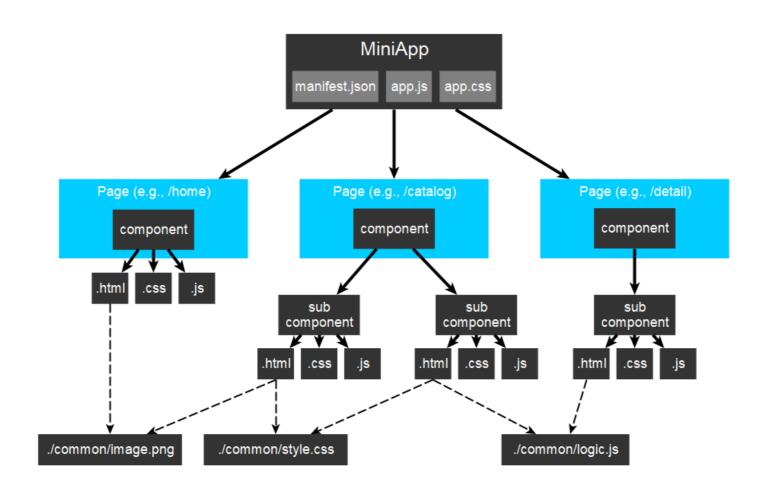
22 Jun 2022

W3C MiniApp CG Monthly Meeting

Background

## MiniApp pages and components

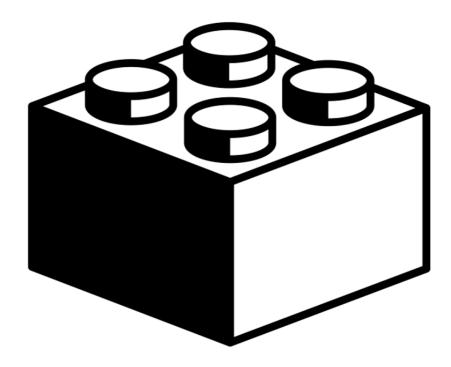


### MiniApp components

- Component is an extensible and reusable high-level building block to create MiniApps
  - > Based on **HTML-like elements** (e.g., <div>, <image>, <text>,...)
  - > Supporting events specific (e.g, click, swipe...)
  - > Supporting a **subset of CSS**
  - Components support data binding, like text interpolation.

#### Concept similar to Web Components

- > Custom HTML Elements
- > HTML modules
- > HTML Templates
- > Shadow DOM
- > CSS



lego by jon trillana from the Noun Project

Note: MiniApp uses Virtual DOM

## MiniApp elements: common attributes

Attribute	Туре	Standard equivalency	Compliant?
id	string	id attribute may be specified on any HTML element	YES
style	string	style attribute may be specified on any HTML element	YES
class	string	class attribute may be specified on any HTML element	YES

#### **Difference with standard approach:**

 None. All these basic attributes are similar (semantics and function) to the HTML element's attributes.

#### **Proposed solution based on standards:**

- Adoption of HTML standard element → MiniApp elements to implement the <u>Element interface</u>
- In this case, MiniApp elements would be compatible with the standard HTML elements

# Element interface adoption Implications



- Reusing the HTMLElement interface, means that any MiniApp element inherits the standard attributes and methods (e.g., addEventListener, parentNode, nextSibling...)
- These elements implement DOM manipulation methods.
- These elements are exposed to the <u>Window interface</u>

```
[Exposed=Window]
interface Element : Node {
 readonly attribute DOMString? namespaceURI;
 readonly attribute DOMString? prefix;
  readonly attribute DOMString localName;
  readonly attribute DOMString tagName;
  [CEReactions] attribute DOMString id;
  [CEReactions] attribute DOMString className;
  [SameObject, PutForwards=value] readonly attribute DOMTokenList classList;
  [CEReactions, Unscopable] attribute DOMString slot;
  boolean hasAttributes();
  [SameObject] readonly attribute NamedNodeMap attributes;
  sequence<DOMString> getAttributeNames();
  DOMString? getAttribute(DOMString qualifiedName);
  DOMString? getAttributeNS(DOMString? namespace, DOMString localName);
  [CEReactions] undefined setAttribute(DOMString qualifiedName, DOMString value);
  [CEReactions] undefined setAttributeNS(DOMString? namespace, DOMString qualifiedName, DOMString value);
  [CEReactions] undefined removeAttribute(DOMString qualifiedName);
  [CEReactions] undefined removeAttributeNS(DOMString? namespace, DOMString localName);
  [CEReactions] boolean toggleAttribute(DOMString qualifiedName, optional boolean force);
  boolean hasAttribute(DOMString qualifiedName);
  boolean hasAttributeNS(DOMString? namespace, DOMString localName);
  Attr? getAttributeNode(DOMString qualifiedName);
  Attr? getAttributeNodeNS(DOMString? namespace, DOMString localName);
  [CEReactions] Attr? setAttributeNode(Attr attr);
  [CEReactions] Attr? setAttributeNodeNS(Attr attr);
  [CEReactions] Attr removeAttributeNode(Attr attr);
  ShadowRoot attachShadow(ShadowRootInit init);
  readonly attribute ShadowRoot? shadowRoot;
  Element? closest(DOMString selectors);
  boolean matches(DOMString selectors);
  boolean webkitMatchesSelector(DOMString selectors); // legacy alias of .matches
 HTMLCollection getElementsByTagName(DOMString qualifiedName);
 HTMLCollection getElementsByTagNameNS(DOMString? namespace, DOMString localName);
 HTMLCollection getElementsByClassName(DOMString classNames);
  [CEReactions] Element? insertAdjacentElement(DOMString where, Element element); // legacy
 undefined insertAdjacentText(DOMString where, DOMString data); // legacy
};
```

## MiniApp elements: common events

Attribute	Interface	Similar DOM Standard Events	Compliant?
click	BasicEvent	Click (PointerEvent)	YES
longpress	BasicEvent	No equivalent standard (it can be implemented as a <a href="CustomEvent">CustomEvent</a> ) Can be implemented based on agnostic pointer events + using the Event. <a href="timestamp">timestamp</a> attribute. (Similar <a href="mailto:example">example</a> )	NO
swipe	BasicEvent	No equivalent standard (it can be created as a CustomEvent) – [Similar example]	NO
touchstart	TouchEvent	pointerdown (PointerEvent) on all HTML elements	NO
touchmove	TouchEvent	pointermove (PointerEvent) on all HTML elements	NO
touchcancel	TouchEvent	pointercancel (PointerEvent) on all HTML elements	NO
touchend	TouchEvent	pointerup (PointerEvent) on all HTML elements	NO

#### **Difference with standard approach:**

- Similar standard approach using W3C DOM standard (PointerEvents)
- Non-standard events (swipe, longpress) can be implemented with existing standards.

#### **Proposed solution based on standards:**

- If MiniApp elements are based on HTML standard elements → MiniApp elements support the standard events and can use the existing equivalent.
- Non-standard events could be proposed for standardization within W3C.

# MiniApp basic elements

Similar semantics

Component	Additional Attributes	Events	HTML	Similar Standard Approach / Comments
div			<div></div>	Constraint of standard attributes and different events
list		scrollend	<u><ul></ul></u>	Element event <u>scroll</u> as the standard. element.scrollHeight - Math.abs(element.scrollTop) === element.clientHeight
list-item			<u>⟨li⟩</u>	
swiper	index, loop, vertical	change	-	loop attribute, similar to Media <u>loop</u> attribute. Similar to <u>change</u> .event.
tabs	index, vertical, disabled		-	OpenUI's issue on tabs for HTML
tab-bar	mode		-	
tab-content	scrollable		-	Standard CSS overflow: scroll
refresh	offset, type, refreshing, lasttime, friction, disabled,		-	No "pull-to-refresh" standard but it could be implemented using CSS overscroll-behavior-y
image	src, alt, disabled	complete, error	<img/>	<img/> element could be used with some changes
progress	Type (circular, lineal), percent		<pre><pre><pre>ogress&gt;</pre></pre></pre>	Element with (max, value, position, labels) attributes
text			<label></label>	(generic element, without semantics, ?, <label>?). Similar to SVG's text.</label>
input	type, placeholder, headericon, disable, focusable	change	<input/>	input element (with all types supported), using attribute disabled, no headericon.
button	type, value, icon, waiting		<button></button>	button element, attribute disabled, no waiting, no icon. (in OpenUI) type attribute is for styles instead of functions.
label	target, disable		<label></label>	Labelable elements: button, input, meter, output, progress, select, textarea for instead of target in the standard element.
select	disable	change	<select></select>	select element. Attribute disabled (in OpenUI)
slider	min, max, value, disable	change	<input/>	input@type="range" (in OpenUI)
switch	checked, showtext, texton, textoff, disable	Change	-	No equivalent. Toggle switch as a checkbox? (in OpenUI)
picker	type (text, date, time, datetime, multi-text), disable		-	input element (date, time, datetime-local) and datalist element (with attribute options).
video	muted, src, autoplay, poster, controls	prepared, seeked,	<video></video>	video element includes all the MiniApp attributes, some differences in events
canvas			<canvas></canvas>	Equivalent

# MiniApp basic elements: direct equivalences

Component	Equivalent HTML Element (w/ minor changes)	OpenUI Component Research	Comments
div	<div></div>		
image	<img/>		
video	<video></video>		
canvas	<canvas></canvas>		
progress	<pre><pre><pre>cprogress&gt;</pre></pre></pre>		
input	<input/>		
picker	<pre><input type="date time"/></pre>		
slider	<pre><input type="range"/></pre>		
label	<label></label>		
button	<button></button>	<pre><button> (research)</button></pre>	
select	<select></select>	<pre><select> (proposal)</select></pre>	
switch		<pre><switch> (research)</switch></pre>	
text		<u>Text</u> (research)	Like <a href="Like"><a href="Like">&lt;</a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a>
tabs		<u>Tabs</u> (research)	
list			
list-item			
swiper (carousel)			
refresh			

Challenges and proposal based on standards

## Challenges and proposals based on standards

Challenges	Potential Solutions
<b>New elements</b> , not included in the HTML specification (but somehow present in some web frameworks, like <swiper>, <tabs>, <switch>,)</switch></tabs></swiper>	<ol> <li>These elements could be defined using stand-alone Web Components, and Custom Elements. Following the OpenUI CG process.</li> </ol>
Similar elements (elements that are semantically similar to HTML standard elements like <image/> <pre>canvas&gt; <slider> <button>)</button></slider></pre>	<ol> <li>Extension of the existing HTML elements (using inheritance). We could define the new attributes, events or redefine the existing ones defining new interfaces in the specification.</li> <li>Reuse and refine the existing HTML elements using the OpenUI CG process. Ideal to leverage the existing Web capabilities.</li> </ol>
Non-standard attributes in MiniApp components (e.g., disable, focusable)	<ol> <li>Adapt and use directly the standard attributes. If new attributes are needed, to propose changes in the standards.</li> <li>Creation of a basic, essential MiniAppElement that extends the HTMLElement with the specific requirements (new attributes, redefinition of existing attributes). Problem: this prevent MiniApps to reuse Web Components.</li> </ol>
Similar events in MiniApp components that are similar to existing standard ones (e.g., touchstart, touchend, etc.)	1. Adoption of the standard version as it is (PointerEvents).
New events for MiniApp components (e.g., longpress, swipe)	<ol> <li>Propose new events to the existing standards if really needed.</li> <li>Definition of new CustomEvents based on the standard DOM interfaces.</li> </ol>

Creation of profiles for **STANDARD HTML**, **CSS** and **DOM** — MiniApps supported by browsers

# How to implement a MiniApp component

**Extension of HTMLEIement** 

Mapping of attributes

Definition of new events

```
class MiniappElement extends HTMLElement {
 constructor() {
    super();
    // Attributes Mapping (e.g., "disable" === "disabled")
    if (this.hasAttribute('disable') && this.getAttribute('disable')==='true') {
      this.setAttribute('disabled', true)
    if (this.hasAttribute('focusable') && this.getAttribute('focusable')==='false') {
      this.setAttribute('inert', true)
    this.addEventListener('pointerdown', (e) => {
     this dataset.timestamp= e.timeStamp
    this.addEventListener('pointerup', (e) => {
      if ((e.timeStamp - this.dataset.timestamp) >= 1000) {
        console.log('Triggering longpress event (over 1s)')
       e.target.dispatchEvent(new CustomEvent('longpress'))
      delete this.dataset.timestamp
    this.addEventListener('longpress', function(e) {
     console.log('long pressed')
    })
  connectedCallback() {
    console.log('MiniappElement connected')
  disconnectedCallback() {
    console.log('MiniappElement disconnected')
```

#### Definition of new elements

Definition of <miniapp-text> as a new element

Extension of MiniAppElement

```
class MiniappText extends MiniappElement
 constructor() {
   super();
 connectedCallback() {
   const template = document.querySelector('template#text-component');
   const cloned = document.importNode(template.content, true);
    // Create a shadow root
   this.attachShadow({ mode: 'open' });
    this.shadowRoot.appendChild(cloned);
   console.log('MiniApp-Text connected');
 disconnectedCallback() {
   console.log('MiniApp-Text disconnected');
customElements.define('miniapp-text', MiniappText);
```

```
Definition of styles
  Template of the <miniapp-text>
<template id='text-component'</pre>
  <style>
      background-color: orange;
  </style>
  <span><slot/></span>
</template>
<miniapp-text>Hello MiniApp</miniapp-text>
```

Use of the <miniapp-text> (developers only need to use this)

Output

Hello MiniApp

### Extension of the existing elements

Definition of the specific behavior for the MiniApp component, with specific attributes, events, etc. Extending the standard element

Using the standard version of the component but indicating the profile (this select is a 'miniapp-select')

Select one option Select one option
Option 1
Option 2
Option 3

https://jsbin.com/getaguy/edit?html,output

```
class MiniappSelect extends HTMLSelectElement {
   constructor() {
     super();
     if (this.hasAttribute('disable') && this.getAttribute('disable')==='true') {
       this.setAttribute('disabled', true)
     if (this.hasAttribute('focusable') && this.getAttribute('focusable')==='false') {
       this.setAttribute('inert', true)
   connectedCallback() {
     const firstOption = this.firstChild;
     const optionDefault = document.createElement('option');
     optionDefault.appendChild(document.createTextNode('Select one option'));
     this.insertBefore(optionDefault, firstOption);
     console.log('MiniApp-Select connected');
     this.selectedIndex = 0;
   disconnectedCallback() {
     console.log('MiniApp-Select disconnected');
```

```
customElements.define('miniapp-select', MiniappSelect, { extends: 'select' });
```

How to write the UI Components specification

### How to write the UI Components specification Case 1: non-standard DOM/HTML

If the MiniApp model doesn't follow the standard DOM/HTML (incompatible with existing Web standards)

- We need to define a new model, or at least a new markup language (for instance, based on XML).
- We should indicate how to interact/manipulate the DOM for our solution (example in the SVG spec.).
- We can use some examples as references: SMIL, TTML, SVG.
- This solution would require at least the definition of XML schemas for the markup language.
- We need to indicate how to bind CSS stylesheets and events (in the SVG spec, we have examples)



#### 6.1 SVG's styling properties

SVG uses styling properties to describe many of its document parameter

- Parameters which are clearly visual in nature and thus lend themse
- . Parameters having to do with text styling such as font family and si
- · Parameters which impact the way that graphical elements are rendered

· animation is discussed in Animation SVG shares many of XSL is the definition 16.2 Complete list of supported events

The following proper

- Font properties
  - 'font-fami
  - 'font-size
  - 'font-size-'font-stret
  - 'font-style

  - o 'font-varia 'font-weight
- Text properties
- 'direction
- 'letter-spa
- 'text-deco

The following aspects of SVG are affected by events: . Using SVG Document Object Model (DOM), a script can register DOM 2 ever

- SVG includes event attributes on selected elements which define script that
- SVG's <u>animation elements</u> can be defined to begin or end based on events.

The following table lists all of the events which are recognized and supported in SV The DOM2 name in the second column is the name to use when defining DOM 2 attributes that can be attached to elements in the SVG language.

Requirements in the table on whether an event of a given type bubbles or is cancellable. method on the DocumentEvent interface can be made to bubble or be cancelable

#### Event name and description

Occurs when an element receives focus, such as when a 'text' becomes selected

Occurs when an element loses focus, such as when a 'text' becomes unselected

activate

# How to write the UI Components specification Case 2: standard DOM/HTML

If the MiniApp model follows the standard DOM/HTML (compatible with existing Web standards)

- > We could define HTML extensions (e.g., <u>EPUB's attributes</u>) if needed.
- > We should present what technologies (HTML, CSS) are supported.
- > We would use HTML and CSS subsets, so we need to indicate these subsets (e.g., EPUB's Deviations & Constraints or CSS).
- > We should define the scripting mechanisms supported in MiniApps (e.g., EPUB's Scripting)
- > Mechanisms to include localized strings.

First step: to organize a meeting with OpenUI CG (I am part of the CG already)

The UI Components specification could be a Group Note to define MiniApp contents, including:

- > Structure of a MiniApp page (how to bind components)
- > Overview of the basic elements (or components) → no technical definition, just high level.
- > What styles are supported (e.g., the CSS subset, how to bind external stylesheets)
- > Scripts supported (e.g., version and limitations for security)
- > Mechanisms to extend the components.

This specification defines a method for semantic inflection using the attribute axis: instead of adding new elements, the epublitype attribute can be appended to existing elements to inflect the desired semantics. A mechanism to identify external vocabularies that provide controlled values for the attributes is also defined. § 2.4.1.2 The epub: type Attribute Attribute Name Global attribute. MAY be specified on all elements A white space-separated list of property [Packages32] values, with restrictions as defined in Vocabulary Association. White space is the set of characters as defined in [XML] The eoub: type attribute inflects semantics on the element on which it appears. Its value is one or more white space-separated terms stemming from external vocabularies associated with the document instance, as define The inflected semantic MUST express a subclass of the semantic of the carrying element. In the case § 3.3 Restrictions on SVG This specification restricts the content model of SVG Content Documents and SVG embedded in XHTML Content Documents as follows: The [SVG] foreignObject element MUST adhere to the following criteria: > It MUST contain either [HTML] flow content or exactly one [HTML] body element in [HTML]. Its content MUST be a valid document fragment that conforms to the XHTML Content Document

EPUB Content Documents MAY contain scripting using the facilities defined for this in the respective underlying specifications ([HTML] and [SVG]). When an EPUB Content Document contains scripting, it is referred to in this

specification as a Scripted Content Document. This label also applies to XHTML Content Documents when they

An instance of the [HTML] script or [SVG] script element included in a Top-level Content Document.

in a parent XHTML Content Document using the [HTML] iframe element

parent XHTML Content Document using the [HTML] iframe element.

An instance of the [HTML] scenar element included in an XHTML Content Document that is embedded

. An instance of the [SVG] script element included in an SVG Content Document that is embedded in a

In both of the above-defined contexts, whether the JavaScript code is embedded directly in the script element

model defined in XHTML Content Documents - Content Conformance

This specification defines two contexts in which scripts MAY appear

§ 5. Scripting

§ 5.1 Scripting Contexts

container-constrained Either of the following:

# How to write the UI Components specification Case 3: Vue.js-like framework

If the MiniApp model follows the existing MVVM (non-compatible with current standards)

- > We could define an informative (non-standard) specification to document the architecture;
- > We should include what technologies (HTML, CSS) are supported and the extensions.
- > We should document the data and event binding.
- > We should define the scripting mechanisms supported in MiniApps.

First step: publicly confirm that MiniApps wouldn't follow the HTML/DOM standards. No WebAPIs available

To avoid friction with other W3C groups, the UI Components specification should be an informative Group Note or similar to define MiniApp contents, including something like the documentation of Vue.js:

- > Structure of a MiniApp page (how to bind components);
- > Explain the differences with HTML.
- What styles are supported (e.g., the CSS subset, how to bind external stylesheets);
- > Scripts supported (e.g., version and limitations for security);
- > Data interpolation, internationalization, etc.



# Thank you.

More info:

https://github.com/w3c/miniapp-components