JavaScript Redirect

Summary: in this tutorial, you will learn how to use JavaScript to redirect to a new URL or page.

Introduction to the JavaScript redirect

Sometimes, you want to redirect users from one page to another page that has a different URL.

For example, you can create an app landing page that purely redirects the users to the Google Play or Apple Store, depending on the operating system that their smartphones have.

To do it, you need to:

- Detect the operating system of the smartphones
- Redirect to Apple Store if the OS is iOS and Google Play if the OS is Android.

The redirection can be done in the web browsers using JavaScript redirection API.

It's important to note that JavaScript redirection runs entirely on web browsers. Therefore it doesn't return the status code 301 (move permanently) like a server redirection.

Therefore, if you move the site to a separate domain or create a new URL for an old page, you should always use the server redirection.

Redirect to a new URL

To redirect web browsers to a new URL from the current page to a new one, you use the `location` object:

```
location.href = 'new_url';
```

For example:

```
location.href = 'https://www.javascripttutorial.net/';
```

Assigning a value to the `href` property of the `location` object has the same effect as calling the `assign()` method of the `location` object:

```
location.assign('https://www.javascripttutorial.net/');
```

Either of these calls will redirect to the new URL and create an entry in the history stack of the browser. It means that you can go back to the previous page via the **Back** button of the browser.

To redirect to a new URL without creating a new entry in the history stack of the browser, you use the `replace()` method of the `location` object:

```
location.replace('https://www.javascripttutorial.net/');
```

The following example creates a page that redirects web browsers to Google Play or Apple Store if the OS is Android or iOS. Otherwise, it'll show a message indicating that the OS is not supported:

index.html

```
<!DOCTYPE html>
<html lang="en">

    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>JavaScript Redirect</title>

    </head>

    <body>
        <div class="message"></div>
        <script src="js/app.js"></script>
    </body>

</html>
```

js/app.js

```
const apps = {
  Android: 'https://play.google.com/',
  iOS: 'https://www.apple.com/store',
};

const platform = () => {
  let userAgent = navigator.userAgent || navigator.vendor || window.opera;
  if (/windows phone/i.test(userAgent)) return 'Windows Phone';
  if (/android/i.test(userAgent)) return 'Android';
  if (/iPad|iPhone|iPod/.test(userAgent) && !window.MSStream) return 'iOS';
  return null;
};

const redirect = () => {
  let os = platform();
  if (os in apps) {
    location.replace(apps[os]);
  } else {
    const message = document.querySelector('.message');
    message.innerText = 'Your OS is not supported';
  }
};

redirect();
```

How it works.

First, define an `apps` object with the keys are OS and values are URLs of the Google Play and Apple Store:

```
const apps = {
  Android: 'https://play.google.com/',
  iOS: 'https://www.apple.com/store',
};
```

Second, define a function that detects the OS:

```
const platform = () => {
  let userAgent = navigator.userAgent || navigator.vendor || window.opera;
  if (/windows phone/i.test(userAgent)) return 'Windows Phone';
  if (/android/i.test(userAgent)) return 'Android';
  if (/iPad|iPhone|iPod/.test(userAgent) && !window.MSStream) return 'iOS';
  return null;
};
```

Third, create the `redirect()` function that detects the OS and redirects the web browser based on the detected OS:

```
const redirect = () => {
  let os = platform();
  if (os in apps) {
    location.replace(apps[os]);
  } else {
    const message = document.querySelector('.message');
    message.innerText = 'Your OS is not supported';
  }
};
```

Finally, call the `redirect()` function:

```
redirect();
```

Redirect to a relative URL

The following script redirects to the `about.html` which is on the same level as the current page.

```
location.href = 'about.html';
```

The following script redirects to `contact.html` page located in the `root` folder:

```
location.href = '/contact.html';
```

Redirect upon page loading

If you want to redirect to a new page upon loading, you use the following code:

```
window.onload = function() {
    location.href = "https://www.javascripttutorial.net/";
}
```

Summary

- To redirect to a new URL or page, you assign the new URL to the `location.href` property or use the `location.assign()` method.
- The `location.replace()` method does redirect to a new URL but does not create an entry in the history stack of the browser.