Полуэктова, Н. Р. Разработка веб-приложений: учебное пособие для вузов / Н. Р. Полуэктова. — Москва: Издательство Юрайт, 2023. — 204 с. — (Высшее образование). — ISBN 978-5-534-13715-6. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/519714

5.5. Строки и массивы JavaScript

Для создания программ на любом современном языке программирования уже недостаточно просто переменных. Все языки «умеют» работать с более сложными типами данных, которые объединяют информацию в массивы, объекты, перечни, последовательности и т. д., и содержат средства эффективной обработки таких сложных структур. Здесь мы ограничимся только рассмотрением строк и массивов JavaScript.

В JavaScript любые текстовые данные являются строками. Не существует отдельного типа «символ», который есть в ряде других языков. Все символы в строке кодируются в формате UTF-16, независимо от кодировки страницы.

Строка в JavaScript — это объект, который имеет множество полезных свойств и методов.

Свойство ***length*** определяет длину строки:

```
let str = 'Hello, world! ';
alert(str.length); // выведет 13
```

Свойство записывается БЕЗ круглых скобок!

Получить символ, который занимает позицию N, можно с помощью квадратных скобок: **[N]**. Первый символ имеет номер 0.

```
let str = 'Hello';
alert( str[0] ); // получаем первый символ H
```

```
// получаем последний символ
alert( str[str.length - 1] ); // о
```

Содержимое строки в JavaScript нельзя изменить. Изменить строку можно только при ее замене с тем же именем.

```
let str = 'Да';
str[0] = 'д'; // ошибка
str = 'д' + str[1]; // заменяем строку
```

Методы ***toLowerCase()*** и ***toUpperCase()*** меняют регистр символов:

```
alert( 'Hello, World!'.toUpperCase() ); // HELLO, WORLD!
alert( "Hello, World!'.toLowerCase() ); // hello, world
```

Для ***поиска подстроки*** (определения того, есть ли такое включение символов в исходной строке) можно использовать метод ***str.indexOf(substr, N)***. Он начинает искать substr в строке str, начиная с позиции N, и возвращает позицию, на которой располагается совпадение, либо -1 при отсутствии совпадений.

В следующем примере в строке «абракадабра» выполняется поиск индексов, начиная с которых в ней содержится строка «абра».

```
let str = 'абракадабра';

let target = 'абра'; // цель поиска

let N = 0; // позиция с которой начинаем поиск
while (true) // цикл, который позволит пройти по всей строке
{
  let N1 = str.indexOf(target, N); // сохраняем позицию
  if (N1 == -1) {
    // если не было включений, выходим из цикла
    break;
  }
  else {
    alert( 'Найдено тут: ${N1}' );
    // если было найдено совпадение, выводим позицию
    N = N1 + 1;
    // продолжаем с позиции, которая следует после заданной
  }
}
```

В JavaScript есть несколько методов для получения подстроки, например, ***substr*** и ***slice***.

Метод slice позволяет «вытащить» символы из исходной строки от начальной позиции до конечной (не включая символ конечной позиции).

```
let str = "абракадабра";
alert( str.slice(1, 5) ); // 'брак', символы от 1 до 5
(не включая 5)
```

Если второй параметр не задан, выбираются все символы до конца строки.

Метод substr позволяет «вытащить» символы из исходной строки от начальной позиции в количестве L.

```
let str = "абракадабра";
alert( str.substr(2, 3) ); // рак, получаем 3 символа,
начиная с позиции 2
```

Задача обработки строковых выражений, которые вводит пользователь на странице является очень распространенной и важной. Поэтому перед началом обработки, во избежание ошибок, необходимо проверить, не является ли строка пустой.

Пример. Пусть необходимо написать функцию, которая переводит первую букву введенной строки в верхний регистр:

```
function ucFirst(str) {
  if (!str) {
    return "Строка пуста";
  }
  else {
    return str[0].toUpperCase() + str.slice(1);
  }
}
let str = "вася";
alert( ucFirst(str) ); // результат: Вася
```

Также, как в любом другом языке высокого уровня, в JavaScript **массивы** (**array**) используются для хранения и обработки наборов каких-либо однородных данных: пользователей, товаров, элементов HTML и т. д.

Для создания нового пустого массива может использоваться конструкция:

```
let arr1 = []; // Пустой массив с именем arr1
let cities = ['Москва', 'Тула', 'Сочи'];
// Массив с названиями городов
```

К отдельным элементам массива обращаются по индексам (первый элемент массива имеет индекс 0):

```
alert(cities[0]); // Результат: Москва
```

Можно изменить любой из элементов массива:

```
cities[2] = 'Калуга'; // Теперь ['Москва', 'Тула', 'Калуга']
```

Можно добавить элемент к имеющемуся массиву:

```
cities[3] = 'Орел'; // Теперь ['Москва', 'Тула', 'Калуга','Орел']
```

Количество элементов массива получают через свойство ***length***:

```
alert(cities.length ); // Результат: 4
```

Наиболее распространенный алгоритм работы с массивом — это ***перебор его значений*** в цикле:

```
let cities = ['Москва', 'Тула', 'Сочи'];
for (let i = 0; i < cities.length; i++)
{
  alert( cities[i] );
}
```

Контрольные вопросы

1. Как кодируются символы в строке в JS?
2. Какие свойства и методы строк использовались в приведенном тексте?

   Ответ: свойство str.length; методы str.toLowerCase(), str.toUpperCase(), str.indexOf(substr, N), str.substr(n, m), str.slice(n,m)

   Прокомментируйте все названные свойства и методы.
3. Как получить элемент строки?
4. Какова позиция последнего символа строки?
5. Можно ли изменить содержимое строки в JS?

   Ответ: нельзя, можно только заменить ее с тем же именем.
6. Написать функцию, которая переводит первый символ строки в верхний регистр.
7. Как задать массив строк?
8. Как получить элемент массива? Как изменить элемент массива? Как добавить элемент в массив?
9. Как перебрать элементы массива?

№	Синтаксис метода или свойства	Пример
1	**length** – длина строки	
2	Получение символа в заданной позиции *N* из строки *str*: str[N]	
3	Получение последнего символа строки:	
4	**toLowerCase()** – перевод символов в верхний регистр. **toUpperCase()** – перевод символов в нижний регистр.	
5	**indexOf(substr,N)** – поиск *позиции* подстроки substr в данной строке; возвращает -1, если подстрока не найдена.	
6	Методы получения подстроки *substr* из исходной строки *str*: – **slice(n, m)** – возвращает подстроку, начиная с позиции *n* до позиции *m* (не включая); если *m* не задано, то до конца строки*;* – **substr(n, L)** – возвращает подстроку, начиная с позиции *n* длиной *L*;	
7	Перебор элементов массива в цикле for:	