НЕТОЛОГИЯ

БАЗОВЫЙ СИНТАКСИС

АЛЕКСЕЙ СУДНИЧНИКОВ

АЛЕКСЕЙ СУДНИЧНИКОВ

Веб-разработчик

alsdew@ya.ru

@avsudnichnikov

ПЛАН ЗАНЯТИЯ

1. Переменные и константы
2. Выражения, операторы и операнды
3. DevTools и `console.log`
4. Числа
5. Строки

ПЕРЕМЕННЫЕ И КОНСТАНТЫ

ПЕРЕМЕННЫЕ

Вы уже знакомы с переменными, которые используются в программировании для хранения информации.

Настало время вспомнить работу с переменными в JS и рассмотреть ограничения при работе с переменными.

ВСПОМНИМ ПРОШЛЫЕ ЗАНЯТИЯ

Ответьте на следующие вопросы:

- Что представляет из себя переменная на ваш взгляд?
- Что представляет из себя процесс объявления переменной?
- С помощью какого ключевого слова происходит объявление переменных?
- Как присвоить переменной `age` значение `25`?
- Можно ли объявить две переменные с одинаковым именем?

ПЕРЕМЕННЫЕ

Переменная — именованный кусочек памяти, который содержит в себе изменяемое значение.

Значение переменной может быть изменено в ходе выполнения программы.

Далее рассмотрим переменные подробнее.

ОБЪЯВЛЕНИЕ И ПРИСВАИВАНИЕ ЗНАЧЕНИЯ ПЕРЕМЕННЫМ

Объявление переменной с помощью ключевого слова `let`, за которым следует название переменной:

```
let age;
```

Использование оператора `=` для присваивания значения переменной:

```
age = 25;
```

Объявление и присваивание значения переменной в одной строке:

```
let year = 1970;
```

ИЗМЕНЕНИЕ ЗНАЧЕНИЯ ПЕРЕМЕННОЙ

```
let speed = 100;
speed = 120;
```

В первой строке объявлена переменная `speed` со значением `100`, а во второй значение переменной `speed` изменено со `100` на `120`.

ОБЪЯВЛЕНИЕ НЕСКОЛЬКИХ ПЕРЕМЕННЫХ С ОДНИМ ИМЕНЕМ

Запрещено объявлять переменные с одинаковыми именами.

Например:

```
let age = 25;
let age = 30;
```

Результатом в данном случае будет ошибка, которая известит вас о том, что переменная `age` уже существует.

КОНСТАНТЫ

Константа — именованный кусочек памяти, который содержит в себе неизменяемое значение.

Значение константы не может быть изменено в ходе выполнения программы.

Далее рассмотрим константы подробнее.

ОБЪЯВЛЕНИЕ И ПРИСВАИВАНИЕ ЗНАЧЕНИЯ КОНСТАНТАМ

Объявление константы с помощью ключевого слова `const`, за которым следует название константы и присвоение значения:

```
const days = 365;
```

Объявление константы без присвоения значения вызовет ошибку.

```
const days; // ОШИБКА
```

Попытка изменения константы также вызовет ошибку.

```
const days = 365;
days = 364; // ОШИБКА
```

КОНСТАНТА И ОБЪЕКТЫ

Константы накладывают ограничения **только** на присваивание значения, то есть оператор `=` и все производные (`+=`, `-=`, `/=` и так далее).

```
1  const ages = [40, 42];
2  ages.push(50);
3
4  console.log(ages); // [40, 42, 50];
```

Это тот же самый экземпляр массива, которые изменился только **внутри**. Присванивание не вызывалось, потому ошибка не случилась.

ТРЕБОВАНИЯ К ИМЕНАМ

- Имя может состоять из букв, цифр, символов `$` и `_`;
- Первый символ не должен быть цифрой;
- Регистр символов имеет значение. `time`, `Time` и `TIME` — разные переменные.

Зарезервированные ключевые слова нельзя использовать в качестве имени переменной. Например, мы не можем создать переменную с именем `let` или `const`.

СОГЛАШЕНИЕ ОБ ИМЕНОВАНИИ

- Используйте английский язык при выборе имени переменной или константы. Никакого транслита.
- Используйте `нижнийВерблюжийРегистр` (`lowerCamelCase`) для переменных, название которых состоит из нескольких слов. В нижнем верблюжьем регистре первое слово пишется с маленькой буквы.
- Самое главное правило: имя переменной должно четко отражать информацию, которая хранится в переменной и быть понятным не только автору кода.

Определение camelCase

В нижнем верблюжьем регистре первое слово пишется с маленькой буквы. Например: `myLetiable`.

Помимо соглашения camelCase также существуют kebab-case, snake-case и другие, но в рамках данного курса мы будем использовать camelCase.

ВЫРАЖЕНИЯ, ОПЕРАТОРЫ И ОПЕРАНДЫ

ВСПОМНИМ ПРОШЛЫЕ ЗАНЯТИЯ

Ответьте на следующие вопросы:

- Для чего предназначены выражения в программировании?
- Какие операторы вы помните?

ВЫРАЖЕНИЕ

Выражение / команда / операция – «атомы», из которых состоят программы. Записав команды в определенной последовательности (при условии, что они понятны интерпретатору) мы получим программу, которая дает желаемый результат.

Интерпретатор — специализированная программа, считывающая и исполняющая исходный код на JavaScript.

ОПЕРАТОР И ОПЕРАНД

Оператор — это команда, которая на письме выглядит как простой символ. С помощью операторов выполняются некоторые действия над данными. Данные, расположенные справа и слева от оператора называются **операндами**.

Операторы бывают:

- Унарные (англ. unary);
- Бинарные (англ. binary);
- Тернарные (англ. ternary).

На следующих слайдах рассмотрим каждый тип подробнее.

УНАРНЫЕ ОПЕРАТОРЫ

Унарные — оперируют только левым или правым операндом.

```
let temp = -10;
```

Унарный минус `-` делает число отрицательным.

В нашем случае `10` является операндом по отношению к унарному минусу.

БИНАРНЫЕ ОПЕРАТОРЫ

Бинарные — оперируют левым и правым операндами.

```
let distance = 1056;
```

Используется бинарный оператор присваивания `=`, а в качестве операндов выступают `distance` и `1056`.

ОПЕРАТОР ПРИСВАИВАНИЯ

Оператор присваивания = бинарный.

С его помощью правый операнд присваивается переменной в левом операнде.

Выражение справа от оператора присваивания сначала будет вычислено, а затем его результат будет присвоен переменной.

```
let time = 1056 / (112 + 73);
let distance = 112 * time;
```

ТЕРНАРНЫЙ ОПЕРАТОР

Единственный в своём роде **тернарный** оператор, который оперирует сразу тремя операндами.

```
let access = true;
let output = access ? 'Доступ разрешен' : 'Доступ запрещен';
```

Три операнда тернарного оператора:

- условие (access);
- выражение 1 ('Доступ разрешен');
- выражение 2 ('Доступ запрещен');

Подробный разбор данного оператора будет произведен в дальнейших лекциях. На данном этапе важно понять, что тернарный оператор взаимодействует сразу с тремя операндами.

МАТЕМАТИЧЕСКИЕ ОПЕРАТОРЫ

Бинарные операторы:

- Оператор сложения `+`;
- Оператор деления `/`;
- Оператор умножения `*`;
- Оператор вычитания `-`;
- Оператор взятия остатка `%`.

Унарный оператор:

- Оператор отрицания `-`.

КОММЕНТАРИИ

В процессе разработки может появиться потребность в том, чтобы добавить комментарий к определённым строкам кода.

Комментарии призваны пояснить, что происходит на описываемом участке кода и почему. Они не оказывают никакого влияния на выполнение программы и могут находиться в любом месте программы.

Однострочный комментарий распространяется от места объявления и до конца строки. Начинается с двойного слэша `//`:

```
// Однострочный комментарий
```

КОММЕНТАРИИ

Многострочные комментарии начинаются слешем-звездочкой `/*` и заканчиваются звездочкой-слэшем `*/`:

```
/*
  Первая строка
  Вторая строка
*/
```

В примерах кода будем использовать однострочный комментарий, для описания того, что будет выводить комментируемая строка.

DEVTOOLS И `console.log`

DEVTOOLS И `console.log`

С целью запуска команд JavaScript, анализа выполняемого кода, поиска ошибок и вывода информации используется консоль браузера.

Консоль находится в компоненте браузера под названием *инструменты разработки* (DevTools).

Для разработки рекомендуется использовать Chrome или Firefox.

ЗАДАЧА

Представьте, что вам необходимо с нуля создать новую страницу и написать к ней скрипт, который будет выполнять определённые действия.

Начинается всё с создания соответствующих файлов. В нашем случае будут созданы:

- `index.html`
- `main.js`

Далее планируется произвести подключение скрипта и проверку произведенного подключения.

С проверкой подключения скрипта нам поможет консоль DevTools.

ПОДГОТОВКА

Для знакомства с DevTools создадим файл `index.html`, со стандартной разметкой. В том же каталоге создадим пустой файл `main.js` и подключим данный скрипт к нашей странице.

В итоге файл `index.html` будет содержать следующий HTML код:

```
<html lang="en">
  <head>
    <title>Document</title>
  </head>
  <body>
    <script src="main.js`"></script>
  </body>
</html>
```

Строка `<script src="main.js"></script>` подключает скрипт `main.js` из текущего каталога к нашей странице.

Теперь можно писать JavaScript код в файле `main.js` и выполнять его отладку с помощью DevTools.

ПИШЕМ JAVASCRIPT

Откроем файл `main.js` и напишем одну строку кода:

```
console.log(100);
```

Код `console.log(100)` выводит число 100 в консоль.

Число выбрано случайное. Можете выбрать для себя любое другое. Основная идея в том, чтобы проверить вывод в консоль.

Далее рассмотрим работу с консолью.

КОНСОЛЬ DEVTOOLS

Файлы `index.html` и `main.js` готовы.

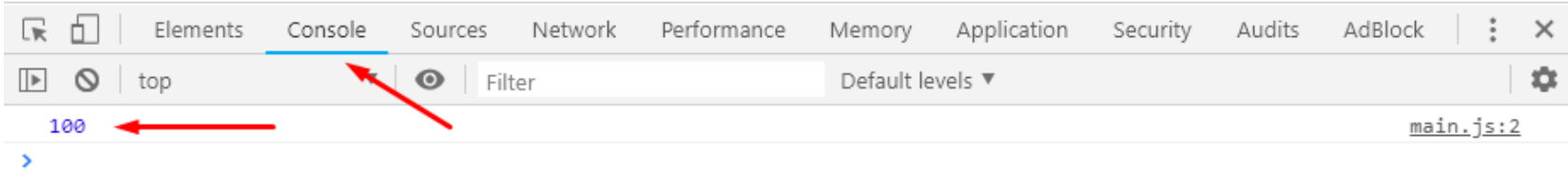
Проверим подключился ли наш скрипт. Ожидаем в консоли увидеть число `100`.

Для этого:

1. Откроем файл `index.html` в браузере;
2. Запустим DevTools. Для этого используйте клавишу `F12` под Windows, а если у вас Mac, то `Cmd+Opt+J`;
3. Выберем вкладку «Console».

КОНСОЛЬ DEVTOOLS

На картинке представлен результат выполненных действий



Консоль нам любезно вывела то, что мы от неё ожидали.

Для вывода информации и отладки кода, вы будите обращаться к консоли DevTools на протяжении всего вашего обучения на данном курсе.

ЗАПУСК ИЗ NODE.JS

А вы знали, что JavaScript код можно запускать не только в браузере?

Для запуска кода в среде Node следует выполнить команду `node ./main.js` (`./main.js` - путь к исполняемому файлу). Если файл находится в текущей директории, то можно использовать `node main.js`.

ЧИСЛА

ЧИСЛА

Этот тип данных встречается очень часто при написании программ. Наша задача о поездах из раздела «Выражения, операторы и операнды» не исключение.

Числа записываются просто цифрами. Числа с плавающей точкой имеют целую и дробную части.

```
let width = 100;
let height = 33.33;
console.log(width); // 100
console.log(height); // 33.33
```

ЗАДАЧА

Рассмотрим простую задачу.

Поезд из Москвы движется со средней скоростью 112 км/ч. Поезд из Самары движется навстречу первому со средней скоростью 73 км/ч. Через какое время они встретятся, если расстояние между Москвой и Самарой 1056 км? На каком расстоянии от Москвы это случится?

РЕШЕНИЕ

Школьная программа толкает нас на мысль применить математические методы вычисления для решения задачи.

Запишем *выражения* для решения:

— Рассчитаем время до встречи:

```
1056 / (112 + 73)
```

— Найдем расстояние от Москвы:

```
112 * (1056 / (112 + 73))
```

ОФОРМЛЕНИЕ КОДА

Для улучшения читаемости кода до и после оператора ставится пробел:

```
let trainMscSpeed = 112;
let trainSmrSpeed = 73;
let distance = 1056;
let time;
time = distance / (trainMscSpeed + trainSmrSpeed);
```

Полное описание стандартов оформления JavaScript кода смотрите по ссылке.

СТРОКИ

ВСПОМНИМ ПРОШЛЫЕ ЗАНЯТИЯ

Ответьте на следующие вопросы:

- Чем нужно обернуть текст, чтобы интерпретировать его как строку?
- С помощью какого специального символа можно указать интерпретатору на перевод строки?
- Что такое конкатенация строк?
- Как можно получить число, указывающее на длину строки?

СТРОКИ

Любой текст, заключенный в одинарные (`' '`) или двойные (`" "`) кавычки, интерпретируется как строка.

На ваше усмотрение вы можете выбрать для себя одинарные или двойные кавычки. Разницы для интерпретатора не будет.

Однако не рекомендуется использовать одновременно оба стиля для описания строк.

СТРОКИ

Добавим к нашим цифрам текстовые сообщения:

```
let trainMscSpeed = 112;
let trainSmrSpeed = 73;
let distance = 1056;
let time = distance / (trainMscSpeed + trainSmrSpeed);
let distanceFromMsc = trainMscSpeed * time;

console.log('Поезда встретятся через');
console.log(time);
console.log('часов');
console.log('Встреча произойдет в');
console.log(distanceFromMsc);
console.log('км');
console.log('от Москвы');
```

СТРОКИ

Результат:

```
Поезда встретятся через
5.708108108108108
часов
Встреча произойдет в
639.3081081081082
12 км
от Москвы
```

КОНКАТЕНАЦИЯ СТРОК

Для конкатенации (объединения) двух или более строк в одну в JavaScript используется оператор `+`.

Используем конкатенацию строк в нашей задаче:

```
let trainMscSpeed = 112;
let trainSmrSpeed = 73;
let distance = 1056;
let time = distance / (trainMscSpeed + trainSmrSpeed);
let distanceFromMsc = trainMscSpeed * time;
console.log('Поезда встретятся через ' + time + ' часов');
console.log('Встреча произойдет в ' + distanceFromMsc + ' км от Москвы');
```

КОНКАТЕНАЦИЯ СТРОК

Результат:

```
Поезда встретятся через 5.708108108108108 часов
Встреча произойдет в 639.3081081081082 км от Москвы
```

СТРОКОВЫЕ ШАБЛОНЫ

Шаблоны — это строки, в которые можно подставить значения переменных.
Для этого используется новый тип кавычек и особый синтаксис:

```
${имяПеременной}
```

Обратите внимание, что для шаблонов используются апострофы `` ` ` ``, а не одинарные кавычки `' '`

СТРОКОВЫЕ ШАБЛОНЫ

Улучшим решение нашей задачи с помощью строковых шаблонов:

```
let trainMscSpeed = 112;
let trainSmrSpeed = 73;
let distance = 1056;
let time = distance / (trainMscSpeed + trainSmrSpeed);
let distanceFromMsc = trainMscSpeed * time;
console.log(`Поезда встретятся через ${time} часов`);
console.log(`Встреча произойдет в ${distanceFromMsc} км от Москвы`);
```

Можно заметить, что код стал более читабелен, чем в варианте с конкатенацией.

ВЫПОЛНЕНИЕ ДОМАШНЕГО ЗАДАНИЯ

- Все домашние задания сдаются через Github.
- Создавать свой репозиторий не нужно.

Нужно сделать форк репозитория с домашними заданиями.

- Все домашние задания структурированы по соответствующим папкам.
- ~~**Важно**~~ **Полезно!!!** Перед домашним заданием ознакомьтесь с отладкой кода.
- Для каждого задания присутствуют тесты, проверяющие правильность вычислений заданий. Откройте файл `test-runner.html` для запуска тестов.
- Все тесты должны корректо выполняться.

ЧЕМУ МЫ НАУЧИЛИСЬ?

1. Переменные и ограничения, связанные с их применением;
2. Операторы, операнды и комментарии, понятия и назначение;
3. Подключение скрипта к странице;
4. Использование консоли DevTools;
5. Числа, `NaN`, `Infinity` и математические операторы;
6. Строки, строковые шаблоны.

ДОМАШНЕЕ ЗАДАНИЕ

Давайте посмотрим ваше домашнее задание.

- Вопросы по домашней работе задаем в чате Slack!
- Задачи можно сдавать по частям.
- Зачет по домашней работе проставляется после того, как приняты **все задачи**.

НЕТОЛОГИЯ

Спасибо за внимание!

Время задавать вопросы ☺

АЛЕКСЕЙ СУДНИЧНИКОВ

 alsdew@ya.ru

 @avsudnichnikov