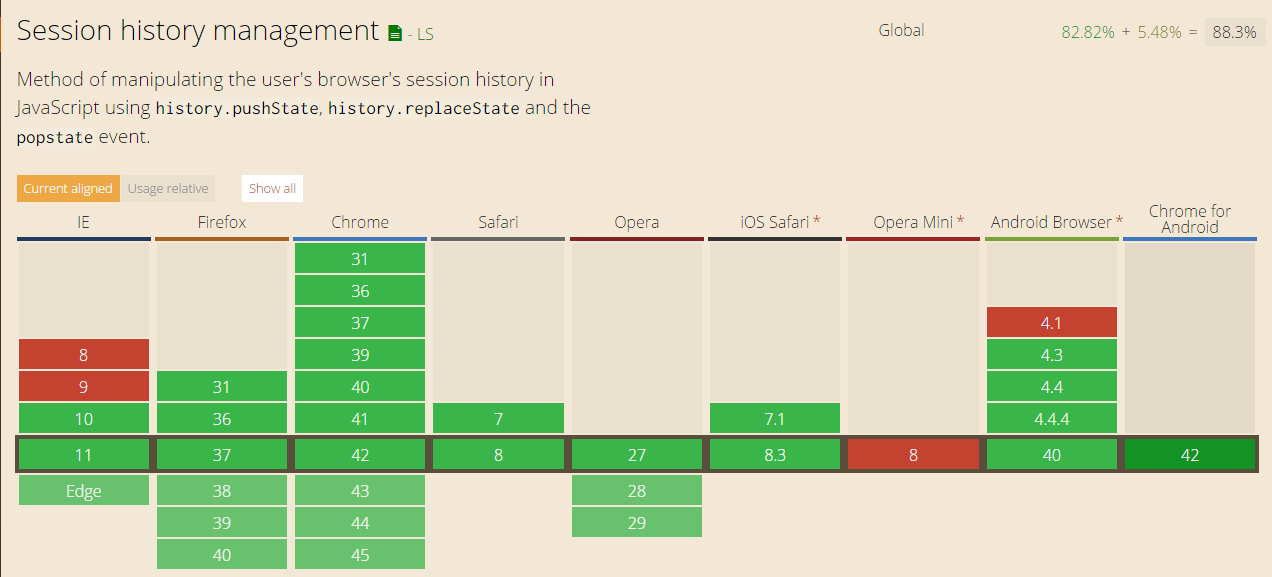
History API



Объект History существовал и до HTML5, но предоставлял меньше возможностей и поддерживался не всеми браузерами.

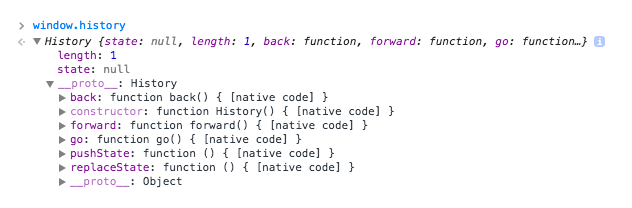
На данный момент history позволяет работать с историей конкретной вкладки (**session history**). Т.е. позволяет манипулировать историей браузера через скрипт.

HTML5 расширяет history API, дает возможность манипулировать URL браузера и поддерживаемым состоянием. Это удобно для перехода по session History из кода и когда нам нужно сохранить какое-то состояния страницы в отдельном url, т.е когда мы хотим сделать доступным состояние страницы (полученное в результате срабатывания ajax запроса, например) после взаимодействия с пользователем с помощью отдельного url)

Важно то, что history API не перезагружает страницу.

Ранее единственным способом изменить URL было изменить window.location, которая всегда перезагружала страницу. (перезагрузка не происходила только в случае если к url прибавлось значение через #) еще можно было использовать hashbang, но оно доступно только на стороне клиента и работает только с js и являлось временным решением

Методы и свойства History API



***Свойства:***

**history.length** – содержит количество записей в session history

## history.state – содержит то, что передается в первом параметре в методах pushState/replaceState

## *Методы*

## history.go() – параметр – целое число, обозначающее на сколько позиций по истории нужно сместиться.

Положительное число – смещение вперед

Отрицательное число – смещение назад

0 – перегрузит текущую страницу

## history.back() & history.forward() – переход на 1 позицию назад и вперед, если в session history нет запрашивамого состояния, ошибки не возникнет, это будет эквивалентно history.go()

## history.pushState() – записывает новое состояние в session history и переходит на него, метод принимает 3 параметра

history.pushState([data], [title], [url]);

1. State object – данные в формате json, которые можно передавать в созданный стейт , если ничего передавать не надо – указывать null, все что передается в первом параметре, будет доступно в объекте события в обработчике события popstate
2. Title – Title новой страницы
3. url – то, что появится в адресной строке

history.pushState(null, null, 'http://www.domain.com/whatever');

После pushState если нажать кнопку назад, мы вернемся на первоначальную страницу

3 параметр, должен соблюдать same origin policy

history.pushState(null, null, 'https://twitter.com/hello');

Сгенерирует ошибку, если б same origin policy не соблюдалось, разработчики могли б подменять url и выдавать 1 сайт за другой

window.addEventListener('popstate', function(e) {

// e.state is equal to the data-attribute of the last image we clicked

});

## history.replaceState() - аналогичен pushState, но вместо того, чтоб создавать новый стейт, он записывает его вместо текущего в session history.

В адресной строке текущий url заменится на указанный.

**Popstate event –** событие срабатывает, при любом изменении цепочки history. Если эти изменения вызваны методами pushState или replaceState, то popstate свойство state содержит копию состояния объекта history

window.addEventListener('popstate', function(e) {

// do stuff here

});

window.addEventListener('popstate', function(e) {

var character = e.state;

if (character == null) {

removeCurrentClass();

textWrapper.innerHTML = " ";

content.innerHTML = " ";

document.title = defaultTitle;

} else {

updateText(character);

requestContent(character + ".html");

addCurrentClass(character);

document.title = "Ghostbuster | " + character;

}

});

Links

<https://css-tricks.com/using-the-html5-history-api/>

<http://code.tutsplus.com/tutorials/a-first-look-at-the-html5-history-api--net-28053>

<http://blog.teamtreehouse.com/getting-started-with-the-history-api>

<http://html5doctor.com/history-api/>

<http://www.impressivewebs.com/html5-history-api-syntax/>