Лекция 14 апреля 2020 года.

JAVASCRIPT. ОСНОВЫ ЯЗЫКА (продолжение).

Преобразование типов. Область видимости переменных и функций. Цепочки областей видимости

материал находится в файле «14 апреля 2020 г – WEB – Лекция. Часть 1.pdf», составлен на основе книги *Флэнаган Дэвид. JavaScript. Подробное руководство.* 6-е изд. – Пер. с англ. – СПб: Символ-Плюс, 2012. – 1080с., ил.

Выражения и операторы.

Выражение – это фраза языка JavaScript, которая может быть вычислена интерпретатором для получения значения. Константа, встроенная в программу, является простейшей разновидностью выражений. Имя переменной также является простейшим выражением, в результате вычисления которого получается значение, присвоенное переменной. Более сложные выражения состоят из простых выражений.

Наиболее типичный способ конструирования сложных выражений из более простых выражений заключается в использовании *операторов*. Операторы объединяют значения своих операндов (обычно двух) некоторым способом и вычисляют новое значение. Иногда для простоты мы говорим, что оператор *возвращает* значение вместо «вычисляет» значение.

Программы на языке JavaScript представляют собой не более чем последовательности выполняемых *инструкций*. По умолчанию интерпретатор JavaScript выполняет эти инструкции одну за другой в порядке их следования. Другой способ сделать так, чтобы «что-то происходило», заключается в том, чтобы влиять на этот порядок выполнения по умолчанию, для чего в языке JavaScript имеется несколько инструкций, или управляющих конструкций, специально предназначенных для этого:

- условные инструкции: if и switch,
- инструкции циклов: while и for,
- инструкции переходов: break, return и throw.

По аналогии с выражениями инструкции можно считать предложениями на языке JavaScript, или командами. Инструкции JavaScript завершаются точками с запятой. Выражения вычисляются и возвращают значение, а инструкции выполняются, чтобы *что-то происходило*.

Чтобы «что-то происходило», можно вычислить выражение, имеющее побочные эффекты. Выражения с побочными эффектами, такие как присваивание и вызовы функций, могут играть роль самостоятельных инструкций – при таком использовании их обычно называют *инструкциями-выражениями*. Похожую категорию инструкций образуют *инструкции-объявления*, которые объявляют новые переменные и определяют новые функции.

Далее материал по темам:

- Обзор операторов: количество операндов, Типы данных операндов и результата, левосторонние выражения, побочные эффекты операторов, приоритет операторов, ассоциативность операторов, порядок вычисления;
- Арифметическое выражение: оператор +;
- Выражения отношений: равенства и неравенства, сравнения,

читай в файле «14 апреля 2020 г – WEB – Лекция. Часть 2.pdf», составлен на основе книги *Флэнаган Дэвид. JavaScript. Подробное руководство.* 6-е изд. – Пер. с англ. – СПб: Символ-Плюс, 2012. – 1080с., ил.

Дополнительно информацию по теме можно прочитать в книге *Закас Николас. JavaScript для профессиональных веб-разработчиков* / [Пер. с англ. А. Лютича]. – СПб.: Питер, 2015. – 960с.: ил. – (Серия «Для профессионалов»):

- Глава 3 . Основы языка
- Глава 4. Переменные, область види мости и память
- Глава 5. Ссылочные типы

Обратите внимание на раздел «Тип Array», который соответствует теме лабораторной работы №3 «Работа с массивами».

Тезисы по теме «Преобразование типов»

Пример. «Простые» примеры

```
10 + " objects" // "10 objects". Число 10 преобразуется в строку
"7" * "4"       // 28: обе строки преобразуются в числа
"7" + "5"       // "75": + - конкатенация строк
("7"+"5")==12   // false
("7"*"5")=="35" // true
("7"*"5")==35   // true
var n=1-"x"; // =>NaN: строка "x" не может быть преобразована в число
n + " objects"  //  => "NaN objects": NaN преобразуется в строку NaN"
```

Значение	Преобразование в:			
	Строку	Число	Логическое значение	Объект
undefined	"undefined"	NaN	false	Ошибка TypeError
null	"null"	0	false	Ошибка TypeError
true	"true"	1		new Boolean(true)
false	"false"	0		new Boolean(false)
"" (пустая строка)		0	false	new String("")
"3.1" (непустая строка, число)		3.1	true	new String("3.1")
"sea" (непустая строка, не число)		NaN	true	new String("sea")
0	"0"		false	new Number(0)
-0	"0"		false	new Number(-0)
NaN	"NaN"		false	new Number(NaN)
Infinity	"Infinity"		true	new Number(Infinity)
-Infinity	"-Infinity"		true	new Number(-Infinity)
13 (конечное, ненулевое)	"13", toString()		true	new Number(13)
{} (любой объект)	toString() valueOf()	valueOf()	true	
[] (пустой массив)	""	0	true	
[7] (один числовой элемент)	"7"	7	true	
['sun'] (любой другой массив)	toString(), join(разделит)	NaN	true	
function(){} (любая функция)	toString()	NaN	true	

Преобразование чисел в строки.

Пример. Число → Строка

```
n=17             // 17
n.toString(2)    // "10001"
n.toString(8)    // "21"
n.toString(16)   // "11"
n.toString(14)   // "13"
n.toString(17)   // "10"
n.toString(18)   // "h"
n.toString(3)    // "122"
```

Преобразование объектов в простые значения.

toString()
valueOf()

Тип объекта	Возвращаемые значения

Object	`a={q:1, w:2}  // Object {q: 1, w: 2}`
	`a.toString()  // "[object Object]"`
	`a.valueOf()  // Object {q: 1, w: 2}`

Array	`b=[1,2]  // [1, 2]`
	`b.toString()  // "1,2"`
	`b.valueOf()  // [1, 2]`
	`b.join("^")  // "1^2"`

Function	`c=function(){alert("Hello!!!");}`
	`// function (){alert("Hello!!!");}`
	
	`c.toString()`
	`// "function (){alert("Hello!!!");}"`
	
	`c.valueOf()`
	`// f (){alert("Hello!!!");}`
	`// c.valueOf()()  -  ???`
	
	`// НО: для функций JavaScript не возвращается:`
	`c.valueOf.toString()`
	`// f valueOf() { [native code] }`

Date	`new Date(2050,1,13).toString()`
	`// "Sun Feb 13 2050 00:00:00 GMT+0300`
	`// (Московское время (зима))"`
	`new Date(2050,1,13).valueOf()  // 2528312400000`

RegExp	`/\d+/g.toString()  // "/\\d+/g"` `/\d+/g.valueOf()  // /\d+/g`

	Объект → Строка	**Объект → Число**
1.	`toString`() возвращает простое значение	`valueOf`() возвращает простое значение
2.	`valueOf`() возвращает простое значение	`toString`() возвращает простое значение
3.	исключение `TypeError`	исключение `TypeError`

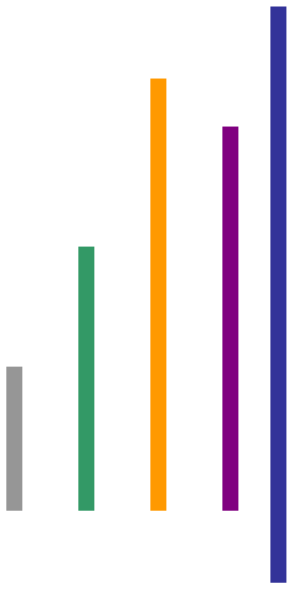
Пример. Каковы результаты выполнения операций?

```
r={x:1};
r+”123”
r+123
12+[1,2]
r-123
12-[1,2]
12-[1.2]
12-[1]
"123"+[7]
12+NaN
14+[NaN]
```

Проверь себя, смотри ответы

Тема. Области действия функций.

Область действия определяется функцией, а не кодовым блоком.
Область «инициализации» - от var до конца функции



```
/* код */
function outer() {
  /* код */
  var a = 1;
  /* код */
  function inner() { /* ------ */}
  var b = 2;
  /* код */
  if (a ==1) {
     var c = 3;
     /* код */
  }
  /* код */
}
/* код */
```

Ответы. Каковы результаты выполнения операций?

Операция	**Результат**
r={x:1};	
r+”123”	“[object Object]123”
r+123	“[object Object]123”
12+[1,2]	“121,2”
r-123	NaN
12-[1,2]	NaN
12-[1.2]	10.8
12-[1]	11
"123"+[7]	"1237"
12+NaN	NaN
14+[NaN]	14NaN

назад