Översikt över widget-API

Bakgrund

Widget-API:et utformades med följande mål i åtanke: dölja komplexiteten i agentapplikationen från widgetarna, presentera ett litet och konsekvent gränssnitt till widgetarna och upprätthålla ett stabilt gränssnitt för widgetarna

Översikt

Widget-API:et är uppdelat i två stora delsystem: händelsesystemet och gränssnitten. Händelsesystemet används för att sända olika meddelanden från tjänsterna till widgetarna och för widget-till-widget-kommunikation. Gränssnitten utgör fasaden för tjänsterna och kärnfunktionen.

Widgetmeddelanden

Som tidigare nämnts finns de externa widgetarna vanligtvis på en annan domän och kan inte komma åt Widget API:et direkt. För att kringgå domänbarriären öppnar agentapplikationens kärna en meddelandekanal med webbläsarens meddelande-API.
Koden nedan visar ett sätt att prenumerera på och hantera meddelanden från API:

```
const origin = 'the origin of the agent application'; let port; window.addEventListener('message', message => { // Make sure that the channel comes from the correct source: if (message.origin !== origin) return; // Setup the communication channel: if (!port) { port = message.ports[0]; port.onmessage = receiver; } }); function receiver(message) { //code here }
```

Meddelandets nyttolast finns i *message.data* . Kärnan lägger till en *egenskap message.data.type* i alla meddelanden.

Förfrågningar till Widget API kan bara skickas genom den angivna porten:

```
port.postMessage(message);
```

Gränssnitt

Om du vill hämta en egenskap eller anropa en metod i Widget API:et ska meddelandeformatet {call, args} användas, där anrop är sökvägen till metoden (eller egenskapen) i API:et.
I händelse av ett metodanrop är args en matris med alla nödvändiga argument för metodanropet. Exempel:

```
port.postMessage({ call: 'tab.setTitle', args: ['new title'] });
```

Om metoden returnerar ett resultat skickas den till den externa widgeten av port.onmessage-hanteraren i formatet {name, value, type}, där name är namnet på den begärda egenskapen eller metoden, value är egenskapens värde eller resultatet av anropet, och type kommer att vara strängens ”resultat”.
Exempelsvar på ett getOption-samtal:

```
{ name: 'widget.getOption', value: 'https://demo.puzzel.com/dev/widgets/external/demo/', type: 'result' }
```

Observera att på grund av hur Meddelande-API:et fungerar finns nyttolasten för meddelandet i message.data-egenskapen.

Om anropsmetoden inte returnerar ett resultat skickar kärnan inget meddelande. Om anropsmetoden returnerar ett löfte skickas meddelandet när löftet är löst eller avvisat. Om löftet löses skickar kärnan ett standardresultatmeddelande, där value innehåller löftets värde. Om löftet avvisas kommer ett felmeddelande att skickas:

```
{ name: 'widget.setOption', value: 'Unexpected end of JSON input', type: 'error' }
```

Om det krävs en matchning av ett anrop till ett resultat kan valfritt id läggas till i förfrågningen. Det kommer att returneras tillbaka:

```
{ call: 'tab.getOption', args: ['option name'], id: '0123456789' }
```

Resultat:

{ name: 'tab.getOption', value: 'option value', id: '0123456789' }

Widgeten kan också observera en egenskap för ändringar genom att skicka ett {watch}-meddelande. Det (watch-fältet) ska innehålla sökvägen till egenskapen i widget-API:et.

Om värdet på egenskapen ändras skickar kärnan meddelandet {name, old, new, type}, där name kommer att vara samma egenskapssökväg, old kommer att bli värdet för den egenskapen före ändringen, new efter ändringen, och type den ”ändrade” strängen.

Events

De externa widgetarna kan prenumerera på Events genom att skicka ett {subcribe, options: {once, address}} meddelande till kärnan. Prenumerationsfältet ska innehålla händelsens namn. Hela alternativfältet är valfritt, liksom dess egenskaper: boolesk en gång och adressträngen. Adressen har samma betydelse som i ExtendedEventAggregators metoder. Inställningenen gång innebär att subscribeOnce-metoden kommer att användas, dvs. den externa widgeten får bara en enda händelse innan prenumerationen avslutar sig själv.
Händelserna tas emot med meddelandet {name, value, type} där name kommer att vara namnet på händelsen, value är nyttolasten och type kommer att vara ”händelse”.

{ name: 'userStatusChanged', value: 'System', type: 'event' }

Den fullständiga API-referensen kommer att göras tillgänglig här

