface elements.	API Reference
px-proudct-switcher	A user interface component that switches between products or applications,	API Reference
px-slider	A user interface element to define a value in a range or a range of values within set boundaries.	API Reference
px-timeline	A data visualization element to display a time- based series of events and documenting details relevant to each event.	API Reference
px-tree	A user interface element that provides an expandable and selectable tree. The contents are controlled primarily through a structured object of branch and leaf nodes.	API Reference
px-view-header	A mobile-responsive user interface element that is located at the top of the page and displays an overview of the content that a user is viewing.	API Reference

Note:

The API reference documentation for individual Predix UI components (https://www.predix-ui.com/) also includes usage instructions and examples.

File a GitHub issue if you encounter a problem using a Predix UI repository. For more information on filing a GitHub issue, see https://guides.github.com/features/issues/.

See the following resources for more information:

- Develop with the Context Browser
- Use the Predix Design System Icon Set
- Develop with the Predix Design System Map Framework
- Use Predix UI Components to Stack Context

Predix UI Datetime Components

The `predixdesignsystem` public GitHub organization stores Predix UI datetime componeents for Predix Design System that control the display of datetime values.

As a developer, you can use the code in the following GitHub repositories to produce these interactions.

Table 7: Predix UI Datetime Components Repositories

GitHub Repository	Description	Documentation
px-calendar-picker	A user interface element that allows the user to select a date or date range.	API Reference
px-datetime-field	A user interface element that allows the user to select a date and time precisely to the second or millisecond.	API Reference
px-datetime-picker	A user interface element that allows the user to select a date and time precisely to the second or millisecond.	API Reference
px-datetime-range-field	A user interface element that allows the user to select a date and time range precisely to the second or millisecond.	API Reference
px-datetime-range-panel	A user interface element that allows the user to select a date and time range precisely to the second or millisecond. .	API Reference
px-rangepicker	A user interface element to set a range of date and time using a calendar-based user interface.	API Reference

Note:

The API reference documentation for individual Predix UI datetime components (https://www.predix-ui.com/) also includes usage instructions and examples.

File a GitHub issue if you encounter a problem using a Predix UI repository. For more information on filing a GitHub issue, see https://guides.github.com/features/issues/.

Predix UI Mobile Components

The `predixdesignsystem` public GitHub organization stores Predix UI mobile components for Predix Design System that provide a foundation for building web applications to look and work like native mobile applications:.

As a developer, you can use the code in the following GitHub repositories to produce these interactions.

Table 8: Predix UI Mobile Components Repositories

GitHub Repository	Description	Documentation
px-table-view	A user interface element that helps organize menus and views into table-style blocks that can be expanded, swiped, and dragged and re-ordered.	API Reference

Note:

The API reference documentation for individual Predix UI mobile components (https://www.predix-ui.com/) also includes usage instructions and examples.

File a GitHub issue if you encounter a problem using a Predix UI repository. For more information on filing a GitHub issue, see https://guides.github.com/features/issues/.

See the following resources for more information:

- Mobile Application Developer Guide
- About Predix SDK for Hybrid

Predix UI Data Visualization Components

The `predixdesignsystem` public GitHub organization stores Predix UI data visualization components for Predix Design System that provide graphical representations of data.

As a developer, you can use the code in the following GitHub repositories to produce these interactions.

Table 9: Predix UI Data Visulization Component Repositories

GitHub Repository	Description	Documentation
px-gauge	A data visualization element that displays a single value between a range by radial. The value is presented by the color of gauge bar and the position of the needle..	API Reference
px-percent-circle	A data visualization element that displays a single ring that fills clockwise based on a percentage.	API Reference
px-simple-area-chart	A data visualization element that displays a simple stacked area chart.	API Reference
px-simple-bar-chart	A data visualization element that displays a series of numeric values as a bar chart, or multiple series of numeric values as a stacked bar chart.	API Reference
px-simple-horizontal-bar-chart	A data visualization element that displays a series of numeric values as a horizontally-oriented bar chart.	API Reference
px-simple-line-chart	A data visualization element that displays a series of numeric values as a line chart.	API Reference
px-simple-win-loss-chart	A data visualization element that displays a series of positive & negative values as a bar chart..	API Reference
px-vis-parallel-coordinates	A data visualization element to draw a parallel coordinates chart	API Reference

GitHub Repository	Description	Documentation
px-vis-pie-chart	A data visualization element to draw a pie or donut chart for some data.	API Reference
px-vis-polar	A data visualization element to draw a phase and amplitude on a polar plot.	API Reference
px-vis-radar	A data visualization element to draw a radar (spider) chart.	API Reference
px-vis-spark	A data visualization element that displays a small line chart without axes or measures that provides a user a glimpse of a trend.	API Reference
px-vis-timeseries	A data visualization element to create an interactive timeseries chart.	API Reference
px-vis-xy-chart	A data visualization element to draw lines, scatter or both against an X and Y axis.	API Reference

Note:

The API reference documentation for individual Predix UI chart elements (https://www.predix-ui.com/) also includes usage instructions and examples.

File a GitHub issue if you encounter a problem using a Predix UI repository. For more information on filing a GitHub issue, see https://guides.github.com/features/issues/.

See the following resources for more information:

- Predix UI Vis Framework on page 35
- Data Visualization Developer Guide

Predix UI Vis Framework

The `predixdesignsystem` public GitHub organization stores the Predix UI Vis charting framework.

The Predix UI Vis charting framework uses the D3.ja data visualization library and Polymer web elements. You do not need to know D3.js to use the Viz framework.

If you have a knowledge of Polymer, you can build a new chart using the building blocks and charts that are provided with the Viz framework. The building blocks and charts are included with the `px-vis` repository availble in the `predixdesignsystem` public GitHub organization (https://github.com/predixdesignsystem/px-vis).

Note: Review the Vis Framework Overview for the overview, task, and reference information needed to the the Vis framework..

Drawing Tool	Description	Documentation
px-vis	A charting framework that uses the D3.ja data visualization library and Polymer web elements. You do not need to know D3.js to use the Viz framework. If you have a knowledge of Polymer, you can build a new chart using the building blocks and charts that are provided with the Viz framework.	API Reference
px-vis-area-svg	An example of an area drawing object	API Reference

Drawing Tool	Description	Documentation
px-vis-axis	An example of an area drawing object	API Reference
px-vis-axis-interaction-space	This drawing tool allows the user to brush an axis in order to filter values.	API Reference
px-vis-bar-svg	An example of a bar drawing object.	API Reference
px-vis-brush	This drawing tool provides user interaction on the px-vis-chart-navigator drawing object.	API Reference
px-vis-canvas	This drawing tool creates a <canvas> element and context.	API Reference
px-vis-chart-navigator	This drawing tool is an example of a chart drawing object.	API Reference
px-vis-clip-path	This drawing tool is a clipping path restricts the region to which paint can be applied.	API Reference
px-vis-clip-path-complex-area	This drawing tool is a clipping path restricts the region to which paint can be applied (currently applies only to px-vis-radar).	API Reference
px-vis-cursor	This drawing tool provides an on-chart reticule on hover, an on-chart reticule on hover..	API Reference
px-vis-cursor-line	This drawing tool provides an on-chart reticule on hover, highlighting an entire line.	API Reference
px-vis-data-converter	This drawing tool converts data from the time series service to the schema required by the Vis framework.	API Reference
px-vis-dynamic-menu	This drawing tool provides a menu, dynamically built depending on the options provided.	API Reference
px-vis-event	An example of a line drawing object	API Reference
px-vis-gridlines	This drawing tool adds gridlines to either the x or the y axis.	API Reference
px-vis-highlight-line-canvas	This drawing tool provides a way to highlight one or more lines of data.	API Reference
px-vis-highlight-line	This drawing tool provides a way to highlight one or more lines of data.	API Reference
px-vis-highlight-point-canvas	This drawing tool provides a way to highlight one or more points of data.. It is intended to be tied to another chart, so hovering over one chart highlights corresponding data on the other chart.	API Reference
px-vis-highlight-point	This drawing tool provides a way to highlight one or more points of data. It is intended to be tied to another chart, so hovering over one chart highlights corresponding data on the other chart	API Reference
px-vis-interactive-axis	This drawing tool lets a user that a user to click and drag a box around a portion of the axis to filter data.	API Reference

Drawing Tool	Description	Documentation
px-vis-interaction-space	This drawing tool provides the basis for all on-chart interaction. It detects mouseovers on the chart and builds the data objects that other components then consume.	API Reference
px-vis-line-canvas	This drawing tool draws line series onto the chart.	API Reference
px-vis-line-svg	This drawing tool is an example of a line drawing object..	API Reference
px-vis-multi-axis	This drawing tool creates creates x & y interpreters, which act as data converters.	API Reference
px-vis-multi-scale	This drawing tool creates creates x & y interpreters, which act as data converters.	API Reference
px-vis-pie	This drawing tool creates an interactive pie or donut chart.	API Reference
px-vis-radar-grid	This drawing tool draws polygonal grid lines.	API Reference
px-vis-radial-gridlines	This drawing tool draws polar grid lines.	API Reference
px-vis-radial-scale	This drawing tool creates x & y interpreters.	API Reference
px-vis-register	This drawing tool displays data from the chart on mouseover.	API Reference
px-vis-scale	This drawing tool converts data into drawing space coordinates.	API Reference
px-vis-scatter	An example of a scatter drawing object.	API Reference
px-vis-scatter-canvas	This drawing tool draws scatter series onto the chart.	API Reference
px-vis-striping	This drawing tool provides a way to highlight a range visually on the chart.	API Reference
px-vis-svg	This component provides the basic framework for drawing objects to be displayed.	API Reference
px-vis-svg-canvas	This drawing tool creates a set of <canvas> and <svg> elements.	API Reference
px-vis-threshold	This drawing tool provides chart thresholds that appear as horizontal lines across the chart.	API Reference
px-vis-toolbar	This drawing tool provides a configurable toolbar, which can be used to add chart interaction and other custom features.	API Reference
px-vis-tooltip	This drawing tool provides data display functionality similar to px-vis-register, but on the chart itself, and without the user interactivity.	API Reference

Note:

The API reference documentation for individual Predix UI chart elements (https://www.predix-ui.com/) also includes usage instructions and examples.

File a GitHub issue if you encounter a problem using the `px-vis` repository. For more information on filing a GitHub issue, see https://guides.github.com/features/issues/.

See the following resources for more information:

- Predix UI Data Visualization Components on page 34
- Data Visualization Developer Guide

Localize Predix UI Components

Localizing Predix UI Components

Change the default language of Predix UI compoments with the Moment.js or D3.js libraries.

Before You Begin

As a developer, you should review the following ressources to understand the conceptual, reference, and task information needed to localize the content of a Predix UI component.

- Predix Design System Internationalization Guidelines
- Predix Design System Localizable Strings

Note: For more information on the Predix UI components, see About Predix UI Components on page 26

Purpose

As a developer, you can localize the Predix UI components for a specific country or region in the following ways:

Localize Text Strings in Predix UI Components

Any hard-coded instance of text strings in the code (such as error messages, alerts, titles, button labels, or text input) has been identified to enable localization so the translation can happen by the developer or through a third-party translation service for such Predix UI components as:

- px-alert-message
- px-context-browser
- px-data-table
- px-login
- px-timeline
- px-vis-parallel-coordinates
- px-vis-pie-chart
- px-vis-radar
- px-vis-radar

Localize Predix UI Datetime Components

Predix UI components that implement date/time/calendar functions as localized strings can be localized with the Moment.js library:

- px-datetime-buttons
- px-datetime-field
- px-datetime-picker
- px-datetime-presets
- px-datetime-range-field
- px-datetime-range-panel
- px-rangepicker

Localize Predix UI Chart Components

Predix UI components that implement date/time and number functions as localized strings with the Vis framework can be localized with the D3.js library:

- px-vis-timeseries

- px-vis-xy-chart
- px-vis-polar

Note: Any hard coded strings for date/time/calendar or number functions will also need to be localized using either the Moment.js or D3.js libraries.

Task Roadmap

The following tasks show how to localize a Predix UI component.

#	Task	Information
1	Set up the Predix Design System developer environment.	See Setting Up the Predix Design System Developer Environment on page 13.
2	Localize hard-coded text strings in aPredix UI component.	See Localizing Text Strings on page 40.
3	Change the locale of a Predix UI date/time component.	See Localizing with the Moments.js Library on page 41.
4	Change the locale of a Predix UI chart component	See Localizing with the D3.js Library on page 44.
5	Change a Predix UI component to a custom locale	See Custom Locale Support on page 46.

Localizing Text Strings

A Predix UI component can specify the text strings to support different locales.

Before You begin

As a developer, you shoul review the overview and reference information in Localizable Strings.

About This Task

A Predix UI component uses the following properties to support different locales:

- `language`: Contains the value of the language. For example:

```
language: {
value: 'en'
}
```

- `resources`: Objects containing a key (a language) that will match the `language` property. Each key/language, contains the 'to be translated' string and its corresponding value (translated string) in that language. For example:

```
{
  'en': {
    'Apply': 'Apply',
    'Cancel': 'Cancel'
  },
  'fr': {
    'Apply': 'Appliquer',
    'Cancel': 'Annuler'
  }
}
```

Text strings can be passed to a Predix UI component in the following ways:

- Loading an external JSON file where the specific resources for each language object is predefined. For example.:

```
attached: function() {
this.loadResources(this.resolveUrl('locales.json'));
}
```

- Loading the resources object in-line within the application. For example:

```
resources: {
value: function() {
                return {
'en': { 'hello': 'My name is {name}.' },
'fr': { 'hello': 'Je m\'apelle {name}.' }
}
}
    }
```

Resources only need to hold the language you currently use (but can include as many languages as you want). Resources need to hold every string that a web application is using and its translation because the string and its translation are passed to every Predix UI component that needs translation.

Localizing with the Moments.js Library

Set up the `bower` task nunner utility to import the correct version of each Predix UI component required to localize a Predix UI component with the Moment.js library using `px-datetime-buttons` as a reference..

Before You begin

- Review the overview, task, and reference information in Predix Design System Internationalization Guidelines to localize a Predix UI component from the `PredixDev` public GitHub organization..
- Review the Predix UI component API documentation for the properties and methods that are available.
- Set up the required services for Predix Design System. See Setting Up the Predix Design System Developer Environment on page 13 for more information.
- Each Predix UI component includes a `bower.json` file that lists other required components (including the `px-moment-import` component). This is displayed as a series of `@import` declarations in the respective file. For more information, see Application Development with the Predix Design System on page 7 for more information.
- Navigate to the project directory of your web application `<web_app_project>`, where the following files will be added:
  - `webcomponents-lite` polyfill
  - `px-datetime-buttons` component

About This Task

The Predix UI date/time components use the Moment.js library (the `px-moment-import` component) to handle date and time. Predix UI keeps its own version of the Moment.js library so that an application can load any version of Moment.js without conflicts. The Predix Design System version of Moment.js can be localized through the convenience function `Px.moment.changeLocale(newLocale, callback, configObject)`.

Note:

You will need to change the `moment` locale before you change the `language` property of a Predix UI component. Changing the `moment` locale does not affect any previously created `moment` object. Predix UI components that depend on the Moment.js library will recreate their `moment` objects after the `language` property changes.

Make sure your build includes the `momentLocale` folder from the `px-datetime-common` component included with the `bower` install of the Predix UI component. By default, the locale will be searched in the `/px-datetime-common/momentLocale/` folder. However, the path can be configured through the `Px.moment.localeURL` variable.

The Predix UI date/time components support strings that can be localized for the following languages:

- Dutch
- English
- French
- German
- Portuguese
- Russian
- Spanish

Note: See Custom Locale Support on page 46 for information to support other languages and regions.

When loading a supported language, you pass a locale name, as well as an optional callback function that can be called after the locale has been effectively loaded (the `configObject` argument is not needed). For example:

```
Px.moment.changeLocale('fr', function() {
  console.log('moment locale changed!');
});
```

This process is asynchronous because the locale needs to be fetched from a script. You can use the callback in `Px.moment.changeLocale` in order to perform actions after the locale has been loaded (such as updating the `language` and `resources` of the Predix UI components).

Procedure

1. Add the `webcomponents-lite` polyfill to your web application as the first line in the `<head>` section of the HTML page to provide polyfill support for older browsers (see.Polymer Library on page 26). For example:

   ```
   <!DOCTYPE html>
    <html>
     <head>
       <script src="bower_components/webcomponentsjs/
   webcomponents.js"></script>
      </head>
    </html>
   ```

2. To clone your copy of the `px-datetime-buttons` component, at a command prompt, enter.

   ```
   git clone https://github.com/PredixDev/px-datetime-buttons.git
   ```

   The command also a `px-datetime-buttons` directory containing the components files.

3. Install the Predix UI component as a web component in the project directory by using the `bower` package-management utility (the `px-datetime-buttons` component is used as an example):

```
bower install px-datetime-buttons --save
```

   The `px-datetime-buttons` component files are installed in the `bower_components` directory.
4. Add the `px-datetime-buttons` component to your web application by adding it to the `<head>` section of the HTML page. For example:

```
<!DOCTYPE html>
 <html>
  <head>
   <script src="bower_components/webcomponentsjs/
webcomponents.min.js"></script>
   <link rel="import" href="bower_components/px-datetime-buttons/px-
datetime-buttons.html"/>
  </head>
 </html>
```

5. Call the Predix UI component in your web application by adding it to the `<body>` section of the HTML page. For example:

```
<!DOCTYPE html>
 <html>
  <head>
   <script src="bower_components/webcomponentsjs/webcomponents.js"></
script>
   <link rel="import" href="bower_components/px-datetime-buttons/px-
datetime-buttons.html"/>
  </head>
  <body>
   <px-datetime-buttons></px-datetime-buttons>
  </body>
 </html>
```

6. Change the date/time locale with the `Px.moment.changeLocale()` function. For example:

```
Px.moment.changeLocale('fr', function() {
  console.log('moment locale changed!');
});
```

Next Steps

See the following resources for more information:

- Moment.js Reference Documentation
- Moment.js Developer Guides

Localizing with the D3.js Library

Set up the `bower` task runner utility to import the correct version of each Predix UI component required to localize a Predix UI component with the D3.js library using `px-vis-polar` as a reference.

Before You begin

- Review the overview, task, and reference information in Predix Design System Internationalization Guidelines to localize a Predix UI component from the `PredixDev` public GitHub organization..
- Review the Predix UI component API documentation for the properties and methods that are available.
- Set up the required services for Predix Design System. See Setting Up the Predix Design System Developer Environment on page 13 for more information.
- Each Predix UI component includes a `bower.json` file that lists other required components (including the `px-d3-import` component). This is displayed as a series of `@import` declarations in the respective file. For more information, see Application Development with the Predix Design System on page 7 for more information.
- Navigate to the project directory of your web application `<web_app_project>`, where the following files will be added:
  - `webcomponents-lite` polyfill
  - `px-vis-polar` component

About This Task

Predix Design System keeps its own version of the D3.js library (the `px-d3-import` component) so that an application can load any version of D3.js as well without conflict. The D3.js library supports two different types of localization: date/time format and number format with two functions:

- `Px.d3.changeNumberLocale(newLocale, callback, configObject)`
- `Px.d3.changeTimeLocale(newLocale, callback, configObject)`

Note:

Make sure your build includes respectively the `timeLocale` and `numberLocale` folders from the `px-d3-imports` included during the `bower` install of the Predix UI component. By default, the locales will be searched in the `/px-d3-imports/timeLocale/` and `/px-d3-imports/numberLocale/` folders. However, the path can be configured through the `Px.d3.timeLocaleURL` and `Px.d3.numberLocaleURL` variables.

The Predix UI components that implement date/time and number functions as localized strings with the D3.js library can be localized for the following languages and regions:

- Dutch
- English
- French
- German
- Portuguese
- Russian
- Spanish

Note: See Custom Locale Support on page 46 for information to support other languages and regions.

When loading a supported language, a locale name is passed, as well as an optional callback function that can be called after the locale has been effectively loaded (the `configObject` argument is not needed). For example:

```
Px.d3.changeTimeLocale('ca-ES', function() {
  console.log('d3 time locale changed!');
});
Px.d3.changeNumberLocale('ar-PS', function() {
  console.log('d3 number locale changed!');
});
```

This process is synchronous because the locale needs to be fetched from a JSON file. You can use the callback in `Px.d3.changeTimeLocale/Px.d3.changeNumberLocale` in order to perform actions after the locale has been loaded (such as updating the `language` and `resources`of the Predix UI components).

Procedure

1. Add the `webcomponents-lite` polyfill to your web application as the first line in the `<head>` section of the HTML page. For example:

   ```
   <!DOCTYPE html>
    <html>
     <head>
       <script src="bower_components/webcomponentsjs/
   webcomponents.js"></script>
      </head>
    </html>
   ```

2. To clone your copy of the `px-vis-polar` component, at a command prompt, enter.

   ```
   git clone https://github.com/PredixDev/px-vis-polar.git
   ```

   The command also creates a `px-vis-polar` directory containing the component files.
3. Install the `px-vis-polar` component as a web component in the project directory by using the `bower` package-management utility:

   ```
   bower install px-vis-polar --save
   ```

   The `px-vis-polar` component files are installed in the `bower_components` directory.
4. Add the `px-vis-polar` component to your web application by adding it to the `<head>` section of the HTML page. For example:

   ```
   <!DOCTYPE html>
    <html>
     <head>
      <script src="bower_components/webcomponentsjs/
   webcomponents.min.js"></script>
      <link rel="import" href="bower_components/px-vis-polar/px-vis-
   polar.html"/>
     </head>
    </html>
   ```

5. Call the Predix UI component in your web application by adding it to the `<body>` section of the HTML page. For example:

```
<!DOCTYPE html>
 <html>
  <head>
   <script src="bower_components/webcomponentsjs/webcomponents.js"></
script>
   <link rel="import" href="bower_components/px-vis-polar/px-vis-
polar.html"/>
  </head>
  <body>
   <px-vis-polar></px-vis-polar>
  </body>
 </html>
```

6. Change the date/time locale with the `Px.d3.changeTimeLocale()` function. For example:

```
Px.d3.changeTimeLocale('ca-ES', function() {
  console.log('d3 time locale changed!');
});
```

7. Change the number locale with the `Px.d3.changeNumberLocale()` function. For example:

```
Px.d3.changeNumberLocale('ar-PS', function() {
  console.log('d3 number locale changed!');
});
```

Next Steps

See the following resources for more information:

- D3.js Reference Documentation
- D3.js Developer Guides

Custom Locale Support

Localization can happen by the developer or through a third-party translation service for a custom language or region with either the Moment.js or D3.js libraries.

Before You Begin

As a developer, you should review the following resources to understand the conceptual, task, and reference information needed to support a custom language or region:

- Localizing Predix UI Components
- Localizing with the Moments.js Library on page 41
- Localizing with the D3.js Library on page 44

Localizing a custom locale with Moment.js

Localizing a custom locale with the Moment.js library requires manually loading or generating a `configObject` (see http://momentjs.com/docs/#/customization/ for more information): For example

```
Px.moment.changeLocale(null, null, yourConfigObject)
```

In this example, the Moment.js locale change is synchronous.

Moment.js Locale Codes

The Moment.js library supports the following locale codes:

- af (Afrikaans)
- ar-dz (Arabic - Algeria)
- ar-ly (Arabic - Libya)
- ar-ma (Arabic - Morocco)
- ar-sa (Arabic - Saudi Arabia)
- ar-tn (Arabic - Tunisia)
- ar (Arabic)
- az (Azeri)
- be (Belarusian)
- bg (Bulgarian)
- bn (Bengali)
- bo (Tibetan)
- bs (Bosnian)
- ca (Catalan)
- cs (Czech)
- cy (Welsh)
- da (Danish)
- de-at (German - Austria)
- de-ch (German - Switzerland)
- de (German)
- el (Greek)
- en-au (English - Australia)
- en-ca (English - Canada)
- en-gb (English - Great Britain)
- en-ie (English - Ireland)
- en-nz (English - New Zealand)
- en-us (English - United States)
- es-do (Spanish - Dominican Republic)
- es (Spanish)
- eu (Basque)
- fa (Farsi - Persian)
- fi (Finnish)
- fo (Faroese)
- fr-ca (French - Canada)
- fr-ch (French - Switzerland)
- fr (French)
- gd (Gaelic)
- he (Hebrew)
- hi (Hindi)
- hr (Croatian)(
- hu (Hungarian)
- hy-am (Armenian)
- id (Indonesian)
- is (Icelandic)
- it (Italian)

- ja (Japanese)
- ka (Georgian)
- kk (Kazakh)
- km (Khmer)
- kn (Kannada)
- ko (Korean)
- lo (Lao)
- lt (Lithuanian)
- lv (Latvian)
- mk (Maori)
- ml (Malayalam)
- mr (Marathi)
- ms-my (Malay - Malaysia)
- ms (Malay)
- my (Burmese)
- nb (Norwegian)
- ne (Nepali)
- nl-be (Dutch - Belgium)
- nl (Dutch)
- pa-in (Punjabi)
- pl (Polish)
- pt-br (Portuguese - Brazil)
- pt (Portuguese)
- ro (Romanian)
- ru (Russian)
- sd (Sindhi)
- sk (Slovak)
- sl (Slovenian)
- sq (Albanian)
- sr-cyrl (Serbian - Cyrillic)
- sr (Serbian)
- sv (Swedish)
- sw (Swahili)
- ta (Tamil)
- te (Telugu)
- th (Thai)
- ttl-phh
- tr (Turkish)
- uk (Ukrainian)
- ur (Urdu)
- uz-latn
- uz (Uzbek)
- vi (Vietnamese)
- yo (Yoruba)
- zh-cn (Chinese - Simplified)
- zh-hk (Chinese - Hong Kong)
- zh-tw (Chinese - Taiwan)

Localizing a custom locale with D3.js

Localizing a custom locale with the D3.js library requires manually loading or generating a `configObject`:

- Date/Time (see https://github.com/d3/d3-time-format/blob/master/README.md#timeFormatDefaultLocale for more information).
- Numbers (see https://github.com/d3/d3-time-format/blob/master/README.md#timeFormatDefaultLocale for more information).

For example:

```
Px.d3.changeTimeLocale(null, null, yourConfigObject);
Px.d3.changeNumberLocale(null, null, yourConfigObject);
```

In this example, the D3.js locale change is synchronous.

D3.js Number Locale Codes

The D3.js library supports the following number locale codes:

- ca-ES (Catalan - Spain)
- cs-CZ (Czech - Czech Republic)
- de-CH (German - Switzerland)
- de-DE (German - Germany)
- en-CA (English - Canada)
- en-GB (English - Great Britain)
- en-US (English - United States)
- es-ES (Spanish - Spain)
- es-MX (Spanish - Mexico)
- fi-FI (Finnish - Finland)
- fr-CA (French - Canada)
- fr-FR (French - France)
- he-IL (Hebrew - Israel)
- hu-HU (Hungarian - Hungary)
- it-IT (Italian - Italy)
- ja-JP (Japanese - Japan)
- ko-KR (Korean - Korea)
- mk-MK (Macedonian - Macedonia)
- nl-NL (Dutch - Netherland)
- pl-PL (Polish - Poland)
- pt-BR (Portuguese - Brazil)
- ru-RU (Russian - Russia)
- sv-SE (Swedish - Sweden)
- uk-UA (Ukrainian - Ukraine)
- zh-CN (Chinese - Simplified)

D3.js Date/Time Locale Codes

The D3.js library supports the following date/time locale codes:

- ca-ES (Catalan - Spain)
- cs-CZ (Czech - Czech Republic)
- de-CH (German - Switzerland)

- de-DE (German - Germany)
- en-CA (English - Canada)
- en-GB (English - Great Britain)
- en-US (English - United States)
- es-ES (Spanish - Spain)
- es-MX (Spanish - Mexico)
- fi-FI (Finnish - Finland)
- fr-CA (French - Canada)
- fr-FR (French - France)
- he-IL (Hebrew - Israel)
- hu-HU (Hungarian - Hungary)
- it-IT (Italian - Italy)
- ja-JP (Japanese - Japan)
- ko-KR (Korean - Korea)
- mk-MK (Macedonian - Macedonia)
- nl-NL (Dutch - Netherland)
- pl-PL (Polish - Poland)
- pt-BR (Portuguese - Brazil)
- ru-RU (Russian - Russia)
- sv-SE (Swedish - Sweden)
- uk-UA (Ukrainian - Ukraine)
- zh-CN (Chinese - Simplified)

Theme Web Applications

Theming Web Applications

Predix Design System uses CSS custom properties to theme web applications.

Before You Begin

As a developer, you should review the following resources to understand the conceptual and reference information needed to create a web application with Predix Design System:

- About the Predix Design System on page 6
- Application Development with the Predix Design System on page 7
- Supported Browsers for Web Applications on page 9

As a developer, you should review the overview, task, and reference information in Theme Component Styles and Colors to apply a theme to a web application

Purpose

As a developer, you can use CSS custom properties to perform the following tasks:

1. Style a Predix UI component by changing the declared value in the HTML page.
2. Apply a theme to a web application by implementing the CSS custom properties for all (or any) Predix UI components.

Task Roadmap

The following tasks show how Predix Design System uses CSS custom properties to theme a web application..

#	Task	Information
1	Set up the Predix Design System developer environment.	See Setting Up the Predix Design System Developer Environment on page 13.
2	Install the `px-spinner` component.	See Using a Predix UI Component in a Web Application on page 27.
3	Change the declared value of a CSS custom property using the `px-spinner` component as a reference.	See Styling a Predix UI Component on page 51.
4	Implement the CSS custom properties for all Predix UI components using the `px-dark-theme` style module as a reference.	See Applying a Theme to a Web Application on page 53.

Styling a Predix UI Component

Change the declared value of a CSS custom property using the `px-spinner` component as a reference.

Before You begin

- Review the `px-spinner` component API documentation for the CSS custom properties that are available for styling.

- Navigate to the project directory of your web application `<web_app_project>` where the `px-spinner` component is installed.

The Predix UI components are implemented using the Polymer library. This library includes a shim that supports CSS custom properties as a style module.

As a developer, you can change the declared values of CSS custom properties to apply styles across the HTML page of a web application by specifying the `:root` selector with the `custom-style` extension of the native `<style>` element in the HTML page.

For example, change the declared value of the CSS custom property `--px-spinner-fill-color` using the `:root` selector to apply across the HTML page of the web application:

```
<!DOCTYPE html>
            <head>
              <script src="bower_components/webcomponentsjs/
webcomponents.js"></script>
              <link rel="import" href="/bower_components/px-spinner/px-
spinner.html"/>
           </head>
           <body>
             <style is="custom-style">
              :root {
                --px-spinner-fill-color: red;
              }
             </style>
             <px-spinner></px-spinner>
         </body>
  </html>
```

The declared value of the CSS custom property `--px-spinner-fill-color` is changed from the default value of `$primary-blue` to `red`.

As a developer, you can also create multiple instances of the `px-spinner` component with different sets of custom properties.

```
<!DOCTYPE html>
  <head>
    <script src="bower_components/webcomponentsjs/webcomponents.js"></
script>
    <link rel="import" href="/bower_components/px-spinner/px-
spinner.html"/>
  </head>
  <body>
   <style is="custom-style">
    :root {
          --px-spinner-fill-color: orange;
          }
          .color-red px-spinner {
             --px-spinner-fill-color: red;
          }
          .color-blue px-spinner {
              --px-spinner-fill-color: blue;
          }
   </style>
   <section class="color-orange">
    <px-spinner></px-spinner>
   </section>
   <section class="color-red">
    <px-spinner></px-spinner>
```

```
    </section>
    <section class="color-blue">
     <px-spinner></px-spinner>
    </section>
   </body>
</html>
```

Three instances of the `px-spinner` component are created with the declared value of the CSS custom property `--px-spinner-fill-color` changed from the default value of `$primary-blue` to `orange`, `red`, and `blue`.

Applying a Theme to a Web Application

Implement the CSS custom properties for all Predix UI components using the `px-dark-theme` style module as a reference.

Before You begin

Navigate to the project directory of your web application `<web_app_project>` where the `px-dark-theme` style module will be installed.

About This Task

The Predix UI components are implemented using the Polymer library. This library includes a shim that supports CSS custom properties as a style module.

The compiled style sheet for the `px-dark-theme-styles.html` style module contains the CSS custom properties for all Predix UI components. wherever such customizations are possible The `px-spinner` componenet is used as an example.

The CSS custom properties are applied to a web application with the `custom-style` extension of the native `<style>` tag in the HTML page.

Procedure

1. Install the `px-dark-theme-style` style module in the project directory `<web_app_project>` by using the `bower` package-management utility:

   ```
   bower install px-dark-theme --save
   ```

   The `px-dark-theme-style` style module which contains the CSS custom properties for all Predix UI components is installed.
2. Call the `px-dark-demo-theme-styles.html` style module and show it in your web application by adding it to the `<head>` section of the HTML page by using the `<style>` tag, which includes the custom extension `custom-style`. For example:

   ```
   <!DOCTYPE html>
       <html>
           <head>
               <script src="bower_components/webcomponentsjs/
   webcomponents.js"></script>
               <link rel="import" href="bower_components/px-spinner/px-
   spinner.html"/>
               <style include="px-dark-demo-theme-styles" is="custom-
   style"></style>
   ```

```
            <link rel="import" href="bower_components/px-dark-
theme/px-dark-demo-theme-styles.html"/>
        </head>
   </html>
```

Note: The include name in the `<style>` tag (for example, `px-dark-demo-theme-styles`) must correspond to the one specified inside the style module you just imported using the `<link>` tag (for example, `px-dark-demo-theme-styles.html`).

The declared values of the CSS custom properties defined for the `px-dark-demo-theme-styles.html` style module are applied to the web application.

CSS Custom Properties Overview

CSS custom properties contain specific values that can be reused throughout a web application.

CSS Default Display Properties and Values

Web applications use Cascading Style Sheets (CSS) to define the styling that is applied to an HTML page, often with repeated values. The following CSS style sheet defines brown as the background color in several places.

```
.one {
  color: white;
  background-color: brown;
  margin: 10px;
  width: 50px;
  height: 50px;
  display: inline-block;
}
.two {
  color: white;
  background-color: brown;
  margin: 10px;
  width: 150px;
  height: 70px;
  display: inline-block;
}
.three {
  color: white;
  background-color: brown;
  margin: 10px;
  width: 100px;
}
```

As a developer, you need to perform a global search and replace to change the value of the background color.

CSS Custom Properties

The future W3C CSS Custom Properties for Cascading Variables specification allows you to define CSS custom properties with the following syntax.

```
element {
  --<custom_property>: <declared_value>;
}
```

For example:

```
.root {
    --main-bg-color: brown; /* the main background color is brown */
}
```

As a developer, you can avoid repetition in a CSS stylesheet by referencing a CSS custom property using the `var()` function in multiple places. For example:

```
:root {
  --main-bg-color: brown;}
}
.one {
  color: white;
  background-color: var(--main-bg-color);
  margin: 10px;
  width: 50px;
}
.two {
  color: white;
  background-color: var(--main-bg-color);
  margin: 10px;
  width: 150px;
}
.three {
  color: white;
  background-color: var(--main-bg-color);
  margin: 10px;
  width: 100px;
}
```

Note: You can use the `:root` CSS selector to match the document root element (in the HTML page, the `:root` element corresponds to the `<html>` tag).

CSS custom properties also act as semantic identifiers that you can understand. For example, the CSS custom property `--main-bg-color` is simpler to understand than the color `brown`, especially if the same color is also used in another context in the HTML document.

CSS Custom Properties Reference

See the following resources for more information on how Predix Design System uses CSS custom properties to theme a web application.

- CSS Custom Properties Reference
- CSS Custom Properties Developer's Guide
- Polymer Library Reference

Get Started with Predix UI CSS Modules

About Predix UI CSS Modules

The Predix Design System includes CSS modules in the `predixdesignsystem` public GitHub organization to customize the base typography, layout, and content design in a web application.

As a developer. you can consume as many CSS modules (https://www.predix-ui.com/) as required to configure the user interface to your specific application requirements. Predix UI also includes collections organized by common functions (such as lists or buttons) for convenient use when customizing a web application. The modular architecture includes defined variables and default variants that provide the flexibility to change and configure existing web applications, rather than use a large, complex, single-component module.

The Predix UI CSS modules are grouped into the following libraries:

- Predix UI CSS Visual Library on page 59
- Predix UI CSS Layout Library on page 60
- Predix UI CSS Utilities Library on page 61

Related concepts

About the Predix Design System on page 6

The Predix Design System gives developers and designers the ability to create and customize the user interface of a modern web application that runs on top of Predix data and services..

Predix UI CSS Module Overview on page 62

As a developer, you should review the conceptual, task, and reference information for the following tools before using a Predix UI Predix UI CSS module in web application:

Getting Started with Predix UI CSS Modules

The Predix UI CSS modules in the `PredixDev` public GitHub organization establish the base typography, layout, and content design in a web application.

Purpose

Use the Predix UI CSS modules to satisfy custom styling requirements in a web application. See About Predix UI CSS Modules on page 56 for more information.

Before You Begin

As a developer, you should review the following resources to understand the conceptual and reference information needed to create a web application with Predix Design System:

- About the Predix Design System on page 6
- Application Development with the Predix Design System on page 7
- Supported Browsers for Web Applications on page 9

Note: For more information on the Predix UI CSS modules, see About Predix UI CSS Modules on page 56.

Task Roadmap

The following tasks show how to use a Predix UI CSS module in a web application with `px-buttons-design` as a reference.

#	Task	Information
1	Set up the Predix Design System developer environment.	See Setting Up the Predix Design System Developer Environment on page 13.
2	Install the Predix UI CSS Starter Kit files.	See Installing the Predix UI CSS Starter Kit on page 57
3	Configure the `px-buttons-design` CSS module to use in a web application.	See Using a Predix UI CSS Module in a Web Application on page 58.

Installing the Predix UI CSS Starter Kit

The `px-starter-kit-design` library serves as a starting point for a new Predix UI CSS project.

Before You begin

- Review https://www.predix-ui.com/#/css/px-starter-kit-design for information on installing the Predix UI CSS starter kit files stored in the `PredixDev` public GitHub organization.
- Set up the required environment for the Predix UI. For more information, see Setting Up the Predix Design System Developer Environment on page 13.
- Navigate to the project directory of your web application `<web_app_project>` where the Predix UI CSS starter kit files will be installed.

Procedure

1. Install the Predix UI CSS starter kit files:

```
bower --save install https://github.com/predixdesignsystem/px-
starter-kit-design
```

   The Predix UI CSS starter kit files are installed in the `bower_components` directory.
2. Create a project SASS file `<web_app_project>.scss` that defines the following order for importing default Predix UI CSS modules:

```
// Settings
   @import "px-colors-design/_settings.colors.scss";
   // Tools
   // Generic
   @import "px-normalize-design/_generic.normalize.scss";
   @import "px-box-sizing-design/_generic.box-sizing.scss";
   @import "px-helpers-design/_generic.helpers.scss";
   // Base
   @import "px-flexbox-design/_base.flexbox.scss";
   @import "px-viewport-design/_base.viewport.scss";
   @import "px-typography-design/_base.typography.scss";
   // Meta
   // Objects
   // Component
   // Your css goes in here
   // Trumps
   @import "inuit-clearfix/_trumps.clearfix.scss";
   @import "px-spacing-responsive-design/_trumps.spacing-
responsive.scss";
   @import "px-widths-responsive-design/_trumps.widths-
responsive.scss";
```

Using a Predix UI CSS Module in a Web Application

The Predix UI CSS modules ensure consistent design in a web applicaion that adapts the presentation to different devices.

Before You begin

- Review https://github.com/PredixDev/px-getting-started/blob/master/README.md for information on using the Predix UI repositories from the `predixdesignsystem` public GitHub organization.
- Review the API reference documentation for each Predix UI CSS module (https://www.predix-ui.com/) for specific configuration information regarding:
  - Feature flags
  - Style variables
  - Import order
- Navigate to the project directory of your web application `<web_app_project>` where the Predix UI starter kit files are installed.

Note: See https://github.com/PredixDev/px-getting-started/blob/master/README.md for more information on using a Predix UI CSS module in the `predixdesignsystem` public GitHub organization.

About This Task

Each Predix UI CSS module includes a `bower.json` file that lists other required components. This is displayed as a series of `@import` declarations in the respective file. For more information, see Application Development with the Predix Design System on page 7 for more information.

Procedure

1. To clone your copy of the `px-buttons-design` CSS module, at a command prompt, enter.

   ```
   git clone https://github.com/predixdesignsystem/px-buttons-design.git
   ```

   The command also creates a `px-spinner` directory containing the component files.
2. Install the `px-buttons-design` CSS module by using the `bower` package-management utility:

   ```
   bower install --save px-buttons-design
   ```

3. In the project SASS file `<web_app_project>.scss`, import the `px-buttons-design` CSS module at the Objects layer:

   ```
   // Settings
      @import "px-colors-design/_settings.colors.scss";
      // Tools
      // Generic
      @import "px-normalize-design/_generic.normalize.scss";
      @import "px-box-sizing-design/_generic.box-sizing.scss";
      @import "px-helpers-design/_generic.helpers.scss";
      // Base
      @import "px-flexbox-design/_base.flexbox.scss";
      @import "px-viewport-design/_base.viewport.scss";
      @import "px-typography-design/_base.typography.scss";
      // Meta
      //Object
      @import "px-buttons-design/_objects.buttons.scss";
      // Component
   ```

```
   // Trumps
   @import "inuit-clearfix/_trumps.clearfix.scss";
   @import "px-spacing-responsive-design/_trumps.spacing-
responsive.scss";
   @import "px-widths-responsive-design/_trumps.widths-
responsive.scss";
```

4. Enable `px-buttons-design` CSS module features. The following example enables the `$inuit-enable-btn--small` feature by setting the feature flag to `true` before the `px-buttons-design` `@import` statement.

```
//Object
$inuit-enable-btn--small : true;
@import "px-buttons-design/_objects.buttons.scss";
```

5. (Optional) Change any available Predix UI CSS module style variable by setting a new value and inserting the style variable before the `@import` statement:

```
//Object
$inuit-btn-color : $black;
@import "px-buttons-design/_objects.buttons.scss";
```

6. In your web application, add the appropriate `px-buttons-design` call to the desired element.

```
<!DOCTYPE html>
    <html>
      <head>
     </head>
     <body>
         <button class="btn btn--small">Small</button>
     </body>
  </html>
```

   For more information, see the `px-buttons-design` CSS module API reference documentation.

Predix UI CSS Visual Library

ThePredix Design System includes a library of CSS visual modules in the `predixdesignsystem` public GitHub organization to customize the base content design of a web application.

As a developer, you can use the code in the following GitHub repositories to produce these interactions.

Note: See Getting Started with Predix UI CSS Modules on page 56 for more information using the Predix UI CSS visual modules in the `predixdesignsystem` public GitHub organization.

Table 10: Predix UI CSS Visual Library

GitHub Repository	Description	Documentation
px-actionable-design	Actionable text applies button style to text string or icon, allowing a user to associate an action with the text.	API Reference
px-button-group-design	Implements button groups (multi- or unique-select).	API Reference
px-buttons-design	Extensible baseline module for building suites of buttons.	API Reference

GitHub Repository	Description	Documentation
px-meta-buttons-design	Shortcut collection of Predix Design System button-related modules.	API Reference
px-code-design	Styling for code snippets.	API Reference
px-colors-design	Assigns SASS variables for the Predix Design System default color palette.	API Reference
px-forms-design	Defines input fields and styles for form elements.	API Reference
px-headings-design	Defines font sizes for base-level heading elements.	API Reference
px-helpers-design	Generic utilities such as `calculatetoRem` and `visually hidden`.	API Reference
px-list-bare-design	Removes bullets and indentations from lists.	API Reference
px-list-inline-design	Displays a list as one horizontal row.	API Reference
px-list-ui-design	Creates blocked, keyline-delimited list items.	API Reference
px-meta-lists-design	Shortcut collection of all Predix Design System list-related modules.	API Reference
px-tables-design	Provides useful helpers for common `<table>` patterns.	API Reference
px-toggle-design	Simple implementation of a toggle switch for on/off states.	API Reference

Note: The API reference documentation for each CSS visual module includes usage instructions and examples.

File a GitHub issue if you encounter a problem using a Predix UI CSS visual module. For more information on filing a GitHub issue, see https://guides.github.com/features/issues/.

Predix UI CSS Layout Library

The Predix Design System includes a library of CSS layout modules in the `predixdesignsystem` public GitHub organization to customize the base layout of a web application.

As a developer, you can use the code in the following GitHub respositories to produce these interactions.

Note: See Getting Started with Predix UI CSS Modules on page 56 for more information using the Predix UI CSS layout modules in the `predixdesignsystem` public GitHub organization.

Table 11: Predix UI CSS Layout Library

GitHub Repository	Description	Documentation
px-box-design	Preconfigured box layout.	API Reference
px-flag-design	Media object containing an image and descriptive content.	API Reference
px-flexbox-design	Arranges, aligns and distributes space among items in a container using a flex layout.	API Reference

GitHub Repository	Description	Documentation
px-layout-design	Grid system definition and layout engine.	API Reference
px-spacing-design	Collection of helper classes for spacings such as margin and padding.	API Reference
px-spacing-responsive-design	Collection of helper classes that provide breakpoint-based classes for nudging margins and paddings around responsively.	API Reference
px-widths-design	Collection of helper classes for use in defining widths for items such as grid systems.	API Reference
px-widths-responsive-design	Collection of helper classes that provide breakpoint-based classes for widths on elements.	API Reference
px-widths-tools-design	Generates a suite of utility classes for sizing pieces of user interface.	API Reference

Note: The API reference documentation for each Predix UI CSS layout module includes usage instructions and examples.

File a GitHub issue if you encounter a problem using a Predix UI CSS layout module. For more information on filing a GitHub issue, see https://guides.github.com/features/issues/.

Predix UI CSS Utilities Library

The Predix Design System includes a library of CSS utilities modules in the `predixdesignsystem` public GitHub organization to customize the base typography, layout, and content design in a web application

As a developer, you can use the code in the following GitHub repositories to produce these interactions.

Note: See Getting Started with Predix UI CSS Modules on page 56 for more information using the Predix UI CSS utilities modules in the `predixdesignsystem` public GitHub organization.

Table 12: Predix UI CSS Utilities Library

GitHub Repository	Description	Documentation
px-box-sizing-design	Restricts all elements to use the `border-box` box model.	API Reference
px-clearfix-design	Minimal Clearfix helper class.	API Reference
px-defaults-design	Contains required variables and settings for using the design libraries.	API Reference
px-functions-design	Contains functions required to use the design libraries. **Note:** It is recommended that you install the `px-defaults-design` module which includes this module as a dependency..	API Reference
px-iconography-design	Defines styles for icon sets.	API Reference
px-mixins-design	Set of functions that can be used across other modules.	API Reference

GitHub Repository	Description	Documentation
px-normalize-design	A set of reset rules which force browsers to render all UI elements more consistently.	API Reference
px-typography-design	Defines styles for basic typography.	API Reference
px-viewport-design	Default viewport parameters.	API Reference

Note: The API reference documentation for each Predix UI CSS utilities module includes usage instructions and examples.

File a GitHub issue if you encounter a problem using a Predix UI module. For more information on filing a GitHub issue, see https://guides.github.com/features/issues/.

Predix UI CSS Module Overview

As a developer, you should review the conceptual, task, and reference information for the following tools before using a Predix UI Predix UI CSS module in web application:

- CSS Framework
- SASS Language
- Bower Manifest File
- BEM Naming Convention
- Default Flags
- Defined Variables
- Import Order

CSS Framework

The Predix UI CSS module libraries are implemented using the `incuit.css` CSS framework that provides standardized code to include in a web application.

The Predix UI CSS module libraries provide the following benefits:

- Web applications that follow consistent UX practices, standards, and specifications.
- Modular architecture that provides the flexibility to change and configure existing web applications.
- Responsive web design that supports laptop, tablet, phone, or large-screen devices.

See the following rewources for more information:

- What is a CSS Framework?
- INUIT.CSS v5.0 The Object Oriented CSS Framework

SASS Language

Each Predix UI CSS module library uses SASS language (`.scss`) files that compile into well-formatted, standard CSS.

The SASS (Syntactically Awesome Style Sheet) preprocessor adds several enhancements available in object-oriented programming languages::

- Preprocessing
- Variables
- Nesting
- Import
- Mixins

- Inheritance
- Operators

See the following resources for more information:

- SASS (Syntactically Awesome StyleSheets)
- SASS Tutorials

Bower Manifest File

The `bower` task-runner utility uses the `bower.json` manifest file to configure a web application.

The `bower.json` file lists the other Predix UI libraries required for successful compilation of the project SASS (`.scss`) files contained in the parent library. These libraries are stored in the **`dependencies`** section of the `bower.json` file. The `bower` task-runner utility uses the `bower.json` file to import the correct version of other required libraries.

Note: For more information, see https://github.com/predixdesignsystem/px-getting-started.

BEM Naming Convention

Every Predix UI CSS module library uses the BEM naming convention to make the code more consistent and informative.

The BEM (block, element, modifier) naming convention follows this pattern:

```
$<block>
$<block>__<element>
$<block>__<element>--<modifier>
```

where:

- `$<block>` represents the higher level of an abstraction or component.
- `$<block>__<element>` represents a descendent of `$<block>` that helps form `$<block>` as a whole.
- `$<block>__<element>--<modifier>` represents a different state or version of `$<block>__<element>`.

For example:

```
$inuit-btn-group-background
$inuit-btn-group-background--hover
```

where:

- `$inuit-btn-group-background` is a block.
- `$inuit-btn-group-background--hover` represents a different state of the `$inuit-btn-group-background` block.

Note: For more information, see http://getbem.com/naming/ .

Default Flags

All Predix UI CSS module library features are disabled by default to reduce the amount of code in the project SASS (`.scss`) file of a web application

To enable a feature, set its flag to `true` before the `@import` statement in the project SASS (`scss`) file of the web application. This reduces the Predix UI size by enabling only the CSS module libraries that you need.

The following example will only enable the CSS for for the full and large button features:

```
$inuit-enable-btn--full:    true;
$inuit-enable-btn--large:   true;
@import "px-buttons-design/_objects.buttons.scss";
```

Note: For more information, see https://github.com/predixdesignsystem/px-getting-started.

Defined Variables

Configure a Predix UI CSS module library by defining its variables immediately before the `@import` declaration in the project SASS (`.scss`) file of a web applicataion.

Modifying Predix UI libraries in this fashion eliminates the need to edit any files directly, and also negates the need to configure an entire library from a large variable file.

For example:

```
$inuit-btn-background:      rgb(247,247,247);
$inuit-btn-color:           rgb(47,47,47);
$inuit-enable-btn--full:    true;
@import "px-buttons-design/_objects.buttons.scss";
$inuit-enable-layout--middle:   true;
@import "px-layout-design/_objects.layout.scss";
$inuit-enable-flag--rev:        true;
$inuit-enable-flag--responsive: true;
@import "px-flag-design/_objects.flag.scss";
```

Note: For more information, see https://github.com/predixdesignsystem/px-getting-started.

Import Order

Import a Predix UICSS module library into the project SASS file in the correct order because it affects system operation.

Each project SASS (`.scss`) of a web application should observe the following import order:

1. Settings – Global variables, site-wide settings, or configuration switches.
2. Tools – Site-wide mixins and functions.
3. Generic – Low-specificity, far-reaching rulesets (such as `normalize.css`).
4. Base – Unclassed HTML elements (such as block quote {}or address {}).
5. Meta – `px-meta-buttons-design` and `px-meta-lists-design` libraries reside here
6. Objects – Objects, abstractions, and design patterns (such as `.media` {}).
7. Components – Discrete, complete UI chunks (such as `.carousel` {}). Predix UI does not provide code for this layer, as it is where your SASS code should reside.
8. Trumps – High-specificity, explicit selectors, including overrides and helper classes (such as `.hidden` {}).

For example:

```
// Settings
@import "px-colors-design/_settings.colors.scss";
// Generic
@import "px-normalize-design/_generic.normalize.scss";
@import "px-box-sizing-design/_generic.box-sizing.scss";
@import "px-helpers-design/_generic.helpers.scss";
// Base
@import "px-flexbox-design/_base.flexbox.scss";
@import "px-viewport-design/_base.viewport.scss";
@import "px-typography-design/_base.typography.scss";
// Trumps
@import "inuit-clearfix/_trumps.clearfix.scss";
@import "px-spacing-responsive-design/_trumps.spacing-responsive.scss";
@import "px-widths-responsive-design/_trumps.widths-responsive.scss";
```

Note: For more information, see the API reference documentation of a Predix UI CSS module library (https://www.predix-ui.com/) under the **Import it in your Sass** section.

Predix Design System Release Notes

Predix Design System Release Notes

Q1 2018 Release Notes

New Features

The following new Predix Design System features have been added.

px-product-switcher

This user interface component switches between products or applications, See the API reference for more details and code examples.

px-notification

This user interface componentt to notify the user of a state change or actions taken on a specific context. See the API reference for more details and code examples.

px-threshold-bar

This user interface componentt displays linear horizontal gauges with configurable threshold areas. See the API reference for more details and code examples.

Enhancements

The following Predix Design System enhancements have been added.

px-dropdown

This user interface component has added a mobile ressponsive view. See the API reference for more details and code examples.

px-modal

This user interface component has added a property and configuration for full-screen application modals . See the API reference for more details and code examples.

Q4 2017 Release Notes

New Features

The following new Predix Design System features have been added.

Polymer 2.x Upgrade

Almost all Predix UI components are upgraded to be Polymer 1.x/2.x hybrid elements, so that they can be used with either Polymer version. Benefits of upgrading to Polymer 2.x include:

- Better interoperability with JavaScript frameworks such as AngularJS, React, and Vue.js.
- Better performance in browsers with native support for web components.
- Better Predix UI component performance by using features not available in Polymer 1.x.
- Continued Google support of Polymer 2.x (e.g.., bug fixes).

Note:

The syntax of light DOM content distribution slots has changed in Polymer 2.0, resulting in the following API breaking changes / major version updates for these Predix UI components (see the API reference of each Predix UI component for more details and code examples):

- px-card (v2)
- px-modal (v3)
- px-view-header (v2)
- px-widget-cards (v4)

Enhancements

The following Predix Design System enhancements have been added.

Data Visualization

The data visualization components include these enhancements:

- Added a flexible API for creating and interacting with chart annotations.
- Added a single-datapoint search option.
- Added a lasso functionality for selection of datapoints.
- The web worker is now a blob so that you no longer needs to include the files in the build process.

Datetime Refactoring

The datetime components include these enhancements:

- All datetime components now support an empty state, where no date or range has been selected.
- A property has been added for "required" datetime fields so that validation will fail if they are not completed.
- The `dateTime` property has been removed for the following datetime components:
  - px-calendar-picker
  - px-datetime-field
  - px-datetime-picker

  The `momentObj` is the new source of truth. To set an initial value, pass in a `moment` object to the `momentObj` property. Listen to the `moment-obj-changed` event to know when the `moment` object has been updated.
- The `range` property has been removed has been removed for the following datetime components:
  - px-datetime-range-field
  - px-datetime-range-panel
  - px-rangepicker

  The `fromMoment` and `toMoment` are the new sources of truth. To set an initial value, pass in `moment` objects to the `fromMoment` and `toMoment` properties. Listen to the `px-datetime-range-submitted` event to know when either one of the `moment` objects has been updated. Another option is to listen to `from-moment-changed` and `to-moment-changed` to know when `fromMoment` and `toMoment` update separately.

px-context-browser

This user interface component has added the ability to set "favorites" for quicker access. See the API reference for more details and code examples.

px-dropdown

This user interface component has added a slot so that any arbitrary element can be used as the dropdown trigger. See the API reference for more details and code examples.

px-inbox

This user interface component has added a property to o make referencing the selected item easier (removed dependency on moment.js). See the API reference for more details and code examples.

px-modal

This user interface component has simplified trigger/modal relationship and configuration of positive/negative action buttons. See the API reference for more details and code examples.

px-tile

This user interface component has updated the visual design, including a new property for text that appears on the bottom of tile instead of on hover. See the API reference for more details and code examples.

Known Issues

The following Predix Design System issues are noted for Q4 2017.

Views Service Behaviors are Deprecated

The Predix Design System Cirrus version of the `px-card` component does not support Views service behaviors to be consistent with other Predix UI components that separates the service layer and UI layer. See Predix Design System Cirrus API Changes on page 19 for more information.

Q3 2017 Release Notes

New Features

The following new Predix Design System features have been added

Data Visualization

Added zooming capability to the following Predix UI data visualization components (see the API reference of each datetime component for more details and code examples):

- px-vis-radar
- px-vis-parallel-coordinates
- px-vis-polar

px-chip

A user interface component that is a representation of an object or a series of information. For example, a chip can be used to represent a contact or a filter. Use a chip in order to represent complex information in a compact way. See Predix UI Templates on page 30 for the specific component API reference that includes more details and code examples. .

px-heatmap-grid

A data visualization component to display a data set based on a heat map scale of colors. See Predix UI Components on page 31 for the specific component API reference that includes more details and code examples. .

px-panel

A user interface component that provides a container for placing controls or content. It can be fixed or relatively positioned, and persistent or shown and hidden. See Predix UI Templates on page 30 for the specific component API reference that includes more details and code examples. .

px-steps

A user interface component that visually represents a sequence of steps. It is useful for creating wizards, multi-step forms, and other step-by-step interactions. See Predix UI Templates on page 30 for the specific component API reference that includes more details and code examples. .

px-tile

A user interface component that displays an image, a title, and a description. It can display an image with text either inside the image or under the image along with customizable caption. See Predix UI Templates on page 30 for the specific component API reference that includes more details and code examples. .

Enhancements

The following Predix Design System enhancements have been added

Data Visualisation

There is now an option to add `event name` to the `px-tooltip` component and customize the location of the chart toolbar submenu

Map Framework

The `px-map-marker` component is extended to allow for custom colors and types.

Icons

A new set of 177 icons for industrial web application needs.

Predix Design System Web Site

The Predix Design System web site (https://www.predix-ui.com/#/home) includes these enhancements:

- Component gallery and filtering
- Added developer & design guidelines
- Added expand/collapse in demos for code
- Added mobile/tablet/desktop indicator.

Q2 2017 Release Notes

New Features

The following new Predix Design System features have been added.

Predix Design System Cirrus

Predix Design System Cirrus brings a more contemporary and sophisticated visual style, and more diversity of design patterns.

Predix Design System Cirrus includes:

- Fresh code for all 117 Predix UI components and CSS modules.
- New visual language for all components.
- Design guidelines.
- Sketch stencils with a Stencil Starter Kit.
- New navigation patterns; horizontal, vertical, and collapsible to improve web app navigation.
- A new single column Context Browser component and separate Breadcrumb component.
- New page header/branding bar.
- New set of 177 icons which replaces the font-awesome icons.
- New animations.

Internationalization

Most Predix UI components can be localized, using one or more of the three following methods: localizing text, localizing date and time and localizing number formatting. Predix Design System does not provide translation for strings, only the hooks for you to provide your own translated resources.

See Localizing Predix UI Components on page 39 for more information.

Map Framework

The `px-map` framework adds new subcomponents:

- The `px-map-layer-geojson` subcomponent can draw GeoJSON geometries as vector shapes on the map. The component allows for complex, feature-level styling. It should be very helpful for anyone looking to draw additional feature/asset types with px-map.
- The `px-map-tile-layer-bing` subcomponent that loads base layer tiles from the Bing Maps API. It handles authentication and communication with the API, making it simple to add Bing's aerial or road imagery to your map.

See Predix UI Components on page 31 for more information.

px-accordion

A user interface component that provides expandable and collapsible subsection headers for a page. An optional "action" icon on the right side will fire an event when pressed.

See Predix UI Components on page 31 for more information.

px-app-header

A user interface component that allows the `px-branding-bar` and `px-app-nav` components to work together.

See Predix UI Components on page 31 for more information.

px-app-route

A user interface component that supports easy binding between various Predix UI components and a web application's URL. It helps keep the user interface state in sync with the URL allowing users to bookmark or share views by copying the URL

See Predix UI Components on page 31 for more information.

px-branding-bar

A user interface component that provides a header area to contain title, logo and branding content.

See Predix UI Components on page 31 for more information.

px-breadcrumb

A user interface component that works with the `px-context-browser` and `px-tree` components to visually represent the asset model hierarchy and jump to levels of the hierarchy.

See Predix UI Components on page 31 for more information.

px-icon-set

A user interface component that serves as a wrapper for the Predix icon set which leverages px-icon. 177 new icons have been added to Predix Design System which replaces the `px-polymer-font-awesome` component.

See Predix UI Components on page 31 for more information.

px-key-value-pair

A user interface component that allows you to prominently display information in a dashboard.

See Predix UI Components on page 31 for more information.

px-tree

A user interface component that provides an expandable and selectable tree. The contents are controlled primarily through a structured object of branch and leaf nodes.

See Predix UI Components on page 31 for more information.

Enhancements

The following Predix Design System enhancements have been added.

Map Framework

The `px-map` framework adds these enhancements:

- Added dark-theme version of `px-map` component (see API documentation).
- Added the ability to bind popups to markers in `px-map-marker-group`. See the API documentation for that component for a thorough guide to setting up your features with marker icons and popups. You can also see `examples/marker-clusters-with-popups.html` for an example of it in use.
- Refactored the `px-map` component to customize how the `fitToMarkers` event happens (`fitToMarkersZoom` and `fitToMarkersPadding`). See the API documentation for details.
- Added tap events to `px-map-marker-group` that fire when clusters or any individual marker inside a cluster is tapped

See Predix UI Components on page 31 for more information.

Predix UI Component Library

The following enhancements improve the information available on the www.predix-ui.com site, including:

- Added ability to see demos in the dark theme.
- Added more guidelines to help designers and developers to implement new designs / features .

px-app-nav

This user interface component has been enhanced to allow horizontal, vertical, and collapsible navigation between major sections of a web application. Navigation items can be nested, with sub-items. Each major navigation item should have a unique and descriptive icon.

See Predix UI Components on page 31 for more information.

px-context-browser

This user interface component has been enhanced to allow single column navigation of a hierarchical model. The context browser allows the user to drill down through a hierarchical menu in order to switch the context of the overall view.

Examples of possible contexts include specific locations organized geographically, physical assets organized by model or other classification, or employees organized by department or function. When a user clicks on the title or arrow, the expanded view of the context browser covers cards and other content on the page.

Selecting an item in the context browser causes the children of that item to show up in the next panel over, and also causes a button to appear within that selected row that allows the user to submit/save that context, which also re-collapses the browser.

See Predix UI Components on page 31 for more information.

px-dropdown

This user interface component was refactored and simplified by making use of iron-dropdown and iron-selector. Several bug fixes and enhancements were added, including an easier method for retrieving selections and the ability to clear selected values.

See Predix UI Components on page 31 for more information.

Vis Framework

The following enhancements are added to the `px-vis` framework:

- The crosshair feature provides users the ability to highlight corresponding datapoints (for example, datapoints which share the same timestamp - though not necessarily the same exact dataset) across multiple charts on a page.
- The `px-vis-timeseries` component adds a striping feature that allows users to highlight a selection of the chart so the application can perform an action with this highlighted data. There is also the ability to cancel the selection and re-highlight.
- Introduced the use of web workers to handle background processing of user-intensive tasks.
- Refactored rendering of event markers for the `ps-vis-timeseries` component to improve performance for use cases with 1000s of event markers.
- Added the ability to mute or unmute the series in the `px-vis-radar` and `px-vis-parallel-coordinates` components when different colors are used for the series.
- Added the ability to have multiple series rendered on the `px-vis-polar-chart` component.

See Predix UI Components on page 31 for more information.

Q1 2017 Release

New Features

The following new Predix Design System features have been added.

Predix Design System Map Framework

The `px-map` user interface component is a framework that includes basic mapping functionality (such as markers, clusters, popups, locator, panning/zooming, and the ability to apply layers).

The `px-map` user interface component provides the following benefits:

- Developers can provide users the ability to visualize the geographic location and other information about assets or asset groups on a map.
- Developers can integrate applications with a preferred mapping/GIS tile service.

See Predix UI Components on page 31 for more information.

Chart and Chart Register Interaction Toolbar Menus

The following data visualization components have added a toolbar menu (in the top right corner of the chart) for chart settings:

- px-vis-timeseries
- px-vis-xy-chart

A developer can specify the number of items that can be configured in the tool bar menu.

See Predix UI Components on page 31 for more information.

Chart Panning and Zooming

The following data visualization components have added controls (at the top of the chart) for a user to pan and zoom on a selected region of the chart:

- px-vis-timeseries
- px-vis-xy-chart

See Predix UI Components on page 31 for more information.

px-gauge

A user interface component that displays a single value between a range by radial. The value is presented by the color of gauge bar and the position of the needle.

See Predix UI Components on page 31 for more information.

px-number-formatter

A user interface component that displays a formatted string of a numeric value, or a numeric value of a formatted string.

See Predix UI Basics on page 29 for more information.

px-toggle

A user interface component that acts as a simple on/off switch to enable and disable things.

See Predix UI Components on page 31 for more information.

Enhancements

The following Predix Design System enhancement have been added.

Vis Framework

The `px-vis` framework is a charting framework that uses the D3.ja (4.4.x) data visualization library and Polymer web components. A major release of the Vis Framework is available for Q1 2017 that includes the following enhancements:

- Performance optimizations
- Framework changes to support new customer feature requests
- Threshold changes
- Scale changes
- Axis changes
- Register changes
- Tooltip changes
- Demo improvements
- Resize method improvements

For more information, see https://github.com/PredixDev/px-vis/blob/master/HISTORY.md.

Theming Guidelines and Demos

The Predix Design System theming guidelines include demos for developers to style a Predix UI component and apply the px-dark-theme style module for all Predix UI components. For more information, see Example: Theme Web Applications with Predix UI.

Predix Design System Component Library

Developers can see API reference information, usage instructions, and examples in a library of Predix UI user interface components that are available (http://predix-ui.com). This library is enhanced for Q1 2017 with a mechanism to add demos and documentation.

px-data-table

A user interface component that displays a data table with the expected data format is updated with the ability to select a single row.

See Predix UI Components on page 31 for more information.

px-dropdown

A user interface component that displays a customizable dropdown list is updated with a search capability.

See Predix UI Components on page 31 for more information.

px-slider

A user interface component to define a value in a range or a range of values within set boundaries. It is refactored for future enhancements beyond the existing ability to specify the minimum and maximum values, step, and default value.

See Predix UI Components on page 31 for more information.

px-vis-timeseries

A data visualization component to create an interactive timeseries chart. It now supports adding, removing, and repositioning (by drag and drop) of Y axes.

See Predix UI Components on page 31 for more information.

px-vis-xy-chart

A data visualization component to draw lines, scatter, or both against an X and Y axis. The Y axis can now be added, removed, and repositioned by drag and drop.

See Predix UI Components on page 31 for more information.

Q4 2016 Release

New Features

The following new Predix Design System features have been added.

Mobile Components

Components that provide a foundation for building web applications to look and work like native mobile applications:

- **px-drawer** - A user interface component that provides an off-screen menu drawer that can be swiped open or closed.
- **px-table-view** - A user interface component that helps organize menus and views into table-style blocks that can be expanded, swiped, and dragged and re-ordered.

See Predix UI Mobile Components on page 33 for more information.

Theming

A dark theme can be applied to user interface components.

px-inbox

A user interface component that displays a list of items with detailed information or actions associated with each item.

See Predix UI Basics on page 29 for more information.

px-kpi

A data visualization component that displays key performance indicators (KPI) in a dashboard layout.

See Predix UI Templates on page 30 for more information.

px-simple-win-loss-chart

A data visualization component that displays a series of positive & negative values as a bar chart..

See Predix UI Components on page 31 for more information.

px-timeline

A data visualization component to display a time-based series of events and document the details relevant to each event.

See Predix UI Components on page 31 for more information.

px-typeahead

A user interface component that searches user input and lists all the suggestions which has the input.

See Predix UI Basics on page 29 for more information.

px-view-header

A mobile-responsive user interface component that is located at the top of the page to display an overview of the content that a user is viewing.

See Predix UI Components on page 31 for more information.

px-vis-parallel-coordinates

A data visualization component to draw a parallel coordinates chart

See Predix UI Components on page 31 for more information.

px-vis-polar

A data visualization component to draw a pie or doughnut chart for polar data sets.

See Predix UI Components on page 31 for more information.

px-vis-radar

A data visualization component to draw a radar (spider) chart.

See Predix UI Components on page 31 for more information.

px-vis-spark

A data visualization component to display a small line chart without axes or measures that provides a user a glimpse of a trend.

See Predix UI Components on page 31 for more information.

px-vis-timeseries

A data visualization component to create an interactive timeseries chart.

See Predix UI Components on page 31 for more information.

px-vis-xy-chart

A data visualization component to draw lines, scatter, or both against an X and Y axis.

See Predix UI Components on page 31 for more information.

Enhancement

The following Predix Design System enhancement has been added.

Predix Design System Component Library

Developers can see API reference information, usage instructions, and examples in a library of Predix UI user interface components that are available (http://predix-ui.com). There is also a set of Predix UI CSS module libraries categorized into the following three groups:

- Visual
- Template
- Utilities

Q3 2016 Release

New Features

The following new Predix Design System features have been added.

Predix Design System Component Library

Developers can see API reference information, usage instructions, and examples in a library of Predix UI user interface components that are available in the `PredixDev` organization in GitHub (http://predixdev.github.io/predix-ui/).

.

Datetime Components

User interface components that allow a user to specify time and date:

- calendar picker
- datetime field
- datetime picker
- datetime range field
- datetime range panel
- rangepicker

See Predix UI Components on page 31 for more information.

Time Series Charts

User interface components that define chart series as:

- bar
- histogram
- line
- scatter

See Predix UI Components on page 31 for more information.

Vis Framework

Set of component basics that provide the ability to create charts:

- pie chart
- time series chart
- bar chart

See Predix UI Components on page 31 for more information.

px-file-upload

User interface component that provides users the ability to upload one or more files, whether as part of a larger form, or as a standalone component.

See Predix UI Basics on page 29 for more information.

px-percent-circle

User interface component that provides the simplest possible doughnut chart - a single ring that fills clockwise based on a percentage.

See Predix UI Components on page 31 for more information.

px-progress-bar

User interface component that indicates the progress of an activity with explicit start and end times, or known duration.

See Predix UI Basics on page 29 for more information.

Q2 2016 Release

Enhancements have been made to these Predix UI components.

px-rangepicker

- UTC support

px-context-browser

- Pre-pass context

px-popover

- Style enhancement for better visibility
- Added support for relative positioned elements
- Added support for transformed parent element

px-tabs

- Event hook on tab switch

px-alert-message

- Optional truncation methods (before, middle, after)