GRT INSTITUTE OF ENGINEERING AND TECHNOLOGY, TIRUTTANI - 631209

Approved by AICTE, New Delhi Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROJECT TITLE

Traffic Management for Internet of Things (IoT)

COLLEGE CODE: 1103

NAME: Janani S

BATCH: 3rd YR, 5th SEM

REG NO.: 110321104048

EMail ID: Sivakumarjanani6@gmail.com

CODING:

getCurrentPosition() method:

The getCurrentPosition(successCallback, errorCallback, options) method steps are:

If the current settings object's relevant global object's associated Document is not fully active:

1) Call back with error errorCallback and POSITION_UNAVAILABLE.
2) Terminate this algorithm.
3) In parallel, request a position passing successCallback, errorCallback, and options.

// A one-shot position request:

```
navigator.geolocation.getCurrentPosition(position => {
  const { latitude, longitude } = position.coords;
  // Show a map centered at latitude / longitude.
});
```

watchPosition() method:

The watchPosition(successCallback, errorCallback, options) method steps are:

If the current settings object's relevant global object's associated Document is not fully active:

1. Call back with error
   passing errorCallback and POSITION_UNAVAILABLE.
2. Return 0.
3. Let watchId be an implementation-defined unsigned long that is greater than zero.
4. Append watchId to this's [[watchIDs]].
5. In parallel, request a
   position passing successCallback, errorCallback, options, and watchId.
6. Return watchId.

Watching a position for repeated updates:

```
const watchId = navigator.geolocation.watchPosition(position => {
  const { latitude, longitude } = position.coords;
  // Show a map centered at latitude / longitude.
});
```

clearWatch() method:

When clearWatch**()** is invoked, the user agent MUST:

1) Remove watchId from this's [[watchIDs]].

Using clearWatch():

```
const watchId = navigator.geolocation.watchPosition(
  position => console.log(position)
```

```
);
function buttonClickHandler() {
  // Cancel the updates when the user clicks a button.
  navigator.geolocation.clearWatch(watchId);
}
```

A HTML button that when pressed stops watching the position.

```
<button onclick="buttonClickHandler()">
  Stop watching location
</button>
```

Handling errors:

```
// Request repeated updates.const watchId =
navigator.geolocation.watchPosition(
  scrollMap, handleError
);
function scrollMap(position) {
  const { latitude, longitude } = position.coords;
  // Scroll map to latitude / longitude.
}
function handleError(error) {
  // Display error based on the error code.
  const { code } = error;
  switch (code) {
    case GeolocationPositionError.TIMEOUT:
```

```
    // Handle timeout.
    break;
  case GeolocationPositionError.PERMISSION_DENIED:
    // User denied the request.
    break;
  case GeolocationPositionError.POSITION_UNAVAILABLE:
    // Position not available.
    break;
 }
}
```

Getting cached position:

```
navigator.geolocation.getCurrentPosition(
 successCallback,
 console.error,
 { maximumAge: 600_000 }
);
function successCallback(position) {
 // By using the 'maximumAge' member above, the position
 // object is guaranteed to be at most 10 minutes old.
}
```

Timing out a position request:

```
// Request a position. We are only willing to wait 10// seconds for it.
```

```
navigator.geolocation.getCurrentPosition(
  successCallback,
  errorCallback,
  { timeout: 10_000 }
);
function successCallback(position) {
  // Request finished in under 10 seconds...
}
function errorCallback(error) {
  switch (error.code) {
    case GeolocationPositionError.TIMEOUT:
      // We didn't get it in a timely fashion.
      doFallback();
      // Acquire a new position object,
      // as long as it takes.
      navigator.geolocation.getCurrentPosition(
        successCallback, errorCallback
      );
      break;
    case "...": // treat the other error cases.
  }
}
function doFallback() {}
```

Enabling the Geolocation API in an iframe:

```
<iframe
  src="https://third-party.com"
  allow="geolocation">
</iframe>
```

Permissions Policy over HTTP:

```
Permissions-Policy: geolocation=()
```

PositionOptions dictionary:

```
PositionOptions {
  boolean enableHighAccuracy = false;
  [Clamp] unsigned long timeout = 0xFFFFFFFF;
  [Clamp] unsigned long maximumAge = 0;
};
```