JAVASCRIPT

Отличные советы

Кайо Рибейру Перейра

JavaScript. Отличные советы

Сборник советов и трюков по JavaScript

Caio Ribeiro Pereira и Alexey Pyltsyn

Эта книга предназначена для продажи на http://leanpub.com/javascript-awesome-tips-russian

Эта версия была опубликована на 2019-03-31

Это книга с Leanpub book. Leanpub позволяет авторам и издателям участвовать в так называемом Lean Publishing - процессе, при котором электронная книга становится доступна читателям ещё до её завершения. Это помогает собрать отзывы и пожелания для скорейшего улучшения книги. Мы призываем авторов публиковать свои работы как можно раньше и чаще, постепенно улучшая качество и объём материала. Тем более, что с нашими удобными инструментами этот процесс превращается в удовольствие.

Я посвящаю эту книгу моей семье, которая всегда поддерживает меня и мотивирует меня с самого начала моей жизни.

Особая благодарность Алине Брандарис Сантос (Aline Brandariz Santos) за то, что она улыбка моей жизни, которая сделала меня счастливым, за её поддержку и надежду.

Отдельное спасибо миссис Шарлотте Бенто де Карвалью (Charlotte Bento de Carvalho), моему двоюродному брату Клаудио Соуза (Cláudio Souza), моим друзьям Леандро Альварес да Коста (Leandro Alvares da Costa) и Бруно Альварес да Коста (Bruno Alvares da Costa), Пауло Сильвейра (Paulo Silveira) и Адриано Алмейда (Adriano Almeida). Они сыграли важную роль в моей жизни, повлияли на то, кем я есть сегодня, а, следовательно, тот, кто написал эту книгу.

В конце концов, спасибо всем читателям сайта https://udgwebdev.com, содержание этой книги основывается на многих постах из этого блога.

И наконец, спасибо тебе, дорогой читатель, за покупку этой книги, надеюсь, тебе она понравится!

Оглавление

Предисловие

JavaScript доминирует в мире

Сегодня JavaScript везде, он широко используется в браузерах, серверах (через Node.js или другие JS-движки для бекенда), мобильных устройствах (через Cordova, Phonegap, React Native и т.д.), IoT, SmartTV и этот список ещё можно продолжать. Если вы веб-разработчик, рано или поздно в своей карьеры вам нужно было хоть немного пописать на JavaScript, чтобы понять, что всё более активно этот язык завоёвывает все больше места в мире. Этот язык с 2015 года получает важные обновления, такие как ES6, ES7, ES8, ES9 и ES10 (он же ECMAScript), которые приносят с собой реализации новых функциональных возможностей, а вместе с ними и новые способы решения проблем более простым способом.

Для кого эта книга

Эта книга предназначена для всех разработчиков, у которых есть опыт работы на JavaScript, по крайней мере навыки основного или среднего уровня, а также для тех, кто стремится узнать больше про некоторые практические советы, трюки и хаки.

Как читать эту книгу

Эта книга представляет ряд проблем и их практических решений на JavaScript, нет необходимости читать каждую главу этой книги в хронологическом порядке. В конце концов, это книга рецептов с коллекцией практических рекомендаций, которые можно применить в своих JavaScript-проектах.

Глава 1: советы по работе со строками

Создание строки для ЧПУ через регулярные выражения

Если нужно создать так называемый «слаг» (slug, описательная часть человекопонятного URL-адреса), например, преобразовать фразу «JavaScript is awesome» в «javascript-is-awesome», то легко сделать с помощью следующей функции:

```
function slugify(content) {
  return content.toLowerCase().replace(/\s/g, '-').trim();
}

slugify("Writing JavaScript Better");
// "writing-javascript-better"
```

Или, если вы хотите внедрить эту функцию в объект String, сделав её встроенной функцией для всех строк:

```
String.prototype.slugify = function() {
  return this.toLowerCase().replace(/\s/g, '-').trim();
}

"Writing JavaScript Better".slugify();
// "writing-javascript-better"
```

Вставка данных в строку

С тех пор, как вышел ES6 (он же ES2015), можно элегантно вставить данные в строку. Вы, вероятно, уже видели много подобного кода:

```
const name = "John Connor";
const message = "I came from future!";
let template = "<p>";
template += "<h3>" + name + "</h3>";
template += "<span>" + message + "</span>";
template += "</p>";
```

Но есть более чистый способ, благодаря возможности ES6 *шаблонные строки (Template Strings)*. Она позволяет вставить данные в строку без сложных выражений с конкатенацией, это исключает использование ненужное использование оператора **+** для этой цели. В качестве можно привести следующий пример использования:

```
const name = "John Connor";
const message = "I came from future!";
const template = `
 <p>
    <h3>${name}</h3>
    <span>${message}</span>
 </p> `;
```

Повтор строк подряд

Когда нужно повторить строку определённое количество раз, наиболее частый способ — написать что-то подобное:

```
let content = '';
const msg = 'Hello! ';
const repeat = 3;
for (let i = 0; i < repeat; i++) {
  content += msg;
}
console.log(content);
// Hello! Hello! Hello!
```

Либо решение в функциональном стиле, немного меньше кода:

```
const repeat = Array(3);
const msg = 'Hello! ';
const content = repeat.map(() => msg).join('');
console.log(content);
// Hello! Hello! Hello!
```

Однако сейчас у ES6 есть простая и подходящая для этой простой задачи функция. Количество повторений указывается в первом параметре, таким образом, за всего за одну строчку кода можно выполнить эту задачу семантическим, читабельным и производительным способом. Достаточно просто использовать функцию `String.prototype.repeat ()`:

```
console.log('Hello!'.repeat(3));
// Hello! Hello! Hello!
```

Разница между `substring()` и `substr()`

Вопрос на засыпку: вы знаете основную разницу между функциями `substring()` и `substr()`?

В сущности, обе функции принимают одни и теж аргументы. Первый аргумент — это **начальный индекс**, но отличие заключается во **втором аргументе** обеих функций, посмотрите на следующий пример:

```
 "JavaScript".substr(4, 6); // "Script"
 "JavaScript".substring(4, 6); // "Sc"
```

- `substr(index, amount)` — второй параметр возвращает **количество символов**, начиная со значения **index**.
- `substring(start, end)` — эта функция возвращает новую строку на основе **начальной и конечной позиции индекса**.

Преобразование в прописные буквы

В JavaScript буквы в строке можно сделать полностью прописными (`String.prototype.toUpperCase()`), либо строчными (`String.prototype.toLowerCase()`), однако нет функции, чтобы превратить только первый символ в верхний регистр, а оставшуюся часть строки — в нижний регистр. Поэтому, чтобы достичь такой задачи, вы можете создать функцию, которая показана ниже:

```
function capitalize(s) {
  return `${s.charAt(0).toUpperCase()}${s.substr(1).toLowerCase()}`;
}

console.log(capitalize('JAVASCRIPT')); // "Javascript"
```

Если вы хотите встроить эту функцию в встроенный объект String, достаточно написать функцию-прототип, например так:

```
String.prototype.capitalize = function() {
  return `${this.charAt(0).toUpperCase()}${this.substr(1).toLowerCase()}`;
}

console.log('JAVASCRIPT'.capitalize()); // "Javascript"
```

Редактирование строк запросов в браузере

Вместо того, чтобы тратить время на ручное редактирование строки URL-адреса, а также увеличивать с этим сложность кода, и всё это для добавления параметров в строку запроса, вы, вероятно, писали либо видели что-то вроде этого:

```
const lang = 'en-us';
const section = 'books';
let url = 'https://crpwebdev.github.io';
url += '?lang ='+ lang;
url += '&section ='+ section;
console.log(url); // "https://crpwebdev.github.io?lang=en&section=books"
```

Но есть безопасный и встроенный в браузере способ редактирования строк URL-адресов с помощью API URL. Можно совершенно безопасно использовать функции `set()` и `get()`, чтобы управлять параметрами запроса. Но учтите, данная возможность существует **только в браузерном JavaScript посредством API HTML5**:

```
const lang = 'en-us';
const section = 'books';
const url = new URL('https://crpwebdev.github.io');
url.searchParams.set('lang', lang);
url.searchParams.set('section', section);
console.log(url); // "https://crpwebdev.github.io?lang=en&section=books"
```

Три способа конвертировать строку в массив

В данный момент существует три способа *разбить строк* в одну строку кода, эта концепция в основном заключается в преобразовании **строки** в **массив символов**. А далее воспользовавшись преимущества массива, использовать соответствующие функции для управления строками:

Использование классического метода `String.prototype.split()` с **пустой строкой** в первом параметре:

```
const title = 'Book';
// Обязательно передать пустую строку
console.log(title.split('')); // ['B', 'o', 'o', 'k']
// Использование этой функции без аргумента или со значением undefined возвращает ма\
ссив с одной строкой
console.log(title.split()); // ['Book']
```

Используя `Array.from()`, передав ей строку в первом параметре:

```
const title = 'Book';
console.log(Array.from(title)); // ['B', 'o', 'o', 'k']
```

Либо можно использовать оператор расширения:

```
const title = 'Book';
console.log([...title]); // ['B', 'o', 'o', 'k']
```

Замена по всей строке

Функция `String.prototype.replace()` крайне полезна, когда нужно заменить определённые символы строки. Кроме того, в этой функции возможно использовать **регулярное выражение**, чтобы производить замены в сложных строках. Эта возможность в свою очередь поддерживает за один вызов заменить несколько вхождений строк, вместо того, чтобы выполнять несколько раз функцию, для этого достаточно использовать `/g` в конце выражения, посмотрите на это в действии:

```
const title = 'JavaJavaScript';
console.log(title.replace(/Java/, '')); // "JavaScript"
console.log(title.replace(/Java/g, '')); // "Script"
```

Поиск слов в атрибутах объекта

Этот совет очень полезен тем, что вы можете искать любое вхождение слова по атрибутам объекта. Это легко добиться всего одной строчкой кода `Object.values().toString().includes('string to search')`:

```
const person = {
  name: 'John Connor',
  twitter: '@john'
};

Object.values(person).toString().includes('Connor'); // true
```

В целом, функция `Object.values(person)` возвращает значения всех атрибутов объекта в виде массива, после которой можно воспользоваться `Array.prototype.toString()`, чтобы объединить элементы массива через запятую в одну строку. Наконец, используем `String.prototype.include()`, чтобы искать слово по всей строке — если оно есть, будет возвращено значение `true`.

Извлечение содержимого из HTML-тегов в строке

Если когда-нибудь понадобится очистить строку от HTML-тегов, то это легко можно сделать используя `String.prototype.replace ()`, как показано ниже:

```
const content = '<h1>JavaScript</h1> <h2>is the best!</h2>';
const text = content.replace(/<[a-zA-Z/][^>]*>/g, '');
console.log(text); // "JavaScript is the best!"
```

Глава 2: Советы по работе с числами

Приведение числа к валюте

В JavaScript ES6 появилась встроенная возможность форматировать числа в виде **денежных единиц**, используя функцию `Number.prototype.toLocaleString()`. Смотрите примеры использования ниже:

```
(10.9).toLocaleString(); // "10.90"
(1002.5).toLocaleString('en-US'); // "1,002.50"
(5.55).toLocaleString('en-US', {
  // Изменение десятичных цифр
  minimumFractionDigits: 2,
  maximumFractionDigits: 2
});
```

Что самое замечательное — теперь, чтобы преобразовать число вам не нужно использовать обходные варианты с использованинем `Math.abs()` или `Number.prototype.toFixed()`. А ещё эта функция защищена от **бага в числах с плавающей точкой**, которые ещё существуют в JavaScript:

```
// Баг с числом с плавающей точкой
0.1 + 0.2 // 0.30000000000000004
// Решение этой проблемы с помощью новой функции
(0.1 + 0.2).toLocaleString(); // "0.3"
```

Для получения более подробной информации о локалях и других параметрах этой функции вы можете обратиться к документации Mozilla: Number.prototype.toLocaleString() - JavaScript | MDN[1]

Приведение строки к числам с помощью оператора +

Подобная магия выглядит круто, вдобавок к тому, что её легко использовать. Код может быть менее лёгким для понимания, чем при использовании `Number('100')` для преобразования

[1] https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Number/toLocaleString

строки в число. Как бы то ни было, всегда полезно узнать что-то новое. И вот, в качестве альтернативе можно использовать оператор +, чтобы преобразовать числовую строку в обычное число. Если строка не является числом, будет возвращено значение `NaN` *(«не число»)*. Взгляните на пример ниже:

```
const stringNumber = '100';
console.log(+stringNumber); // 100
```

Приведение даты к числа, используя снова оператор +

Подобно тому, как оператор + может приводить *строковое число* к *числу*, вы также можете использовать этот же оператор, чтобы конвертировать объект `Date` в число, которое будет означать дату в миллисекундах. Данный хак может быть весьма полезным, когда нужно получить временную метку даты:

```
console.log(+new Date()); // 1410480611962
```

Примечание: Если вы хотите использовать более читабельную реализацию, нежели чем оператор +, вы всегда можете использовать `Number(new Date())`, `Date.now()` или `Date.prototype.getTime()`, чтобы получить миллисекунды с даты.

Округление чисел с помощью оператора ~~

Мне кажется, использование оператора ~~ для округления чисел выглядит странным решением, но тем не менее это альтернатива использованию функции `Math.floor()`:

```
console.log(Math.floor(10.99)); // 10
console.log(~~10.99); // 10
```

Проверка, является ли число нечётным или чётным

Для определения, является ли число чётным либо нечётным, можно использовать оператор `%`. Этот оператор отвечает за получение остатка от деления между двумя числами, и в случае, если остаток от числа, разделённого на 2, равен нулю, то число является чётным, а если оно равно единице, то соответственно число нечётное:

```
// Odd number
console.log(12 % 2 === 0); // true
// Even number
console.log(13 % 2 === 1); // true
```

Подсчёт возраста круче всех

Узнав про оператор `+`, как решение для преобразования типа `String` или `Date` в тип `Number`, а также применение странного оператора `~~` для округления числа, можно создать минималистическую функцию для вычисления возраста. И достичь этого можно всего за одну строчку производительного кода, посмотрите на пример ниже:

```
function calculateAge(date) {
  return ~~((Date.now() - date.getTime()) / 31557600000);
}
// Я пишу этот код, когда на дворе 2019/03/03
console.log(calculateAge(new Date(1990, 5, 4))); // 28
```

Примечание: А что означает число `31557600000`? Это всего лишь продолжительность года в миллисекундах. Если вам интересно узнать производитель этой функции с использованием этого магического числа, вы можете посмотреть результат по этой ссылке — jsperf.com/birthday-calculation[2]

[2] https://jsperf.com/birthday-calculation