screen

> *Retrieve information about screen size, displays, cursor position, etc.*

Process: Main

This module cannot be used until the `ready` event of the `app` module is emitted.

`screen` is an EventEmitter.

Note: In the renderer / DevTools, `window.screen` is a reserved DOM property, so writing `let { screen } = require('electron')` will not work.

An example of creating a window that fills the whole screen:

```
const { app, BrowserWindow, screen } = require('electron')

let win
app.whenReady().then(() => {
  const { width, height } = screen.getPrimaryDisplay().workAreaSize
  win = new BrowserWindow({ width, height })
  win.loadURL('https://github.com')
})
```

Another example of creating a window in the external display:

```
const { app, BrowserWindow, screen } = require('electron')

let win

app.whenReady().then(() => {
  const displays = screen.getAllDisplays()
  const externalDisplay = displays.find((display) => {
    return display.bounds.x !== 0 || display.bounds.y !== 0
  })

  if (externalDisplay) {
    win = new BrowserWindow({
      x: externalDisplay.bounds.x + 50,
      y: externalDisplay.bounds.y + 50
    })
    win.loadURL('https://github.com')
  }
})
```

Events

The `screen` module emits the following events:

Event: 'display-added'

Returns:

- `event` Event
- `newDisplay` Display

Emitted when `newDisplay` has been added.

Event: 'display-removed'

Returns:

- `event` Event
- `oldDisplay` Display

Emitted when `oldDisplay` has been removed.

Event: 'display-metrics-changed'

Returns:

- `event` Event
- `display` Display
- `changedMetrics` string[]

Emitted when one or more metrics change in a `display`. The `changedMetrics` is an array of strings that describe the changes. Possible changes are `bounds`, `workArea`, `scaleFactor` and `rotation`.

Methods

The `screen` module has the following methods:

`screen.getCursorScreenPoint()`

Returns `Point`

The current absolute position of the mouse pointer.

Note: The return value is a DIP point, not a screen physical point.

`screen.getPrimaryDisplay()`

Returns `Display` - The primary display.

`screen.getAllDisplays()`

Returns `Display[]` - An array of displays that are currently available.

`screen.getDisplayNearestPoint(point)`

- `point` Point

Returns `Display` - The display nearest the specified point.

`screen.getDisplayMatching(rect)`

- `rect` Rectangle

Returns `Display` - The display that most closely intersects the provided bounds.

`screen.screenToDipPoint(point)` *Windows*

- `point` Point

Returns `Point`

Converts a screen physical point to a screen DIP point. The DPI scale is performed relative to the display containing the physical point.

`screen.dipToScreenPoint(point)` *Windows*

- `point` Point

Returns `Point`

Converts a screen DIP point to a screen physical point. The DPI scale is performed relative to the display containing the DIP point.

`screen.screenToDipRect(window, rect)` *Windows*

- `window` BrowserWindow | null
- `rect` Rectangle

Returns `Rectangle`

Converts a screen physical rect to a screen DIP rect. The DPI scale is performed relative to the display nearest to `window`. If `window` is null, scaling will be performed to the display nearest to `rect`.

`screen.dipToScreenRect(window, rect)` *Windows*

- `window` BrowserWindow | null
- `rect` Rectangle

Returns `Rectangle`

Converts a screen DIP rect to a screen physical rect. The DPI scale is performed relative to the display nearest to `window`. If `window` is null, scaling will be performed to the display nearest to `rect`.