Лекция 1
Знакомимся с JavaScript

Matthew Levin
.NET full stack /
front-end
developer

Правила оформления кода

1. Для отступов в начале строки есть табуляция
2. Никаких пробелов в конце строки
3. Табуляция до 5 уровней вложенности
4. Длина строки до 80 символов
5. Выравнивание при переносе, к примеру, аргументов лучше выполнять по первому аргументу
6. if/else/for/while/try многострочные и с фигурными скобками
7. Унарные операторы отделяются пробелами
8. Операторы «,» и «;» не выделяются запятыми
9. «:» после имени должны отделяться 1 пробелом
10. Тернарный оператор ? и : должен иметь пробелы с обеих сторон.
11. Не использовать пробелы в пустых конструкторах, таких как {}, [], fn()
12. 1 Пробел между аргументами и выражением

Предисловие

Предисловие

- Код — это набор особых инструкций, сообщающих компьютеру какие задачи нужно сделать. Обычно код сохраняют в текстовый файл, хотя в случае JavaScript можно писать код прямо в консоли разработчика в браузере, чего мы кратко коснемся далее.

Предисловие

- Правила допустимого формата и комбинаций операторов называются язык программирования, иногда их соотносят с его синтаксисом, аналогично английскому языку, где правила говорят вам как произносить слова и как составлять правильные предложения используя слова и знаки препинания.

Операторы

Операторы

- В языке программирования группа слов/ чисел / операций

Пример:

a = b * 2;

a,b – переменные

2- литеральное значение

‘=‘ и ‘*’ – операции.

Переменные

1. Имеют название
2. Имеют значение
3. Т.к. JS – язык без строгой типизации, конкретный тип хранимых данных в переменной не указывается

Литеральное значение / Литерал

1. константы, включаемые непосредственно в текст программы
2. не могут быть изменены в тексте программы

Операции

- Выполняют действия со значениями и переменными
- Действия бывают следующих видов:
  1. Математические
  2. Логические
  3. Действия присваивания

В результате

Оператор «a= b* 2;» сообщает компьютеру, что ему необходимо взять текущее значение из переменной b, умножить его на 2, и сохранить результат в другую переменную, которую называется

a.

Выражения

1. Операторы состоят из одного или более выражений
2. Выражение —это любая ссылка на переменную или значение или набор переменных и значений, объединенных операциями

Пример

- a=b*2;
- У этого оператора 4 выражения:
- 2—это выражение литерального значения
- b—это выражение переменной, которое тут означает извлечение его текущего значения
- b* 2—это арифметическое выражение, в данном случае выполнение умножения
- a= b* 2—это выражение присваивания, в данном случае это присвоить результат выражения b* 2 переменной a

подробнее о выражениях далее

Оператор-выражение

- Выражение, которое является законченным

Пример:

```
a*4;
```

Выполнение программы

Инструменты:

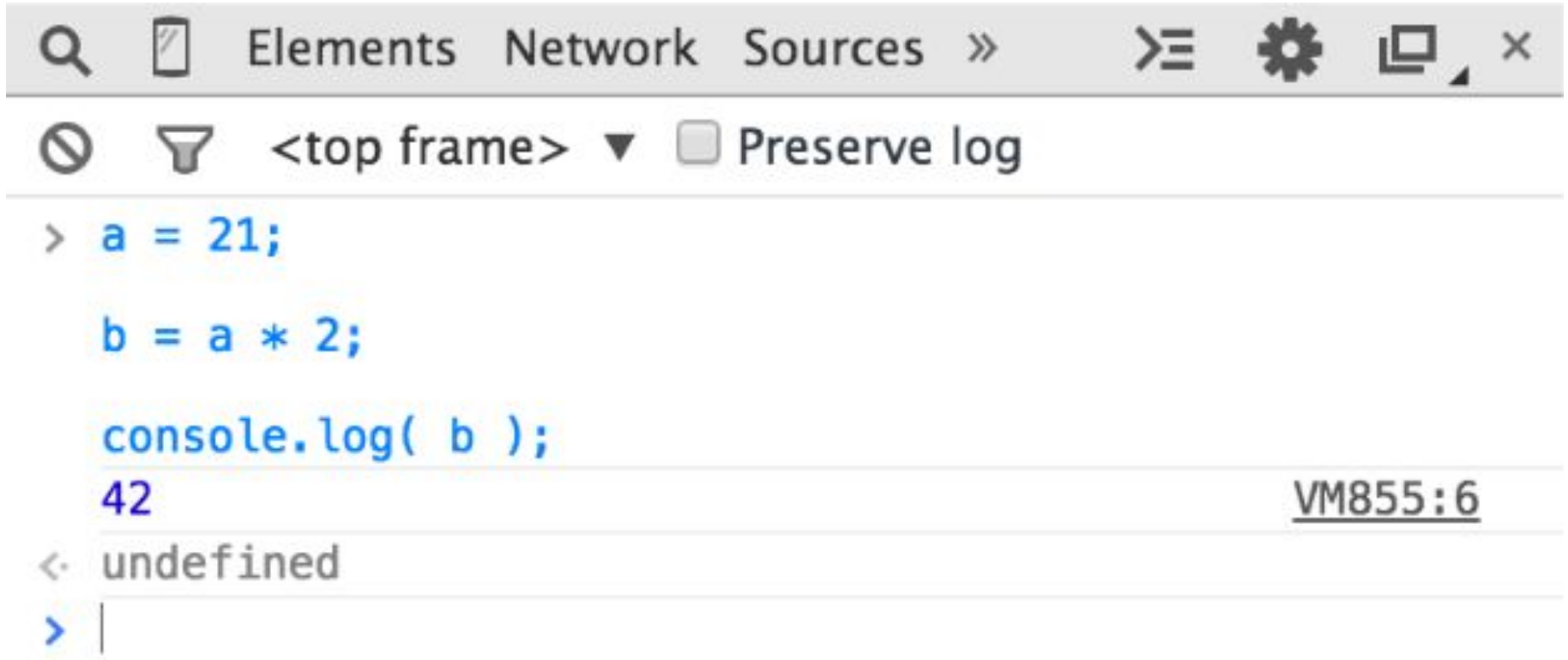
Для знакомства с языком JavaScript понадобиться открыть консоль в средствах разработки в ближайшем браузере (Firefox, Chrome, IE и т.п.).

По шагам:

- Ввести в адресную строку «about:blank» для открытия пустой страницы
- Нажать клавишу f12
- Перейти во вкладку «консоль»

*комбинация <shift> + <enter> используется для многострочного скрипта

Пример


Elements
Network
Sources
<top frame>
Preserve log
> a = 21;
b = a * 2;
console.log(b);
42
VM855:6
undefined
>

console.log()

- Оператор, предназначенный для вывода значения в консоли разработчика
- log(b) – функция, в которую передаётся значение, которое нужно вывести
- console. —это ссылка на объект, где расположена функция log(..)

alert()

- Еще один путь вывести информацию —запустить оператор alert(..). Например:

```
alert( b);
```

Данный оператор показывает всплывающее окно с кнопкой «ОК» и содержимым переменной b

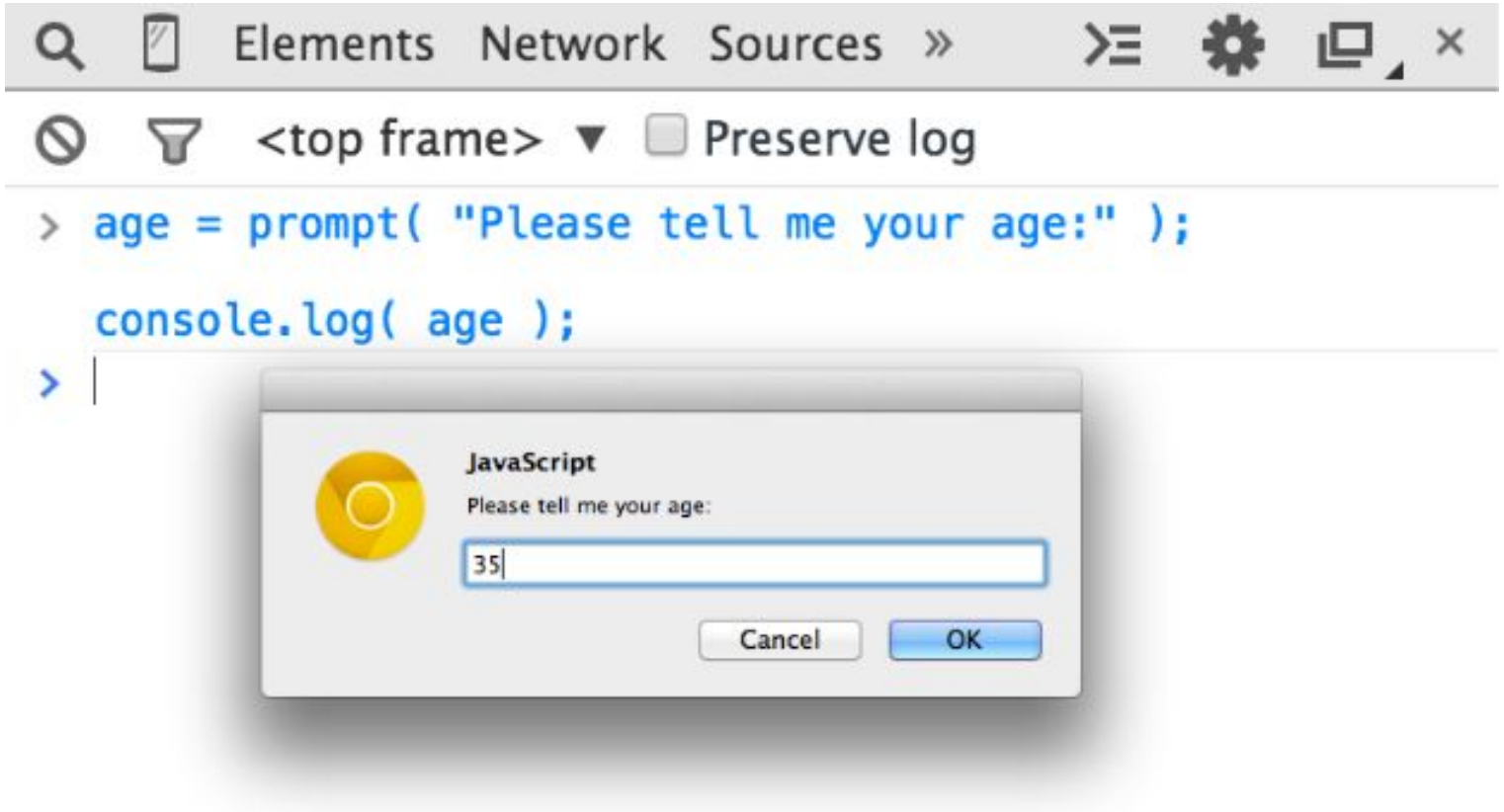
prompt()

- Оператор, предназначенный для простого диалога с пользователем по средствам модального окна. Пример:

```
age =prompt( "Please tell me your age:");
console.log( age);
```

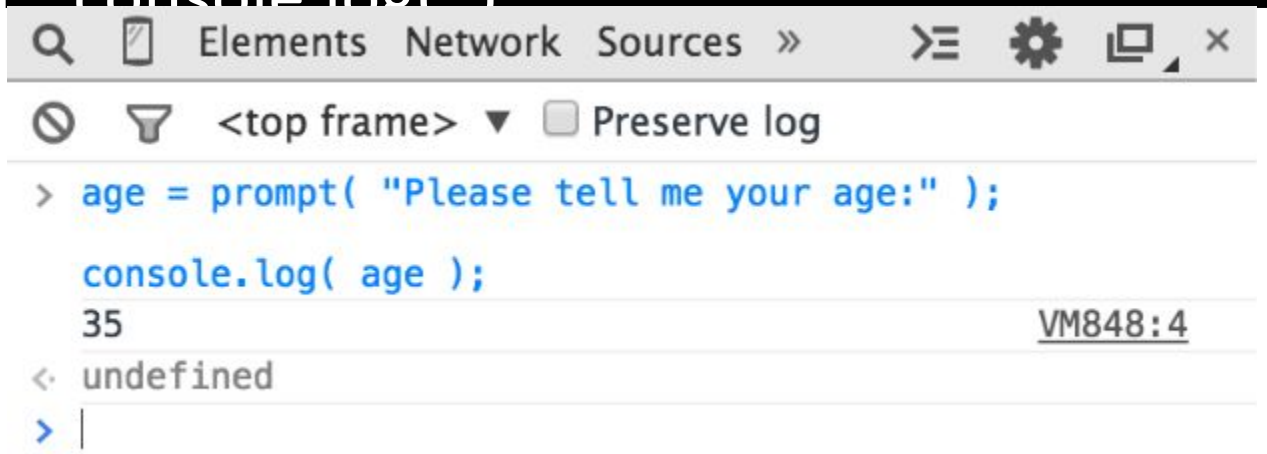
Сообщение, которое вы передаете в prompt(..), в данном случае «Please tell me your age:» выводится во всплывающем окне.

prompt()


Elements
Network
Sources
<top frame>
Preserve log
> age = prompt("Please tell me your age:");
console.log(age);
>
JavaScript
Please tell me your age:
35
Cancel
OK

prompt()

- Как только вы подтвердите ввод текста щелкнув по «ОК», вы заметите, что введенное значение теперь хранится в переменной age, которую мы затем выведемс помощью console.log( );



Операции

Операции —это те действия, которые мы выполняем над переменными и значениями. Мы уже видели две операции JavaScript, = и *.

Операция * выполняет математическое умножение.

Операция = используется для присваивания—сначала мы вычисляем значение с правой стороны(исходное значение) от = , а затем записываем его в переменную, которую мы указываем с левой стороны(переменная назначения)

var

Хоть и не являющееся технически операцией, вам необходимо ключевое слово var в любой программе, поскольку это основной способ, с помощью которого вы объявляете (т.е. создаете) переменные (сокращение от variables).

Вы всегда должны объявить переменную с именем до того, как начнете её использовать. Но вам достаточно объявить переменную всего раз для каждой области видимости, а затем пользоваться ею столько раз, сколько нужно.

var

```
// Пример
var a =20;
a =a +1;
a =a *2;
console.log( a );// 42
```

Операции

1. Присваивание: = как в «a = 2».
2. Математические: +(сложение), -(вычитание), *(умножение) и /(деление), как в «a* 3».
3. Составное присваивание: +=, -=, *=, и /= — это составные операции, которые объединяют математическую операцию с присваиванием, как в a += 2 (эквивалентно «a= a+ 2»).
4. Инкремент/Декремент: ++(инкремент), --(декремент), как в «a++»(эквивалентно «a=a+ 1»).
5. Доступ к свойству объекта: «.» как в console.log().
6. Объекты —это значения, которые хранят другие значения под своими именами, называемые свойства.
7. obj.a означает значение из объекта obj из его свойства a. Еще один способ доступа к свойствам —obj["a"].
8. Равенство: ==(нестрогое), ===(строгое), !=(нестрогое неравенство), !==(строгое неравенство), как в «a == b».
9. Сравнение: <(меньше чем), >(больше чем), <=(меньше или нестрого равно), >=(больше или нестрого равно), как в «a<= b».
10. Логические: &&(и),||(или), как в «a||b», которое выбирает или a, или(or) b. Эти операции используются для создания составных условных конструкций. Например: если либо a либо (or)b—истина

Значения и типы

- В JavaScript есть встроенные типы для каждого из этих так называемых примитивных значений:
- когда вам нужно работать с математикой, вам нужно число.
- когда вам нужно вывести значение на экран, вам нужна строка(один или несколько символов, слов, предложений).
- когда вам нужно принять решение в своей программе, вам нужно логическое значение (true (истина) или false (ложь)).

Дополнительные типы данных

1. Специальное значение «null»
2. Специальное значение «NAN»
3. Специальное значение «undefined»
4. Объекты «object»

Для определения типа переменной можно использовать оператор **typeof**

Литералы

Значения, непосредственно включаемые в исходный код, называются литералы. Строковые литералы заключаются в двойные кавычки "...« или одинарные ('...’) — единственная разница в них — это ваши стилистические предпочтения. Литералы

Числа и логического зннначения пишутся как есть.

Литералы

1. "Я -строка";
2. 'Я -тоже строка';
3. 42;
4. true;
5. false;

Приведение типов

Преобразование между типами

Преобразование 1 типа к другому в программировании называется приведением.

JavaScript предоставляет несколько различных возможностей принудительного приведения между типами. Например:

```
var a ="42";

var b =Number( a );

console.log( a );// "42"

console.log( b);// 42
```

Явное приведение

Пример:

Использование Number(..)(встроенная функция)

Неявное приведение

При сравнении строки "99.99" с числом 99.99 многие согласятся, что они равны. Но они ведь не совсем одно и то же, не так ли? Это одно и то же значение в двух разных представлениях, двух разных типов

.

Неявное приведение

если вы используете операцию нестрогого равенства == для сравнения "99.99" == 99.99, JavaScript преобразует с левой стороны "99.99"в его числовой эквивалент 99.99. После этого сравнение превращается в 99.99 == 99.99, которое конечно является истинным.

Комментарии в коде

Комментарии в коде

Правила:

- Код без комментариев не оптимален.
- Слишком много комментариев (по одному на каждую строку кода, например) возможно являются признаком плохо написанного кода.
- Комментарии должны объяснять почему, а не что. Они могут дополнительно объяснять как, когда код особенно сложен.

Комментарии в коде

Пример:

```
// Это -однострочный комментарий
/* А это -многострочный
комментарий.*/
```

Переменные

Переменные

JavaScript использует динамическую типизацию, что означает, что переменные могут хранить значения любого типа без какого-либо контроля типов.

Как уже упоминалось ранее, мы объявляем переменную используя оператор var, заметьте, что при этом нет больше никакой другой информации о типе в объявлении.

Обратите внимание на эту простую программу:

```
1. var amount=99.99;
2. amount=amount*2;
3. console.log( amount);// 199.98
4. // преобразует `amount` в строку и
5. // добавляет "$" в начало
6. amount ="$"+String( amount );
7. console.log( amount );// "$199.98"
```

Переменная amount начинает свой жизненный цикл с хранения числа 99.99, а затем хранит числовой результат amount* 2, который равен 199.98.

Переменные

- Первая команда console.log(..) должна неявно привести это Числовое значение к строке, чтобы вывести его в консоль. Затем оператор amount= "$" + String(amount) явно приводит значение 199.98 к строке и добавляет символ "$« в начало. С этого момента, amount хранит строковое значение "$199.98", поэтому второму оператору console.log(..)не нужно выполнять никакого приведения, чтобы вывести его в конс

Условные конструкции

Оператор if

```
var a=1;
if(a<2){
a++;
}
```

Оператор if..else

```
var age=17;
if(age<18){
console.log(“sorry, you are so young”);
}
else{
console.log(“welocome!”);
}
```

Конструкция switch ..case

```
1.    switch(x) {
2.      case 'value1':  // if (x === 'value1')
3.        ...
4.        [break]
5.      case 'value2':  // if (x === 'value2')
6.        ...
7.        [break]
8.      default:
9.        ...
0.        [break]
1.    }
```

Циклы

Оператор for

```
for(var a=1,a<18,a++)
{
console.log(“sorry, you are so young”);
}
console.log(“welocome!”);
```

Оператор while

```
var a=1;
while(a<18)
{
console.log(“sorry, you are so young”);
a++;
}
console.log(“welocome!”);
```

Оператор do..while

```
var a=1;
do
{
console.log(“sorry, you are so young”);
a++;
}
while(a<18);
console.log(“welocome!”);
```

Функции

Синтаксис

```
function functionName(param1,param2 ..){
…//тело функции
}
```

Оператор return

```
1. function functionName(param1,param2 ..){
2. …//тело функции
3. return a;
4. }
```

Область видимости

Локальные переменные

```
function showMessage() {
  var message = 'Привет, я - Вася!'; // локальная переменная

  alert( message );
}

showMessage(); // 'Привет, я - Вася!'

alert( message ); // <-- будет ошибка, т.к. переменная видна только внутри
```

Важно помнить

1. Блоки if/else, switch, for, while, do..while не влияют на область видимости переменных.
2. Неважно, где именно в функции и сколько раз объявляется переменная. Любое объявление срабатывает один раз и распространяется на всю функцию.

Внешние переменные

```
1. var userName = 'Вася';
2. function showMessage() {
3.   var message = 'Привет, я ' + userName;
4.   alert(message);
5. }
6. showMessage(); // Привет, я Вася
```

Внешние переменные

```
1. var userName = 'Вася';
2. function showMessage() {
3.   userName = 'Петя'; // (1) присвоение во внешнюю переменную
4.   var message = 'Привет, я ' + userName;
5.   alert( message );
6. }
7. showMessage();
8. alert( userName ); // Петя, значение внешней переменной изменено функцией
```

События в браузере

Типы событий

1. События мыши
2. События документа Window
3. События клавиатуры
4. События формы и ЭУ
5. События буфера обмена
6. События перетаскивания
7. События медиа
8. События CSS

События мыши

click	нажатие на элемент левой кнопкой мыши
dblclick	двойное нажатие на элемент левой кнопкой мыши
contextmenu	нажатие на элемент правой кнопкой мыши
mouseover	при наведении мыши на элемент
mousedown	при нажатии мышью на элемент
mouseup	при отжатии мыши на элементе
mousemove	при движении мыши
onwheel	при передвижении колеса мыши над элементом

События документа Window

DOMContentLoaded	HTML загружен и обработан, DOM полностью построена и доступна
load	браузер загрузил все ресурсы
beforeunload/unload	при уходе со страницы
resize	браузер изменил размеры окна
error	при запуске/загрузке ресурса происходит ошибка

События форм и элементов управления

blur	при потере элементов фокуса
change	по окончании изменении значения элемента формы
focus	при получении элементов фокуса
invalid	при получении элементом статуса invalid
select	при выборе пользователем элемента
submit	при отправке формы на сервер
input	срабатывает тут же при изменении значения текстового элемента


JavaScript
The Definitive Guide
David Flanagan
O'REILLY
Unearthing the Excellence in JavaScript
JavaScript:
The Good Parts
O'REILLY
YAHOO! PRESS
Douglas Crockford

Дома:

- Самостоятельно изучить методы получения доступа к объектам страницы
  - Самостоятельно изучить методы изменения стиля по средствам JS (минимум – 1, 2)
  - Методы переадресация по средствам JS (к примеру эти)
  - Изучение методов изменения HTML – страницы по средствам JS(к примеру этого)
  - Разработать страницу с тремя кнопками:
    - При нажатии на первую - переход на другую(любую) страницу
    - При нажатии на вторую – смена стиля страницы
    - При нажатии на третью – удаление содержимого со страницы и отрисовка любого макета с flexbox