

JavaScript: The Good Parts Bookclub

Chapter 1: Good Parts

History:

[JavaScript Timeline](#)

[Evolution of the Web](#)

[Browser Timeline](#)

[Dynamic vs Static languages](#)

Browser Dev Status:

[Chrome](#)

[Mozilla](#)

[Webkit](#)

[Edge](#)

Browser Implementation Status:

[What the Web Can Do Today](#)

[Browserscope](#)

[Quirksmode](#)

JavaScript Status:

[ES6 Perf](#)

[ES6 Compat](#)

Browser Release Status:

[Chrome Channel Releases](#)

[Safari Technology Preview](#)

[Firefox Channel](#)

[Edge Changelog](#)

Chapter 2: Grammar

Strings:

[Pile of poo test](#)

Expressions:

<http://rainsoft.io/javascriptss-addition-operator-demystified/>

[Wat](#)

[Wat Explained](#)

[ToPrimitive](#)

[JS equality table](#)

[JS comparison example](#)

Try and Catch:

[Asynchronous Problem Example](#)

[Promises](#)

[Async / Await](#)

Chapter 3: Objects

Enumeration:

[The Dictionary Pattern](#)

Chapter 4: Functions

Constructor Invocation Pattern:

[Video tries to 'fix' problems with the function constructor](#), how did it fail?

Closures:

[Cute Closure Analogy](#)

Chapter 5: Inheritance

Pseudoclassical:

```
Super = function () {...};  
Super.prototype.constructor=Super;  
Super.prototype.methodA = ...;  
Super.prototype.methodB = ...;  
  
Sub = function () {...};  
Sub.prototype = Object.create(Super.prototype);  
Sub.prototype.constructor = Sub; //need to fix manually  
Sub.prototype.methodB = ...;  
Sub.prototype.methodC = ...;
```

Source: The Awesome [Speaking JS](#) Book Free Online

Sub.prototype

methodA	...
methodB	...
methodC	...

[[Prototype]]	●
prop1	...
prop2	...
prop3	...
prop4	...

subInstance

Super.prototype

methodA	...
methodB	...

[[Prototype]]	●
prop1	...
prop2	...

superInstance



Chapter 5: Inheritance

Pseudoclassical: Instance of Super Constructor Used As Prototype Antipattern

Before ECMAScript 5 and `Object.create()`, an often-used solution was to create the subprototype by invoking the superconstructor:

`Sub.prototype = new Super(); // Don't do this`

This is not recommended under ECMAScript 5. The prototype will have all of Super's instance properties, which it has no use for. Instead use `Object.create` to make a new instance on the prototype chain:

`Sub.prototype = Object.create(Super.prototype); // Do this`

Don't forget to fix constructor:

`Sub.prototype.constructor = Sub;`

Chapter 5: Inheritance Example

<https://rajsite.github.io/good-parts-notes/inheritance/>