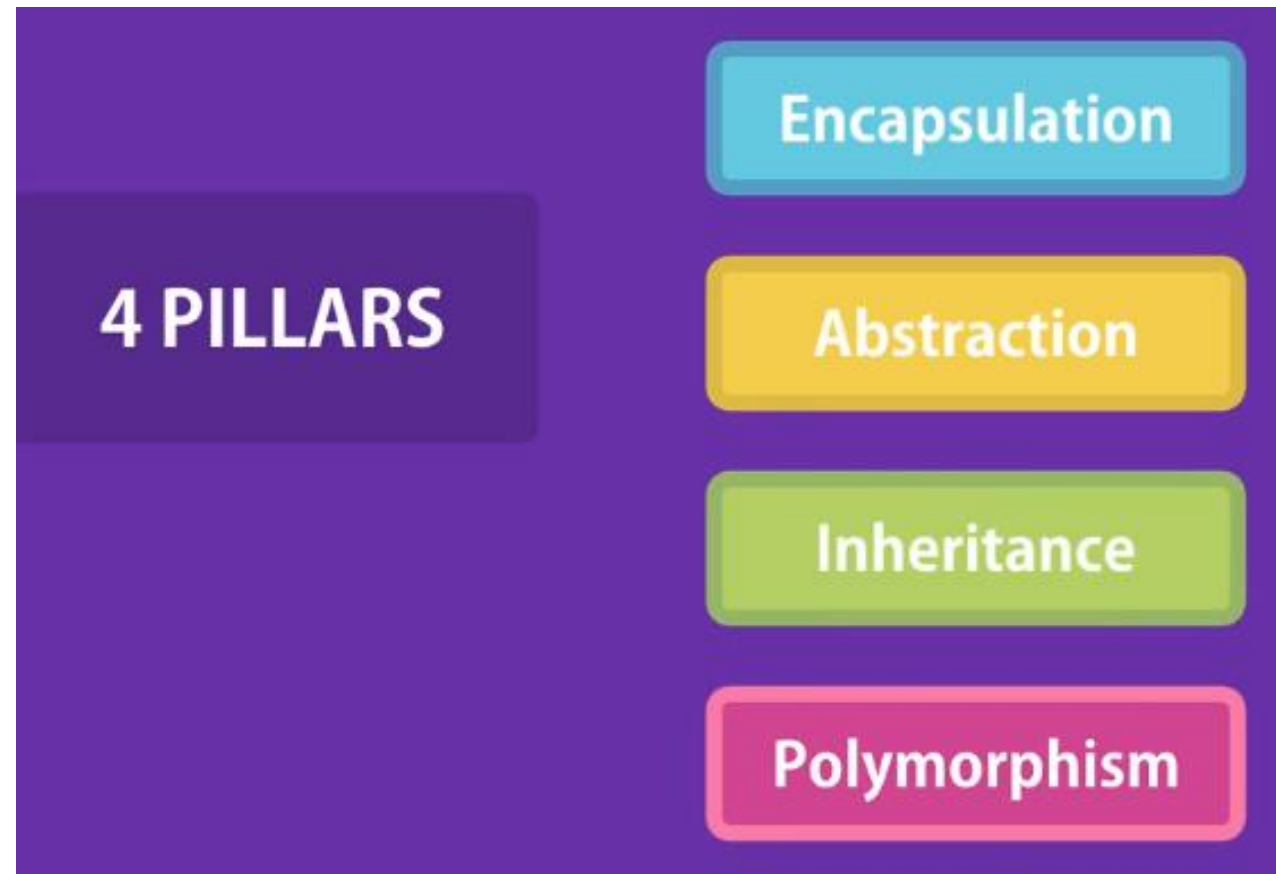


Node

JS OOP

4 Pillars of OOP



Encapsulation

**“The best functions are those
with no parameters!”**

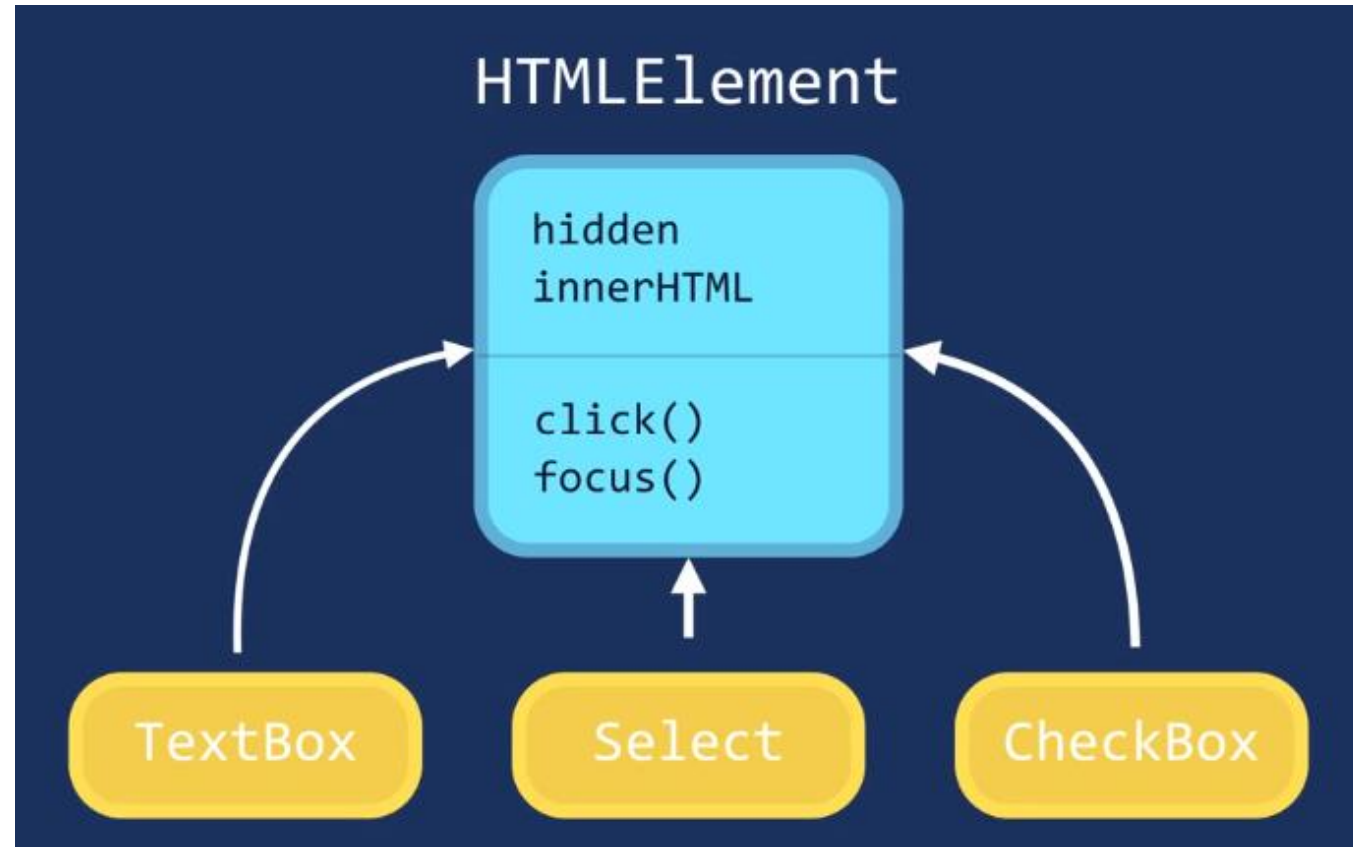
Uncle Bob - Robert C Martin

Abstraction

Hide Complex Implementation Details

- Clean your interface

Inheritance

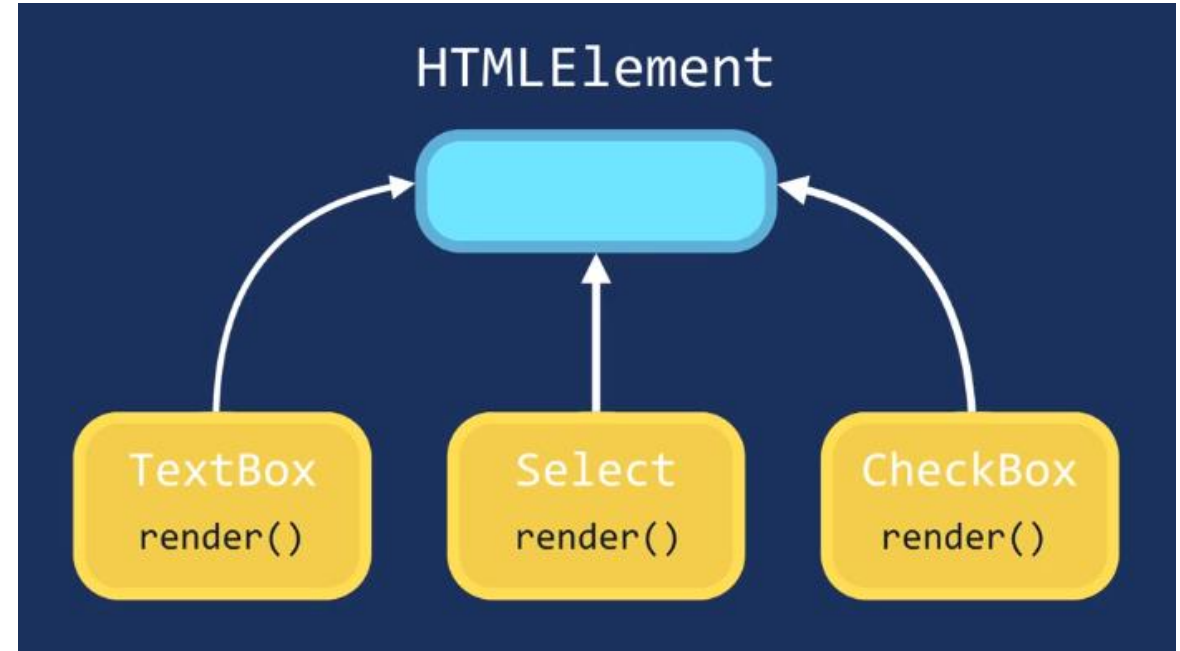


Poly Morphism

```
switch (...) {  
  case 'select': renderSelect();  
  case 'text': renderTextBox();  
  case 'checkbox': renderCheckBox();  
  case ...  
  case ...  
  case ...  
}
```

Poly Morphism

```
switch (...) {  
  case 'select': renderSelect();  
  case 'text': renderTextBox();  
  case 'checkbox': renderCheckBox();  
  case ...  
  case ...  
  case ...  
}
```



Why OOP

Encapsulation

Reduce complexity + increase reusability

Abstraction

Reduce complexity + isolate impact of changes

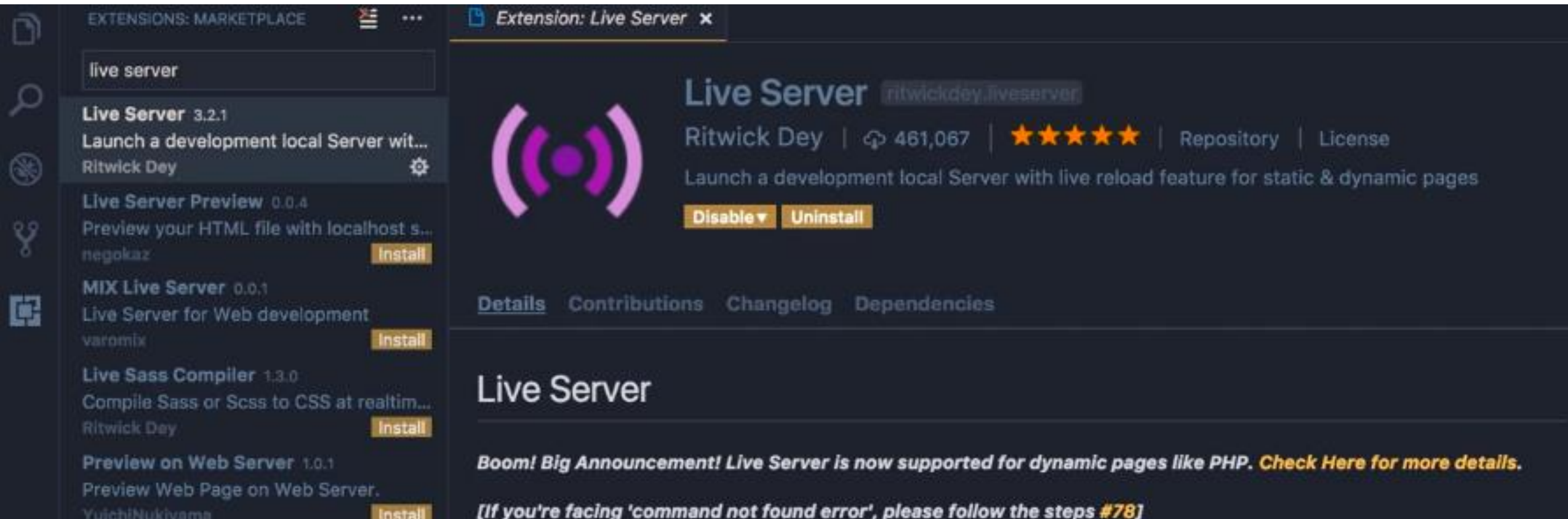
Inheritance

Eliminate redundant code

Polymorphism

Refactor ugly switch/case statements

Development Environment



The screenshot displays the Visual Studio Code interface with the Extensions Marketplace open. The search bar contains 'live server'. The left sidebar lists several extensions, with 'Live Server 3.2.1' by Ritwick Dey at the top. The main panel shows the details for the 'Live Server' extension, including its icon (a purple signal tower), the author 'Ritwick Dey', a download count of 461,067, and a five-star rating. Below the rating are buttons for 'Disable' and 'Uninstall'. The extension's description states: 'Launch a development local Server with live reload feature for static & dynamic pages'. At the bottom of the main panel, there is a large announcement: 'Boom! Big Announcement! Live Server is now supported for dynamic pages like PHP. Check Here for more details.' followed by a note: '[If you're facing 'command not found error', please follow the steps #78]'. The sidebar also shows other extensions like 'Live Server Preview', 'MIX Live Server', 'Live Sass Compiler', and 'Preview on Web Server'.

EXTENSIONS: MARKETPLACE

live server

Live Server 3.2.1
Launch a development local Server wit...
Ritwick Dey

Live Server Preview 0.0.4
Preview your HTML file with localhost s...
negokaz

MIX Live Server 0.0.1
Live Server for Web development
varomix

Live Sass Compiler 1.3.0
Compile Sass or Scss to CSS at realtim...
Ritwick Dey

Preview on Web Server 1.0.1
Preview Web Page on Web Server.
YuichiNukiyama

Live Server ritwickdey.liveserver

Ritwick Dey | 461,067 | ★★★★★ | Repository | License

Launch a development local Server with live reload feature for static & dynamic pages

Disable Uninstall

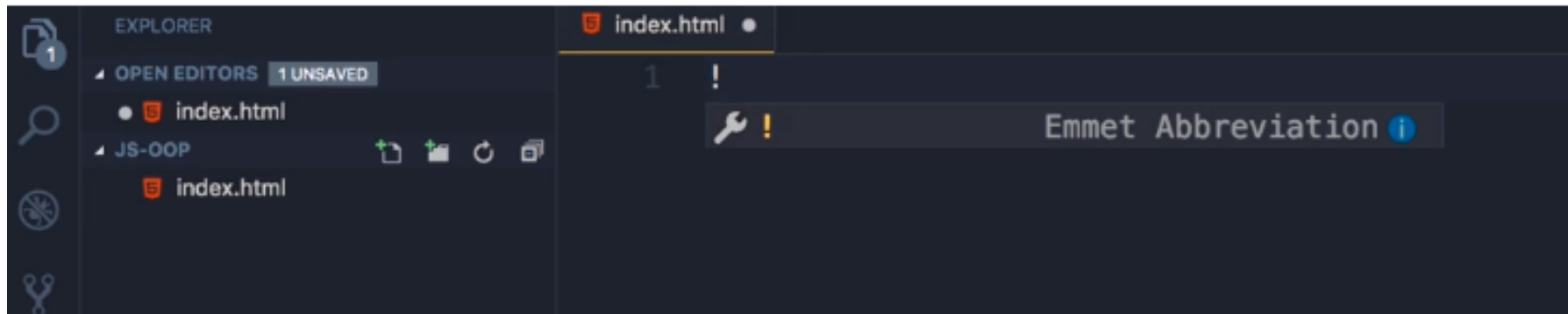
[Details](#) [Contributions](#) [Changelog](#) [Dependencies](#)

Live Server

Boom! Big Announcement! Live Server is now supported for dynamic pages like PHP. Check Here for more details.

[If you're facing 'command not found error', please follow the steps #78]

Live Templates in CS Code (!) press tab



Object Literal

```
let circle = {  
  radius: 1,  
  border: 2,  
}
```

Object Literal

```
let circle = {  
  radius: 1,  
  border: 2,  
  location: {  
    x: 45,  
    y: 35  
  }  
}
```

Object Literal

```
let circle = {  
  radius: 1,  
  draw: function () {  
    console.log('draw');  
  }  
}  
circle.draw();
```

Factory Function

```
// Factory Function
function createCircle(radius) {
  return {
    radius,
    draw: function() {
      console.log('draw');
    }
  };
}

const circle = createCircle(1)
circle.draw();
```

Constructor Function

```
function Circle(radius) {  
  this.radius = radius;  
  this.draw = function () {  
    console.log("Draw: r=" + radius);  
  }  
}  
const c = new Circle(5); //new Object  
c.draw();
```

Don't Miss It

this

Refers to the object calling current function

Constructor property

```
let x = {}
```

```
// let x= new Object()
```

//factory functions use default constructor

//check from browser by

object.constructor

Value vs Reference Types

Value Types

Number

String

Boolean

Symbol

undefined

null

Reference Types

Object

Function

Array

Value vs Reference Types

```
let x = 10;
```

```
let y = x;
```

```
x = 20;
```

```
//y will have 10
```

```
let x = {value:10}
```

```
let y = x;
```

```
x.value = 20;
```

```
//y.value will have 20
```

Primitives are copied by their **value**

Objects are copied by their **reference**

What will be the output

```
let x = 10;
function increase(x) {
  x++;
}
Increase(x);
console.log(x);
//10
```

```
let y = { value: 10 };
function increaseObj(y) {
  y.value++;
}
increaseObj(y);
console.log(y.value);
//11
```

Cheat Sheet

<https://1drv.ms/u/s!AtGKdbMmNBGdhQqT7nVD8sP5MIW2>