Set up

Install viewportsCore from npm:

```
npm install @meddream/viewports-core
```

To access viewports core functions you will need to use global `viewportsCore` variable.

To create simple viewer first of all you should create html file like this:

```
<!--index.html file-->
<style>
    .viewports-layout {
        display: flex;
        flex: 1;
        flex-wrap: wrap;
        background: #000;
    }
    .viewport-container {
        display: flex;
        flex: 1;
        flex-wrap: wrap;
        position: relative;
        border: 1px solid #323237;
    }
    .viewport-container.active {
        border: 1px solid #e51b48;
    }
</style>
...
<body>
    <div id="viewports-layout" class="viewports-layout">
        <div id="viewport-container-1-1" class="viewport-container"/>
    </div>
</body>
```

In this example layout container has optional .viewports-layout class, but it is required to have .viewport-container class for each viewport container inside the layout.

Then you need to initialize viewportsCore and add your created layout container:

```
viewportsCore.create();
viewportsCore.layoutController.addLayoutContainer(`viewport-container-1-1`, true);
```

Viewports core functions

Here you will find description about viewportsCore functions.

Viewports core

Create

To create viewportsCore use this function:

```
viewportsCore.create(properties);
```

Parameter description:

- `properties` (object) - parameters of viewports (not required). Possible options:
  - `enabledScrollBetweenSeries` (boolean) - this parameter allows scrolling between series. Default value: `false`.
  - `autoOpenImageType` (string) - parameter defines which way to use for the study auto open. Possible values: `none`, `first`, `always`. Default values: `none`.

Load study with HIS

Load study with HIS to memory.

```
viewportsCore.loadStudyWithHIS(studyUid);
```

Parameter description:

- `studyUid` (string) - study unique identification.

Load studies with HIS

Load multiple studies with HIS to memory.

```
viewportsCore.loadStudiesWithHIS(studiesToLoad);
```

Parameter description:

- `studiesToLoad` (array) - studies array with needed information.
    - `studyUid` (string) - study unique identification.
    - `callback` (function) - loaded study callback. Callback parameters are:
        - `study` (object) - loaded study data.

Example of studiesToLoad array:

```
const studiesToLoad = [
    {
        studyUid: '1.2.840.113619.2.55.3.4271045733.996.1449464144.595',
        callback: (study) => {console.log(study)}
    },
    {
        studyUid: '1.2.840.113619.2.55.3.4271045733.996.1449464144.595',
        callback: (study) => {console.log(study)}
    },
];
```

Reference lines

Reference line actions.

```
viewportsCore.enableReferenceLines();
viewportsCore.disableReferenceLines();
viewportsCore.isEnableReferenceLines();
```

Cache study

Cache study information to the viewportsCore.

```
viewportsCore.cacheStudy(studyUid, storageId);
```

Enable/Disable loaders

Enable or disable loader and progress bar visibility.

```
viewportsCore.enableViewportLoaders();
viewportsCore.disableViewportLoaders();
```

Layout controller

Add layout container

Add layout container to viewportsCore and register needed container events.

```
viewportsCore.layoutController.addLayoutContainer(containerId, selectContainer);
```

Parameters description:

- `containerId` (string) - new layout container id.
- `selectContainer` (boolean) - select new layout container.

Remove layout container

Remove layout container from viewportsCore.

```
viewportsCore.layoutController.removeLayoutContainer(containerId);
```

Parameter description:

- `containerId` (string) - layout container id which has to be removed.

Select layout container

Select layout container as active.

```
viewportsCore.layoutController.selectLayoutContainer(containerId);
```

Parameter description:

- `containerId` (string) - layout container id which has to be selected.

Deselect active layout container

Deselect layout container.

```
viewportsCore.layoutController.deselectActiveLayoutContainer();
```

Get layout containers

Get list of registered layout containers.

```
viewportsCore.layoutController.getLayoutContainers();
```

Is layout container available

Returns if layout container is available for viewportsCore usage.

```
viewportsCore.layoutController.isLayoutContainerAvailable(containerId);
```

Parameter description:

- `containerId` (string) - container id.

Get active container

Returns active container information.

```
viewportsCore.layoutController.getActiveContainer();
```

Get active container id

Returns active layout container id.

```
viewportsCore.layoutController.getActiveContainerId();
```

Get active viewport

Returns active viewport.

```
viewportsCore.layoutController.getActiveViewport();
```

Get active viewport id

Returns active viewport id.

```
viewportsCore.layoutController.getActiveViewportId();
```

Open instance to active viewport container

Opens instance to currently active viewport container.

```
viewportsCore.layoutController.openInstanceToActiveViewportContainer(instanceUid);
```

Parameter description:

- `instanceUid` (string) - instance uid.

Open instance to viewport container

Opens instance to provided layout container.

```
viewportsCore.layoutController.openInstanceToViewportContainer(containerId, instanceUid);
```

Parameters description:

- `containerId` (string) - container id.
- `instanceUid` (string) - instance uid.

Viewports controller

Delete viewport

Deletes viewport container by viewport id.

```
viewportsCore.controller.deleteViewport(viewportId);
```

Parameter description:

- `viewportId` (string) - viewport id.

Delete viewport by container id

Deletes viewport container by layout container id.

```
viewportsCore.controller.deleteViewportByContainerId(containerId);
```

Parameter description:

- `containerId` (string) - container id.

Get viewport

Returns viewport by viewport id.

```
viewportsCore.controller.getViewport(viewportId);
```

Parameter description:

- `viewportId` (string) - viewport id.

Get viewport by container id

Returns viewport by layout container id.

```
viewportsCore.controller.getViewportByContainerId(containerId);
```

Parameter description:

- `containerId` (string) - container id.

Reset all viewports renders

Resets all viewports to original state.

```
viewportsCore.controller.resetAllViewportsRenders();
```

Change viewport orientation

Changes viewport Mpr orientation.

```
viewportsCore.controller.changeViewportOrientation(containerId, orientation);
```

Parameters description:

- `containerId` (string) - container id.
- `orientation` (string) - orientations: `CORONAL`, `AXIAL`, `SAGITTAL`.

Generate Mpr

Generates 3 Mpr projections to provided layout containers.

```
viewportsCore.controller.generateMpr(containerId, layoutContainers);
```

Parameters description:

- `containerId` (string) - source container id.
- `layoutContainers` (array) - array of target containers for 3 Mpr projections.

Viewport actions

Fusion

Enables fusion with default `Hot Iron` palette.

```
viewportsCore.getActiveViewport().enableFusion(seriesId);
```

Changes color palette.

```
viewportsCore.getActiveViewport().applyFusionClut(name);
```

Parameter description:

- `name` (string) - name of color palette.

Possible color palettes names: `HOT_IRON`, `PET`, `HOT_METAL_BLUE`, `PET_20_STEP`, `RAINBOW`.

Fusion opacity

Fusion opacity from CT (0%) to PT (100%)

```
viewportsCore.getActiveViewport().applyFusionOpacity(value);
viewportsCore.getActiveViewport().getFusionOpacity();
```

Parameter description:

- `value` (number) - range of numbers from 0.0 to 1.0.

Window leveling

Window leveling actions.

```
viewportsCore.getActiveViewport().applyWL(w,l);
viewportsCore.getActiveViewport().applyDefaultWL();
viewportsCore.getActiveViewport().applyAutoWL();
viewportsCore.getActiveViewport().applyInvert(true|false);
viewportsCore.getActiveViewport().isInverted();
```

Transformations of viewport

Viewport transformation actions.

```
viewportsCore.getActiveViewport().applyRotation(degrees);
viewportsCore.getActiveViewport().applyRightRotation();
viewportsCore.getActiveViewport().applyLeftRotation();
viewportsCore.getActiveViewport().applyHorizontalFlip();
viewportsCore.getActiveViewport().applyVerticalFlip();
viewportsCore.getActiveViewport().applyTransformationCleanup();
```

Parameter description:

- `degrees` (number) - instance rotation by degrees.

Zoom operations

Zoom related operation actions.

```
viewportsCore.getActiveViewport().applyFitToScreen();
viewportsCore.getActiveViewport().applyOriginalResolution();
viewportsCore.getActiveViewport().applyZoom(scale, point);
```

Parameters description:

- `scale` (number) - scale number.
- `point` (object) - point object with x and y coordinates. For example: {x: 242, y: 236}.

Scale range restriction actions.

```
viewportsCore.getActiveViewport().setScaleRange(min, max);
viewportsCore.getActiveViewport().getScaleRange();
viewportsCore.getActiveViewport().clearScaleRange();
```

Parameters description:

- `min` (number) - minimal scale value.
- `max` (number) - maximum scale value.

To receive current viewport get fit to screen scale value:

```
viewportsCore.getActiveViewport().getFitToScreenScaleValue();
```

Pan operation

Pan operation action.

```
viewportsCore.getActiveViewport().applyPan(position);
```

Parameter description:

- `position` (object) - object with x and y position coordinates. For example: {x: 10, y: 15}.

Reset render

Reset active viewport render to original state.

```
viewportsCore.getActiveViewport().resetRender();
```

Actions with measurements

Measurements id: `none`, `ruler`, `angle`, `polyline`, `oval`, `volume`, `VTI`, `cobb angle`, `roi`; Measurements related functions:

```
viewportsCore.setActiveMeasurementId(`ruler`);
viewportsCore.getActiveMeasurementId();
viewportsCore.getActiveViewport().deleteMeasurements();

viewportsCore.enableAngleBetweenIntersectingRulers();
viewportsCore.disableAngleBetweenIntersectingRulers();
viewportsCore.isEnableAngleBetweenIntersectingRulers();

viewportsCore.enableIntensity();
viewportsCore.disableIntensity();
viewportsCore.isEnableIntensity();
```

Mouse actions

To bind action to mouse button, your need to provide button number and mouse action constant:

```
VIEWPORT_FUNCTIONS = {
    NONE: 0,
    WL: 1,
    ZOOM: 2,
    PAN: 3,
    SCROLL: 4,
    MEASURE: 5,
    ROTATE: 6,
    CROSSHAIR: 8,
};

viewportsCore.setMouseButtonFunction(1|2|3, VIEWPORT_FUNCTIONS.WL);
```

Synchronization

Enable synchronization

Enables synchronization.

```
viewportsCore.enableSync(yourCallback);
```

Synchronization example

```
viewportsCore.enableSync((functionName, parameters) => console.log(functionName, parameters));
```

Parameters description:

- `functionName` (string) - sync event function name. Available event names: `image-position-changed`, `zoom-changed`, `container-size-changed`.
- `parameters` (object) - object of provided event function parameters.

Example of parameters object:

```
- `image-position-changed`: {"imagePosition": {"x": 0.031, "y": 0.097}, "containerId": "viewport-container-1-1"};
- `zoom-changed`: {"scale": 1.2529411185263977, "point": { "x": 253.15", "y": 389}, "containerId": "viewport-container-1-1"};
- `container-size-changed`: {"width": 531, "height": 883, "scale": 1.09375, "containerId": "viewport-container-1-1"}.
```

Disable synchronization

Disables synchronization.

```
viewportsCore.disableSync(yourCallback);
```

Add to synchronization

This function can be used to add desired containers to the synced containers list.

```
viewportsCore.addToSync(containersId);
```

Parameter description:

- `containersId` (array) - Array of containers. Example: (['viewport-container-1-1', 'viewport-container-1-2']).

Remove from synchronization

This function can be used to remove desired containers from the synced containers list.

```
viewportsCore.removeFromSync(containersId);
```

Parameters description:

- `containersId` (array) - Array of containers. Example: (['viewport-container-1-1', 'viewport-container-1-2']).

Get locked viewport containers

Returns list of synced viewport containers.

```
viewportsCore.getLockedViewportContainers();
```

Apply zoom for locked views

When sync is enabled you can use zoom function on locked containers.

```
viewportsCore.applyZoomForLockedViews(containerId, scale, point);
```

Parameters description:

- `containerId` (string) - container id from which zoom operation was performed (to ignore provided container).
- `scale` (number) - scale number.
- `point` (object) - point object with x and y coordinates. For example: {x: 242, y: 236}.

Apply pan for locked views

When sync is enabled you can use pan function on locked containers:

```
viewportsCore.applyPanForLockedViews(containerId, positionX, positionY);
```

Parameters description:

- `containerId` (string) - container id from which pan operation was performed (to ignore provided container).
- `positionX` (number) - position x coordinates.
- `positionY` (number) - position y coordinates.

Landmark

Enable landmark

Enables landmark functionality and add callback event.

```
viewportsCore.landmarksController.enable(yourCallback);
```

Enable landmark example

```
viewportsCore.landmarksController.enable((landmarkData) => console.log(landmarkData));
```

Parameter description:

- `landmarkData` (object) - object contains information about added landmark. Landmark information:
  - `studyUid` (string) - study unique identification;
  - `frameOfReferenceUid` (string) - frame of reference unique identification;
  - `points` (array) - 3 landmark points array with x, y, z coordinates. Array example: [[-7.2, 1.8, -2.5], [-6.5, -1.5, -282.5], [5.5, 8.3, -29.5]].

Disable landmark

Disables landmark.

```
viewportsCore.landmarksController.disable(yourCallback);
```

Create landmark

Enables landmark tool and starts drawing landmark.

```
viewportsCore.landmarksController.createLandmark();
```

Add landmark

Creates landmark in viewports by stored landmark data.

```
viewportsCore.landmarksController.addLandmark(landmarkData);
```

Parameter description:

- `landmarkData` (object) - object with landmark data information:
  - `studyUid` (string) - study unique identification;
  - `frameOfReferenceUid` (string) - frame of reference unique identification;
  - `points` (array) - 3 landmark points array with x, y, z coordinates. Array example: [[-7.2, 1.8, -2.5], [-6.5, -1.5, -282.5], [5.5, 8.3, -29.5]].

Remove landmark

Removes created or added landmark.

```
viewportsCore.landmarksController.removeLandmark(referenceUid);
```

Parameter description:

- `referenceUid` (string) - frame of reference uid.

Get landmark data

Returns landmark data by frame of reference uid.

```
viewportsCore.landmarksController.getLandmarkData(referenceUid);
```

Parameter description:

- `referenceUid` - frame of reference uid.

Main data objects

Study data object

To get study data use this function:

```
viewportsCore.getStudy(studyUid);
```

Series data object

To get series data use this function:

```
viewportsCore.getSeries(seriesUid);
```

Instance object

To get instance data use this function:

```
viewportsCore.getInstance(instanceUid);
```

Events

Register event

To register event use:

```
viewportsCore.registerEvent(`event-constant`, callback);
```

Remove event

To remove remove:

```
viewportsCore.removeEvent(`event-constant`, callback);
```

Active container changed event

Triggered when active container has changed. Event constant: `active-container-changed` or `viewportsCore.getConstants().CORE_EVENTS.ACTIVE_CONTAINER_CHANGED`.

To use this event:

```
viewportsCore.registerEvent(`active-container-changed`, (containerId) => console.log(containerId));
```

Callback parameter:

- `containerId` (string) - active container id.

Active viewport changed event

Triggered when active viewport has changed. Event constant: `active-viewport-changed` or `viewportsCore.getConstants().CORE_EVENTS.ACTIVE_VIEWPORT_CHANGED`.

To use this event:

```
viewportsCore.registerEvent(`active-viewport-changed`, (viewportId) => console.log(viewportId));
```

Callback parameter:

- `viewportId` (string) - active viewport id.

Viewport instance changed event

Triggered when new instance has opened to viewport. Event constant: `viewport-instance-changed` or `viewportsCore.getConstants().CORE_EVENTS.VIEWPORT_INSTANCE_CHANGED`.

To use this event:

```
viewportsCore.registerEvent(`viewport-instance-changed`, (viewportId, instanceUid) => console.log(viewportId, instanceUid));
```

Callback parameters:

- `viewportId` (string) - viewport id.
- `instanceUid` (string) - instance uid.

Set fullscreen container event

Triggered when fullscreen container has been set. Event constant: `set-fullscreen-container` or `viewportsCore.getConstants().CORE_EVENTS.SET_FULLSCREEN_CONTAINER`.

To use this event:

```
viewportsCore.registerEvent(`set-fullscreen-container`, (containerId) => console.log(containerId));
```

Callback parameter:

- `containerId` (string) - active container id.

Image rotation changed event

Triggered when image rotation has changed. Event constant: `image-rotation-changed` or `viewportsCore.getConstants().CORE_EVENTS.IMAGE_ROTATION_CHANGED`.

To use this event:

```
viewportsCore.registerEvent(`image-rotation-changed`, ({degrees, containerId}) => console.log(degrees, containerId));
```

Callback parameters:

- `degrees` (number) - instance rotation degrees.
- `containerId` (string) - container id.

Image position changed event

Triggered when image position has changed. Event constant: `image-position-changed` or `viewportsCore.getConstants().CORE_EVENTS.IMAGE_POSITION_CHANGED`.

To use this event:

```
viewportsCore.registerEvent(`image-position-changed`, ({imagePosition, containerId}) => console.log(imagePosition, containerId));
```

Callback parameters:

- `imagePosition` (object) - instance position coordinates.
- `containerId` (string) - container id.

Zoom changed event

Triggered when synced images zoom has changed. Event constant: `zoom-changed` or `viewportsCore.getConstants().CORE_EVENTS.ZOOM_CHANGED`.

To use this event:

```
viewportsCore.registerEvent(`zoom-changed`, ({scale, point, containerId}) => console.log(scale, point, containerId));
```

Callback parameters:

- `scale` (number) - scale size.
- `point` (object) - the point coordinates from which to zoom in or out.
- `containerId` (string) - container id.

Viewport scale changed event

Triggered when viewport scale has changed. Event constant: `viewport-scale-changed` or `viewportsCore.getConstants().CORE_EVENTS.VIEWPORT_SCALE_CHANGED`.

To use this event:

```
viewportsCore.registerEvent(`viewport-scale-changed`, ({scale, containerId}) => console.log(scale, containerId));
```

Callback parameters:

- `scale` (number) - scale size.
- `containerId` (string) - container id.

Delete viewports event

Event gives callback when the viewport is deleted or when the viewport disappears on layout change. Event constant: `container-cleaned` or `viewportsCore.getConstants().CORE_EVENTS.CONTAINER_CLEANED`.

To use this event:

```
viewportsCore.registerEvent(`container-cleaned`, (container) => console.log(container));
```

Callback parameter:

- `container` (object) - object with container information:
  - `containerId` (string) - cleaned container id.
  - `viewportId` (string) - cleaned viewport id.

Viewport loader event

Triggered on viewport loader change. Event constant: `viewport-loader` or `viewportsCore.getConstants().CORE_EVENTS.VIEWPORT_PRELOADER`.

To use this event:

```
viewportsCore.registerEvent(`viewport-loader`, (object) => console.log(object));
```

Callback parameter:

- `object` (object) - object with loader information:
  - `containerId` (string) - container id.
  - `status` (bool) - status of loader.
  - `progress` (string) - loader progress status "Loading" or "Finished".

Viewport progress loader event

Triggered on viewport progress loader change. Event constant: `progress-loader` or `viewportsCore.getConstants().CORE_EVENTS.PROGRESS_LOADER`.

To use the event:

```
viewportsCore.registerEvent(`progress-loader`, (object) => console.log(object));
```

Callback parameter:

- `object` (object) - object with progress loader information:
  - `containerId` (string) - container id.
  - `status` (bool) - status of loader.
  - `progress` (number) - loader progress value.

Viewport resized event

Triggered when viewport container size changed and viewport had to be resized. Event constant: `viewport-resized` or `viewportsCore.getConstants().CORE_EVENTS.VIEWPORT_RESIZED`.

To use this event:

```
viewportsCore.registerEvent(`viewport-resized`, (containerId, viewportId) => console.log(containerId, viewportId));
```

Callback parameters:

- `containerId` (string) - container id.
- `viewportId` (string) - viewport id.