网站开发基础

- Javasrcipt



前端最强攻！没有之一

- Javascript 介绍
- 基本语法
- 基本库 Jquery
- 框架 AngularJS, ReactJS
- 最强可视化 D3JS

Javascript 案例分析

- 吴亦凡广告案例，
- http://wefire.qq.com/act/a20150826kris/m/index.htm?ADTAG=a20150826kris.tiaozhuan.wx

Javascript 介绍

- Created in 10 days in May 1995 by Brendan Eich
- Originally developed as a prototype language for web browser (Client-side).
- Now used in server-side (Node.js) as well.
- Not related to Java, just named similarly for marketing purpose.
- C style syntax but got inspiration from Functional programming
- for, while, continue, break, if/else, switch are similar to C
- operators (+,-,*,/,%) are also similar (except ==,!=,||)
- include function operations such as map, reduce, forEach.

Note

- Javascript Next include versions of new syntax release annually (ES2015, ES2016, and ES2017). To use these new syntax, take a look at Babel.
- Typescript is a version of Javascript with static typing, built by Microsoft.

基本语法

- JavaScript and using the console.
- Variables， store and use numbers.
- Conditional, True and false
- Loop
- Function
- Objects
- Arrays

Data Type

- Numbers - 42, 3.14159
- Logical - true, false
- Strings - "Hello", 'Hello’
- null
- undefined* - Yes. undefined is not null!
- Objects
- functions

Variables

```
var a;
console.log("The value of a is " + a); // logs "The value of a is undefined"
console.log("The value of b is " + b); // throws ReferenceError exception
b = 5 // declare global variable  ... equivalent to window.b = 5
console.log(b, window.b) // 5, 5
```

Javascript uses dynamic typing.

```
x = "The answer is " + 42; // "The answer is 42"
"37" - 7; // 30
"37" + 7; // "377"
"1.1" + "1.1"; //= "1.11.1"
(+"1.1") + (+"1.1"); // 2.2
+"1.1" // 1.1 "+" operator converts string to number
```

Tips: operator can return Object too.

```
y = null || "String"
y // "String"
```

Control Flow

```
for (i=0; i<10; i++) {
  if (condition) {
      statement_1_runs_if_condition_is_true();
      break;
  } else {
      statement_2_runs_if_condition_is_false();
      continue;
  }
}
```

Object

- An object in javascript is an associative array of property names (Strings) and values (Objects).
- Everything except null and undefined) can be treated as objects.
- Object can be used as hashmaps.
- Object can be created using (a) literals (b) new

Object

```
//create object using an object literal
var apple = {
  // quotes not required for a property name
  state: "CA",
  // unless the name contains space(s) or precedes with number or the name is reserved word
  "famous founder": "Steve Jobs",
  // property value can be any type including function
  getCity: function(){return "Cupertino";},
  // nested object
  boards: { "CEO": "Tim Cook"}
};
apple.state; // "CA"
apple["state"]; // "CA"
apple.getCity(); // "Cupertino"
apple['getCity'](); // "Cupertino"
apple.boards.CEO // "Tim Cook"

//create object using new instead
var microsoft = new Object();
microsoft.state = "WA";
microsoft['famous founder'] = "Bill Gates";
microsoft.getCity = function(){ return "Redmond";};
```

=== and ==

- The == operator will compare for equality after doing any necessary type conversions (with very weird conversion rules).

- The === operator will not do the conversion, so if two values are not the same type === will simply return false.

for .. in

```
var obj = {a:1, b:2, c:3}, key;
Object.prototype.crazy = 4;
for (key in obj) {
  if(obj.hasOwnProperty(key)){ //avoid the inherited obj.crazy=4
    console.log(key, obj[key]); // "a, 1" "b, 2", "c, 3"
  }
}
```

Arrays

```
var numbers = [ 5, 10, 15, 20, 25];
numbers[1] // 10
numbers[2] // 15
numbers.length = 5
2 in numbers // true
5 in numbers // false
numbers.a = 5 // Array is an object
numbers.a // 5
'a' in numbers // true

var numbers = [ 5, 10, , 20, 25, , ];
numbers[1] // 10
numbers[2] // undefined
numbers.length //6 last empty comma is automatically ignored

// array with multiple types of values
var mashup = [ 1,3, 4.5, 5.6, "string", null, undefined, true ];
```

Function

```
function foo(a1, a2){
  // code
  console.log(a1+a2);
}
```

same as

```
var foo = function(a1, a2){
  // code
  console.log(a1+a2);
};

foo(a1,a2); // call

function g(f){ f(1,2); }
g(foo) // 3
```

Function level SCOPE

```
var name = "Andy"
var pet = function(name) {       // The outer function defines a variable called "name"
  var getName = function() {
    return name;                 // The inner function has access to the "name" variable of the outer
  }

  return getName;                // Return the inner function, thereby exposing it to outer scopes
};
var myPet = pet("Vivie");
myPet();                         // Returns "Vivie"
console.log(name);               // "Andy"
```

玩转DOM

- The DOM as a tree of elements.
- Listening for events

DOM Tree

- Document and Window Objects
- document.getElementById
- document.createElement
- document.createNode
- document.querySelector,document.querySelectorAll

IIFE, Immediately Invoke Function

```
var elems = document.getElementsByTagName( 'a' )

for ( var i = 0; i < elems.length; i++ ) {

  elems[ i ].addEventListener( 'click', function
    e.preventDefault();
    alert( 'I am link #' + i );
  }, 'false' );
}
```

Jquery

- DOM library
- $(). That is all.
- $.get, $.post, and $.ajax.
- jQuery: Other Tricks:

- Inserting elements into the DOM

JS + HTML5

- Canvas
- Local Storage
- Regular Expressions

Angular Fight with ReactJS

- Angular: As above, but with Angular.

画图谁最牛？

- http://uwdata.github.io/d3-tutorials/
- http://uwdata.github.io/d3-tutorials/fundamental.html
- https://bl.ocks.org/mbostock

异类！！NodeJS

- Node.js: Javascript… on the server? What is this madness?

```
var http = require('http');

var server = http.createServer(function
(req, res) {
    res.writeHead(200, {'Content-Type':
'text/plain'});
    res.end('Hello World\n');
})

server.listen(1337, '127.0.0.1');

console.log('Server running at
http://127.0.0.1:1337/');
```

谢谢

beijing@dataapplab.com