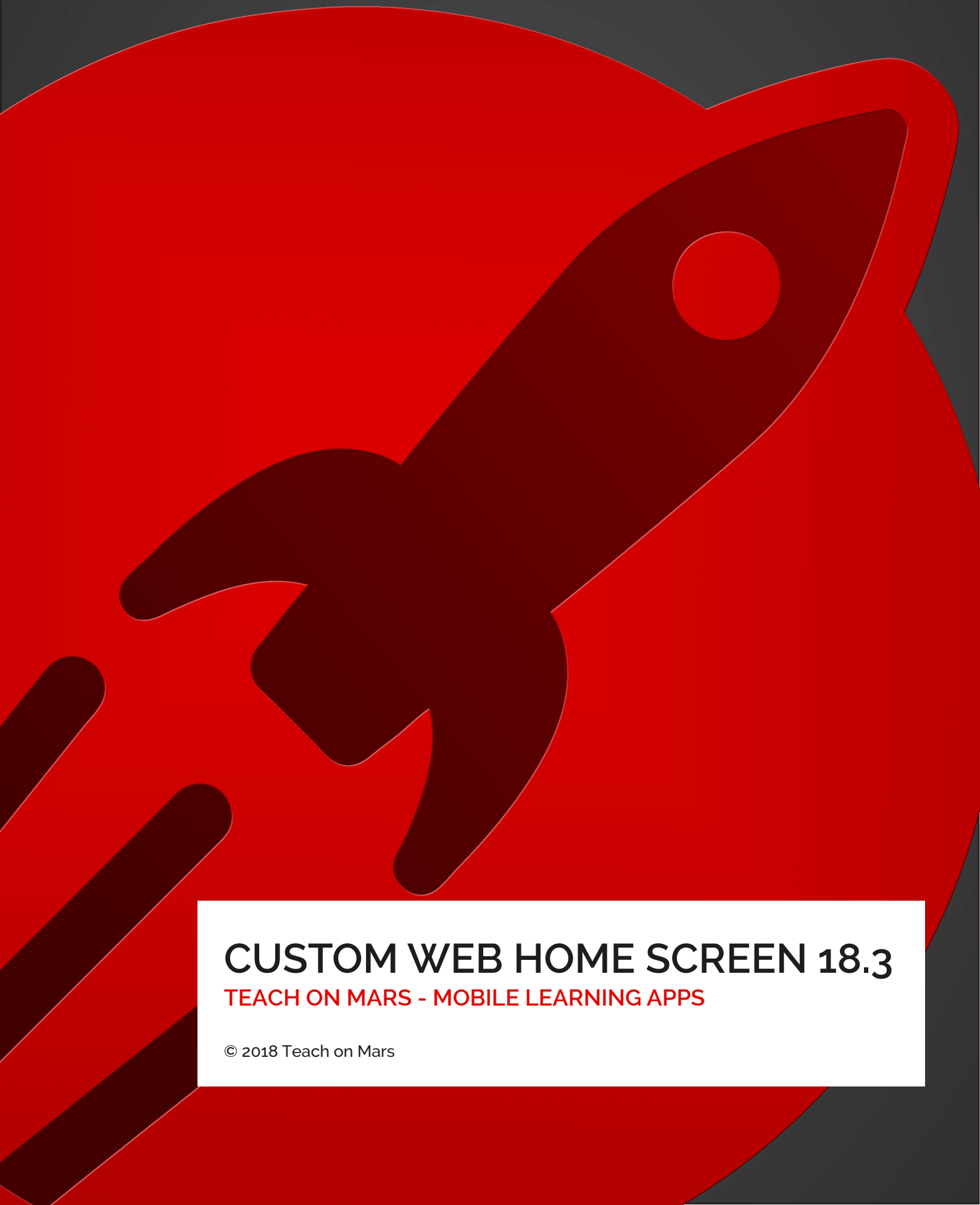
TEACH ON MARS



CUSTOM WEB HOME SCREEN 18.3

TEACH ON MARS - MOBILE LEARNING APPS

Integration Guide

Design guidelines

Interact with the Teach on Mars app

Introduction

The goal of this document is to provide a complete guide to the integration of a custom home screen for a Teach on Mars mobile or web application. It is intended for Teach on Mars clients and web agencies hired to create an original home interface.

Why a custom home screen?

The Teach on Mars application is great out of the box, and its home screen may already be configured to suit your needs and graphical charter. But we thought some of you might like something more.

So here comes the *Custom web home* that allows you to create pretty much any kind of home screen as long as it is web based. This web home will be able to interact with the standard part of the application.

Why a "web" home screen?

HTML5, CSS and Javascript are both very common and versatile technologies. HTML5 and CSS allows you to be very creative and build a unique user experience that includes your brand graphical identity. You will find many agencies and freelance developers that will gladly create your custom web home screen to enhance your learners' experience.

Design guidelines

Here are some quick tips to make sure the user experience of your home screen is perfect.

Think mobile first!

Your home screen will be displayed on a large variety of devices and screen sizes. So you need to create a responsive design that will work on different device dimensions and width/height ratios.

Questions that might help you design your home screen:

- Do you know what kind of device your users will use?
- Is it only tablets? Phones? Both?
- Are these standard devices? Does each user bring his own device?

The *Mobile first* rule applies to the graphic design, but also to the UX design.

Mobile first also means offline

Internet connectivity is a real issue with mobile devices, so designing mobile first also means designing something that can work offline. So, all files required for your custom home screen must be embedded in the application.

Mobile first but not only

If your design is so mobile first that it becomes hard to adapt it to a desktop sized web app, you might consider creating two custom home screen designs: one optimized for mobile devices and the other adapted to desktop sized screens.

Consistency

Even though the first impression of the user is very important, let's not forget the whole user experience. The home screen is the entry point to the application but behind it, the other screen that the user will browser through are standard Teach on Mars design. So, it is important to keep consistency between the home screen and the rest of the application.

Think about the font family you are using, the style and size of the interface labels, the style of the buttons, etc. Also keep consistency between the color palette you are using and the one that has been configured in the application.

Test

It might be obvious to say this but it is essential to test the display of your custom home on as many device as possible. And not only the display, but also the action of the links or buttons that you've put in your custom home screen.

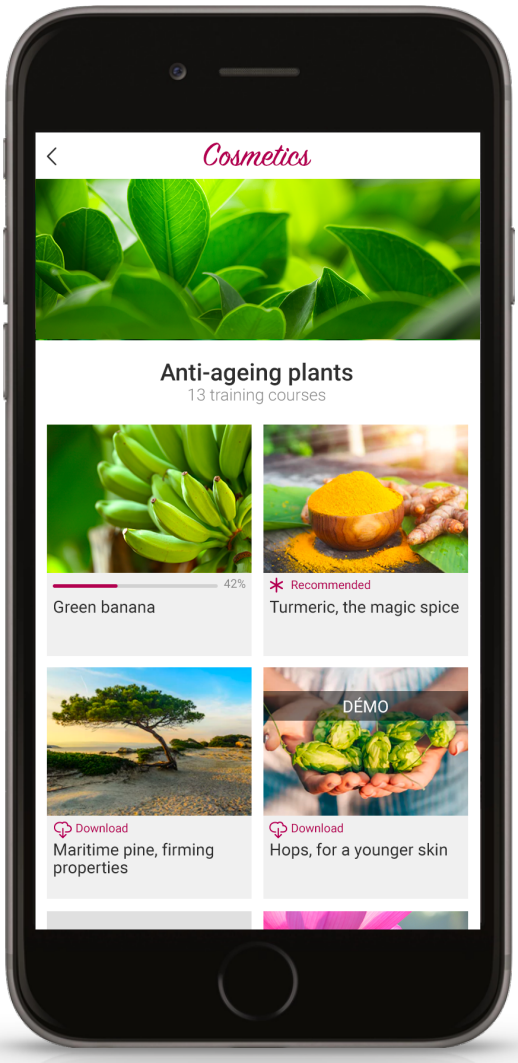
You don't have to test on every single device, but try to do it on a representative collection of devices (device platform and screen sizes). If you know what kind of device the target population uses, it is obvious what your tests should focus on. If you don't, there are plenty of statistics showing what are the most commonly used devices, so you should start with that.

Navigation

With the release of version 18.3, the navigation pattern in the Teach on Mars application has changed. There is now a *tab bar* that allows the user to navigate through the different *main screens* of the application:

- Wall
- Catalog
- Profile
- Search
- Apps

Any other screen accessed from one of these has a *back button* in the top left hand corner to navigate back to the related main screen.



Example: in a category screen the back button leads back to the catalog screen

Alright, but why is it important ?

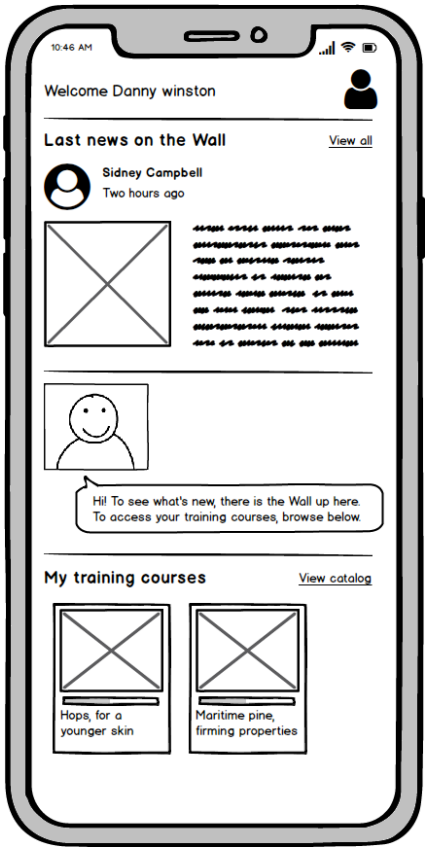
Well, with this new pattern there is simply no main home screen left.

It's gone!

But don't worry, you can still have your custom web home screen. It just requires a few things to keep the user experience complete. But we'll get back to it.

First, Let's go through the screen workflow in details with a custom web home screen.

When the application is launched the custom web home screen is displayed in *full screen* (i.e. without the tab bar and without the top bar) to offer the client a complete freedom of customization. The home screen must provide links to the rest of the application.



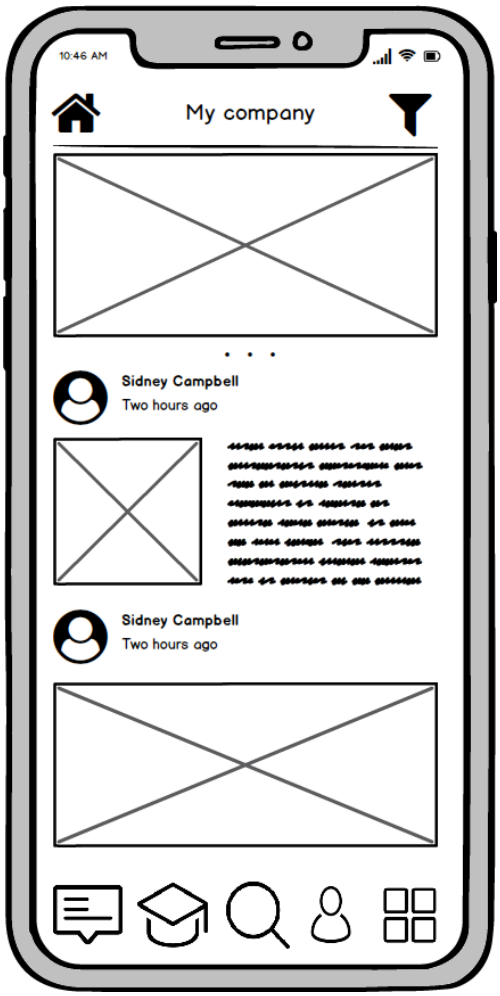
A custom home wireframe with links to the learner's profile, the wall and the catalog

Important notice

At least one of the links must lead to one of the main screens to allow the user to navigate through all the application.

This is the only requirement of the custom web home screen.

The user will be able to go back to the home screen by tapping the *home* icon on the top left hand corner of each main screen (this icon is not present when no custom home screen is configured).



Here's a wireframe of the wall screen
You can notice that there is now a Back icon
which leads back to the Custom Home Screen

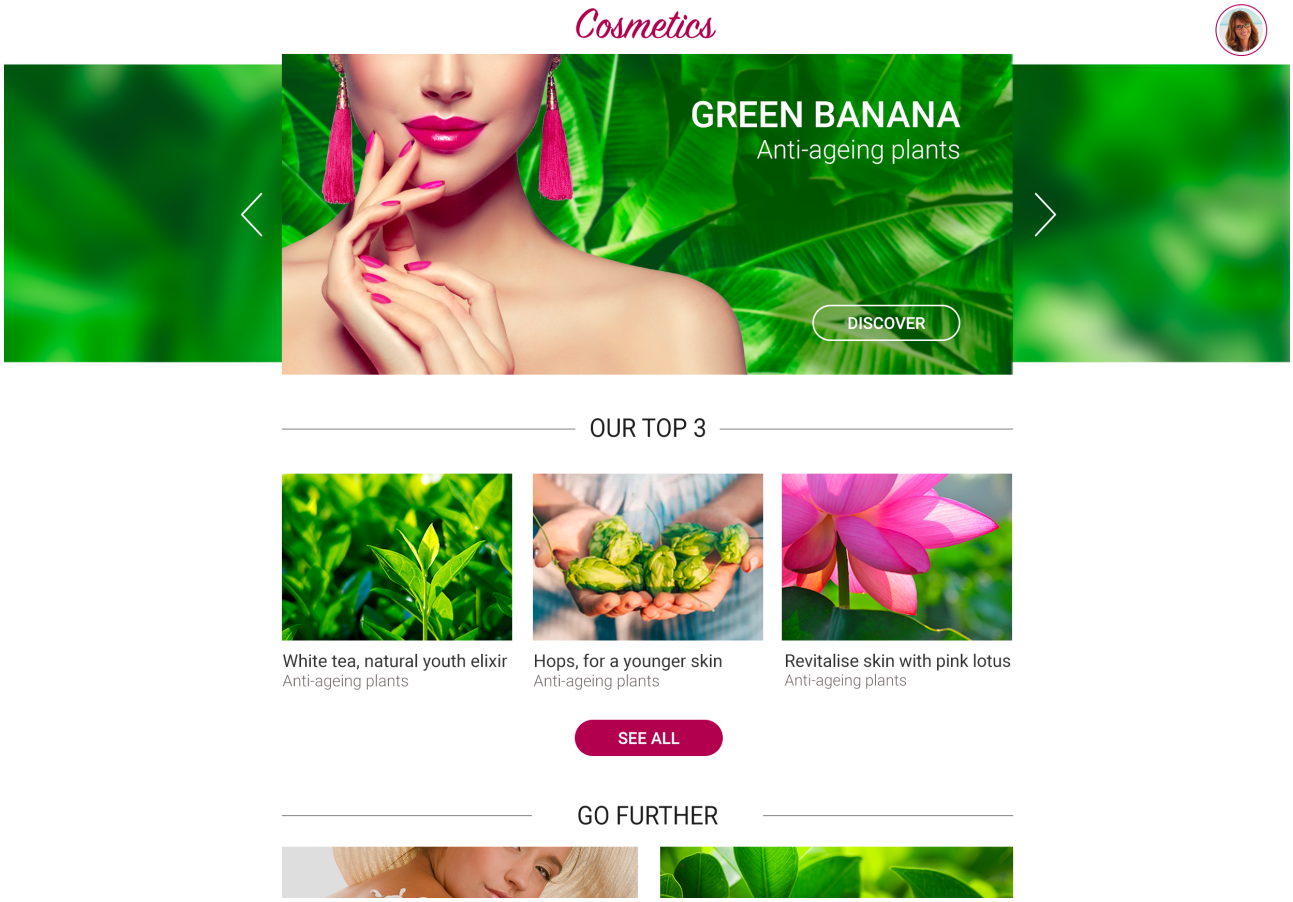
The rest of the navigation stays the same. If the learner uses the tab bar to navigate to others *Main screens* the *Back* icon is still here to offer a way back to the home screen.

Navigation in the web app

In the web app, the rules are pretty much the same as in the mobile app, but here are some details anyway.

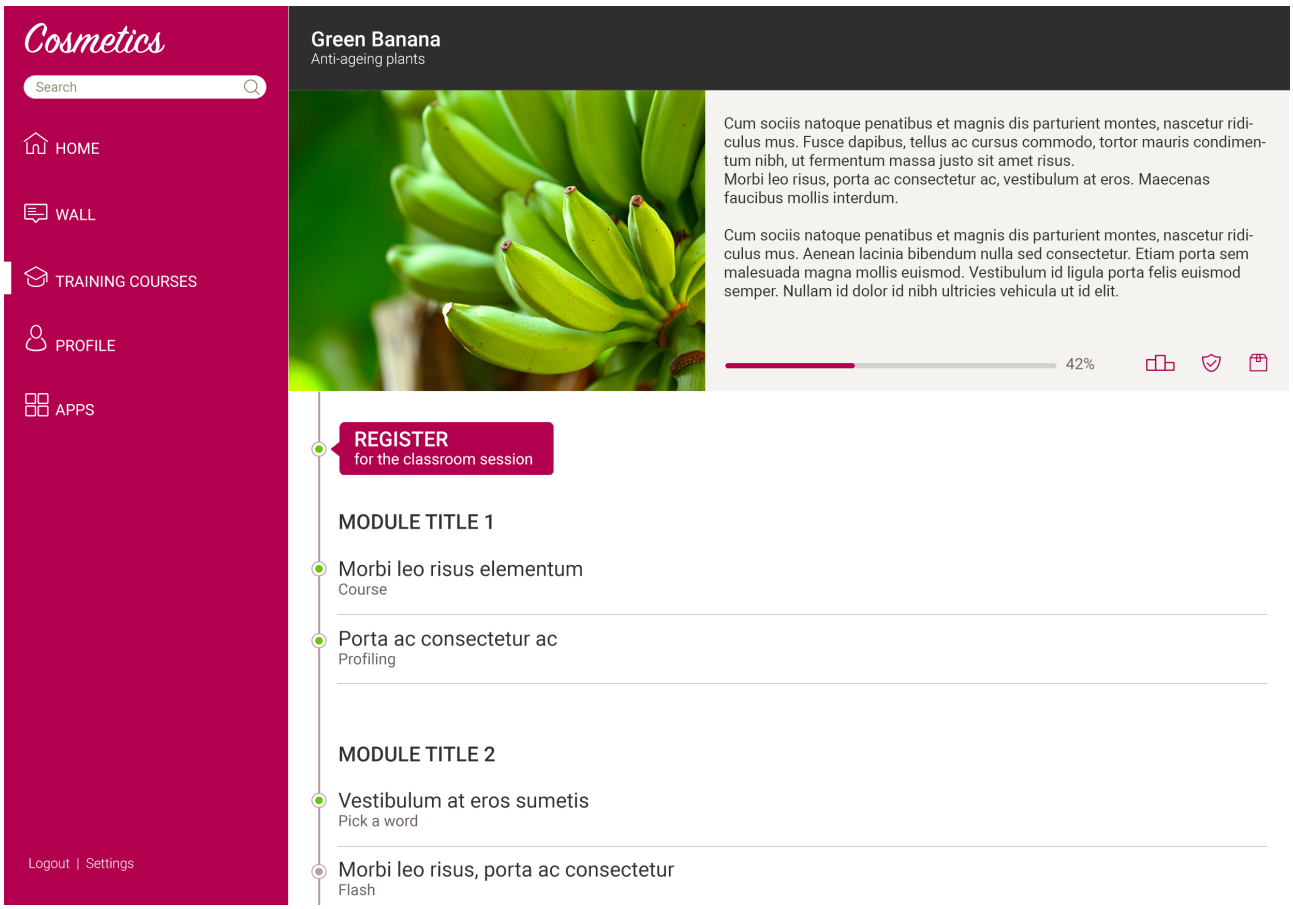
When the web app is launched, the custom web home is displayed in full screen. It literally covers the all window area. So you are free to use that space as you wish.

The sky's the limit.



Here's an example of custom home in a web app.
Nothing of the original webapp screen is visible.

But as there are navigation rules in the mobile app, there are also rules in the web app. The Custom home must provide a link to one of the main screen of the web app.



The user will then be able to navigate through the different tabs of the web app, and the web app will provide a way for the user to go back to the custom home through the *Home* link in the side bar.

Interact with the application

It is possible to use a JS library to fetch useful data from the Teach on Mars application. This way, your custom home can become dynamic and keep up with the updates that are made from the Mission Center.
In this section, we will cover how to use the ToM Home API to integrate data from the Mission Center in you custom home screen.

So here we go.

How to call these methods

The ToM Javascript library is organized in namespaces that each have a purpose.

First things first, there is some basic principles to know about the **ToM.appContent** and **ToM.navigation** library namespaces. All its methods actually return a *Promise*, so they should be called this way:

```
ToM.appContent.getCategories()
.then(function(categories) {
      _.each(categories, function(category) {
            // Now insert the HTML for the category in your home
            // with your own method
            insertCategoryToHome(category);
      }
})
```

This example uses lodash methods, but it's up to the developper to choose what's best.

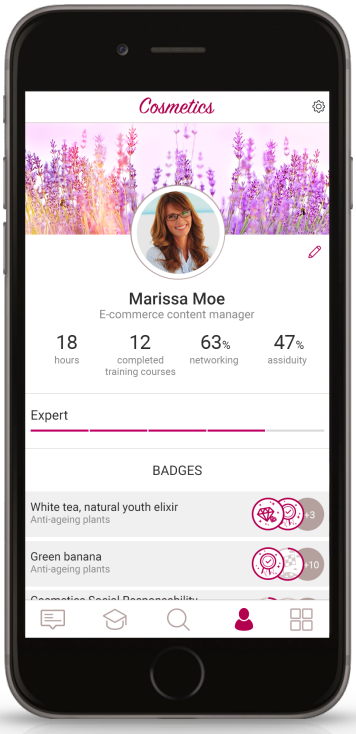
In the webapp
The custom web home in the web app is displayed in an iframe, so before you can start calling the following methods in the webapp, you need to get a link to the library object called *ToM* in your environment. To do that, you need to reference the *ToM* object in the parent window like this:

```
window.addEventListener('load', function () {
  window.ToM = window.ToM || (window.top && window.top.ToM) || {};
});
```

Navigation to main screens

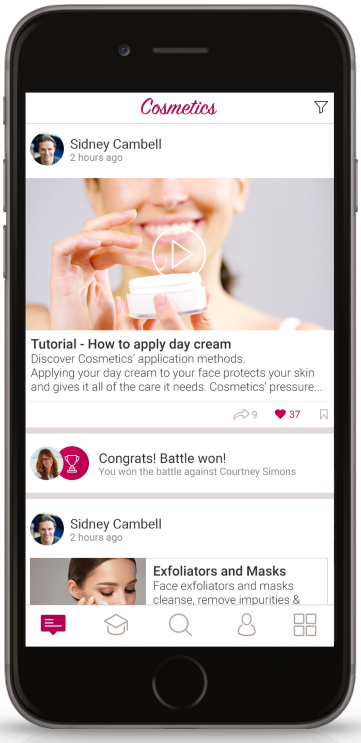
ToM.navigation.displayProfile()

Navigates the application to the logged in learner profile screen.



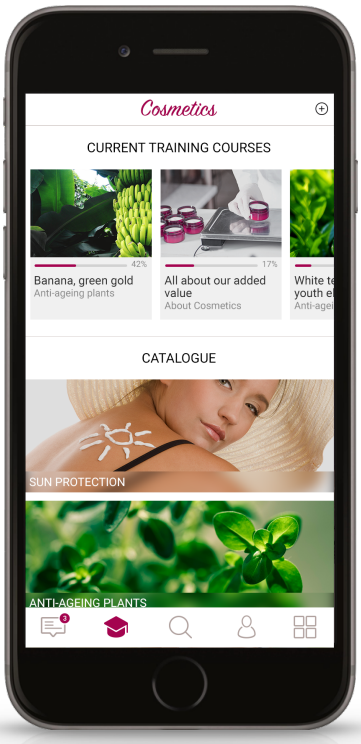
ToM.navigation.displayWall()

Navigates the application to the Wall screen.



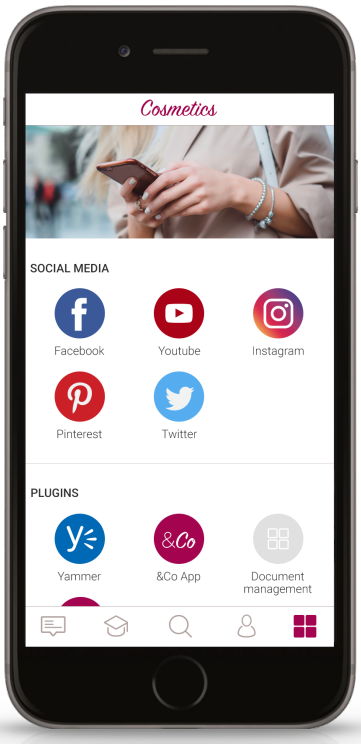
ToM.navigation.displayCatalog()

Navigates the application to the Catalog screen.



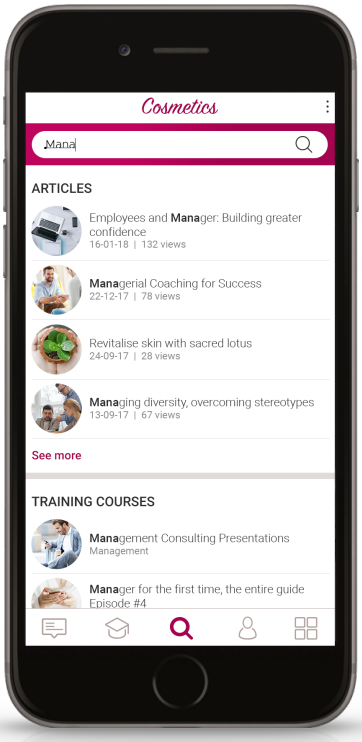
ToM.navigation.displayApps()

Navigates the application to the Apps screen.



ToM.navigation.displaySearch()

Navigates the application to the Search screen.



This method may also be called with an argument: the search query.

ToM.navigation.displaySearch("Mana") will redirect the application to the *Search* screen and launch a search query with *Mana*.

Catalog

ToM.appContent.getCategories()

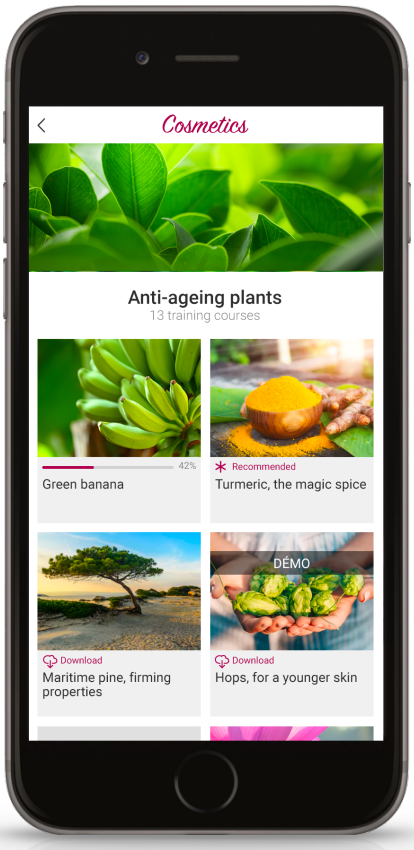
Returns a flat collection of categories that the learner have access to. Categories describe the tree-

```
[
  {
    "id" : "anti-ageing-12654726584",
    "title" : "Anti-ageing plants",
    "image" : "http://foo.com/products.jpg",
    "children" : [
      {
        "id" : "sub-category-87436536234",
        "title" : "Sub category",
        "image" : "http://foo.com/sub-category.jpg",
        "parent" : "anti-ageing-12654726584",
      }
    ]
  }, ...
]
```

like structure of the catalog and contain all the training courses.

ToM.navigation.displayCategory(categoryId)

Navigates to the defined category screen. This allows you to create a link from your custom home to a category screen, showing the training courses in this category.



ToM.appContent.getTrainingCoursesForLearner()

Returns a collection of training courses accessible to the current learner.


```
[
  {
    "id": "my-training-course",
    "description": "Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Quisque sed pellentesque tellus. Donec condimentum
eget mi id venenatis. Integer convallis ex quis felis hendrerit
dapibus. Phasellus turpis nisi, pretium id vehicula eget, porttitor
a arcu. Phasellus odio elit, dictum sit amet accumsan in, maximus
sed augue.",
    "title": "My Training Course",
    "cover": {
        "url": "https://my-userdata.teachonmars.com/api/v3/
training/my-training-course/image/cover/
b38b16f4e478455a1d5456873581963aa285bf547ff71d2f3ed5a42d9acd4289"
    },
    "thumbnail": {
        "url": "https://my-userdata.teachonmars.com/api/v3/
training/my-training-course/image/thumbnail/
7f01eb819246ea218ed7e6f37631d69b0a643cf9d0e5aba5b3b81a8a31b113c3"
    },
    "categories": ["category-1"],
    "demo": false,
    "recommended": false,
    "sandbox": false,
    "updateAvailable": false,
    "accessible": true,
    "progress": 20,
    "points": 75,
    "completed": false,
    "certified": false
  },
  ...
]
```

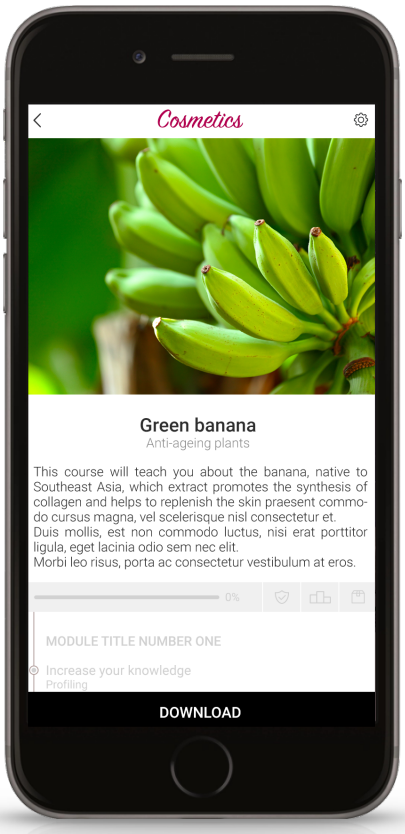

ToM.appContent.getTrainingCoursesForCategory(categoryId)

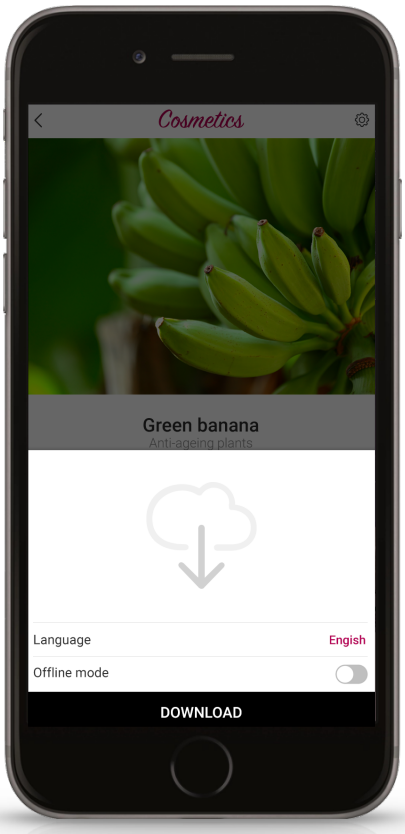
Returns a collection of training courses that are in the given category.
Same format as the *ToM.appContent.getTrainingCoursesForLearner()* method.

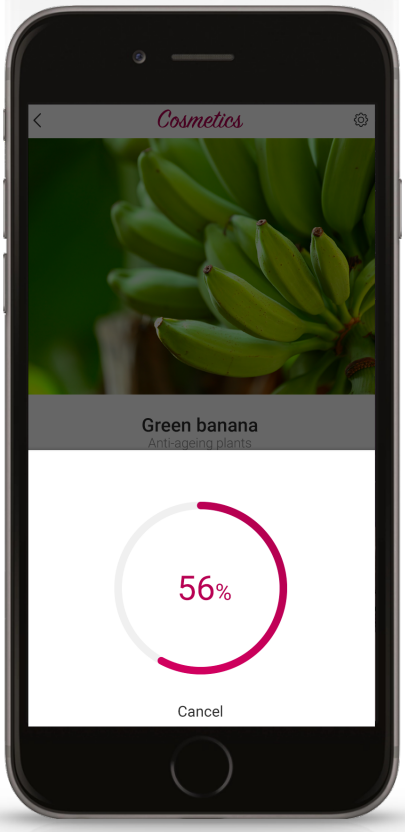
ToM.navigation.displayTrainingCourse(trainingCourseId)

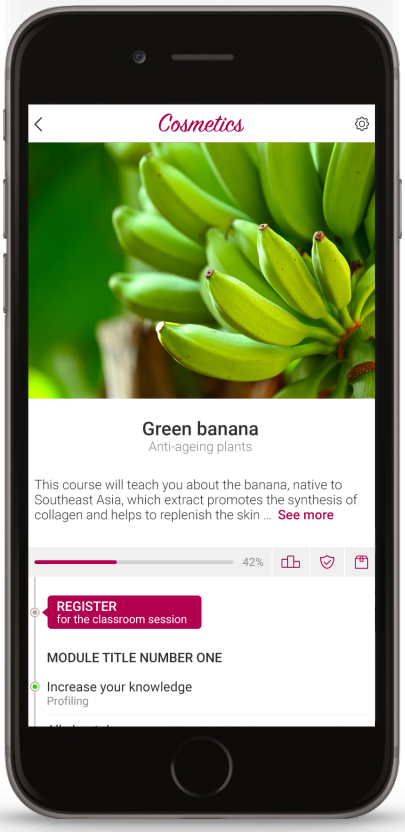
Navigates to the given training course screen. If the training course content needs to be downloaded, the application will automatically show the training path with a *Download* button at the bottom of the screen.

On the next page are some screenshots of the complete training course download process. Of course, if the user has already downloaded the training course, this method navigates straight to the last step.









Communications

ToM.appContent.getFeaturedCommunications()

Returns a collection of communications that have been tagged as *featured in the slider* in the Mission Center.


```
[
  {
    "id": "4714",
    "type": "article",
    "author": {
        "avatarUrl": "https://my-userdata.teachonmars.com/api/v3/
communication/avatar/longtext/7a49e231f69fb51e07a5b11843bcdf8a",
        "displayName": "Robert Alfred"
    },
    "article": [],
    "bookmarked": false,
    "description": "Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Morbi viverra eleifend turpis eu tincidunt.
Quisque varius nisi elementum dolor posuere luctus. In quis
vestibulum arcu, eu tincidunt tellus. Sed at sem et neque suscipit
volutpat. Pellentesque in lectus id metus ultricies rutrum.",
    "featured": true,
    "featuredImage": {
      "size": {
        "height": 545,
        "width": 750
      },
      "url": "https://my-userdata.teachonmars.com/api/v3/media/
e3f1c026-819e-4c10-9b4b-5b40f90ab4bf"
    },
    "image": {
      "size": {
        "height": 270,
        "width": 750
      },
      "url": "https://my-userdata.teachonmars.com/api/v3/media/
b56f6007-d9cc-45e0-8a0f-857e62790846"
    },
    "layout": "top",
    "likeEnabled": true,
    "liked": false,
    "likesCount": 37,
    "shareEnabled": false,
    "shared": false,
    "sharesCount": 0,
    "title": "Lorem ipsum dolor sit amet",
    "viewed": true,
    "viewsCount": 135
  }
]
```

Each type of communication will provide different data. This was an example of an *Article*.

And now here's an example of a training course communication.


```
[
  {
    "id": "4404",
    "type": "training",
    "trainingId": "quiz-contest-1",
    "author": {
        "avatarUrl": "https://my-userdata.teachonmars.com/api/v3/
communication/avatar/longtext/7a49e231f69fb51e07a5b11843bcdf8a",
        "displayName": "Robert Alfred"
    },
    "bookmarked": false,
    "description": "Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Morbi viverra eleifend turpis eu tincidunt. Quisque
varius nisi elementum dolor posuere luctus. In quis vestibulum arcu,
eu tincidunt tellus. Sed at sem et neque suscipit volutpat.
Pellentesque in lectus id metus ultricies rutrum.",
    "featured": true,
    "featuredImage": {
      "size": {
        "height": 545,
        "width": 750
      },
      "url": "https://my-userdata.teachonmars.com/api/v3/media/
e3f1c026-819e-4c10-9b4b-5b40f90ab4bf"
    },
    "image": {
      "size": {
        "height": 545,
        "width": 750
      },
      "url": "https://my-userdata.teachonmars.com/api/v3/media/
b56f6007-d9cc-45e0-8a0f-857e62790846"
    },
    "layout": "top",
    "likeEnabled": true,
    "liked": false,
    "likesCount": 37,
    "shareEnabled": false,
    "shared": false,
    "sharesCount": 0,
    "title": "Quiz contest!",
    "viewed": true,
    "viewsCount": 135
  }
]
```

ToM.navigation.displayCommunication(communicationId)

Displays a given communication. What is displayed depends on the type of communication.

Type	Behavior
Article	Displays the article with its header image on a new screen
Training course	Displays the training course screen just like ToM.home.displayTraining()
Web Link	Opens the link in a web view. The web view can be closed and refreshed
Image	Opens the image full screen.
Short text	Nothing happens.

Profile

ToM.learner.getProfile()

Returns information about the learner and some basic learning data.


```
{
  "id": "5205dda0-db62-11e7-a799-25541a29a5fd",
  "firstname": "John",
  "lastname": "Doe",
  "title": "Project manager",
  "lang": "en",
  "avatar": {
    "url": "https://my-userdata.teachonmars.com/api/v3/device/
learner/5205dda0-db62-11e7-a799-25541a29a5fd/avatar/956872937255892"
    },
  "completedTrainingsCount": 3,
  "points": 45,
  "badgesCount": 3,
  "time": 349
}
```


Shake'n'Learn

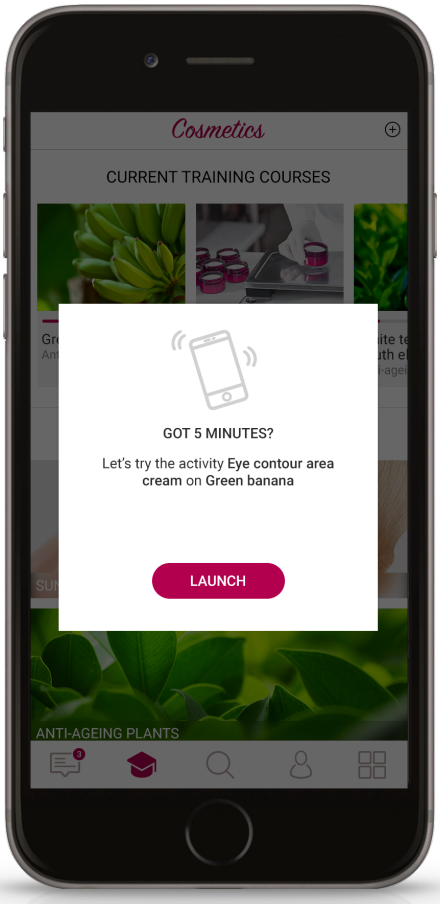
Shake'n'Learn is a feature of the Teach on Mars Enterprise solution that allow the user to quickly launch a scored activity that has already been attempted by shaking their mobile phone while on the home screen. It is a fun way to review a random lesson.

ToM.home.isShakeAndLearnPossible()

Return true or false whether an activity can be launched through the Shake'n'Learn feature.

ToM.home.shakeAndLearn()

Launches a random scored activity that have already been completed before. It is a way to go over some gamified content again. This replaces the shake gesture and launches the Shake'n'Learn action.



Other methods

ToM.env.getAll()

Returns all environment variables in a single object

Variable	Description
APPLICATION_SERVER_ID	Application server ID
APPLICATION_VERSION	Application version

Variable	Description
EMAIL	Email for current logged in user
FIRST_NAME	First name for current logged in user
LANGUAGE	Language setted in the application
LAST_NAME	Last name for current logged in user
LEARNER_ID	The unique identifier for current logged in user
LOGIN	Login of current logged in user
USER_OS	OS on witch the application is running
metadata.CODE_META	Display the value of a metadata (CODE_META is case sensitive

ToM.env.get(variable)

Returns the value of the given variable.

How and to update the data with callbacks

In the webapp, the custom home is reloaded every time it is displayed. So, anytime the user navigates to the home, the page reloads completely as if it was loaded from the start.
But in the mobile applications, the state of the custom home is preserved when the user navigates to other screens. So when the user gets back to the home, it will not reload on its own.

To update the home after some actions from the user, you can implement *callbacks.* They are methods that will be called by the app or the webapp on certain events that will be detailed below.

ToM.callback.beforeHomeDisplayed

The callback is called before the home is displayed.

Examples:
- the user is navigating back from the Wall (or any other screen in the app) to the home
- the user launches the app and arrives on the home

You can use this callback to execute updates in the home page just before it is displayed with this kind of code:

```
ToM.callback.beforeHomeDisplayed = function() {

    // Here you can do stuff before the home is displayed
    console.log("The home screen is displayed");
    myCustomMethodToDisplayStuffInTheHome();
};
```

ToM.callback.beforeAppBackground

This callback is called when the user is on the home and the app goes to background mode (during a phone call, when the user navigates to the phone home screen, ...)

```
ToM.callback.beforeAppBackground = function() {

    // Here you can do stuff before the app goes to background mode
    console.log("The app goes to background");
    myCustomMethodBeforeBackground();
};
```

ToM.callback.beforeAppForeground

This callback is called when the user is on the home and the app goes to back from background mode (after a phone call, when the user navigates back to the app after a background mode, ...)

```
ToM.callback.beforeAppForeground = function() {

    // Here you can do stuff before the app goes to background mode
    console.log("The app goes back from background");
    myCustomMethodBeforeForeground();
};
```

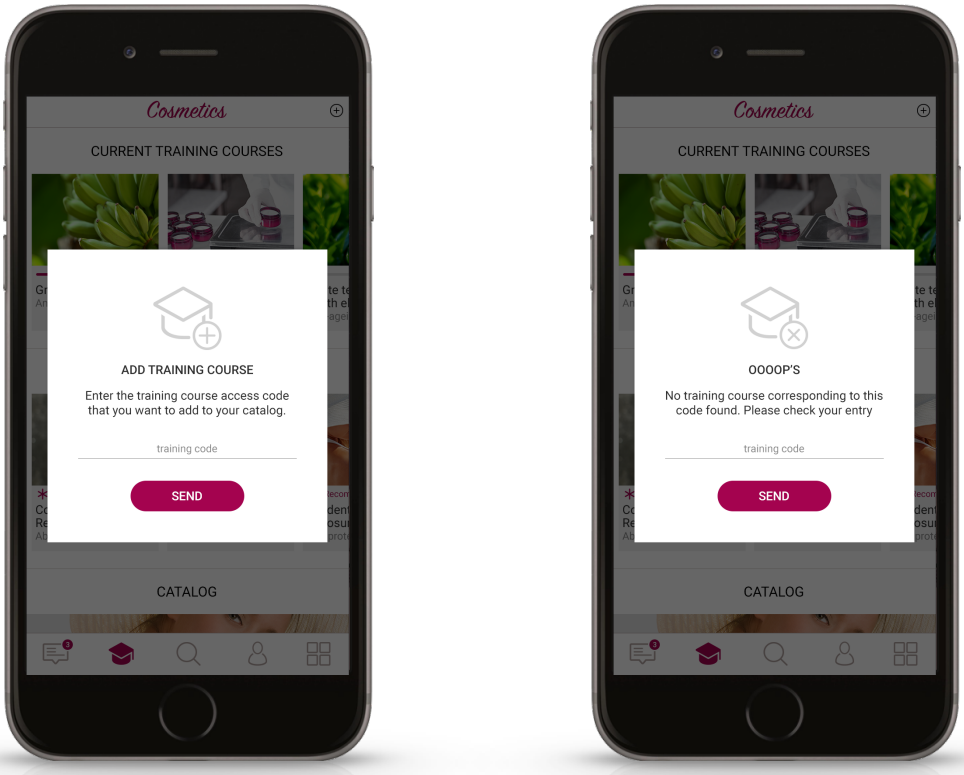
ToM.callback.onDataUpdated

This callback is called when the app has updated some data like trainings, featured communications or the user profile. This allows you to update parts of your home screen without reloading the whole page.
The callback is called with an argument that describes which type of data was just updated.

```
ToM.callback.onDataUpdated = function(what) {

    // Here you can do stuff when some data was updated
    switch(what) {
        case 'profile':
            updateProfileContainer();
            break;

        case 'catalog':
            updateCategoriesContainer();
            break;

        case 'featuredCommunications':
            updateCommunicationsContainer();
            break;
    }
};
```

ToM.appUtils.showTrainingCourseEnrollment()

Learners may enroll to a training course by typing a code linked to the training course. This method displays the dialog in which the learner is able to do just that.



If the code is valid and the enrollment successful, the learner will be redirected to the training course screen.

ToM.appUtils.logout()

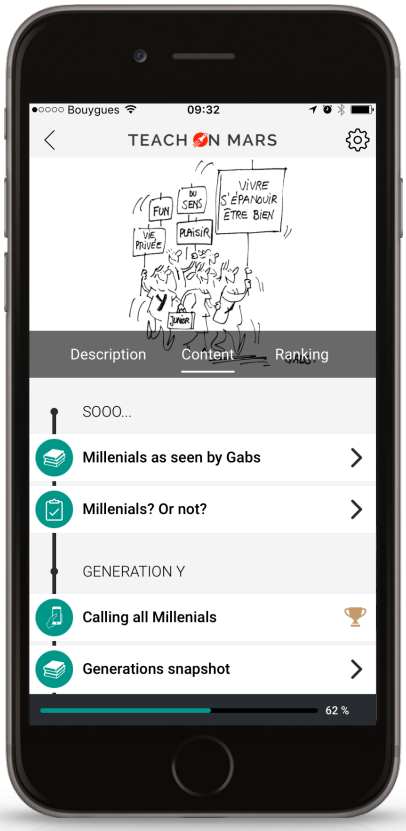
This method is pretty obvious, when it is called, the current learner is logged off the application and redirected to the login screen.

Migrating from 18.1

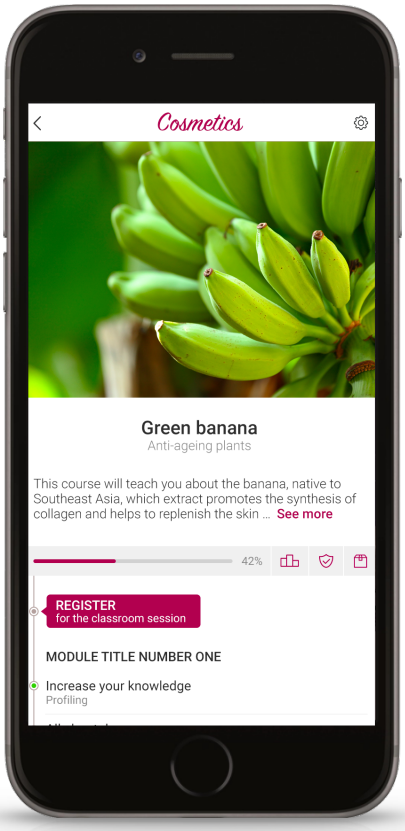
Here is a migration guide in case your application already featured a custom web home screen in version 18.1 and you are now upgrading to version 18.3.

Design

The design of the standard part of the application has changed. As mentioned in the first part of this documentation (Design guidelines), it is important to keep consistency between your custom home and the rest of the application. Therefore, you need to make sure that the custom home design you will use in version 18.3 is consistent with the 18.3 look.



Training course screen in 18.1



Training course screen in 18.3

Navigation

In version 18.1, the custom home was displayed beneath the top bar which contained the logo and the burger menu in the top left hand corner. It is no longer the case in 18.3, the custom home is displayed full screen. It implies that no standard navigation element is present when the custom home screen is displayed ; there is no longer a burger menu, and the tab bar from version 18.3 is not displayed.

You need to make sure that your custom home contains at least one link to any main screen of the application (as mentioned in the *Design guidelines > Navigation* part of this documentation).

Full screen display

As mentioned before, there is no longer a top bar and the custom home is now displayed full screen. This implies that the aspect ratio of your custom home will change when displayed on devices you used for testing.

You might want to make sure that this detail does not affect your design even if it should be responsive to fit any commonly used device ratio.