The Jasmine Effect

Automate; Don't Panic.

Jasmine's Got Your Back

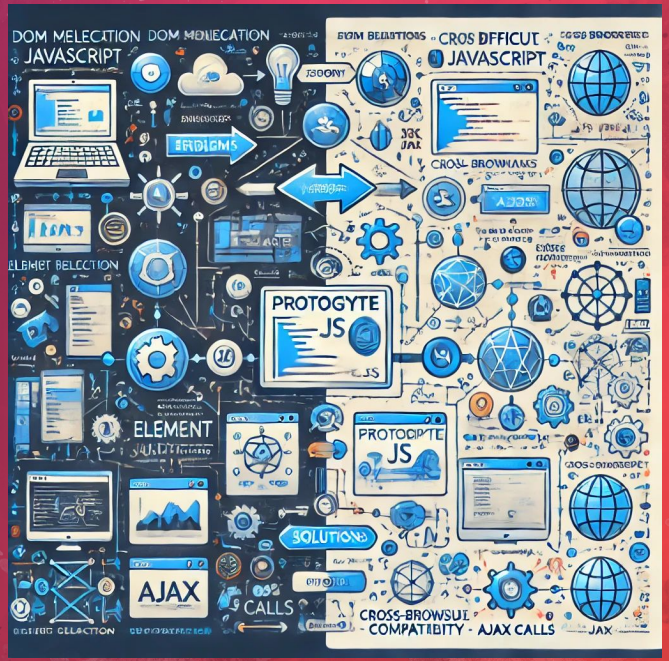
Jasmine is a lightweight testing framework for JavaScript.
Think of it as a detective who catches bugs before they cause trouble.

- Automated, consistent testing.
- Catching bugs early in development.
- Improving script quality and maintainability.

Prototype

Created by **Sam Stephenson** in 2005 to simplify and improve the process of writing JavaScript for web development.
It provided various utilities and extensions to the JavaScript language, such as:

- Extending built-in JavaScript objects like Array, String, and Function.
- Easier DOM traversal and manipulation.
- Cross-browser support.



ServiceNow's Secret Sauce

Enter Prototype.js Code

```
var Person = Class.create();
Person.prototype = {
  initialize: function(name) {
    this.name = name;
  },
  say: function(message) {
    return this.name + ': ' + message;
  }
};

var guy = new Person('Miro');
guy.say('hi');
```

```
Miro: hi
```

```
1  var Person = Class.create();
2  Person.prototype = {
3      initialize: function(personName) {
4          this.personName = personName;
5      },
6      say: function(message) {
7          return this.personName + ': ' + message;
8      },
9
10     type: 'Person'
11 };
```

```
1  var guy=new Person("Miro");
2  gs.info(guy.say("hi"));
```

```
*** Script: Miro: hi
```

ServiceNow's Secret Sauce

Enter Prototype.js Code

```
var Person = Class.create();
Person.prototype = {
  initialize: function(name) {
    this.name = name;
  },
  say: function(message) {
    return this.name + ': ' + message;
  }
};

var Pirate = Class.create();

// inherit from Person class:
Pirate.prototype = Object.extend(new Person(), {

  // redefine the speak method
  say: function(message) {
    return this.name + ': ' + message + ', yarr!';
  }

});

var john = new Pirate('Long John');
john.say('ahoy matey');
```

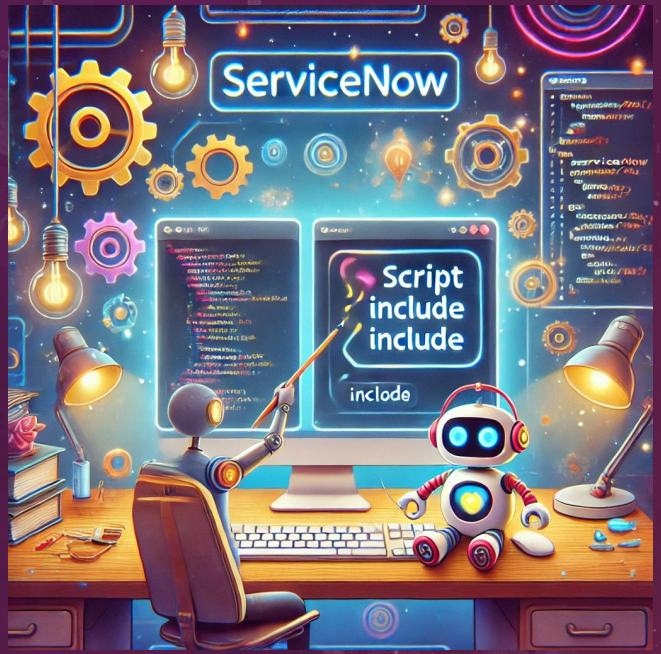
```
Long John: ahoy matey, yarr!
```

```
1  var Pirate = Class.create();
2  Pirate.prototype = Object.extendsObject(Person, {
3
4      say: function(message) {
5          return this.personName + ': ' + message + ', yarr!';
6      },
7
8      type: 'Pirate'
9  });
```

```
1  var john = new Pirate('Long John');
2  gs.info(john.say("ahoy matey"));
```

```
*** Script: Long John: ahoy matey, yarr!
```

Script Includes



- Script Includes are reusable server-side script logic that define a function or class.
- Script Includes execute their script logic only when explicitly called by other scripts.
- They are a powerful way to write code that can be called from multiple places, making your instance more modular, maintainable, and efficient.

Script Includes Patterns

```
var ParentSI = Class.create();
ParentSI.prototype = {
    initialize: function() {
        gs.info("initialize: ParentSI method.");
    },
    parentFunc: function() {
        gs.info("parentFunc: ParentSI method.");
    },
    funcToBeOverriden: function() {
        gs.info("funcToBeOverriden: ParentSI method.");
    },

    type: 'ParentSI'
};
```

```
function sumTwoNums(int1, int2) {
    var mySum = int1 + int2;
    gs.info("The sum of " + int1 + " + " + int2 + "= " + mySum);
}
```

Script Includes Patterns

```
var Person = Class.create();
Person.prototype = {
    initialize: function(personName) {
        this.personName = personName;
    },
    say: function(message) {
        return this.personName + ': ' + message;
    },
    type: 'Person'
};
```

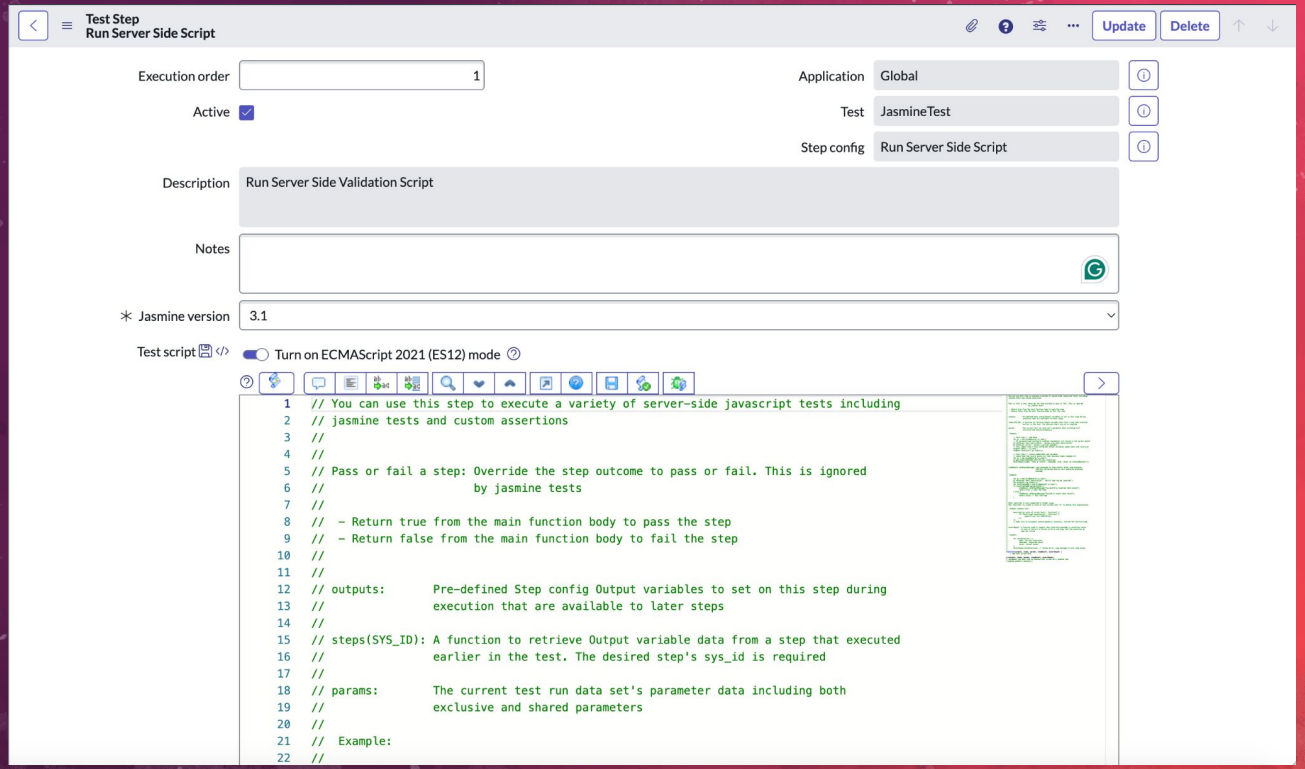
```
var Pirate = Class.create();
Pirate.prototype = Object.extendsObject(Person, {
    say: function(message) {
        return this.personName + ':' + message + ', yarr!';
    },
    type: 'Pirate'
});
```

Script Includes Patterns

```
var ChildSI = Class.create();
ChildSI.prototype = Object.extendsObject(ParentSI, {

    initialize: function() {
        gs.info("initialize: ChildSI method.");
        ParentSI.prototype.initialize.call(this);
    },

    childFunc: function() {
        gs.info("childFunc: ChildSI method.");
    },
    funcToBeOverriden: function() {
        gs.info("funcToBeOverriden: ChildSI method.");
    },


    type: 'ChildSI'
});
```

```
var ConstantsSI = Class.create();
ConstantsSI.myConstVar = "myConstVar";
ConstantsSI.prototype = {
    myVar: "myVar",
    initialize: function() {},
    myFunc: function() {
        gs.info("myFunc");
    },

    type: 'ConstantsSI'
};
ConstantsSI.prototype.varFunc = function() {
    gs.info("varFunc");
};
ConstantsSI.constFunc = function() {
    gs.info("constFunc");
};
```

Run Server Side Script



References

- http://prototypejs.org/
- https://beingfluid.github.io/ptototypejstest/
- Objects: Making It A Little Easier...
- Script Includes
- NOWCommunity Live Stream - Code Decoded - Using a Constants script include
- Live Coding Happy Hour for 2017-02-10 - Server Scripts with Automated Test Framework
- Unit testing your Javascript with jasmine
- https://jasmine.github.io/