Практическое занятие №9 Введение в JS

Задание №1

Проверьте запущен ли у Вас Open Server. Папка с материалами урока (js9.loc) должна быть размещена в директории /domains сервера. Заходим на http://js9.loc и начинаем урок.

Задание №2

Существует несколько способов подключения JS. В этом задании мы их рассмотрим.

Встраивание (embedding) - код сценариев пишется непосредственно между тэгами <script>. Это самый простой способ. Код JS располагается непосредственно на самой html странице. Открываем файл index-2.html и размещаем тег <script>, как показано ниже:

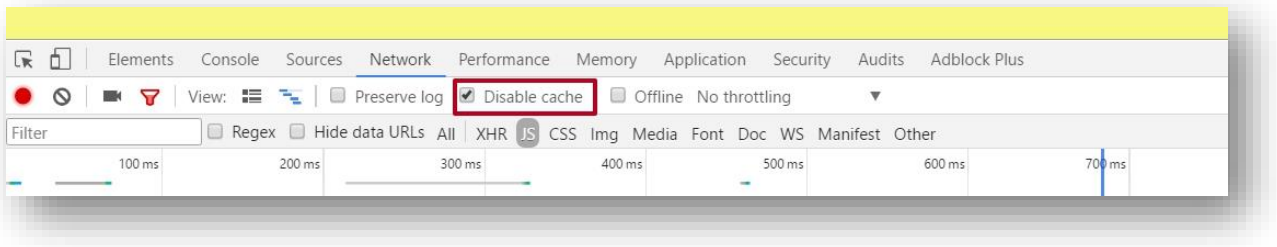
```
<script type="text/javascript">
    console.log('встроенный код JS'); // вызываем консоль браузера
</script>
```

Этот код может быть размещён как в секции <head> так и <body>, на данный момент разницы для нас нет.

console.log() - Выводит значения в консоль браузера. Подробнее об этом методе Вы узнаете на следующих занятиях.

Как увидеть результат работы скрипта? Перезагрузить страницу. Открыть консоль браузера (DevTools, вкладка «Console»).

ЭТО ВАЖНО! Отключите кеширования в настройках DevTools (Вкладка Network->Disable cache)!



Вложение (inline) - код сценариев JS пишется в значении атрибутов событий (изучается на следующих занятиях).

Для того что бы Вы имели начальное представление о вложениях мы напишем сейчас небольшой пример вложения и убедимся, что он работает. Создаем еще один блок «.jsblock__inner» сразу после первого и в нем создайте кнопку с атрибутом onclick, как показано ниже:

```
<div class="jsblock__inner">
    <button onclick="console.log('Клик!');" >Кнока</button>
</div>
```

Если Вы сделали все верно, то при клике на кнопку в консоли будет выводиться новое сообщение. Не забудьте перезагрузить страницу после написания кода.

Связывание (linking) – с помощью атрибута src тега <script> указывается адрес внешнего файла. Этот вид внедрения сценариев предпочтителен и обладает преимуществами перед методом встраивания. У элемента <script> используется атрибут type="text/javascript" для обозначения типа скрипта. Значение этого атрибута указывает браузеру на то, каким модулем обрабатывать этот контент (его можно встретить в таких элементах, как <style>, <object>, <embed>, <link>). Раньше использовался атрибут language, но в настоящее время использовать его не рекомендуется.

В папке задания создайте файл custom.js с одной строчкой JS кода:

```
console.log('custom.js');
```

и подключите его к странице в разделе <head>:

```
<script src="/tasks/task-2/custom.js" type="text/javascript"></script>
```

Перезагрузите страницу и в консоли должны быть сообщения, сгенерированные в custom.js.

В этом задании мы рассмотрели способы подключения JS к HTML странице. Идем дальше.

Задание №3

Создайте файл JS в папке урока и подключите его к странице урока (index-3.html). Создайте пять переменных, с произвольными именами. Присвойте им произвольные строковые значения. Затем выведите переменные в консоль браузера.

```
var name1 = 'Лена';
```

ЭТО ВАЖНО! Не забывайте использовать разделитель инструкций «;»!

ЭТО ВАЖНО! Не забывайте про правила наименования переменных!

Убедитесь, что в консоли у Вас не ошибок и все данных выведены.

Задание №4

В этом задании мы узнаем, как создавать строковый тип данных в JS.

Создайте файл JS в папке урока и подключите его к странице урока.

Строки в JS это объекты определенного типа, создать строку можно с помощью вызова класса-конструктора или с помощью строкового литерала, выполните код как показано ниже:

```
var stringObject = new String('эта строка - обьект');
var string = String('просто строка');
var string2 = 'еще одна строка'; /* предпочтительный способ создания строк */
```

Теперь проверим тип созданных нами переменных stringObject, string с помощью оператора typeof:

```
console.log(typeof stringObject);
console.log(typeof string);
console.log(typeof string2);
```

Теперь выведем переменные напрямую в консоль:

```
console.log(stringObject);
console.log(string);
console.log(string2);
```

Какие выводы мы можем сделать? В переменной stringObject находится объект, а в string и string2 – строки. Объект имеет сложную структуру, строки – простую. Это связано с тем что объект содержит в себе данные и информацию как работать с этими данными (методы). Базовая структура объекта показана на рисунке ниже.

В строковом объекте находится сама строка, это его данные (левая желтая часть на рисунке), а также в нем находятся методы, которые могут работать с этой строкой (правая часть) с ними мы познакомимся подробнее в следующих заданиях (также более подробно будет рассмотрена объектная модель в JS).

Задание №5

В JS строку можно создать, используя двойные и одинарные кавычки. Это дело вкуса. Правило одно внутри двойных кавычек можно свободно использовать одинарные, и наоборот.

В файле задания у Вас уже есть закомментированные шесть строк, нужно их раскомментировать (убрать символы // в начале каждой строки) и поместить каждую строку в отдельную переменную, затем вывести переменные в консоль браузера. Обратите внимание, что некоторые строки содержат внутри кавычки.

Задание №6

Особое значение имеет символ “\” (обратная косая черта или обратный «слеш»). В совокупности с некоторыми другими символами он образует так называемые «управляющие последовательности». Эти последовательности могут обозначать непечатаемые символы и знаки. Далеко не все управляющие последовательности нужны для отображения на мониторе. Некоторые требуются для управления выводом на других типах устройств, например, на принтере.

С помощью управляющих последовательностей можно внутри строки, ограниченной двойными кавычками, поставить двойную кавычку \". Этот прием называется «экранированием».

Сделайте то же самое, что и в предыдущем задании, только используя экранирование. Например, если в строке встречается двойная кавычка, то строку мы обрамляем двойными кавычками, а внутри кавычки экранируем.

Список управляющих последовательностей

Последовательность	Что означает
\0	Символ NULL (\u0000)
\b	«Забой» (\u0008)
\t	Горизонтальная табуляция (\u0009)
\n	Перевод строки (\u000A)
\v	Вертикальная табуляция (\u000B)
\f	Перевод страницы (\u000C)
\r	Возврат каретки (\u000D)
\"	Двойная кавычка (\u0022)
\'	Одинарная кавычка (\u0027)
\\	Обратная косая черта или обратный «слеш» (\u005C)
\xXX	Символ Latin-1, заданный двумя шестнадцатеричными цифрами
\uXXXX	Символ Unicode, заданный четырьмя шестнадцатеричными цифрами

Задание №7

Свойство строки ***length*** - возвращает длину строки, вызываются свойства как показано ниже:

```
var $string = 'my string';

// <переменная>.<свойство> – обратите внимание на точку между переменной и свойством!
$string.length; // вернёт длину строки
```

Скопируйте все свои переменные со строками из предыдущего задания и выведите в консоль длины их строк с помощью свойства ***length***.

ЭТО ВАЖНО! Как это возможно? Очень просто! Когда мы пытаемся вызвать метод или свойство у простой строки JavaScript «в тихую» создаёт объект и уже от имени этого объекта вызывает метод или свойство, после чего объект уничтожается.

Задание №8

В этом задании мы проведём обзор основных методов типа данных String. Работаем по простой схеме: сначала идёт описание работы метода, и за тем демонстрационный пример. Вы набираете демо примеры и убеждаетесь в их работоспособности и так далее.

.concat(string1 [, string2]) - Объединяет исходную строку и строки-аргументы. На практике используют оператор сложения “+”;

```
var str1 = 'Привет, ';
var str2 = 'Вася ';
var str3 = 'удачного дня!';

var fullString = str1.concat(str2, str3);

console.log(fullString);
```

Тоже самое только используя оператор сложения строк «+» (конкатенацию).

```
var str1 = 'Привет, ';
var str2 = 'Вася ';
var str3 = 'удачного дня!';

var fullString = str1+str2+str3;

console.log(fullString);
```

.charAt(n) – возвращает символ строки по указанной позиции.

```
var str = 'abcd';
console.log(str.charAt(1)); // вернёт "b"
```

.indexOf(text[, start]) - возвращает позицию первого вхождения указанного текста в строке (>=0) или -1, если ничего не найдено. Ищет с начала строки. Если стартовая позиция не указана, поиск идет с первого символа.

```
console.log('Голубой кит'.indexOf('Голубой'));   // вернёт  0
console.log('Голубой кит'.indexOf('Голуббой')); // вернёт -1
console.log('Голубой кит'.indexOf('кит', 0));    // вернёт  8
```

.replace(text1, text2) - возвращает строку, в которой произведена замена первого аргумента на второй.

```
var str = ' var str = "string";';

console.log( str.replace('var ', 'var $') );
```

.slice(begin[, end]) – вырезает часть строки между позициями begin и end. Отрицательное значение будет считаться от конца строки.

```
var str = 'Добрый вечер';

console.log( str.slice(1, 11) ); // вернёт "обрый вече"
console.log( str.slice(-5) ); // вернёт "вечер"
console.log( str.slice(3) ); // вернёт "рый вечер"
```

.split([separator[, limit]]) – разбивает строку по разделителю. Возвращает массив строк.

```
var str = 'Маша,Лена,Сара,Трисс';
console.log( str.split(',') );
```

.toLowerCase() и . toUpperCase() – преобразует строку в нижний или верхний регистр.

```
var str = 'Маша,Лена,Сара,Трисс';
console.log( str.toLowerCase() );
console.log( str.toUpperCase() );
```

Задание №9

Пользуясь методами и свойствами из предыдущих уроков сделайте первую букву в строке «ann» заглавной.

Задание №10

Из строки в переменной «str» нужно вырезать спам. Спамом считается строка «XXX» в любом регистре.

Задание №11

Аналогично строкам, числа в JS можно создавать различными способами. Создайте числа различными способами и выведите получившиеся переменные в консоль браузера:

```
var num1 = new Number(5);
var num2 = Number(5);
var num3 = 5; // лучше использовать это способ

console.log(num1, num2, num3);
```

Операторы для работы с числами

Оператор	Как называется	Как применяется	Результат
+	оператор сложения	8 + 7	15
–	оператор вычитания	11 – 6	5
/	оператор деления	21 / 7	3
*	оператор умножения	4 * 6	24
%	оператор остатка от деления	5 % 3	2
++	инкремент (применяется к переменной)	myVar++, ++myVar	myVar + 1
--	декремент (применяется к переменной)	myVar--, --myVar	myVar – 1

Сокращенная запись операторов

Полная форма записи	Сокращенная форма записи
x = x + y	x += y
x = x – y	x –= y
x = x / y	x /= y
x = x * y	x *= y
x = x % y	x %= y

Все приведённые в таблице выше операторы должны быть Вам знакомы со школы. Интерес представляют операторы инкремента (++) и декремента (--). В зависимости от места указания инкремента и декремента в инструкции, результат выполнения именно этой инструкции может быть различным. Разберем пример:

```
//Постфиксное написание
 var x = 7;
 var y = x++;
 console.log(x);
 console.log(y); //покажет 7 - в "y" попало значение x, потом значение x увеличилось


//Префиксное написание
var x = 7;
var y = ++x;
console.log(x);
console.log(y); //покажет 8 - значение x увеличилось, потом в "y" попало значение x
```

Из примеров видно, что по отношению к операнду инкремент и декремент работают одинаково независимо от формы записи. Разница видна в результатах операции присвоения.

Знак “–” (минус), записанный перед числом, делает его отрицательным (смена знака). Знак “+” (минус), записанный перед именем или значением переменной, пытается превратить значение переменной в число.

Задание №12

В этом задании мы проведём обзор основных методов типа данных Number. Работаем по простой схеме: сначала идёт описание работы метода, и за тем демонстрационный пример. Вы набираете демо примеры и убеждаетесь в их работоспособности и так далее.

.toFixed([n]) - форматирует число, используя запись с фиксированной запятой, где n - необязательный параметр. Количество цифр после десятичной запятой; может быть значением между 0 и 20 включительно, хотя реализации могут поддерживать и больший диапазон значений. Если аргумент опущен, он считается равным 0.

```
var pi = 3.1416181;
console.log( pi.toFixed(2) );
```

.toPrecision([точность]) - возвращает строку, представляющую объект Number с указанной точностью.

```
var pi = 3.1416181;
console.log( pi.toPrecision(2) );
```

Задание №13

Используя JS сделайте расчет сколько секунд в одной неделе и выведите результат в консоль.

Задание №14

Используя JS сделайте расчет сколько часов прошло, начиная с года Вашего рождения (без учета високосных годов, то есть принять год равным 365 дней).

Задание №15

Дано значение температуры в градусах Фаренгейта $T_F = 100$, найдите значение температуры в градусах Цельсия с помощью JS, если они связаны следующим соотношением:

$$T_C = (T_F - 32) \cdot 5/9.$$

Выведите получившиеся значение в консоль.

Задание №16

Глобальный объект JS обладает следующими свойствами для работы с числами:

Свойства глобального объекта

Название	Что означает
-Infinity	Отрицательная бесконечность – возвращается при переполнении
Infinity	Положительная бесконечность – возвращается при переполнении
NaN	«Нечисло» – возвращается при ошибке арифметической операции

Уникальность NaN – это значение не равно ничему, даже самому себе. Для проверки «нечисла» используется функция глобального объекта isNaN().

В этом задании мы проведём обзор основных методов глобального объекта для работы с числами. Работаем по простой схеме: сначала идёт описание работы метода, и за тем демонстрационный пример. Вы набираете демо примеры и убеждаетесь в их работоспособности и так далее.

isNaN() - определяет является ли литерал или переменная не числовым значением. По сути это проверка значения на «не число».

```
// не числа, вернёт true
console.log( isNaN(NaN));
console.log( isNaN(undefined));
console.log( isNaN({}));
console.log( isNaN('строка'));
console.log( isNaN('12ABC'));

// числа, вернёт false
console.log( isNaN(true));
console.log( isNaN(null));
console.log( isNaN(37));
console.log( isNaN(37.78797));
console.log( isNaN(1 / 8));
console.log( isNaN('12'));
console.log( isNaN('12.89'));
```

parseFloat(string) -если строка начинается с цифр, извлекает вещественное число и возвращает его, иначе возвращает NaN.

```
console.log( parseFloat('12.7 xxxxxxxxxxxxxxxxxxxxx xxxxxxxx xxxxx') );
console.log( parseFloat('xxxx 17.67 xxxx') );
```

parseInt(string) – тоже самое что parseFloat, только возвращает целые числа.

```
console.log( parseInt('12.7 xxxxxxxxxxxxxxxxxxxxx xxxxxxx xxxxx') );
console.log( parseInt('xxxx 17.67 xxxx') );
```

Задание №17

Экземпляры логического типа данных Boolean имеют два возможных значения – true и false. Этот тип данных используется там, где есть проверка на соответствие условию. JavaScript легко преобразовывает типы данных из одного в другой, что следует учитывать в таких операциях.

```
var bool = new Boolean (true);

var bool2 = Boolean (true);
var bool3 = true; // предпочтительный способ

console.log(typeof bool);
console.log(typeof bool2);
console.log(typeof bool3);

console.log(bool);
console.log(bool2);
console.log(bool3);
```

Операторы сравнения возвращают своим результатом значение логического типа.

Операторы сравнения

Оператор	Называется	Что делает
==	Равенство	Проверяет равенство значений двух объектов Если равны – возвращает true, нет – false
===	Идентичность	В отличие от сравнения сначала проверяет соответствие типов данных двух объектов Если объекты идентичны (тип одинаков и значения после преобразования равны) – возвращает true, нет – false
!=	Неравенство	Проверяет, не равны ли два объекта Если не равны – возвращает true, нет – false
!==	Неидентичность	Проверяет, не идентичны ли два объекта Если не идентичны – возвращает true, нет – false
>	«Больше»	Если левый объект строго больше правого, возвращает true, нет – false
>=	«Больше или равно»	Если левый объект больше или равен правому, возвращает true, нет – false
<	«Меньше»	Если левый объект строго меньше правого, возвращает true, нет – false
<=	«Меньше или равно»	Если левый объект меньше или равен правому, возвращает true, нет – false

Наберите и протестируйте несколько примеров операторов сравнения:

```
// истинные значения true
console.log( 5 == 5 );
console.log( 5 === 5 );
console.log( 1 < 2 );

// ложные значения
console.log( 5 != 5 );
console.log( 5 !== 5 );
console.log( 1 > 2 );
```

Логические операторы

! - логическое «НЕ». Возвращает обратное значение (инвертирует его).

&& - логическое «И». Пытается преобразовать значение каждого операнда в логическое значение. Прекращает работу, когда значение операнда преобразуется в false. Возвращает значение последнего операнда или того, на котором прекратилось выполнение оператора. Перебирает все операнды, пока не встретится значение, которое нельзя преобразовать в true.

|| - логическое «ИЛИ». Пытается превратить значение каждого операнда в логическое значение. Возвращает значение первого операнда, значение которого удалось преобразовать в true или значение последнего операнда. Перебирает все операнды, пока не встретится значение, которое нельзя преобразовать в true.

Наберите и протестируйте несколько примеров логических операторов (с булевым типом):

```
console.log(!true);
console.log(!!true);

console.log( false || false || false );
console.log( false || true );

console.log( false && false );
console.log( false && true );
```

Несколько примеров с не булевыми типами:

```
var number = 5;
var number2 = 0;
var number3 = 6;
var string = '';

console.log(number && number3); //выведет 6
console.log(number && number2 && string); //выведет 0

console.log(string || number2); //выведет 0
console.log(string || number2 || number); //выведет 5
```

Задание №18

Преобразования значений в JavaScript производятся очень часто.

Следует понимать, что при таких фоновых преобразованиях значение, хранящееся в переменной, не меняется – создается временное значение с типом, который нужен для спешного выполнения операции. Например, сравниваются строка и число – эти значения приводятся к одному типу данных, сравнение

возвращает логическое значение, преобразованные временные значения удаляются. Все преобразования зависят от контекста – операторов, которые выполняют над ними действия.

Ниже представлена таблица преобразования в различные контексты:

Тип значения	Контекст, в котором используется значение			
	Строковый	Числовой	Логический	Объектный
Неопределенное значение	'undefined'	NaN	false	Error
null	'null'	0	false	Error
Пустая строка	Как есть	0	false	Объект String
Непустая строка	Как есть	Числовое значение строки или NaN	true	Объект String
0	'0'	Как есть	false	Объект Number
NaN	'NaN'	Как есть	false	Объект Number
Infinity	'Infinity'	Как есть	true	Объект Number
-Infinity	'-Infinity'	Как есть	true	Объект Number
Любое другое число	Строковое представление числа	Как есть	true	Объект Number
true	'true'	1	Как есть	Объект Boolean
false	'false'	0	Как есть	Объект Boolean
Объект, функция	toString()	valueOf(), toString() или NaN	true	Как есть

Существует несколько интересный методов преобразования типов в JS. Например, из любого типа данных можно получить логический тип данных с помощью двойного отрицания:

```
var var1 = 5;
console.log(!!var1); //  true
```

Выполните преобразование значений переменных от b1 до b8 в bool тип и выведите значения в консоль.

Получить строку из любого объекта можно двумя способами:

```
var x = 15;
var string = x.toString();
var string = x + '';  //более короткий способ
```

Выполните преобразование значений переменных от s1 до s8 в строковый тип и выведите значения в консоль.

Получение числа из строкового представления в 10-й системе счисления:

```
var x = '25';
var number = x * 1; //числовой контекст - оператор умножения
var number = + x; //унарный плюс
var number = parseInt(x);
```

Выполните преобразование значений переменных от n1 до n8 в числовой тип и выведите значения в консоль.

Задание №19

Очень часто JS автоматически производит нужные преобразования типов, рассмотрим и протестируем несколько примеров. Все строки ниже будут преобразованы в числа на "лету" (метод isNaN вернёт false) проверьте это:

```
console.log( isNaN("37") );
console.log( isNaN("37.37") );
console.log( isNaN("") );
console.log( isNaN(" ") );
```