AD1026
Front-end Development Advance
Session 3: JavaScript

Rex Chen @ 2011/12/26
Enterprise, Sr. Engineer

TREND MICRO

Writing a reusable component

Basic Knowledge

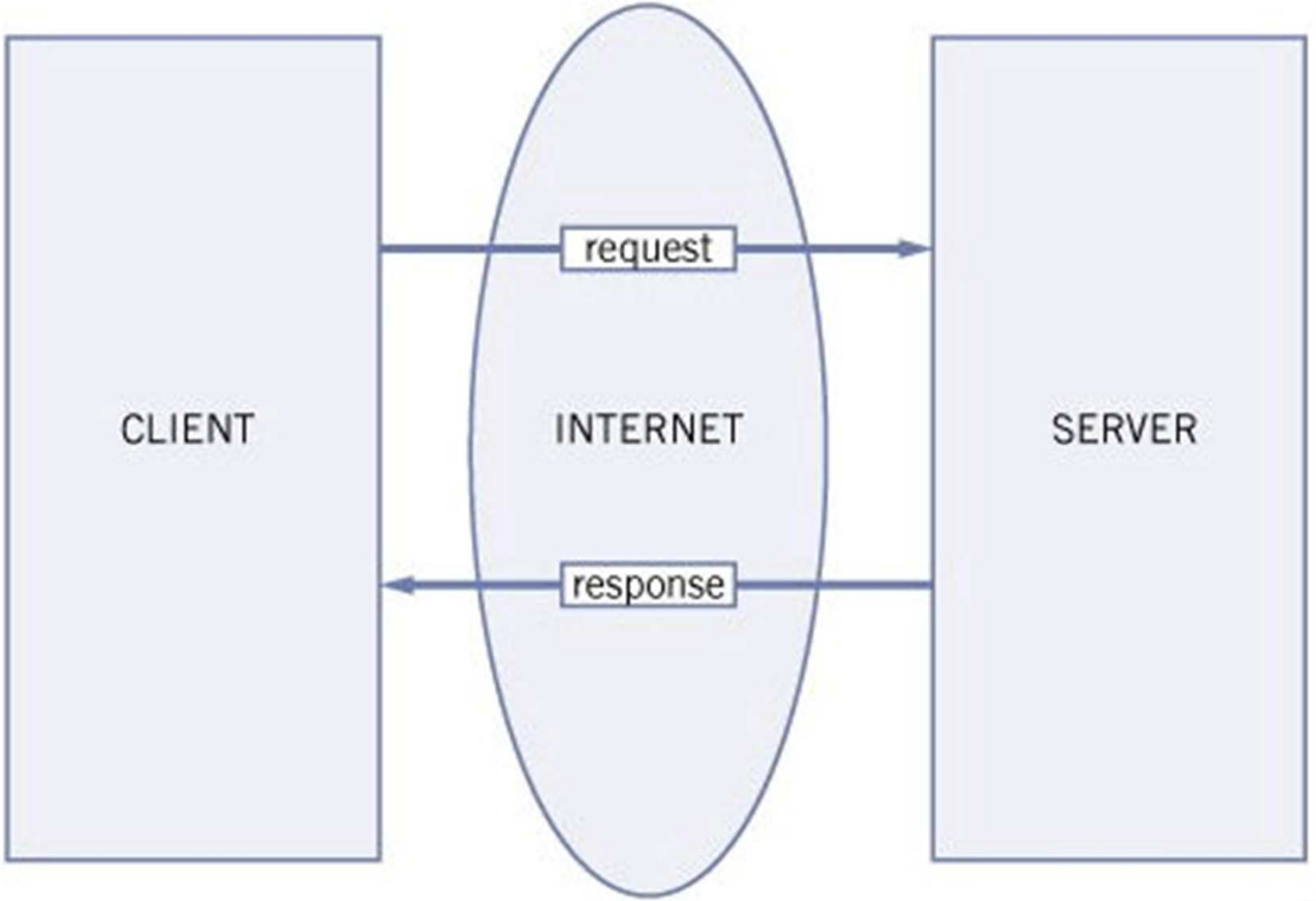
Popularity of Programming Languages

Position Dec 2011	Position Dec 2010	Delta in Position	Programming Language	Ratings Dec 2011	Delta Dec 2010	Status
1	1	=		17.561%	-0.44%	A
2	2	=		17.057%	+0.98%	A
3	3	=		8.252%	-0.76%	A
4	5	↑		8.205%	+1.52%	A
5	8	↑↑↑		6.805%	+3.56%	A
6	4	↓↓		6.001%	-1.51%	A
7	7	=		4.757%	-0.36%	A
8	6	↓↓		3.492%	-2.99%	A
9	9	=		2.472%	+0.14%	A
10	12	↑↑		2.199%	+0.69%	A

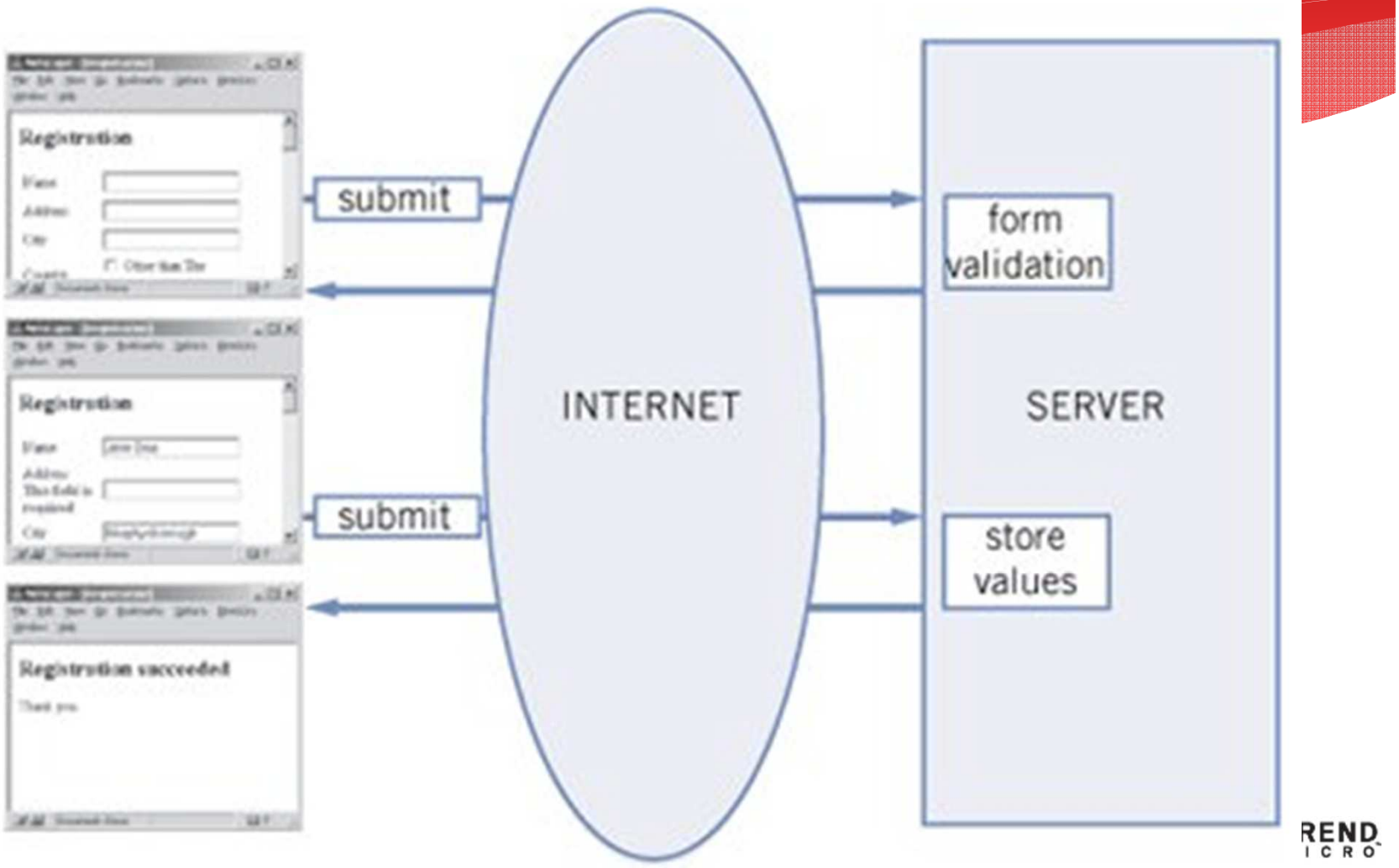
資料來源: TIOBE
http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

TREND MICRO

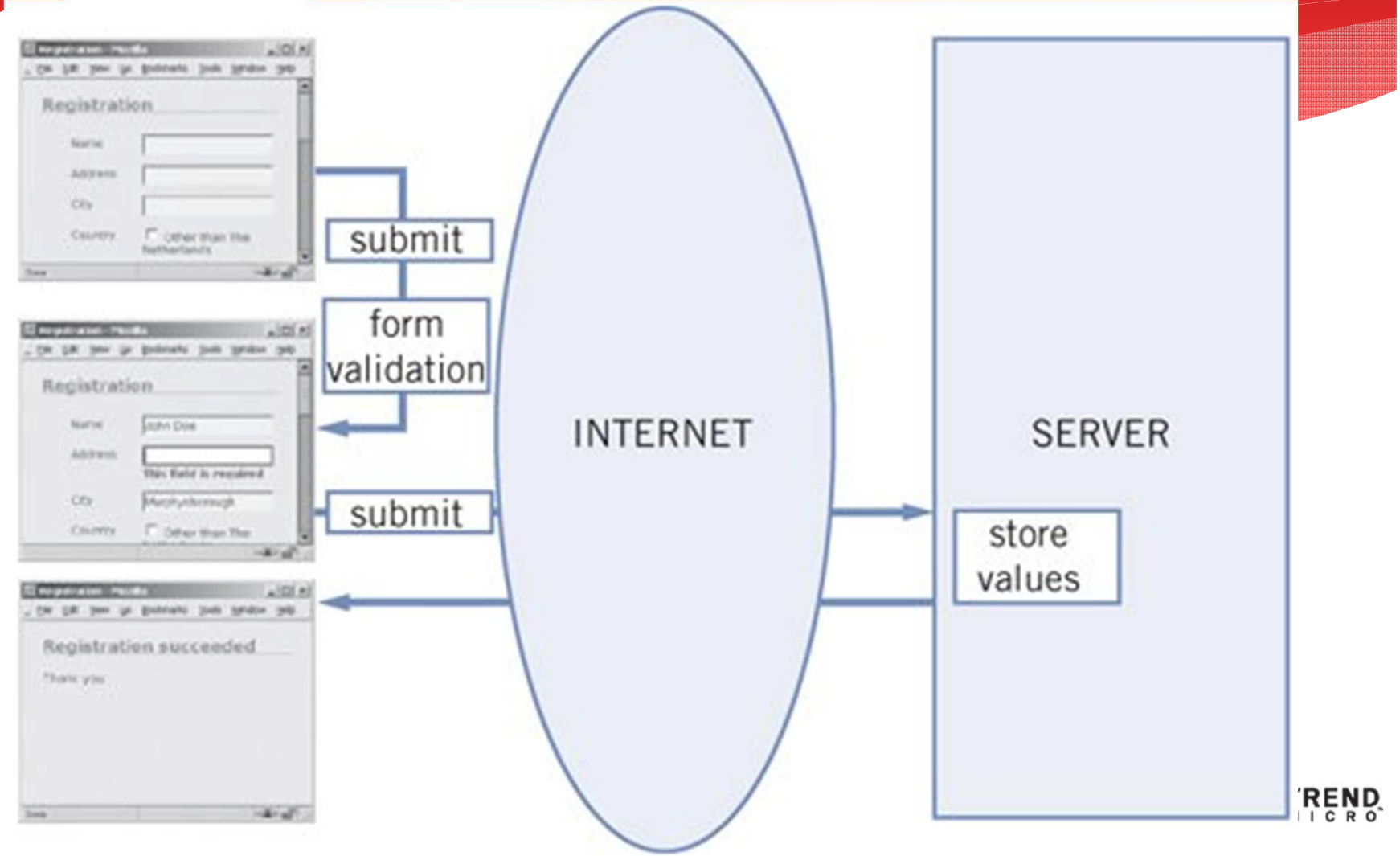
Client-server communication



Form validation without JavaScript



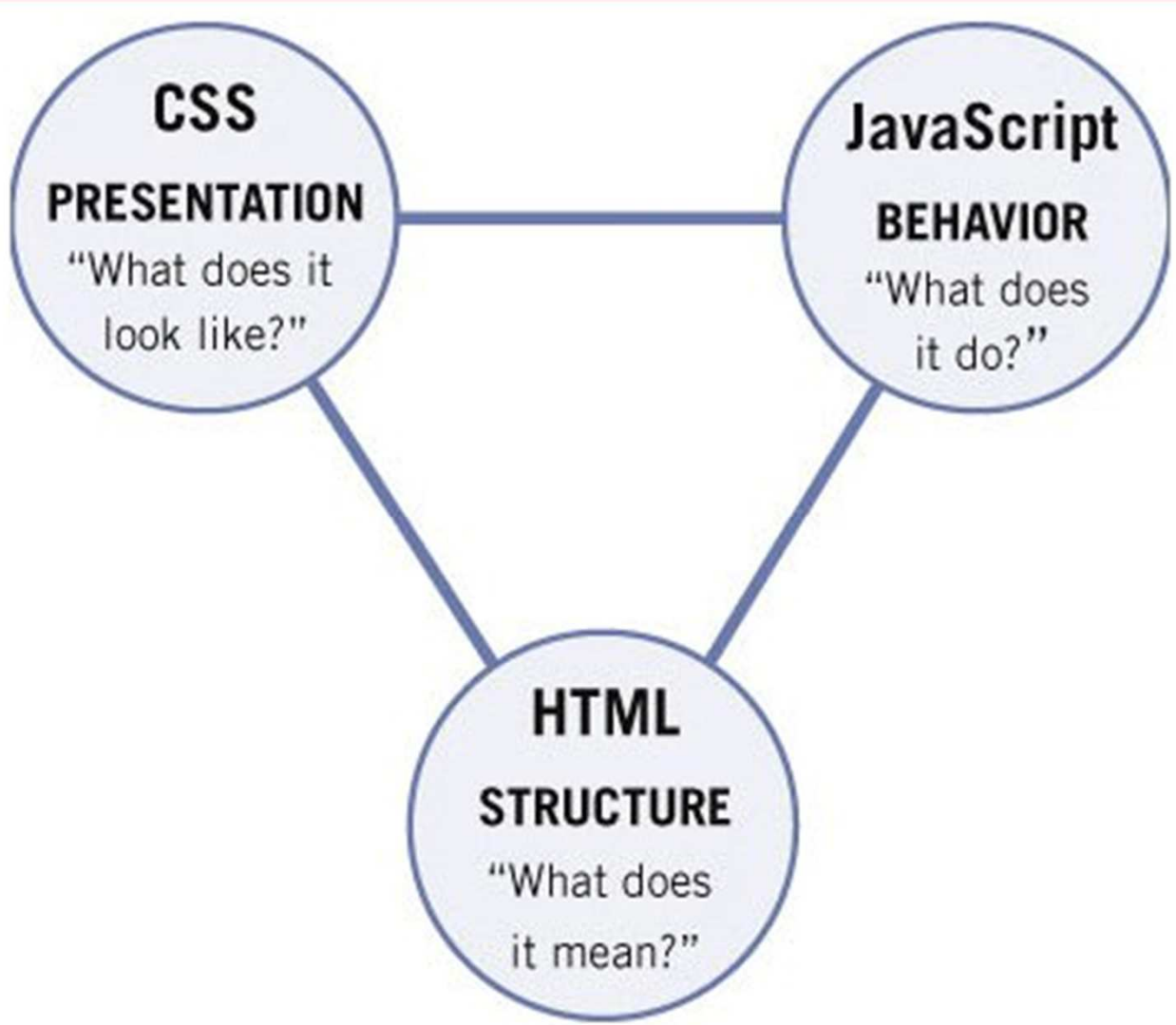
Form validation with JavaScript



Which approach is better?

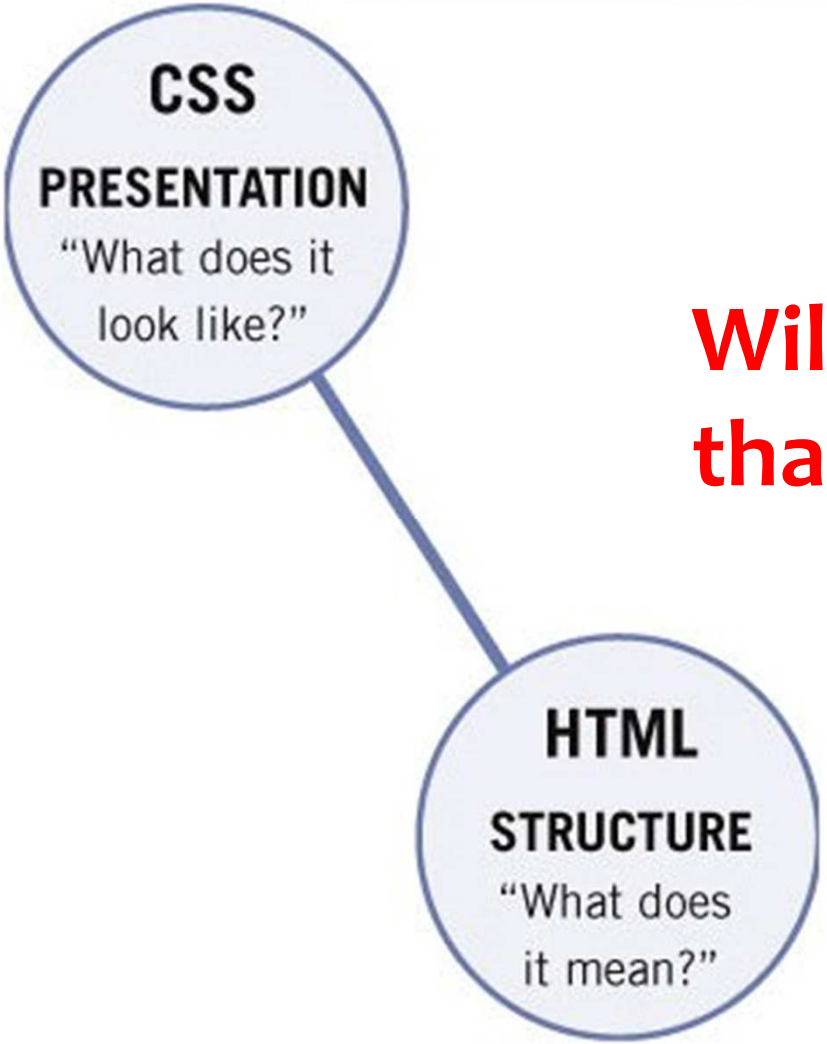
It depends.

The three layers of a Web page



TREND MICRO

If Javascript is disabled



Will your site survive that experience?

TREND MICRO

Bad Sample

```
< !-- tightly coupled HTML/JavaScript using < script > -- >
< script type=”text/javascript” >
document.write(“Hello world!”);
< /script >
< !-- tightly coupled HTML/JavaScript using event handler attribute -- >
< input type=”button” value=”Click Me” onclick=”doSomething()” / >
```

Bad Sample

```
//tight coupling of HTML to JavaScript
function insertMessage(msg){
   var container = document.getElementById(“container”);
container.innerHTML = “ < div class=\”msg\” > < p class=\”post\” >
” + msg + “ < /p > ” +
“ < p > < em > Latest message above. < /em > < /p > < /div > ”;
}
```

Bad Sample

```
//tight coupling of CSS to JavaScript
element.style.color = “red”;
element.style.backgroundColor = “blue”;
```

TREND MICRO

Practice 3 layers structure

TREND MICRO

Rules – Best Practice

Minimizing Globals

- A third-party JavaScript library
- Scripts from an advertising partner
- Code from a third-party user tracking and analytics script
- Different kinds of widgets, badges, and buttons
- Only one global variable

var Declaration

- never forgetting var declaration

TREND MICRO

var Declaration

a, b, sum, myobject, i, j

TREND MICRO

```
function func() {
    var a = 1,
    b = 2,
    sum = a + b,
    myobject = {},
    i,
    j;
    // function body...
}
```

Single var Pattern

- Provides a single place to look for all the local variables needed by the function
- Prevents logical errors when a variable is used before it's
- Helps you remember to declare variables and therefore minimize globals
- Is less code

TREND MICRO

for Loops

```
//sample for loop
for (var i = 0; i < myarray.length; i++) {
   // do something with myarray[i]
}
```

for Loops efficiency

TREND MICRO

for Loops

```
while (i--) {
   // do something with myarray[i]
}
```

* Use one less variable(no max)
* Count down to 0, which is usually faster because it's more efficient to compare to 0 than to the length of the array or to anything other than 0

TREND MICRO

Switch Pattern

```
var inspect_me = 0,
    result = '';
switch (inspect_me) {
case 0:
    result = "zero";
    break;
case 1:
    result = "one";
    break;
default:
    result = "unknown";
}
```

Avoid Implied Typecasting

```
var zero = 0;
if (zero === false) {
   // not executing because zero is 0, not false
}
// antipattern
if (zero == false) {
   // this block is executed...
}
```

Avoid eval()

* eval() is evil.

parseInt()

* parseInt(number, radix)

* The problems occur when the string to parse starts with 0

TREND MICRO

Coding Conventions

function and method

* Each function or method should include a comment that describes its purpose and possibly the algorithm being used to accomplish the task. It ' s also important to state assumptions that are being made, what the arguments represent, and whether or not the function returns a value (since this is not discernible from a function definition).

Large sections of code

* Multiple lines of code that are all used to accomplish a single task should be preceded with a comment describing the task.

Complex algorithms

* If you ’ re using a unique approach to solve a problem, explain how you are doing it as a comment. This will not only help others who are looking at your code, but will also help you the next time you look at it.

Hacks

* Because of browser differences, JavaScript code typically contains some hacks. Don't assume that someone else who is looking at the code will understand the browser issue that such a hack is working around. If you need to do something differently because one of the browsers can ’ t use the normal way, put that in a comment. It reduces the likelihood that someone will come along, see your hack, and “ fix ” it, inadvertently introducing the bug that you had already worked around.

Indentation

* 4 spaces

TREND MICRO

Opening Brace Location

```
if (true) {
    alert("It's TRUE!");
}
Or:
if (true)
{
    alert("It's TRUE!");
}
```

TREND MICRO

```
function func() {
    return
    {
        name: "Batman"
    };
}
```

```
function func() {
    return undefined;
    {
        name: "Batman"
    };
}
```

White Space

* After the semicolons that separate the parts of a for loop: for example, for (var i = 0; i < 10; i += 1) {...}
* Initializing multiple variables (i and max) in a for loop:

  for (var i = 0, max = 10; i < max; i += 1) {...}
* After the commas that delimit array items: var a = [1, 2, 3];
* After commas in object properties and after colons that divide property names and their values: var o = {a: 1, b: 2};

White Space

* Delimiting function arguments: myFunc(a, b, c)
* Before the curly braces in function declarations:

  function myFunc() {}
* After function in anonymous function expressions:

  var myFunc = function () {};

Use Array and Object Literals

```
//four statements to create and initialize array - wasteful
var values = new Array();
values[0] = 123;
values[1] = 456;
values[2] = 789;
//four statements to create and initialize object - wasteful
var person = new Object();
person.name = “Nicholas”;
person.age = 29;
person.sayName = function(){
   alert(this.name);
};
```

Use Array and Object Literals

```
//one statement to create and initialize array
var values = [123, 456, 789];
//one statement to create and initialize object
var person = {
    name : "Nicholas",
    age : 29,
    sayName : function(){
        alert(this.name);
    }
};
```

Naming Conventions

Naming Conventions

* Variable names should be nouns such as car or person .
* Function names should begin with a verb such as getName() . Functions that return Boolean values typically begin with is , as in isEnabled() .
* Use logical names for both variables and functions, without worrying about the length. Length can be mitigated through post - processing and.

TREND MICRO

Variable Type Transparency

```
//variable type indicated by initialization
var found = false; //Boolean
var count = -1; //number
var name = “”; //string
var person = null; //object

//Hungarian notation used to indicate data type
var bFound; //Boolean
var iCount; //integer
var sName; //string
var oPerson; //object
```

Naming Conventions

* Capitalizing Constructors

  var rex = new Person();
* Separationg Words

  getFirstName();
* CONSATNT

  var MAX_WIDTH = 800;

Naming Conventions

- Private function

```
var Person = {
    getName: function () {
        return this._getFirst() + ' ' + this._getLast();
    },
    _getFirst: function () {
        // ...
    },
    _getLast: function () {
        // ...
    }
};
```

TREND MICRO

Naming Conventions

* Writing Comments
* Writing API Docs

```
/**
* Reverse a string
*
* @param {String} input String to reverse
* @return {String} The reversed string
*/
var reverse = function (input) {
   // ...
   return output;
};
```

TREND MICRO

Object Creation Patterns

```
// global object
var MYAPP = {};
// constructors
MYAPP.Parent = function () {};
MYAPP.Child = function () {};
// a variable
MYAPP.CONSTANT = 1;
// an object container
MYAPP.modules = {};
// nested objects
MYAPP.modules.Module1 = {};
MYAPP.modules.Module1.data = {a: 1, b: 2};
MYAPP.modules.Module2 = {};
```

Namespace Pattern

TREND MICRO

```
var MYAPP = MYAPP || {};
MYAPP.namespace = function (ns_string) {
    var parts = ns_string.split('.'),
        parent = MYAPP,
        i;
    // strip redundant leading global
    if (parts[0] === "MYAPP") {
        parts = parts.slice(1);
    }
    for (i = 0; i < parts.length; i += 1) {
        // create a property if it doesn't exist
        if (typeof parent[parts[i]] === "undefined") {
            parent[parts[i]] = {};
        }
        parent = parent[parts[i]];
    }
    return parent;
};
```

Practice
Namespace Pattern

TREND MICRO

Module Pattern

TREND MICRO

Module Pattern

* Namespaces
* Immediate functions
* Private and privileged members
* Declaring dependencies

```
MYAPP.utilities.array = (function () {
    // dependencies
    // private properties
    // private method

    // public API
    return {
        inArray: function (needle, haystack) {
            // ...
        },
        isArray: function (a) {
            // ...
        }
    };
}());
```

Inherit

Default Pattern

```
function inherit(Child, Parent) {
    Child.prototype = new Parent();
}
```

TREND MICRO

Practice Inherit

Singleton Pattern

```
var obj = {
    property: 'value'
};
```

TREND MICRO

DOM access

```
// antipattern
for (var i = 0; i < 100; i += 1) {
    document.getElementById("result").innerHTML += i + ", ";
}
// better - update a local variable
var i, content = "";
for (i = 0; i < 100; i += 1) {
    content += i + ",";
}
document.getElementById("result").innerHTML += content;
```

TREND MICRO

DOM Manipulation

```
var p, t;
p = document.createElement('p');
t = document.createTextNode('first paragraph');
p.appendChild(t);
document.body.appendChild(p);

p = document.createElement('p');
t = document.createTextNode('second paragraph');
p.appendChild(t);
document.body.appendChild(p);
```

DOM Manipulation

```
var p, t, frag;
frag = document.createDocumentFragment();
p = document.createElement('p');
t = document.createTextNode('first paragraph');
p.appendChild(t);
frag.appendChild(p);

p = document.createElement('p');
t = document.createTextNode('second paragraph');
p.appendChild(t);
frag.appendChild(p);
document.body.appendChild(frag);
```

JSLint

http://www.jslint.com/

YUI Compressor

http://developer.yahoo.com/yui/compressor/

TREND MICRO

Conclusion

The Maintainable code means code that:

* Is readable
* Is consistent
* Is predictable
* Looks as if it was written by the same person
* Is documented
* Is extendable
* Is debuggable

TREND MICRO

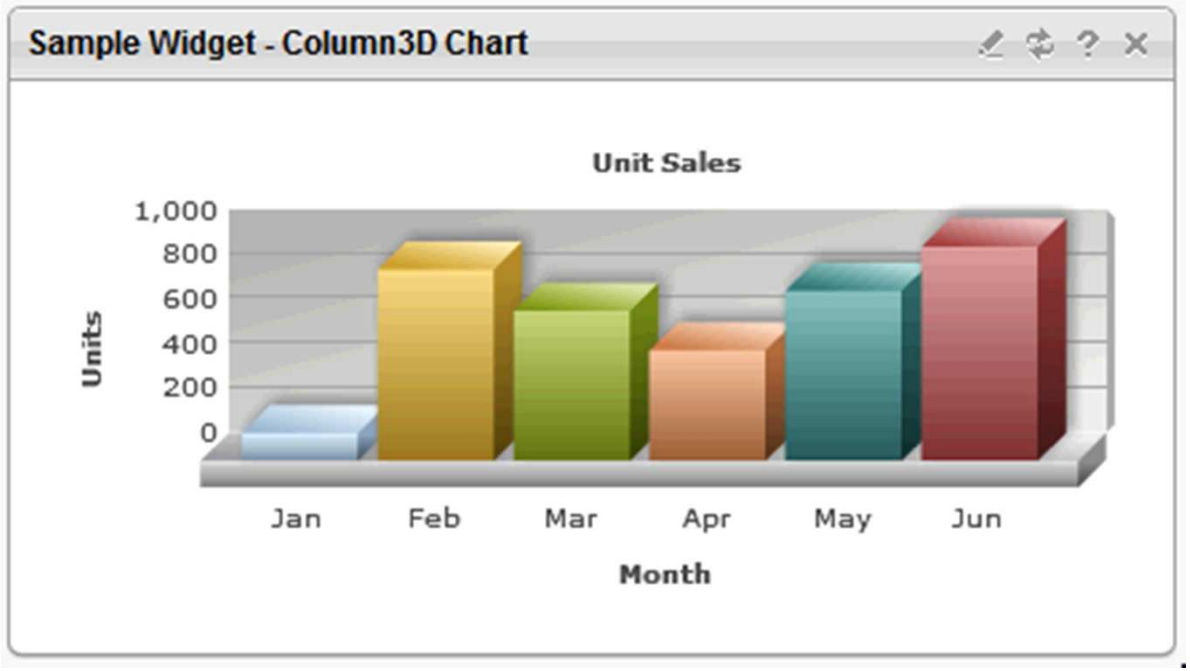
Questions?

TREND MICRO

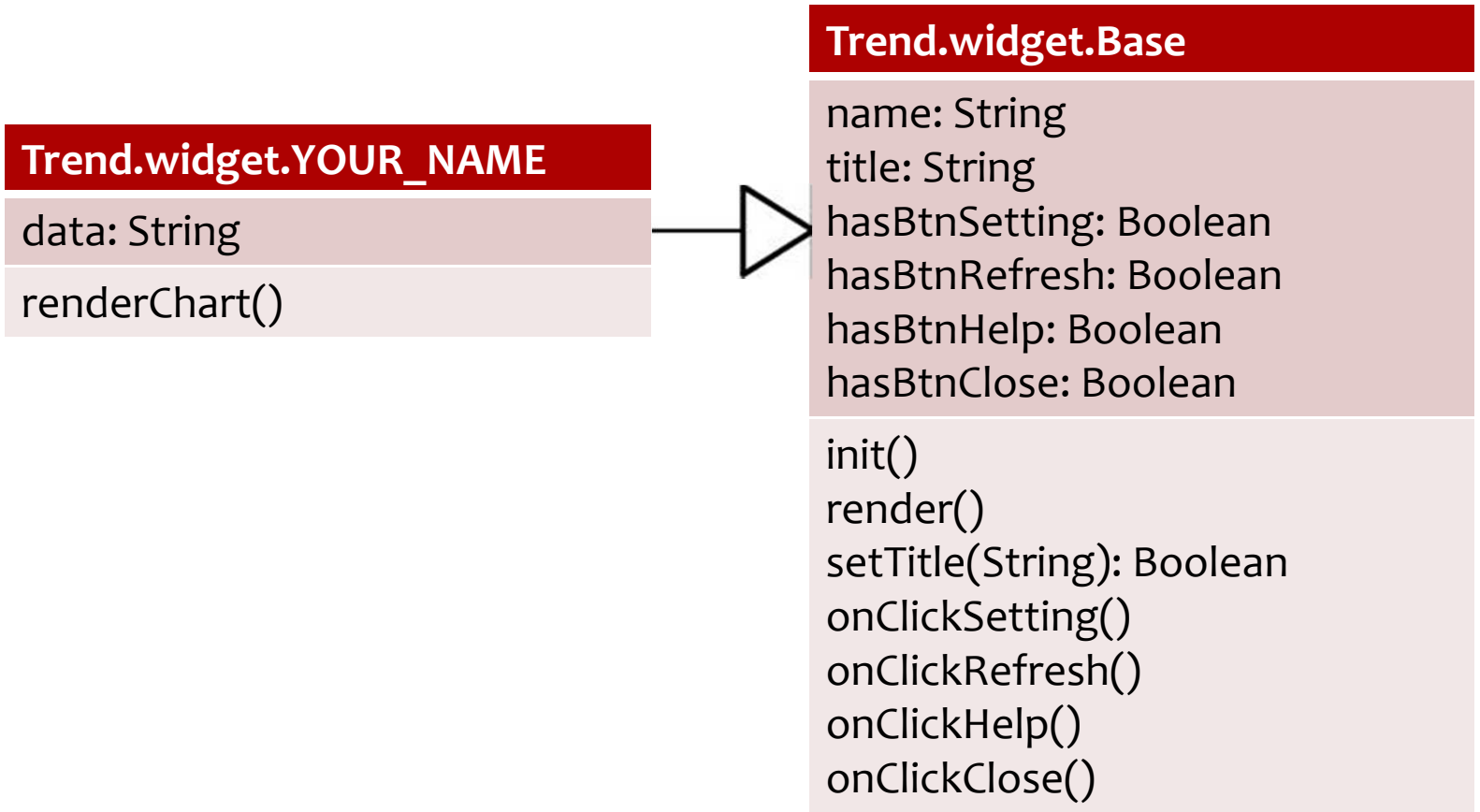
Homework

TREND MICRO

Mockup


Sample Widget - Column3D Chart
Unit Sales
1,000
800
600
400
200
0
Units
Jan
Feb
Mar
Apr
May
Jun
Month

Class Diagram



TREND MICRO

Commit before 2012/01/02 00:00

TREND MICRO