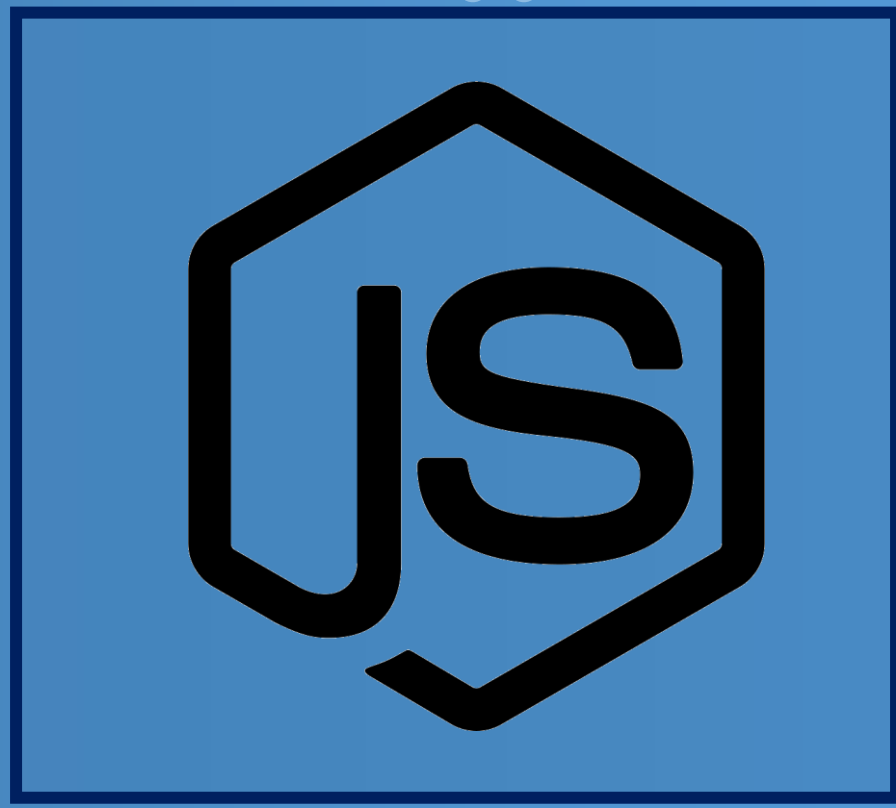
STEP IT ACADEMY

Занятие №1

Введение, переменные и условия



ЧТО МЫ ЗНАЕМ

1. КАКИЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ ВЫ ЗНАЕТЕ ?
2. ИЗ ЧЕГО СОСТОИТ СТРАНИЦА HTML ?
3. ЧТО ТАКОЕ АТРИБУТЫ?
4. ЧТО ТАКОЕ CSS ?

HTML + CSS + JAVASCRIPT

Когда создался язык JavaScript, он предназначался для динамичности web страницы.

То есть страница так же была написана на HTML со стилем CSS, но теперь к ним добавился кусочек кода, который был написан на JavaScript

LIVESCRIPT -> JAVASCRIPT

Первоначально язык назывался **LiveScript**, но на волне популярности в тот момент другого языка **Java LiveScript** был переименован в **JavaScrip**.

Однако данный момент до сих пор иногда приводит к некоторой путанице: некоторые начинающие разработчики считают, что **Java** и **JavaScript** чуть ли не один и тот же язык. Нет, это абсолютно два разных языка, и они связаны только по названию.

JAVASCRIPT в html

Внедрить код сценария в HTML документ можно двумя способами.

- Первый — указать теги в теле документа и между ними поместить код.
- Второй — написать код в виде отдельного файла (например, file.js) и подключить его в документ, указывая файл-источник .

ОТДЕЛЬНЫЙ ФАЙЛ JS

Вынесенный код JS имеет ряд преимуществ:

- Можем **повторно использовать**
- Внешние файлы JavaScript бразуер **может кэшировать**
- Код веб-страницы становится "**чище**".

ГДЕ ПИСАТЬ ПРОГРАММЫ НА JAVASCRIPT

Для написания и тестирования программ на JavaScript нам потребуются две вещи: **текстовый редактор** и **веб-браузер**.

В качестве текстового редактора можно взять любой, который нравится - Atom, Sublime Text, Visual Studio Code, Notepad++ и другие.

Я буду использовать **Visual Studio Code**, поскольку он удобный.

В качестве браузера также можно взять последние версии любого предпочтительного веб-браузера.

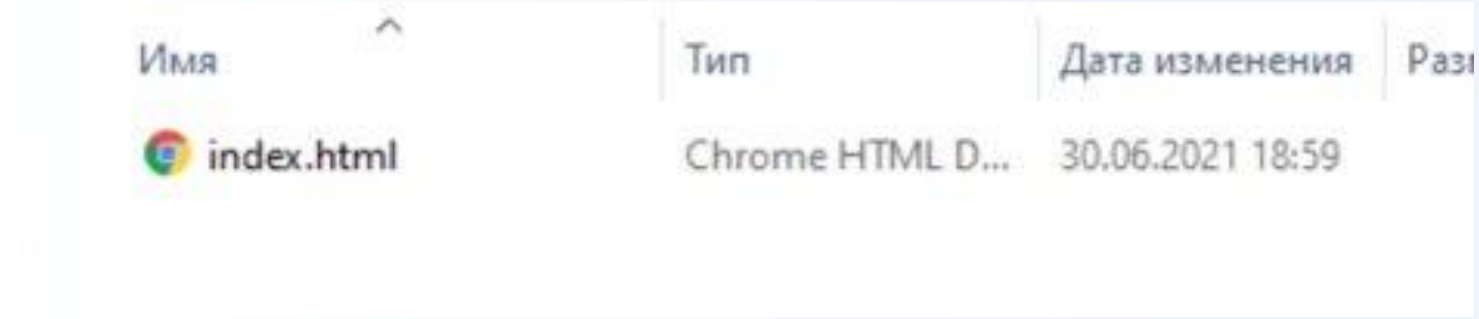
Я буду использовать **Google Chrome**.

ПЕРВОЕ ПРИЛОЖЕНИЕ

Для начала откройте на своих компьютерах программу Visual Studio Code.

Далее нажмите создать новый файл (New File..)

Сохраните его под названием "index" на рабочем столе с расширением HTML.



Внутри body пишем тег script

```
<body>
    <script>
        document.write("текст на старнице")
    </script>
</body>
```

Сохраняем и запускаем

ОСНОВЫ СИНТАКСИСА

Обязательно ставьте в конце строки точку с запятой!

Язык JavaScript относится к **регистрозависимым**.

Переменные name и Name будут считаться разными.

Также нельзя давать переменным такие имена, которые совпадают с зарезервированными ключевыми словами (пройдем их ниже).

Кроме исполнимого кода в состав программы включаются еще и авторские примечания — комментарии. В этом языке они ставятся двойным слэшом - **//** , а для больше части кода используется **/**/**.

КЛЮЧЕВЫЕ И ЗАРЕЗЕРВИРОВАННЫЕ СЛОВА

В языках программирования также существуют слова, смысл которых установлен их разработчиками. Такие слова называют ключевыми.

await, break, case, catch, class, const, continue, debugger, default, delete, do, else, export, extends, finally, for, function, if, import, in, instanceof, let, new, return, static, super, switch, this, throw, try, typeof, var, void, while, with, yield.

ПЕРЕМЕННЫЕ.
ПРАВИЛА ИМЕНОВАНИЯ ПЕРЕМЕННЫХ

Принципы и ограничения, накладываемые на возможные имена переменных, называют правилами именования.

- Для имен переменных запрещается использование ключевых или зарезервированных слов
- В именах переменных не допускаются пробельные и разделительные символы
- Имена переменных не могут содержать символы операций
- В именах переменных не допускаются специальные символы операторов, например «.», «{}» или «?:», а также кавычки всех типов.
- В именах переменных допускаются только «_» (нижнее подчеркивание) и «$» (знак доллара).
- Имя переменной не может начинаться с цифры

ОБЪЯВЛЕНИЕ ПЕРЕМЕННЫХ

Для создания переменных применяются операторы **var** и **let**.

var аналогичное определение переменной с помощью оператора **let**

Через запятую можно определить сразу несколько переменных

```
var username, age, heigt;
let a, b, c;
```

```
var username;
```

```
let username;
```

```
$commision
someVariable
product_Store
income2
myIncome_from_deposit
```

Помним, что можно, а что нельзя прописывать в названиях переменных!

ОБЪЯВЛЕНИЕ КОНСТАНТ

Для создания переменных применяются операторы **var** и **let**.

Значения обычных переменных можно изменять в любой части кода.

Но есть переменные, которые называются КОНСТАНТЫ.

Их значение присваивается только ОДИН раз и не изменяется никогда

```
const username = "Tom";
```

```
$commision
someVariable
product_Store
income2
myIncome_from_deposit
```

ТИПЫ ДАННЫХ

- **String**: представляет строку
- **Number**: представляет числовое значение
- **BigInt**: предназначен для представления очень больших целых чисел
- **Boolean**: представляет логическое значение **true** или **false**
- **Undefined**: представляет одно специальное значение - **undefined** и указывает, что значение не установлено
- **Null**: представляет одно специальное значение - **null** и указывает на отсутствие значения
- **Symbol**: представляет уникальное значение, которое часто применяется для обращения к свойствам сложных объектов
- **Object**: представляет комплексный объект

«UNDEFINED» И «NULL»

Тип «**Undefined**» имеет только одно значение «**undefined**». Этот тип имеет любая переменная, которой еще не было присвоено значение.

Тип «**Null**» также имеет единственное значение «**null**». Это значение применяется там, где ожидается получение объекта, но по каким-либо причинам данный объект не был получен. Другими словами, переменная была создана, но значение в ней отсутствует. Если «**undefined**» соотносится с переменной, которая не участвовала в операции присваивания, то «**null**» возникает в результате неудачной операции присваивания

«NUMBER»

Тип «**Number**» предназначен для хранения чисел. Стандартом установлено, что этот тип имеет ровно 18437736874454810627 различных значений, представляющих число двойной точности в формате 64 бит. Кроме самих чисел среди этих значений есть несколько специальных:

- «**NaN**» (Not-a-Number) применятся при невозможности преобразовать результат к числу, по сути, обозначает ошибку;

ПРИСВОЕНИЕ ПЕРЕМЕННОЙ ЗНАЧЕНИЯ

Процесс присвоения переменной начального значения называется **инициализацией**.

После определения переменной ей можно присвоить какое-либо значение. Для этого применяется **оператор присваивания** (=)

```
var username;
username = "Tom";
```

Можно сразу присвоить переменной значение при ее определении:

```
var username = "Tom";
let userage = 37;
```

МАТЕМАТИЧЕСКИЕ ОПЕРАЦИИ

Сложение:

```
let x = 10;
let y = x + 50;
```

Вычитание:

```
let x = 100;
let y = x - 50;
```

Деление по модулю (оператор %) возвращает остаток от деления:

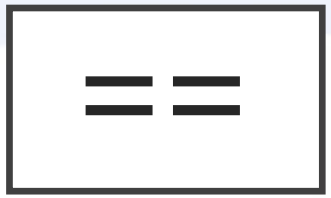
```
let x = 5;
let y = 2;
let z = x % y;
console.log(z); // 1
```

Умножение:

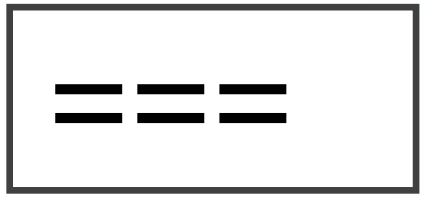
```
let x = 4;
let y = 5;
let z = x * y;
```

Деление:

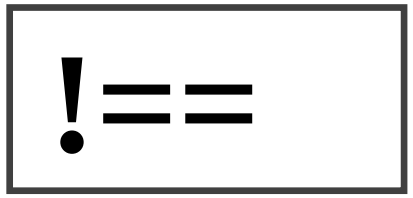
```
let x = 5;
let y = 2;
let z = x / y;
console.log(z); // 2.5
```



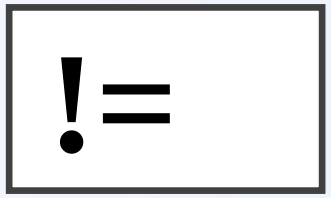
Оператор равенства сравнивает два значения, и если они равны, возвращает true, иначе возвращает false



Оператор тождественности также сравнивает два значения и их тип, и если они равны, возвращает true, иначе возвращает false



Сравнивает два значения и их типы, и если они не равны, возвращает true, иначе возвращает false



Сравнивает два значения, и если они **не** равны, возвращает true, иначе возвращает false

```
let income = 100;
let strIncome = "100";
let result = income == strIncome;
console.log(result); //true
```

Переменная **result** здесь будет равна **true**, так как фактически и **income**, и **strIncome** представляют число 100.
Но оператор тождественности возвратит в этом случае false, так как данные имеют разные типы.

```
let income = 100;
let strIncome = "100";
let result = income === strIncome;
console.log(result); // false
```

ОПЕРАЦИИ ПРИСВАИВАНИЯ

= += -=

*= **= /=

`=` `+=` `-=`

```
let x = 5;
```

- Приравнивает переменной определенное значение:

```
let a = 23;
a += 5; // аналогично a = a + 5
console.log(a); // 28
```

- Сложение с последующим присвоением результата.

```
let a = 28;
a -= 10; // аналогично a = a - 10
console.log(a); // 18
```

- Вычитание с последующим присвоением результата.

*= **= /=

```
let x = 20;
x *= 2; // аналогично x = x * 2
console.log(x); // 40
```

- Умножение с последующим присвоением результата:

```
let x = 5;
x **= 2;
console.log(x);  // 25
```

- Возведение в степень с последующим присвоением результата:

```
let x = 40;
x /= 4; // аналогично x = x / 4
console.log(x); // 10
```

- Деление с последующим присвоением результата:

ОПЕРАТОР TYPEOF

С помощью оператора **typeof**

можно получить тип переменно

Для значения **null**

оператор **typeof**

возвращает значение "**object**".

```
let id;
console.log(typeof id);        // undefined

id = 45;
console.log(typeof id);    // number

id = 45n;
console.log(typeof id);      // bigint

id = "45";
console.log(typeof id);      // string
```

ВВОД/ВЫВОД ДАННЫХ

Чтобы взаимодействовать с пользователем необходимо использовать диалоговые окна или принимать от пользователя данные:

alert, **prompt** и **confirm**.

```
alert(сообщение)
```

```
result = confirm(question);
```

```
result = prompt(title, default);
```

alert выводит на экран окно с сообщением и приостанавливает выполнение скрипта, пока пользователь не нажмёт «**ОК**»

```
alert( "Привет" );
```

```
var years = prompt('Сколько вам лет?', 100);

alert('Вам ' + years + ' лет!')
```

prompt принимает 2 аргумента

Функция выводит окно с заголовком **title**, полем для ввода текста **default** и кнопками **OK/CANCE**L.

confirm выводит окно с вопросом **question** с двумя кнопками:

OK и **CANCEL**.

Результатом будет:

True при нажатии **OK**

False – при **CANCEL**(Esc).

```
var isAdmin = confirm("Вы - администратор?");

alert( isAdmin );
```

ЗАДАНИЕ

Запрашивать данные у пользователя необходимо с помощью **prompt**(), а выводить результат с помощью **alert**().

1. Запросите у пользователя число, возведите это число во 2-ю степень и выведите на экран.

2. Реализуйте конвертор из километров в мили (пользователь вводит километры, программа выводит мили). 1 км = 0,621371 миль. Это значение укажите в коде как константу.

3. Реализуйте калькулятор. Пользователь вводит два числа, а программа выводит результаты действий + - * / между этими числами.

4. Запросите у пользователя текущее время (часы и минуты) и выведите, сколько часов и минут осталось до следующего дня.

5. Зарплата работника составляет $250 + 10% от продаж. Запросите общую сумму продаж за месяц и посчитайте зарплату.

КОНСТРУКЦИЯ IF..ELSE

Конструкция **if..else** проверяет некоторое условие и если это условие верно, то выполняет некоторые действия.

После ключевого слова **if** в круглых скобках идет условие, а после условия - блок кода с некоторыми действиями. Если это условие истинно, то затем выполняются действия, которые помещены в блоке кода

ВЫРАЖЕНИЕ ELSE И ELSE IF

```
const income = 50;
if(income > 50) {
    console.log("Доход больше 50");
}
else if(income === 50){
    console.log("Доход равен 50");
}
else{
    console.log("Доход меньше 50");
}
```

ЗАДАНИЕ

Задания, в которых необходимо использовать **IF**.

1. Запросить у пользователя число и определить, оно положительное, отрицательное или ноль.

2. Запросить у пользователя его возраст и проверить корректность введенных данных (0–120 лет).

3. Запросить у пользователя число и вывести его модуль:

(|7| = 7, |-7| = 7)

4. Запросить у пользователя время (часы, минуты, секунды) и проверить корректность введенных данных.

5. Запросить координаты точки (x, y) и определить номер четверти, в которую попала эта точка. Необходимо учесть случаи попадания точки на оси X или Y или в начало координат.

ТЕРНАРНЫЙ ОПЕРАТОР

После символа «**?**» следует значение, которое будет возвращено при **истинном** результате условия.

После символа «**:**» будет возвращено при **ложном** результате условия.

```
parity = (x % 2 == 0) ? “even” : “odd”
```

Простыми слова:
Если **ЧЕТНОЕ**, то результат "**EVEN**",
Иначе **НЕЧЕТНОЕ** и результат "**ODD**"

ЗАДАНИЕ

Задания, в которых необходимо использовать тернарный оператор.

1. Запросить 2 числа и вывести большее из них.

2. Запросить 1 число и проверить,

оно кратно 5 или нет.

3. Запросить у пользователя название планеты. Если пользователь ввел «Земля» или «земля», то вывести «Привет, землянин!», в остальных случаях вывести «Привет, инопланетянин!»

SWITCH

Конструкция **switch..case** является альтернативой использованию конструкции **if..else** и также позволяет обработать сразу несколько условий

Switch последовательно сравнивается со значениями, помещенными после оператора **case**.
Если совпадение будет найдено, то будет выполняться определенный блок **case.**
Иначе будет обрабатываться блок **default.**

После ключевого слова **switch** в скобках идет сравниваемое выражение.

Если совпадение будет найдено, то будет выполняться блок **case.**
Иначе будет блок **default.**
В конце каждого блока **case** ставится оператор **break** (означает СТОП выполнения конструкции)

```
const income = 300;
switch(income){

    case 100 :
        console.log("Доход равен 100");
        break;
    case 200 :
        console.log("Доход равен 200");
        break;
    case 300 :
        console.log("Доход равен 300");
        break;
    default:
        console.log("Доход неизвестной величины");
        break;
}
```

ЗАДАНИЕ

Задания, в которых необходимо использовать **SWITCH**

1. Запросить у пользователя номер месяца и вывести на экран его название.

2. Реализовать калькулятор. Пользователь вводит 2 числа и знак (+ - * /). В зависимости от введенного знака решить пример и вывести результат.

ВОПРОСЫ

ПРАКТИКА

Давайте подведем итоги.