Лекция №9

Modern JS

Strings

String.prototype.indexOf()

Сводка

Метод **`indexOf()`** возвращает индекс первого вхождения указанного значения в строковый объект `String`, на котором он был вызван, начиная с индекса `fromIndex`. Возвращает -1, если значение не найдено.

String.prototype.slice()

Сводка

Метод **`slice()`** извлекает часть строки и возвращает новую строку без изменения оригинальной строки.
Метод `slice()` извлекает текст из одной строки и возвращает новую строку. Изменения текста в одной строке не влияют на другую строку.

Метод `slice()` извлекает все символы до индекса *`endIndex`*, не включая сам этот индекс. Вызов *`str`*`.slice(1, 4)` извлечёт символы со второго по четвёртый (символы под индексами `1`, `2` и `3`).

К примеру, вызов `str.slice(2, -1)` извлечёт символы с третьего по второй с конца строки.

String.prototype.toLocaleLowerCase()

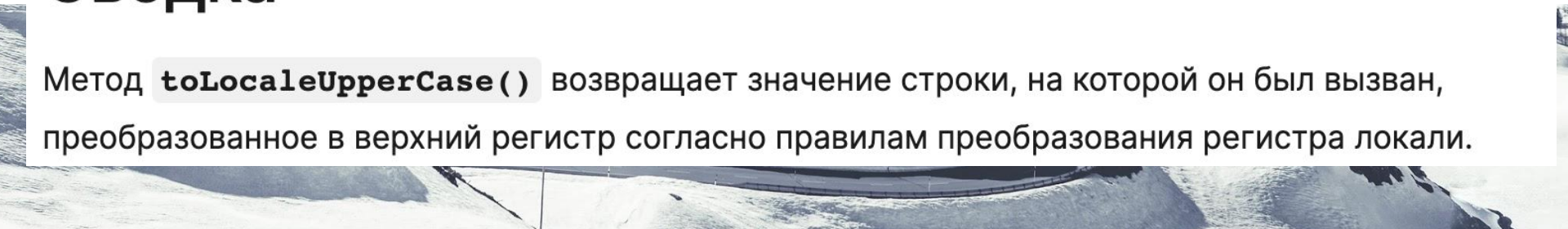
Сводка

Метод `toLocaleLowerCase()` возвращает значение строки, на которой он был вызван, преобразованное в нижний регистр согласно правилам преобразования регистра локали.

String.prototype.toLocaleUpperCase()

Сводка

Метод `toLocaleUpperCase()` возвращает значение строки, на которой он был вызван, преобразованное в верхний регистр согласно правилам преобразования регистра локали.



String.prototype.trim()

Сводка

Метод `trim()` удаляет пробельные символы с начала и конца строки. Пробельными символами в этом контексте считаются все собственно пробельные символы (пробел, табуляция, неразрывный пробел и прочие) и все символы конца строки (LF, CR и прочие).

String.prototype.replace()

Сводка

Метод `replace()` возвращает новую строку с некоторыми или всеми сопоставлениями с шаблоном, заменёнными на заменитель. Шаблон может быть строкой или регулярным выражением, а заменитель может быть строкой или функцией, вызываемой при каждом сопоставлении.

String.prototype.includes()

Метод **`includes()`** проверяет, содержит ли строка заданную подстроку, и возвращает, соответственно `true` или `false`.

String.prototype.startsWith()

Сводка

Метод **`startsWith()`** помогает определить, начинается ли строка с символов указанных в скобках, возвращая, соответственно, `true` или `false`.

String.prototype.endsWith()

Сводка

Метод **`endsWith()`** позволяет определить, заканчивается ли строка символами указанными в скобках, возвращая, соответственно, `true` или `false`.

String.prototype.split()

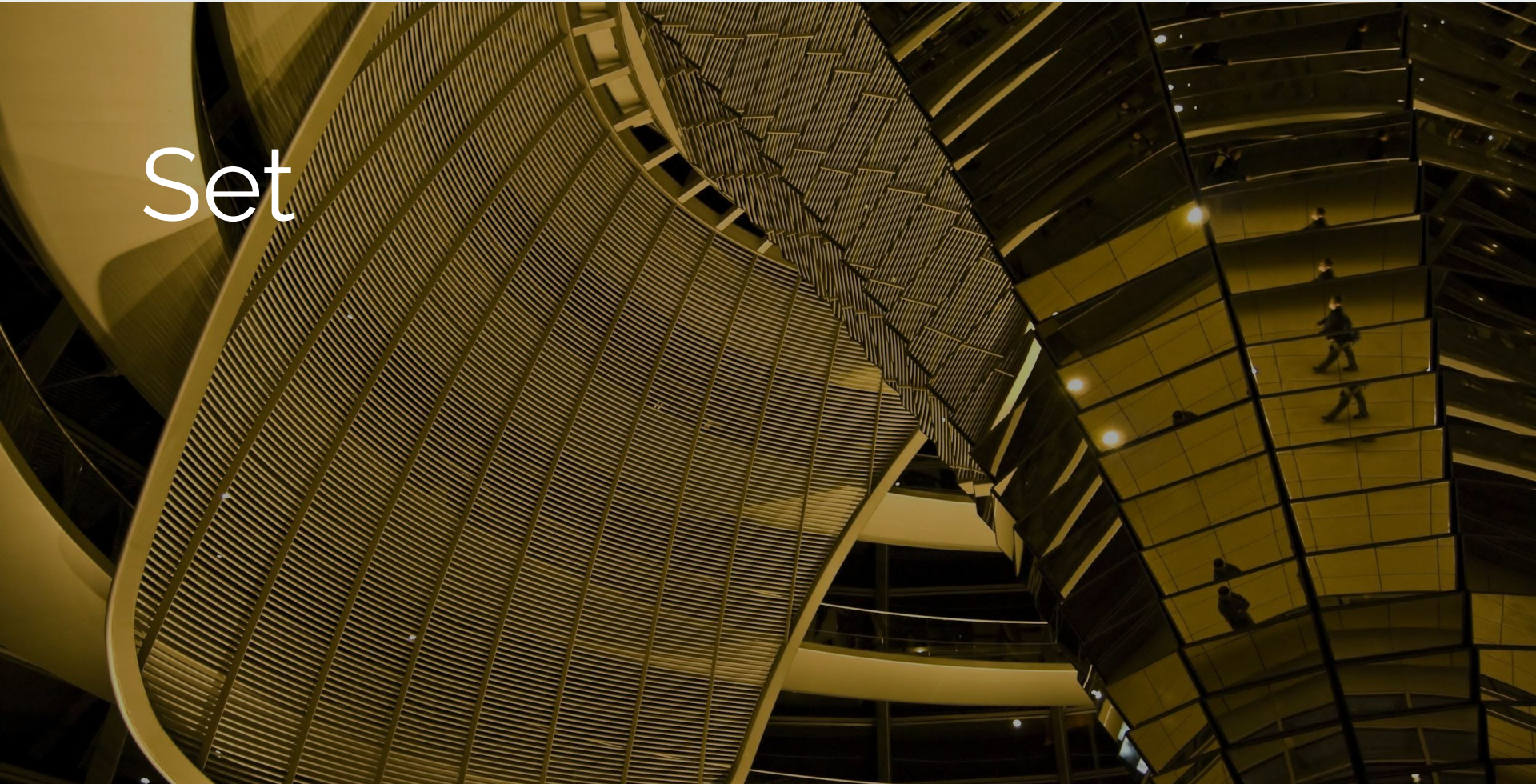
Сводка

Метод `split()` разбивает объект `String` на массив строк путём разделения строки указанной подстрокой.

Optional chaining

Оператор **опциональной последовательности** `?.` позволяет получить значение свойства, находящегося на любом уровне вложенности в цепочке связанных между собой объектов, без необходимости проверять каждое из промежуточных свойств в ней на существование. `?.` работает подобно оператору `.`, за исключением того, что не выбрасывает исключение, если объект, к свойству или методу которого идёт обращение, равен `null` или `undefined`. В этих случаях он возвращает `undefined`.

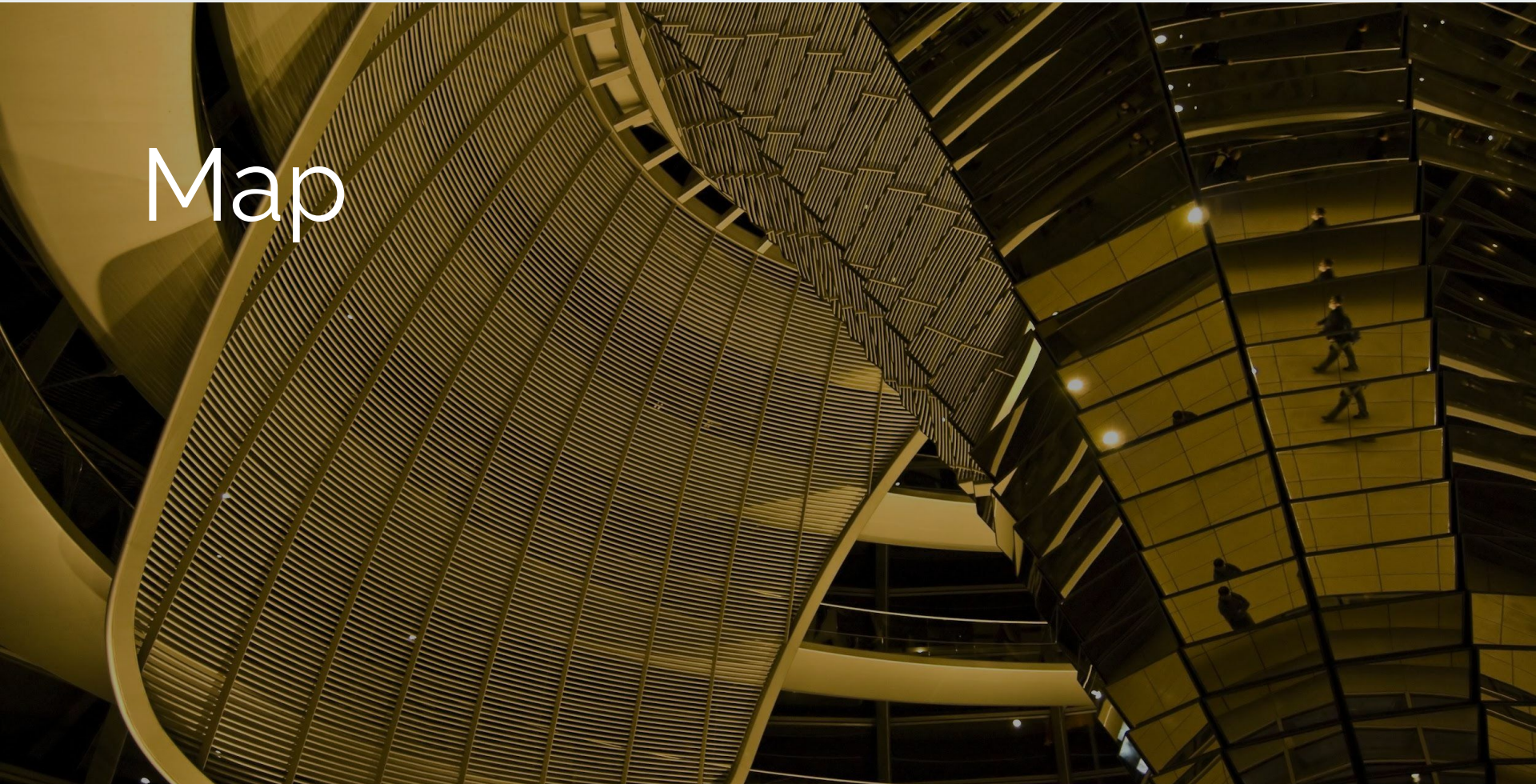
Таким образом, мы получаем более короткий и понятный код при обращении к вложенным по цепочке свойствам объекта, когда есть вероятность, что какое-то из них отсутствует.


Set

Описание

Объекты `Set` представляют коллекции значений, по которым вы можете выполнить обход в порядке вставки элементов. Значение элемента в `Set` может присутствовать **только в одном экземпляре**, что обеспечивает его уникальность в коллекции `Set`.

Map



Map

Map – это коллекция ключ/значение, как и `Object`. Но основное отличие в том, что `Map` позволяет использовать ключи любого типа.

Методы и свойства:

- `new Map()` – создаёт коллекцию.
- `map.set(key, value)` – записывает по ключу `key` значение `value`.
- `map.get(key)` – возвращает значение по ключу или `undefined`, если ключ `key` отсутствует.
- `map.has(key)` – возвращает `true`, если ключ `key` присутствует в коллекции, иначе `false`.
- `map.delete(key)` – удаляет элемент по ключу `key`.
- `map.clear()` – очищает коллекцию от всех элементов.
- `map.size` – возвращает текущее количество элементов.


LocalStorage,
sessionStorage

LocalStorage, sessionStorage

Объекты веб-хранилища `localStorage` и `sessionStorage` позволяют хранить пары ключ/значение в браузере.

Что в них важно – данные, которые в них записаны, сохраняются после обновления страницы (в случае `sessionStorage`) и даже после перезапуска браузера (при использовании `localStorage`). Скоро мы это увидим.

Демо localStorage

Основные особенности `localStorage`:

- Этот объект один на все вкладки и окна в рамках источника (один и тот же домен/протокол/порт).
- Данные не имеют срока давности, по которому истекают и удаляются. Сохраняются после перезапуска браузера и даже ОС.

Например, если запустить этот код...

```
1  localStorage.setItem('test', 1);
```

Только строки

Обратите внимание, что ключ и значение должны быть строками.

Если мы используем любой другой тип, например число или объект, то он автоматически преобразуется в строку:

```
1 sessionStorage.user = {name: "John"};
2 alert(sessionStorage.user); // [object Object]
```

sessionStorage

Объект `sessionStorage` используется гораздо реже, чем `localStorage`.

Свойства и методы такие же, но есть существенные ограничения:

- `sessionStorage` существует только в рамках текущей вкладки браузера.
  - Другая вкладка с той же страницей будет иметь другое хранилище.
  - Но оно разделяется между ифреймами на той же вкладке (при условии, что они из одного и того же источника).
- Данные продолжают существовать после перезагрузки страницы, но не после закрытия/открытия вкладки.

Событие storage

Когда обновляются данные в `localStorage` или `sessionStorage`, генерируется событие storage со следующими свойствами:

- `key` – ключ, который обновился (`null`, если вызван `.clear()`).
- `oldValue` – старое значение (`null`, если ключ добавлен впервые).
- `newValue` – новое значение (`null`, если ключ был удалён).
- `url` – url документа, где произошло обновление.
- `storageArea` – объект `localStorage` или `sessionStorage`, где произошло обновление.

Важно: событие срабатывает на всех остальных объектах `window`, где доступно хранилище, кроме того окна, которое его вызвало.

```
window.onstorage = event => {
  if (event.key != 'now') return;
  alert(event.key + ':' + event.newValue + " at " + event.url);
};

localStorage.setItem('now', Date.now());
```

Полезные ссылки

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Operators/Optional_chaining
https://learn.javascript.ru/localstorage

Спасибо!