W3C Vehicle API

Kevron Rees - ALS 2014

Agenda

1. History of Automotive BG
2. Organization of specifications
3. availability
4. get/set
5. subscribe/unsubscribe
6. history
7. path to standardization

History

http://www.w3.org/community/autowebplatform/

- Business group began in 2013.
- Initial charter to write a vehicle data spec

Business Groups help provide input into what may become standard.

Working Groups create standards

Vehicle API

- Vehicle API

https://rawgit.com/w3c/automotive-bg/master/snapshots/vehicle_spec_snapshot_latest.html

- Vehicle Data

https://rawgit.com/w3c/automotive-bg/master/snapshots/data_spec_snapshot_latest.html

Vehicle and VehicleInterface

Vehicle is top-level object:
navigator.vehicle

VehicleInterface is interface to specific DataType (see data types in data spec)

Vehicle and VehicleInterface

All attributes on vehicle are VehicleInterface objects allowing access to specific data types:

vehicle.vehicleSpeed

Implementation of specific types is optional

Vehicle Interface

```
[NoInterfaceObject]
interface VehicleInterface {
    Promise        get (optional Zone zone);
    Promise        set (object value, optional Zone zone);
    unsigned short subscribe (VehicleInterfaceCallback callback, optional
Zone zone);
    void          unsubscribe (unsigned short handle);
    readonly    attribute Zone[] zones;
};
```

Availability

- allows developers to discover whether a data type is supported and if not, why not
- Allows developers to be notified when support changes

Availability

```
partial interface VehicleInterface {
    Availability available ();
    readonly    attribute boolean supported;
    short       availabilityChangedListener
(AvailableCallback callback);
    void        removeAvailabilityChangedListener (short
handle);
};
```

get()

```
Promise        get (optional Zone zone);
```

get() example

```
var vehicle = navigator.vehicle;

vehicle.vehicleSpeed.get().then(function(data) {
  console.log("vehicle speed: " + data.speed);
},
function(error) {
  console.log("There was an error");
});
```

set()

```
Promise        set (object value, optional Zone zone);
```

set() example

```
var zone = Zone
vehicle.door.set({"lock" : true}, zone.driver).then(resolve, reject);
function resolve()
{
  /// success
}
function reject(errorData)
{
  console.log("Error occurred during set: " + errorData.message + " code: " +
errorData.error);
}
```

subscribe()/unsubscribe()

Allows developers to be notified when a specific data item changes.

```
unsigned short subscribe (VehicleInterfaceCallback callback, optional Zone zone);

    void            unsubscribe (unsigned short handle);
```

subscribe()/unsubscribe() example

```
var vehicleSpeedSub = vehicle.vehicleSpeed.subscribe(function
(vehicleSpeed) {
  console.log("vehicle speed changed to: " + vehicleSpeed.speed);
  vehicle.vehicleSpeed.unsubscribe(vehicleSpeedSub);
});
```

History API

optional API that allows developers to access logged data

```
partial interface VehicleInterface {
    Promise getHistory (Date begin, Date end, optional
Zone zone);
    readonly    attribute boolean isLogged;
    readonly    attribute Date ?  from;
    readonly    attribute Date ?  to;
};
```

History API example

```
if(vehicle.vehicleSpeed.isLogged)
{
  /// get all vehicleSpeed since it was first logged:
  vehicle.vehicleSpeed.getHistory(vehicle.vehicleSpeed.from, vehicle.
vehicleSpeed.to).then( function ( data ) {
    console.log(data.length);
  });
}
```

Path to standardization

- Need to form automotive working group
- Implementations (Tizen+crosswalk has partial implementation)
- Participate! Subscribe to mailing list and get involved in the API discussion!

Question?

Kevron Rees - kevron.m.rees@intel.com

mailing list: http://lists.w3.org/Archives/Public/public-autowebplatform/