9-го октября 2020

Открытие окон и методы window

Всплывающее окно («попап» – от англ. Popup window) – один из древнейших способов показать пользователю ещё один документ.

Достаточно запустить:

```
window.open('https://javascript.info/')
```

… и откроется новое окно с указанным URL. Большинство современных браузеров по умолчанию будут открывать новую вкладку вместо отдельного окна.

Попапы существуют с доисторических времён. Они были придуманы для отображения нового контента поверх открытого главного окна. Но с тех пор появились другие способы сделать это: JavaScript может загрузить содержимое вызовом fetch и показать его в тут же созданном `<div>`, так что попапы используются не каждый день.

Кроме того, попапы не очень хороши для мобильных устройств, которые не умеют показывать несколько окон одновременно.

Однако, для некоторых задач попапы ещё используются, например для OAuth-авторизации (вход через Google/Facebook/…), так как:

1. Попап – это отдельное окно со своим JavaScript-окружением. Так что открытие попапа со стороннего, не доверенного сайта вполне безопасно
2. Открыть попап очень просто.
3. Попап может производить навигацию (менять URL) и отсылать сообщения в основное окно.

Блокировка попапов

В прошлом злонамеренные сайты заваливали посетителей всплывающими окнами. Такие страницы могли открывать сотни попапов с рекламой. Поэтому теперь большинство браузеров пытаются заблокировать всплывающие окна, чтобы защитить пользователя.

Всплывающее окно блокируется в том случае, если вызов window.open произошёл не в результате действия посетителя (например, события `onclick`).

Например:

```
// попап заблокирован
window.open('https://javascript.info');

// попап будет показан
button.onclick = () => {
  window.open('https://javascript.info');
};
```

Таким образом браузеры могут защитить пользователя от появления нежелательных попапов, при этом не отключая попапы полностью.

Что, если попап должен открываться в результате `onclick`, но не сразу, а только после выполнения `setTimeout`? Здесь все не так-то просто.

Запустим код:

```
// откроется через 3 секунды
setTimeout(() => window.open('http://google.com'), 3000
```

Попап откроется в Chrome, но будет заблокирован в Firefox.

Но если мы уменьшим тайм-аут до одной секунды, то попап откроется и в Firefox:

```
// откроется через 1 секунду
setTimeout(() => window.open('http://google.com'), 1000
```

Мы получили два разных результата из-за того, что Firefox «допускает» таймаут в 2000 мс или менее, но все, что свыше этого – не вызывает его доверия, т.к. предполагается, что в таком случае открытие окна происходит без ведома пользователя. Именно поэтому попап из первого примера будет заблокирован, а из второго – нет.

Полный синтаксис window.open

Синтаксис открытия нового окна: `window.open(url, name, params)`:

url

URL для загрузки в новом окне.

name

Имя нового окна. У каждого окна есть свойство `window.name`, в котором можно задавать, какое окно использовать для попапа. Таким образом, если уже существует окно с заданным именем – указанный в параметрах URL откроется в нем, в противном случае откроется новое окно.

params

Строка параметров для нового окна. Содержит настройки, разделённые запятыми. Важно помнить, что в данной строке не должно быть пробелов. Например `width=200,height=100`.

Параметры в строке `params`:

- Позиция окна:
  - `left/top` (числа) – координаты верхнего левого угла нового окна на экране. Существует ограничение: новое окно не может быть позиционировано вне видимой области экрана.
  - `width/height` (числа) – ширина и высота нового окна. Существуют ограничения на минимальную высоту и ширину, которые делают невозможным создание невидимого окна.
- Панели окна:
  - `menubar` (yes/no) – позволяет отобразить или скрыть меню браузера в новом окне.
  - `toolbar` (yes/no) – позволяет отобразить или скрыть панель навигации браузера (кнопки вперёд, назад, перезагрузки страницы) нового окна.
  - `location` (yes/no) – позволяет отобразить или скрыть адресную строку нового окна. Firefox и IE не позволяют скрывать эту панель по умолчанию.
  - `status` (yes/no) – позволяет отобразить или скрыть строку состояния. Как и с адресной строкой, большинство браузеров будут принудительно показывать её.
  - `resizable` (yes/no) – позволяет отключить возможность изменения размера нового окна. Не рекомендуется.
  - `scrollbars` (yes/no) – позволяет отключить полосы прокрутки для нового окна. Не рекомендуется.

Помимо этого существует некоторое количество не кроссбраузерных значений, которые обычно не используются. Найти примеры таких свойств можно по ссылке.

Пример: минималистичное окно

Давайте откроем окно с минимальным набором настроек, просто чтобы посмотреть, какие из них браузер позволит отключить:


```
let params = `scrollbars=no,resizable=no,status=no,loca
width=0,height=0,left=-1000,top=-1000`;

open('/', 'test', params);
```

В этом примере большинство настроек заблокированы и само окно находится за пределами видимой области экрана. Посмотрим, что получится в результате. Большинство браузеров «исправит» странные значения – как, например, нулевые `width/height` и отрицательные `left/top`. Например, Chrome установит высоту и ширину такого окна равной высоте и ширине экрана, так что попап будет занимать весь экран.

Давайте исправим значения и зададим нормальные координаты (`left` и `top`) и значения размеров окна (`width` и `height`):

```
let params = `scrollbars=no,resizable=no,status=no,loca
width=600,height=300,left=100,top=100`;

open('/', 'test', params);
```

Большинство браузеров выведет окно с заданными нами настройками.

Правила для опущенных параметров:

- Если третий аргумент при вызове `open` отсутствует или он пустой, будут использованы настройки окна по умолчанию.
- Если строка параметров передана, но некоторые параметры yes/no пропущены, то считается, что указано `no`, так что соответствующие возможности будут отключены, если на это нет ограничений со стороны браузера. Поэтому при задании параметров убедитесь, что вы явно указали все необходимые yes.
- Если координаты `left/top` не заданы, браузер попытается открыть новое окно рядом с предыдущим открытым окном.
- Если не заданы размеры окна `width/height`, браузер откроет новое окно с теми же размерами, что и предыдущее открытое окно.

Доступ к попапу из основного окна

Вызов `open` возвращает ссылку на новое окно. Эта ссылка может быть использована для управления свойствами окна, например, изменения положения и др.

Например, здесь мы генерируем содержимое попапа из JavaScript:

```
let newWin = window.open("about:blank", "hello", "width

newWin.document.write("Hello, world!");
```

А здесь содержимое окна модифицируется после загрузки:

```
let newWindow = open('/', 'example', 'width=300,height=
newWindow.focus();

alert(newWindow.location.href); // (*) about:blank, заг

newWindow.onload = function() {
  let html = `<div style="font-size:30px">Добро пожалов
  newWindow.document.body.insertAdjacentHTML('afterbegi
};
```

Обратите внимание: сразу после `window.open` новое окно ещё не загружено. Это демонстрируется в строке `(*)`. Так что нужно ждать `onload`, чтобы его изменить. Или же поставить обработчик `DOMContentLoaded` на `newWin.document`.

Политика одного источника

Окна имеют свободный доступ к содержимому друг друга только если они с одного источника (у них совпадают домен, протокол и порт (protocol://domain:port).

Иначе, например, если основное окно с `site.com`, а попап с `gmail.com`, это невозможно по соображениям пользовательской безопасности. Детали см. в главе Общение между окнами.

Доступ к открывшему окну из попапа

Попап также может обратиться к открывшему его окну по ссылке `window.opener`. Она равна `null` для всех окон, кроме попапов.

Если вы запустите код ниже, то он заменит содержимое открывшего (текущего) окна на «Тест»:

```
let newWin = window.open("about:blank", "hello", "width
newWin.document.write(
  "<script>window.opener.document.body.innerHTML = 'Tec
);
```

Так что связь между окнами двусторонняя: главное окно и попап имеют ссылки друг на друга.

Закрытие попапа

Чтобы закрыть окно: `win.close()`

Для проверки, закрыто ли окно: `win.closed`.

Технически метод `close()` доступен для любого окна, но `window.close()` будет игнорироваться большинством браузеров, если `window` не было создано с помощью `window.open()`. Так что он сработает только для попапов.

Если окно закрыто, то его свойство `closed` имеет значение `true`. Таким образом можно легко проверить, закрыт ли попап (или главное окно) или все ещё открыт. Пользователь может закрыть его в любой момент, и наш код должен учитывать эту возможность.

Этот код откроет и затем закроет окно:

```
let newWindow = open('/', 'example', 'width=300,height=
newWindow.onload = function() {
  newWindow.close();
  alert(newWindow.closed); // true
};
```

Прокрутка и изменение размеров

Методы для передвижения и изменения размеров окна:

`win.moveBy(x,y)`

Переместить окно относительно текущей позиции на `x` пикселей вправо и `y` пикселей вниз. Допустимы отрицательные значения (для перемещения окна влево и вверх).

`win.moveTo(x,y)`

Переместить окно на координаты экрана `(x,y)`.

`win.resizeBy(width,height)`

Изменить размер окна на указанные значения `width/height` относительно текущего размера. Допустимы отрицательные значения.

`win.resizeTo(width,height)`

Изменить размер окна до указанных значений.

Также существует событие `window.onresize`.

> **Только попапы**
>
> Чтобы предотвратить возможные злоупотребления, браузер обычно блокирует эти методы. Они гарантированно работают только с попапами, которые мы открыли сами и у которых нет дополнительных вкладок.

> **Нельзя свернуть/развернуть окно**
>
> Методами JavaScript нельзя свернуть или развернуть («максимизировать») окно на весь экран. За это отвечают функции уровня операционной системы, и они скрыты от фронтенд-разработчиков.
>
> Методы перемещения и изменения размера окна не работают для свернутых и развёрнутых на весь экран окон.

Прокрутка окна

Мы уже говорили о прокрутке окна в главе Размеры и прокрутка окна.

`win.scrollBy(x,y)`

Прокрутить окно на `x` пикселей вправо и `y` пикселей вниз относительно текущей прокрутки. Допустимы отрицательные значения.

`win.scrollTo(x,y)`

Прокрутить окно до заданных координат `(x,y)`.

`elem.scrollIntoView(top = true)`

Прокрутить окно так, чтобы `elem` для `elem.scrollIntoView(false)` появился вверху (по умолчанию) или внизу.

Также существует событие `window.onscroll`.

Установка и потеря фокуса

Теоретически, установить попап в фокус можно с помощью метода `window.focus()`, а убрать из фокуса – с помощью `window.blur()`. Также существуют события `focus/blur`, которые позволяют отследить, когда фокус переводится на какое-то другое окно.

Раньше на «плохих» сайтах эти методы могли становиться средством манипуляции. Например:

```
1  window.onblur = () => window.focus();
```

Когда пользователь пытается перевести фокус на другое окно, этот код возвращает фокус назад. Таким образом, фокус как бы «блокируется» в попапе, который не нужен пользователю.

Из-за этого в браузерах и появились ограничения, которые препятствуют такого рода поведению фокуса. Эти ограничения нужны для защиты пользователя от назойливой рекламы и «плохих» страниц, и их работа различается в зависимости от конкретного браузера.

Например, мобильный браузер обычно полностью игнорирует такие вызовы метода `window.focus()`. Также фокусировка не работает, когда попап открыт в отдельной вкладке (в отличие от открытия в отдельном окне).

Но все-таки иногда методы фокусировки бывают полезны. Например:

Редактировать на GitHub

- Когда мы открываем попап, может быть хорошей идеей запустить для него `newWindow.focus()`. Для некоторых комбинаций браузера и операционной системы это устранит неоднозначность – заметит ли пользователь это новое окно.
- Если нужно отследить, когда посетитель использует веб-приложение, можно отслеживать `window.onfocus/onblur`. Это позволит ставить на паузу и продолжать выполнение анимаций и других интерактивных действий на странице. При этом важно помнить, что `blur` означает, что окно больше не в фокусе, но пользователь может по-прежнему видеть его.

Итого

Всплывающие окна используются нечасто. Ведь загрузить новую информацию можно динамически, а показать – в элементе `<div>`, расположенным над страницей (`z-index`). Ещё одна альтернатива – тег `<iframe>`.

Если мы открываем попап, хорошей практикой будет предупредить пользователя об этом. Иконка открывающегося окошка на ссылке поможет посетителю понять, что происходит и не потерять оба окна из поля зрения.

- Новое окно можно открыть с помощью вызова `open(url, name, params)`. Этот метод возвращает ссылку на это новое окно.
- По умолчанию браузеры блокируют вызовы `open`, выполненные не в результате действий пользователя. Обычно браузеры показывают предупреждение, так что пользователь все-таки может разрешить вызов этого метода.
- Вместо попапа открывается вкладка, если в вызове `open` не указаны его размеры.
- У попапа есть доступ к породившему его окну через свойство `window.opener`.
- Если основное окно и попап имеют один домен и протокол, то они свободно могут читать и изменять друг друга. В противном случае, они могут только изменять положение друг друга и взаимодействовать с помощью сообщений.

Чтобы закрыть попап: метод `close()`. Также попап может закрыть и пользователь (как и любое другое окно). После закрытия окна свойство `window.closed` имеет значение `true`.

- Методы `focus()` и `blur()` позволяют установить или убрать фокус с попапа. Но работают не всегда.
- События `focus` и `blur` позволяют отследить получение и потерю фокуса новым окном. Но, пожалуйста, не забывайте, что окно может остаться видимым и после `blur`.

Комментарии

перед тем как писать...