tt(fp)

FUNCTIONAL PROGRAMMING

DEBUG & REFACTOR

SCHEDULE

now

I. refactor

II. debug

REFACTOR

I/II

I/II REFACTOR

re•fac•tor

/ˌrā'faktər/

1. Improve without altering external behavior of (computer hardware or software).
   “games are the worst to refactor”

I/II REFACTOR

LIKE THIS

```
console.log(fibonacci(5)) // 8

function fibonacci(num) {
  var a = 1, b = 0, temp
  while (num >= 0) {
    temp = a
    a = a + b
    b = temp
    num--
  }
  return b
}
```

I/II REFACTOR

```
function fibonacci(num) {
  var a = 1, b = 0, temp
  while (num >= 0) {
    temp = a
    a = a + b
    b = temp
    num--
  }
  return b
}

console.log(fibonacci(5)) // 8
```

I/II REFACTOR

FUNCTION EXPRESSIONS

```
var fibonacci = function (num) {
  var a = 1, b = 0, temp
  while (num >= 0) {
    temp = a
    a = a + b
    b = temp
    num--
  }
  return b
}

console.log(fibonacci(5)) // 8
```

I/II REFACTOR

LOOSE VARIABLE DECLARATIONS

```
console.log(fibonacci(5)) // 8

function fibonacci(num) {
  var a = 1
  var b = 0
  var temp
  while (num >= 0) {
    temp = a
    a = a + b
    b = temp
    num--
  }
```

I/II REFACTOR

SPACES EVERYWHERE

```
console.log( fibonacci( 5 ) ) // 8

function fibonacci ( num ) {
  var a = 1, b = 0, temp
  while ( num >= 0 ) {
    temp = a
    a = a + b
    b = temp
    num --
  }
  return b
}
```

I/II REFACTOR

SEMICOLONS EVERYWHERE

```
console.log(fibonacci(5)); // 8

function fibonacci(num) {
  var a = 1, b = 0, temp;
  while (num >= 0) {
    temp = a;
    a = a + b;
    b = temp;
    num--;
  }
  return b;
}
```

I/II REFACTOR

Linters / Formatters

- prettier (JS, CSS, etc) – Opinionated code formatter
- standard (JS) – Standard style
- xo (JS) – Happiness style
- eslint (JS) – Fully pluggable style
- stylelint (CSS) – Mighty, modern linter

DEBUG

11/11

II/II DEBUG

de•bug

/dē'bəg/

1. Identify and remove errors from (computer hardware or software)
   “games are the worst to debug”

DEBUG

```
<script src=index.js></script>
```

```
Failed to load resource: the server responded with a status
of 404 (HTTP/2.0 404)
```

II/II DEBUG

```
<h1 id=title>This is fine…</h1>


// JS
document.getElementByID('title').textContent = 'Fixed!'
```

```
TypeError: document.getElementByID is not a function. (In
'document.getElementByID('title')', 'document.getElementByID'
is undefined)
```

11/11 DEBUG

```
<h1>This is fine…</h1>
// JS
update()
var update = function () {
  document.querySelector('h1').textContent = 'Fixed!'
}
```

```
TypeError: update is not a function. (In 'update()', 'update'
is undefined)
```

II/II DEBUG

```
<h1>This is fine…</h1>


// JS
document.querySelector('h1').textContent = 'Its
  Fixed!'
```

```
SyntaxError: Unexpected EOF
```

11/11 DEBUG

JS

```
<h1>This is fine…</h1>


// JS
document.querySelector('h1').textContent = 'It's Fixed!'
```

```
SyntaxError: Unexpected identifier 's'
```

II/II DEBUG

JS

```
<h1>This is fine…</h1>


// JS
document.querySelector('h1').textContent = ['It’s' 'fixed!'].join(' ')
```

```
SyntaxError: Unexpected string literal "fixed!". Expected
either a closing ']' or a ',' following an array element.
```

II/II DEBUG

```
<h1 style=color:red>This is fine…</h1>
// JS
document.querySelector('h1').onclick = function () {
  this.onclick = null
  setTimeout(function () {
    this.textContent = 'Fixed!'
    this.style.color = 'green'
  })
```

```
TypeError: undefined is not an object (evaluating
'this.style.color = 'green'')
```

II/II DEBUG

```
var image = document.createElement('img')
var width

image.onload = function () {
  width = this.width
}

image.src = 'cmd.png'

console.log(width) // ?
```

II/II DEBUG

BUGS A: ACTUAL

```
var image = document.createElement('img')
var width

image.onload = function () {
  width = this.width
}

image.src = 'cmd.png'

console.log(width) // undefined
```

II/II DEBUG

BUGS A: FIX

```
var image = document.createElement('img')
var width

image.onload = function () {
  width = this.width
  console.log(width) // 400
}

image.src = 'cmd.png'
```

II/II DEBUG

BUGS B: EXPECT

```
// HTML
<h1>?</h1>

// JS
var title = document.querySelector('h1')

for (var index = 0; index < 2; index++) {
  setTimeout(function () {
    title.textContent = index === 1 ? 'Fixed!' : 'This is fine…'
  }, 0)
}
```

II/II DEBUG

BUGS B: ACTUAL

```
// HTML
<h1>This is fine…</h1>

// JS
var title = document.querySelector('h1')

for (var index = 0; index < 2; index++) {
  setTimeout(function () {
    title.textContent = index === 1 ? 'Fixed!' : 'This is fine…'
  }, 0)
}
```

II/II DEBUG

BUGS B: FIX

```
// HTML
<h1>Fixed!</h1>

// JS
var title = document.querySelector('h1')

for (var index = 0; index < 2; index++) {
  setTimeout((function (i) {
    return function () {
      title.textContent = i === 1 ? 'Fixed!' : 'This is fine…'
    }
  })(index), 0)
}
```

11/11 DEBUG

```
// HTML
<h1>?</h1>

// JS
var title = document.querySelector('h1')

title.textContent = ok() || 'This is fine…'

function ok() {
  return
    'It’s' +
    'fixed!'
```

II/II DEBUG

BUGS C: ACTUAL

```
// HTML
<h1>This is fine…</h1>

// JS
var title = document.querySelector('h1')

title.textContent = ok() || 'This is fine…'

function ok() {
  return
    'It’s' +
    'fixed!'
```

11/11 DEBUG

```
// HTML
<h1>It’s fixed!</h1>

// JS
var title = document.querySelector('h1')

title.textContent = ok() || 'This is fine…'

function ok() {
  return 'It’s' +
    'fixed!'
}
```

11/11 DEBUG

BUGS D: EXPECT

```
// HTML
<h1>?</h1>

// JS
var fine = false
var title = document.querySelector('h1')

if (fine = true) {
  title.textContent = 'This is fine…'
} else {
  title.textContent = 'It’s fixed!'
}
```

II/II DEBUG

BUGS D: ACTUAL

```
// HTML
<h1>This is fine…</h1>

// JS
var fine = false
var title = document.querySelector('h1')

if (fine = true) {
  title.textContent = 'This is fine…'
} else {
  title.textContent = 'It’s fixed!'
}
```

II/II DEBUG

BUGS D: FIX

```
// HTML
<h1>It’s fixed!</h1>

// JS
var fine = false
var title = document.querySelector('h1')

if (fine === true) {
  title.textContent = 'This is fine…'
} else {
  title.textContent = 'It’s fixed!'
}
```

I/II REFACTOR

TOOLS & TIPS

Developer tools

- Chrome
- Firefox
- Safari

Tips

- console.log everything, even if you're sure it works (it probably doesn't)
- use linters, read docs, check stackoverflow
- talk with a duck

rubber duck debugging