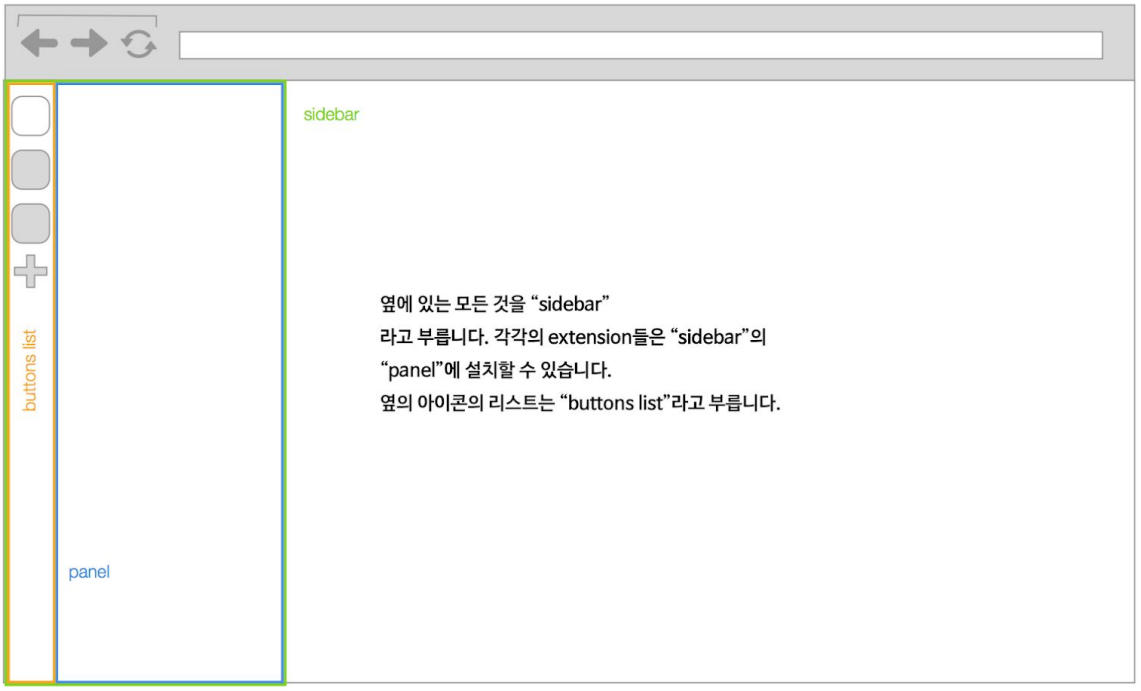
Opera Branding Guidelines - (OPG)
Editor : Chanuim
2020.04.10

extension -> 확장팩 혹은 확장프로그램으로 번역했습니다.

## Introduction

브라우저의 사이드바에 위치한 확장 프로그램을 만들기 위해서, sidebarActionAPI를 어떻게 사용하는지에 관해 설명한 아티클 입니다.

## The Sidebar




오페라30이 되고, 브라우저 사이드바가 나왔읍니다.
사이드바는 브라우저의 왼쪽에 위치해 있으며, 사용자를 위한 추가적인 기능을 붙인 공간이 있는 곳입니다.
사이드바의 왼쪽에는 아이콘 리스트가 있는데 아이콘을 클릭하면 그에 맞는 패널이 열립니다.

패널은 메인 컨텐츠를 보관하는 확장프로그램의 안에 명시된 HTML 페이지입니다.
모든 사이드바 액션은 반-드시 패널 페이지가 명시되어 있어야 합니다.

## The Manifest

사이드바 확장프로그램의 manifest가 따라야 할 몇 가지들을 언급하기 때문에 이 부분은 중요합니다.

```
“sidebar_action”: {
        // 필요한 부분이에요~
        “default_icon”: “icon.png”,
        // 옵션이에요~
        “default_title”: “My Sample Extension”,
        // 필요한 부분이에요~
        “default_panel”: “panel.html”
}
```

물론 위에 적은 한 개의 아이콘 대신에 일련의 사이즈와 아이콘도 key-value 조합으로 명시할 수 있습니다.

```
“sizebar_action”: {
        “default_icon”: {
                “19”: “images/19.png”,
                “38”: “images/38.png”
        }
}
```

## The sidebarAction API

사이드바 액션 API는 브라우저 액션 API와 비슷하게 만들었습니다. 그래서 확장프로그램 개발자는 쉽게 API를 이해할 수 있습니다.
그리고 기존의 확장프로그램에서 쉽게 사이드바를 이식할 수 있습니다.

구조면에서는 비슷해 보이지만, 사이드바 확장 프로그램은 브라우저액션을 사용하는 확장프로그램과는 다르게 보여야 됩니다.
사이드바의 확장프로그램은 더 오랫동안 살아있습니다.

## Maintaining state

패널 페이지는 다른 웹 페이지와 비슷한 방식으로 작동한다.
그래서 만약 사용자가 패널을 닫는다면, 다른 어떤 페이지의 탭이닫힌 것마냥 같은 방식으로 작동한다.
즉, 사용자가 패널을 연다면, 웹 페이지가 새로고침한것 마냥  처음부터 시작된다는 말이다.

하지만, 패널은 명줄이 긴 앱으로 만들어졌다면, 상태를 유지할 방법이 있어야 한다.
이 뜻은 사용자가 패널을 열고 닫음에 상관없이 데이터가 계속되어야 한다는 말이다.
팝업에 입력했던 데이터를 background script로 보냄으로써 이게 실현가능해졌다.

textarea로 한번 해보자.
우린 유저가 입력한 모든 데이터가 패널을 열거나 닫음에 상관없이 여전히 거기 있었으면 한다.
작성 코드는 이렇다.

```
<h1>Saving state</h1>
<p>
        The stuff typed in the textarea should still be there
        exactly as you had typed it, even if you close and re-open the panel.
</p>
<textarea id=”maintext”></textarea>

let maintext = document.querySelector(“#maintext”);
let theValue;

maintext.onchange = function() {
        save();
}

function save() {
        theValue = maintext.value;
```

```
        chrome.extension.getBackgroundPage().setValue(theValue);
}

function show() {
        theValue = chrome.extension.getBackgroundPage().getValue();
        if(!theValue){
                theValue = '';
        }
        maintext.value = theValue;
}

document.addEventListener('DOMContentLoaded', show, false);
```

onchange이벤트를 listening하고 있어서 text area가 바뀔 때마다, save() 함수를 부르는 걸 볼 수 있습니다.

DOMContentLoaded이벤트를 listen하고 있는 페이지가 로드될 때마다, background 스크립트에 있던 값을 되찾아오고, textarea에 다시 넣는다.

이렇게 할 수도 있읍니다.

```
let value;

function getValue() {
        value = localStorage.getItem("maintext");
        return value;

function setValue(theValue) {
        localStorage.setItem("maintext", theValue);
}
```

패널이 열려있지않다면 닫혔다고 간주 할 수 있지만, 백그라운드는 여전히 돌아가고 있다는 걸 잊지 마세요!
그래서 localStorage에 데이터를 저장하거나 아님 스크립트 그 자체에 저장하거나 하는 등의 행동으로 상태를 유지할 수 있습니다.

## Detecting user focus on the panel

사용자가 패널에 focus중인가 아닌가를 감지하는걸 원하는 상황 일 수도 있다.
패널 페이지에 window요소의 onfocus와 onblur 이벤트를 listen할 수 있음.
그렇게 함으로써 사용자가 패널에 포커싱했는지 아닌지를 확인 할 수 있음.

```
window.onfocus = function() {
        console.log('The user is focussed on the panel page');
}
window.onblur = function() {
        console.log('The user has left focus from the panel page');
}
```