3. Introduction to Browser APIs

* Browser APIs

- also known as web APIs
- are Application Programming Interfaces (APIs) that are built into a web browser and provide native features.
- That can be used in web apps.
- can expose data from the browser and computer environment and allow developers to implement features with fewer lines of code.
- example, make network requests, manage storage retrieve device media streams.

Intersection Observer

- Is an API used to detect the intersection of a target element with it's ancestor element / the document viewport.
- example, we want to detect if some element is visible in the viewport.

Use cases:

1) Lazy loading images
2) Detect if an element is in the viewport or not
3) Auto-play a video if in the viewport, otherwise pause the video.
4) Infinite scrolling.

How to use:

- Can be used to observe an element.
- takes two inputs:

1) A callback function:

- receives a list of entries (elements) that are to be observed by an ancestor or document viewport.
- has the property isIntersecting - used to determine if the target entry is visible or not
  - True - Target is visible
  - False - Target is not visible.

2) An object with properties root, threshold and rootMargin.

- root - tell the element that is used as the viewport.
  - checking the visibility of the target element.
  - must be the ancestor of the target element.
  - if not specified then document viewport is the default value.

- threshold - can be a number or an array of numbers.
  - tell how much of the target element should be visible when the callback function gets triggered.
  - Default is 0, - as soon as the target element is visible the call back function will be triggered.

0.5 - triggers callback when 50% of target element visible

[0.25, 0.5] - triggers callback when 25% and 50% of target element is visible.

- rootMargin - same as CSS's margin property
  - can take either one value or multiple values for the individual margins.
  - can be used to grow / shrink the container viewport.
  - Default - 0.

This API returns an object which has a property observe which can be used to observe our desired target element.

View Transitions

- Provides a mechanism for easily creating animated transitions between different website views.
- incl. animating between DOM states in a single-page app (SPA).
  animating the navigation between documents in a multi-page app (MPA).

Concept and usage:

View transitions are popular choice for
- reducing user's cognitive load
- help them to stay in context.
- reducing perceived loading latency as they move between states or views of an application.

However, creating view transitions on the web has historically been difficult:

Transitions between states in single-page apps (SPAs) involve writing significant CSS and JS TO
1) Handle the loading and positioning of old and new content.

2) Animate the old and new states to create the transition.

3) Stop accidental user interactions with the old content from causing problems.

4) Remove the old content once the transition is complete.

- Cross - document view transitions (i.e. across ~~nog~~ ~~a~~ navigations between different pages in MPAs) have been impossible historically.

- View transitions API provides easy way of handling the required view changes and transition animations for both the above use cases.

Interfaces:

1) View Transition:
Represents a view transition and provides functionality to react to the transition reaching different states or skip the transition altogether.

2) Document. startViewTransition()
Start a new same-document (SPA) view transition and returns a viewTransition object to represent it.

3) PageRevel Event
- Event object for the pagereveal event.
- During a cross-document Navigation, it allows to manipulate the related view transition
- providing access to the relevant viewTransition object
- from the document being navigated to
- if view transition was triggered by navigation.

4) PageSwapEvent :

- event object for pageswap event.
- During cross-document navigation, it allows you to manipulate the related view transition.
- providing access to the relevant viewTransition object.
- from the document being navigated from.
- if view transition was triggered by the navigation
- also provides access to information on the navigation type and current and destination document history entries.

5) Window pagereveal event

- fired when a document is first rendered.
- either when loading a fresh document from the network or activating a document.

6) Window pageswap event.

- fired when a document is about to be unloaded due to a navigation.

HTML Additions :

1) <link rel = "expect" >

- Identifies the most critical content in the associated document for the user's initial view.
- Document rendering will be blocked untill the critical content has been parsed.
- ensures a consistent - first paint
- and so, view transition - across all supporting browsers.

CSS additions:

At-rules

@view-transition

In case of cross-document navigation, it is used to opt in the current & destination documents to undergo a view transition.

Properties

view-transition-name.

Provides the selected element with a separate identifying name and causes it to participate in a separate view transition from the root view transition - or no view transition if the none value is specified.

Pseudo-elements

1) ::view-transition

- The root of view transitions overlay
- Contains all view transitions
- sits over the top of all other page content.

2) view-transition-group()

The root of single view transition.

3) ::view-transition-image-pair()

- The container for a view-transitions old and new views.
- before & after the transition.

4) ::view-transition-old()

A static snapshot of the old view, before the transition.

5) ::view-transition-new()

A live representation of the new view, after the transition.

Smooth transitions with the view transition API

- has power to create seamless transitions between different views on website.
- creates more visually engaging user experience for users as they navigate the site.
- regardless of whether it's built as a multi-page application (MPA) or a single-page application (SPA).

* Situations where view transitions can be used:

1) A thumbnail image on a product listing page that transitions into a full-size product image on the product detail.

2) A fixed navigation bar that stays in place as you navigate from one page to another.

3) Grid with items moving positions as you filer through.

* Implement view transitions.

- not tied to specific application architecture or framework.
- can be triggered not only on a single document and also between two different documents

building blocks and principles:

1) The browser takes snapshot of the old and new states.
2) The DOM gets updated while rendering is suppressed.
3) The transitions are powered by CSS animations.

Same-document view transitions:

- runs on a single document.
- case in SPAs - single page applications.
- supported in chrome from chrome 111.
- triggered by calling $\boxed{\text{document.startViewTransitions}}$.

```
function handleClick(e){
    if(! document.startViewTransition){
        update the DOMSomehow();
        return;
    }
•   document.startViewTransition(
            () => update the DOM Somehow());
}
```

cross-document view transitions

- transition occurs between two different documents.
- typical for MPAs
- supported in Chrome 126 & greater.

How to trigger.

- triggered by a same-origin cross-document navigation.
- both pages opted in
- no API to call a start a cross-document view transitions.
- when a user clicks a link, the click triggers the view transition.

```
@view-transition{
        navigation: auto;
}.
```