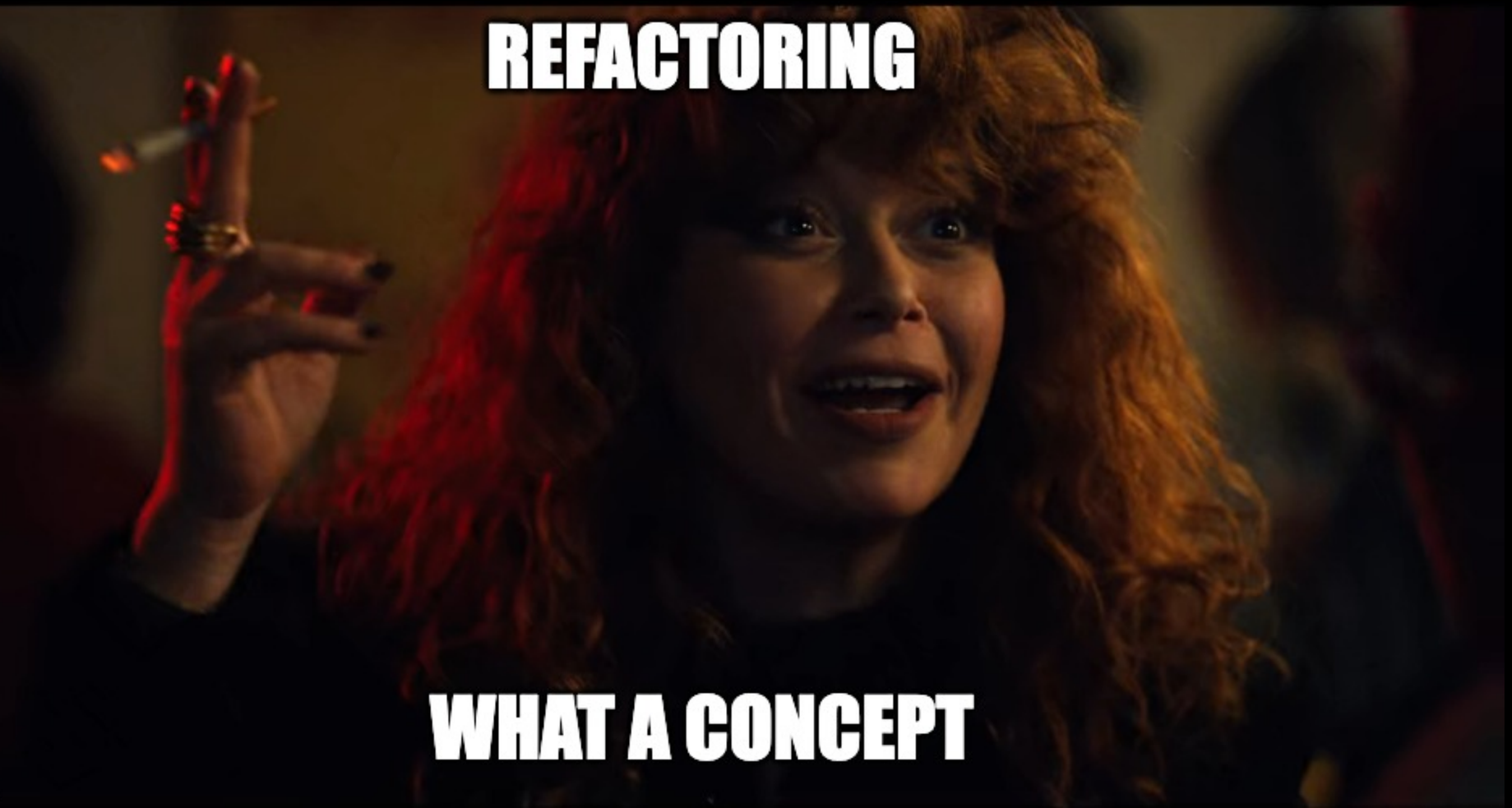
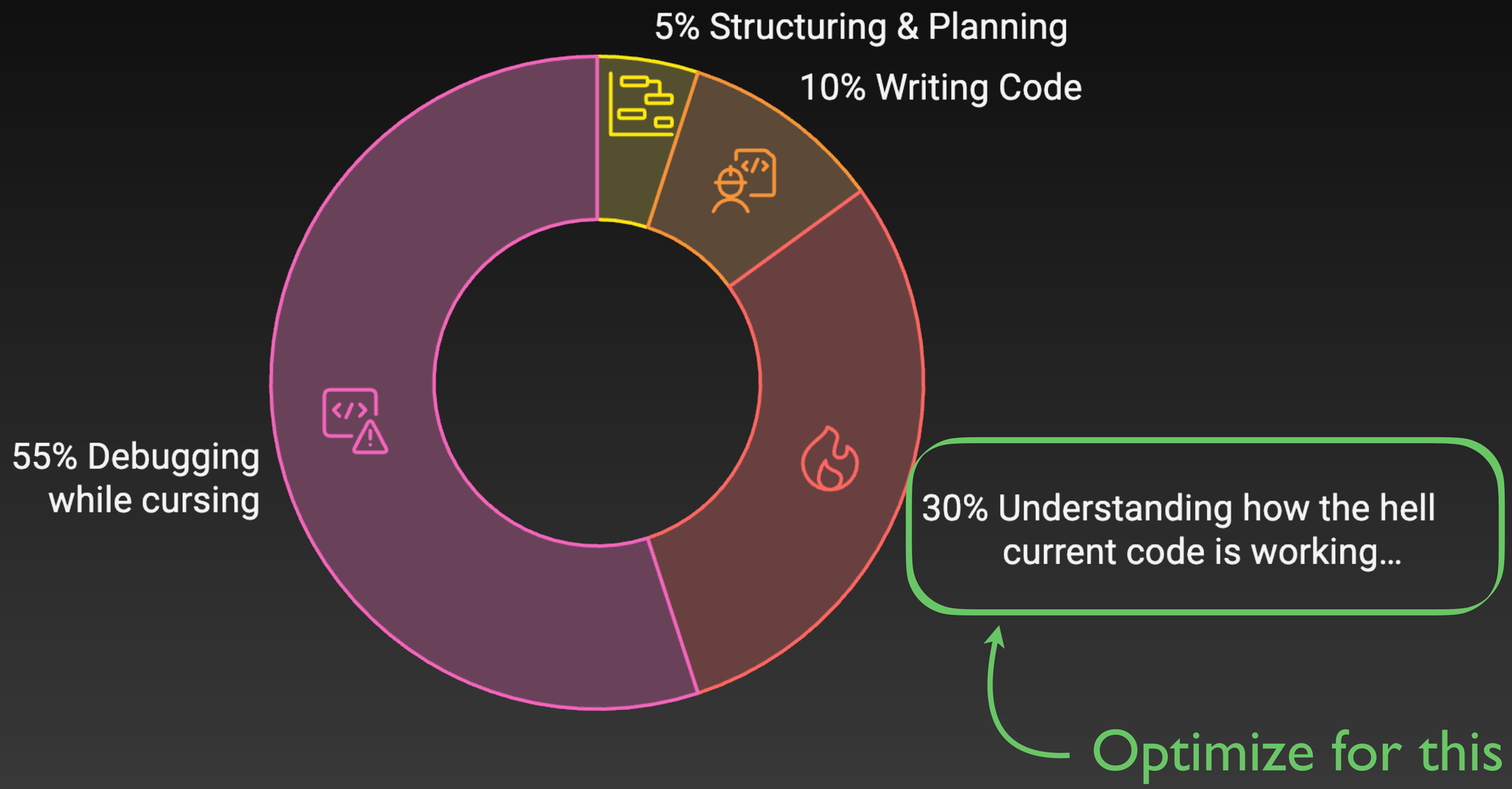

6 Safe Refactorings
for Untested Legacy Code
by Nicolas Carlo
ConFoo 2025

"**Change made to the internal structure** of software to make it easier to understand and cheaper to modify **without changing its observable behavior**"

— Martin Fowler



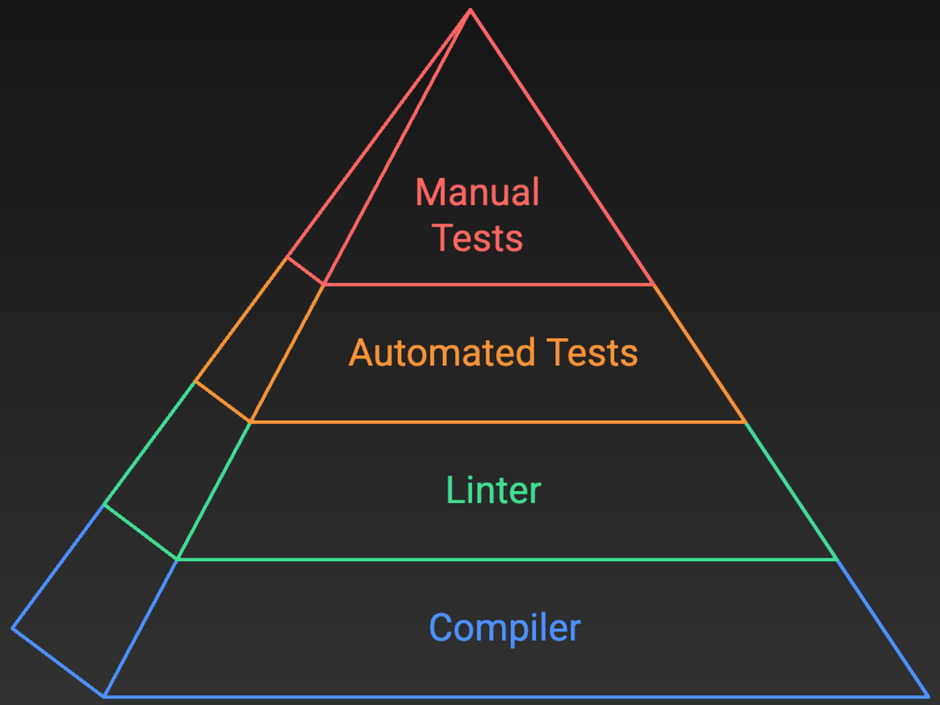
Refactor to make code easier to read

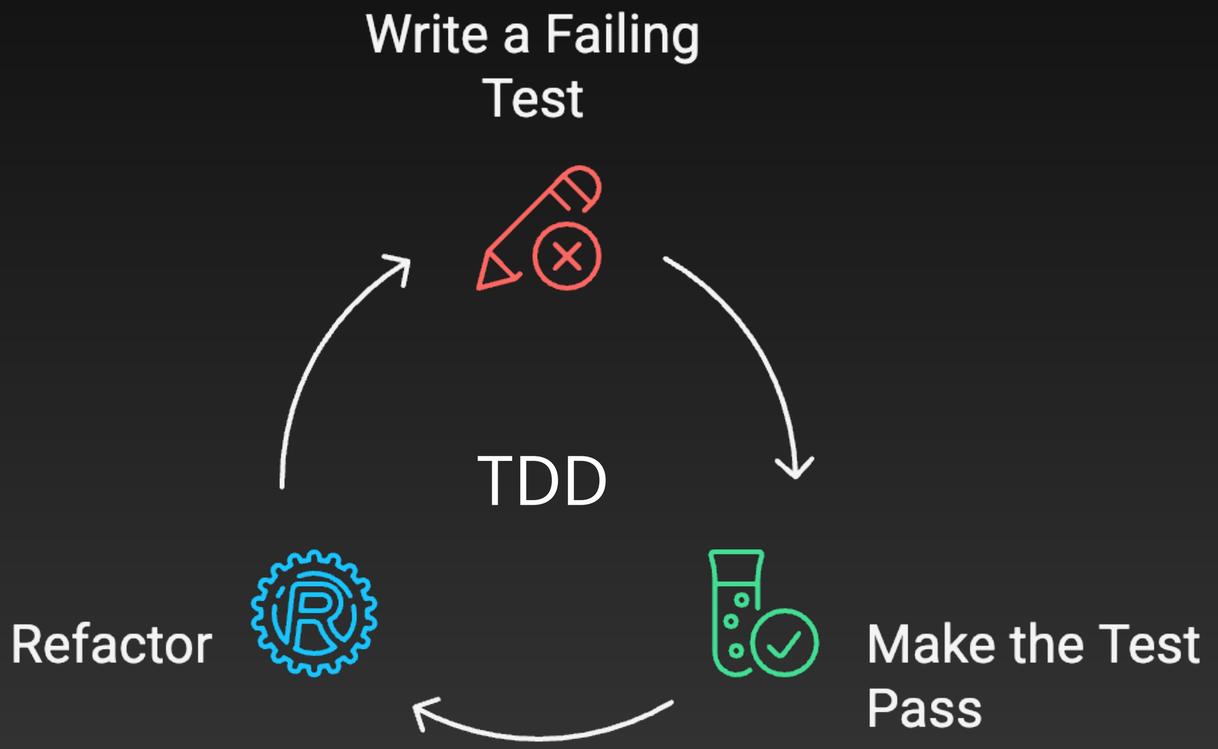


Refactor to make code easier to change

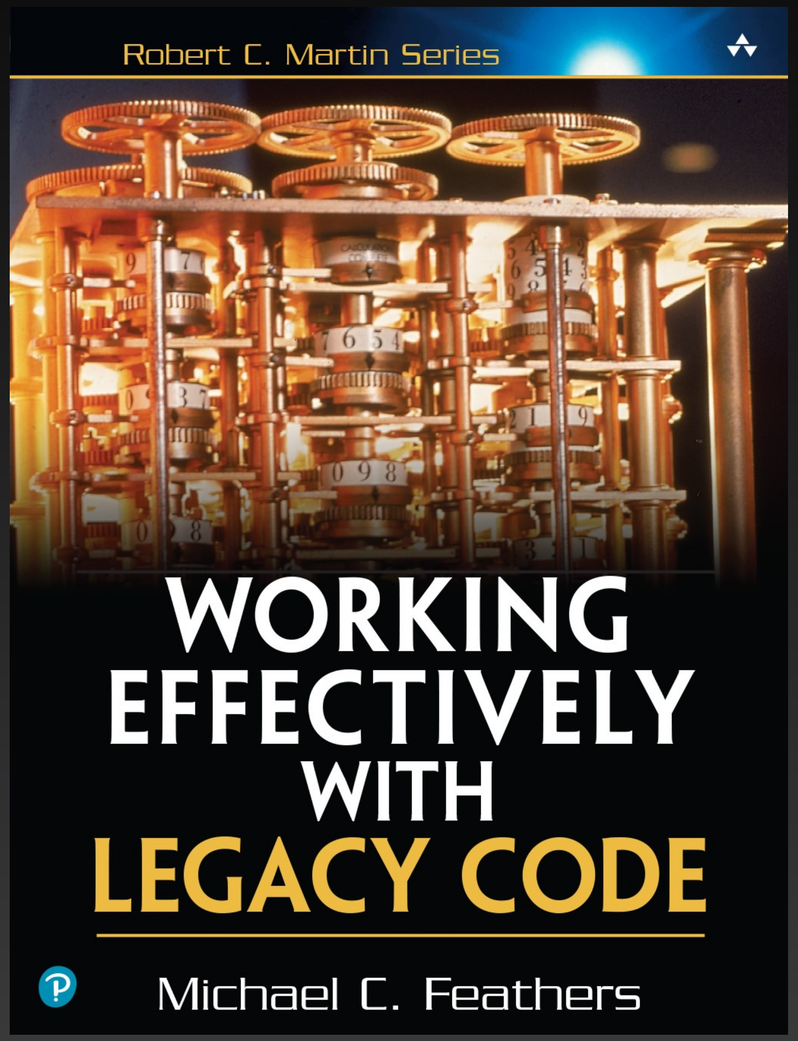
How do you know it was a refactoring?

Test before & after





What if the code is NOT tested?



You are dealing with **Legacy Code**

Don't worry, you are not alone ✊

Nicolas Carlo

- Freelance Web Developer
- Legacy Code specialist
- Author of Abracadabra for VS Code
- Software Crafters & React MTL guilds

The paradox of Legacy Code

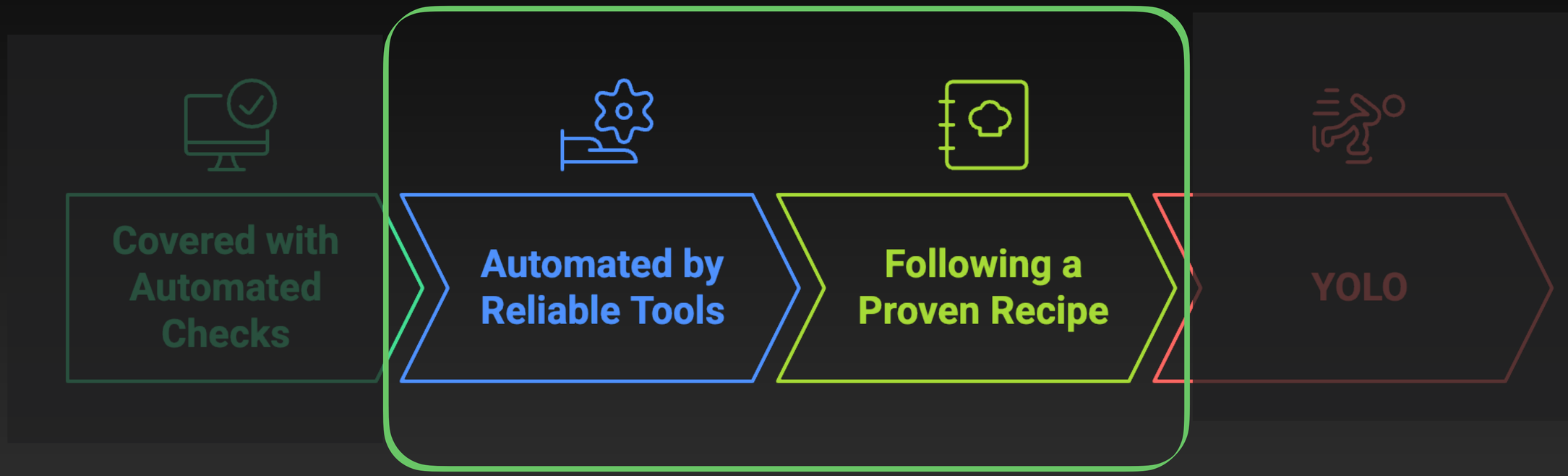
Refactoring

Necessary to write tests

Automated Tests

Essential for safe refactorings

Refactoring strategies, from safest to riskiest



6 Refactorings for Legacy Code

Inline

Extract

Distill & Extract

Rename

Change Signature

Extend & Override

6 Refactorings for Legacy Code

Inline

Extract

Distill & Extract

Rename

Change Signature

Extend & Override

Inline

```
export class Shop {
  updateItemQuality(item) {
    if (
      item.name !== "Aged Brie" &&
      item.name !== "Backstage passes to a TAFKAL80ETC concert"
    ) {
      this.decreaseQuality(item);
    } else {
      // ... more logic
    }
  }

  decreaseQuality(item) {
    if (item.quality > 0 && item.name !== "Sulfuras, Hand of Ragnaros") {
      item.quality -= 1;
    }
  }

  // ... more logic
}
```

We realize `decreaseQuality` prevents a better refactoring… let's inline!

Inline

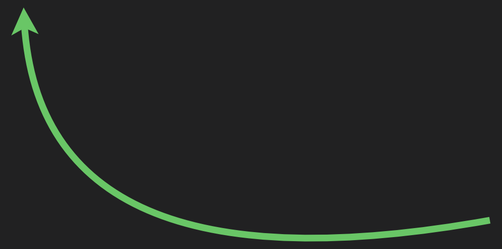
```
export class Shop {
  updateItemQuality(item) {
    if (
      item.name !== "Aged Brie" &&
      item.name !== "Backstage passes to a TAFKAL80ETC concert"
    ) {
      this.decreaseQuality(item);
    } else {
      // ... more logic
    }
  }

  decreaseQuality(item) {
    if (item.quality > 0 && item.name !== "Sulfuras, Hand of Ragnaros") {
      item.quality -= 1;
    }
  }

  // ... more logic
}
```

First, replace the call with the function body

Inline

Repeat for each occurrence until the function becomes unused

```
export class Shop {
  updateItemQuality(item) {
    if (
      item.name !== "Aged Brie" &&
      item.name !== "Backstage passes to a TAFKAL80ETC concert"
    ) {
      if (item.quality > 0 && item.name !== "Sulfuras, Hand of Ragnaros") {
        item.quality -= 1;
      }
    } else {
      // ... more logic
    }
  }

  decreaseQuality(item) {
    if (item.quality > 0 && item.name !== "Sulfuras, Hand of Ragnaros") {
      item.quality -= 1;
    }
  }

  // ... more logic
}
```



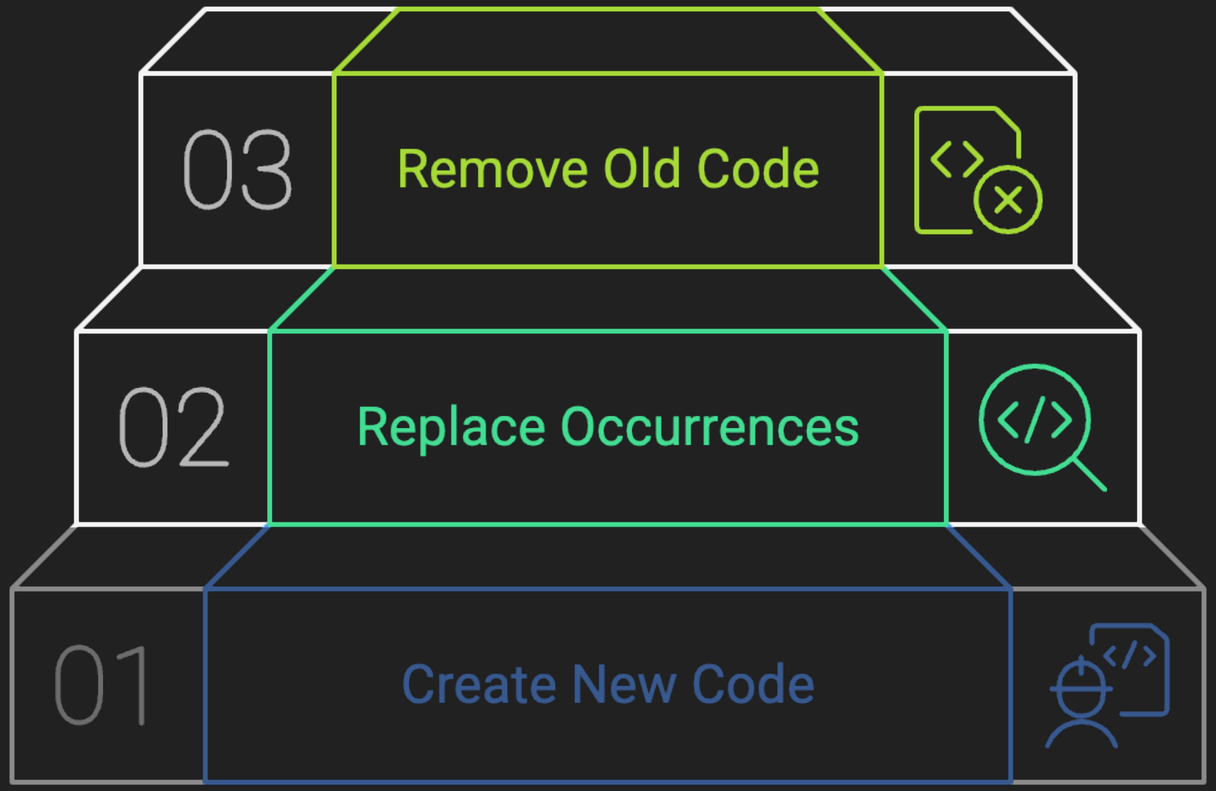
Then, we can delete the function

Inline

```
export class Shop {
  updateItemQuality(item) {
    if (
      item.name !== "Aged Brie" &&
      item.name !== "Backstage passes to a TAFKAL80ETC concert"
    ) {
      if (item.quality > 0 && item.name !== "Sulfuras, Hand of Ragnaros") {
        item.quality -= 1;
      }
    } else {
      // ... more logic
    }
  }

  // ... more logic
}
```

Inline

Remove inconvenient abstractions



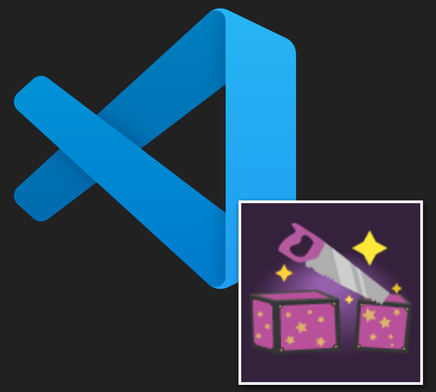
A consistent recipe

Inline

Remove inconvenient abstractions

Ctrl + Alt + N

⌘ ⌥ N



+ Abracadabra extension

6 Refactorings for Legacy Code

Inline

Extract

Distill & Extract

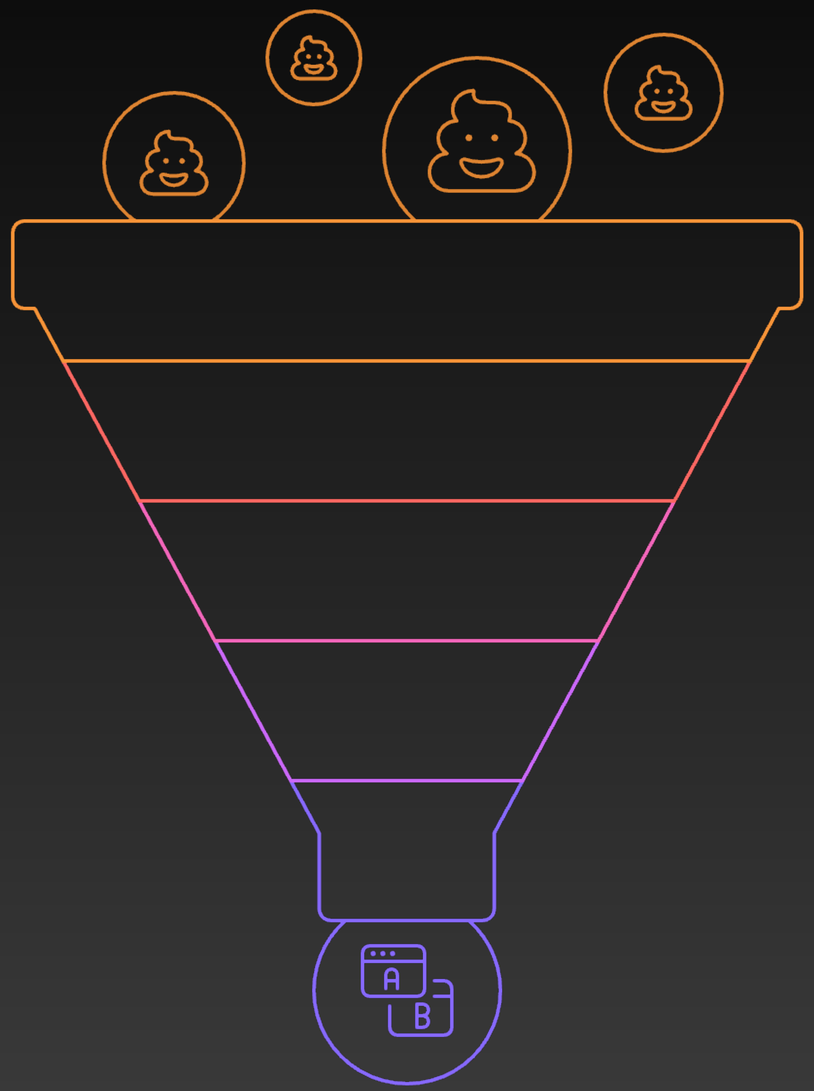
Rename

Change Signature

Extend & Override

Rename

Naming is hard, make it iterative



Misleading Names
parseXML()

Obvious Nonsense
appleSauce()

Honest Names
parseXMLAndSaveFlightToDBAndShowOnScreen()

Intention Revealing
beginTrackingFlight()

Domain Abstraction
screen.add(flight)

Naming as a process
— Arlo Belschee

Rename

```
const dfrq = {};
for (const i of DICE_VALUES) {
  dfrq[i] = 0;
}

for (const d of dice) {
  dfrq[d] = (dfrq[d] || 0) + 1;
}

let score;
switch (category) {
  case YatzyCategory.YATZY:
    if (Object.values(dfrq).includes(5)) {
      score = 50;
    }
    break;

  case YatzyCategory.ONES:
    score = dfrq[1];
    break;

  // … more code
}
```

Rename

What is dfrq?

```
const dfrq = {};
for (const i of DICE_VALUES) {
  dfrq[i] = 0;
}

for (const d of dice) {
  dfrq[d] = (dfrq[d] || 0) + 1;
}

let score;
switch (category) {
  case YatzyCategory.YATZY:
    if (Object.values(dfrq).includes(5)) {
      score = 50;
    }
    break;

  case YatzyCategory.ONES:
    score = dfrq[1];
    break;

  // … more code
}
```

Rename

Oh, it's the *dice frequency*!

```
const dfrq = {};
for (const i of DICE_VALUES) {
  dfrq[i] = 0;
}

for (const d of dice) {
  dfrq[d] = (dfrq[d] || 0) + 1;
}

let score;
switch (category) {
  case YatzyCategory.YATZY:
    if (Object.values(dfrq).includes(5)) {
      score = 50;
    }
    break;

  case YatzyCategory.ONES:
    score = dfrq[1];
    break;

  // … more code
}
```

R Rename

First, declare a new variable with the name you want

```
const dfrq = {};
for (const i of DICE_VALUES) {
  dfrq[i] = 0;
}

for (const d of dice) {
  dfrq[d] = (dfrq[d] || 0) + 1;
}

let score;
switch (category) {
  case YatzyCategory.YATZY:
    if (Object.values(dfrq).includes(5)) {
      score = 50;
    }
    break;

  case YatzyCategory.ONES:
    score = dfrq[1];
    break;

  // … more code
}
```

Rename

Then, replace each occurrence

```
const dfrq = {};
const diceFrequencies = dfrq;
for (const i of DICE_VALUES) {
  dfrq[i] = 0;
}

for (const d of dice) {
  dfrq[d] = (dfrq[d] || 0) + 1;
}

let score;
switch (category) {
  case YatzyCategory.YATZY:
    if (Object.values(dfrq).includes(5)) {
      score = 50;
    }
    break;

  case YatzyCategory.ONES:
    score = dfrq[1];
    break;

  // … more code
}
```

Rename

Then, replace each occurrence

```
const dfrq = {};
const diceFrequencies = dfrq;
for (const i of DICE_VALUES) {
  diceFrequencies[i] = 0;
}

for (const d of dice) {
  dfrq[d] = (dfrq[d] || 0) + 1;
}

let score;
switch (category) {
  case YatzyCategory.YATZY:
    if (Object.values(dfrq).includes(5)) {
      score = 50;
    }
    break;

  case YatzyCategory.ONES:
    score = dfrq[1];
    break;

  // … more code
}
```

Rename

Then, replace each occurrence

```
const dfrq = {};
const diceFrequencies = dfrq;
for (const i of DICE_VALUES) {
  diceFrequencies[i] = 0;
}

for (const d of dice) {
  diceFrequencies[d] = (dfrq[d] || 0) + 1;
}

let score;
switch (category) {
  case YatzyCategory.YATZY:
    if (Object.values(dfrq).includes(5)) {
      score = 50;
    }
    break;

  case YatzyCategory.ONES:
    score = dfrq[1];
    break;

  // … more code
}
```

Rename

Then, replace each occurrence

```
const dfrq = {};
const diceFrequencies = dfrq;
for (const i of DICE_VALUES) {
  diceFrequencies[i] = 0;
}

for (const d of dice) {
  diceFrequencies[d] = (diceFrequencies[d] || 0) + 1;
}

let score;
switch (category) {
  case YatzyCategory.YATZY:
    if (Object.values(dfrq).includes(5)) {
      score = 50;
    }
    break;

  case YatzyCategory.ONES:
    score = dfrq[1];
    break;

  // … more code
}
```

R Rename

Finally, Inline Variable

```
const dfrq = {};
const diceFrequencies = dfrq;
for (const i of DICE_VALUES) {
  diceFrequencies[i] = 0;
}

for (const d of dice) {
  diceFrequencies[d] = (diceFrequencies[d] || 0) + 1;
}

let score;
switch (category) {
  case YatzyCategory.YATZY:
    if (Object.values(diceFrequencies).includes(5)) {
      score = 50;
    }
    break;

  case YatzyCategory.ONES:
    score = diceFrequencies[1];
    break;

  // … more code
}
```

Rename

```
const diceFrequencies = {};
for (const i of DICE_VALUES) {
  diceFrequencies[i] = 0;
}

for (const d of dice) {
  diceFrequencies[d] = (diceFrequencies[d] || 0) + 1;
}

let score;
switch (category) {
  case YatzyCategory.YATZY:
    if (Object.values(diceFrequencies).includes(5)) {
      score = 50;
    }
    break;

  case YatzyCategory.ONES:
    score = diceFrequencies[1];
    break;

  // … more code
}
```

Rename

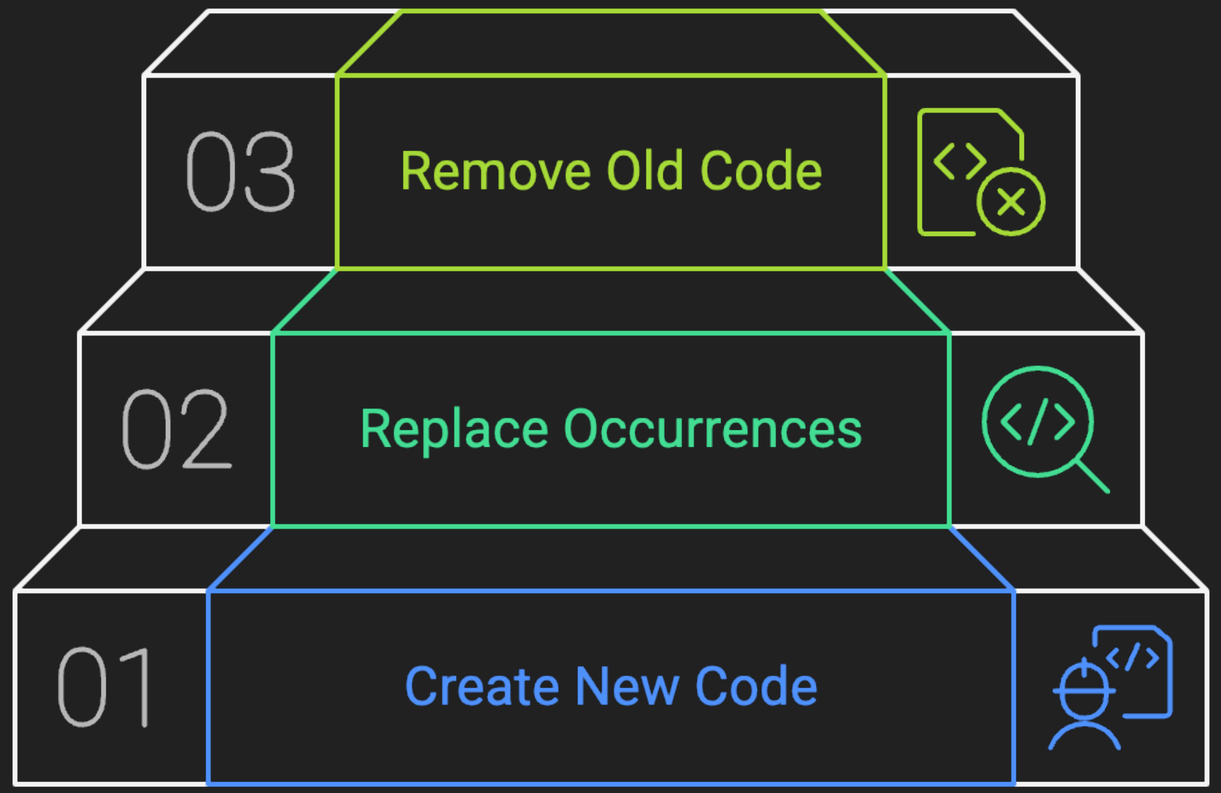
Improve the clarity

⇧ F6

F2

Rename

Improve the clarity



A consistent recipe

6 Refactorings for Legacy Code

Inline

Extract

Distill & Extract

Rename

Change Signature

Extend & Override

Extract

Use-case 1: Clarify the intent

```
function calculateScore(player1Score, player2Score) {
  if (player1Score < 4 && player2Score < 4 && !(player1Score + player2Score === 6)) {
    const score = SCORE_NAMES[player1Score];
    return (player1Score === player2Score)
      ? score + '-All'
      : score + '-' + SCORE_NAMES[player2Score];
  } else {
    // … more code
  }
}
```

Extract

Use-case 1: Clarify the intent

```
function calculateScore(player1Score, player2Score) {
  if (player1Score < 4 && player2Score < 4 && !(player1Score + player2Score === 6)) {
    const score = SCORE_NAMES[player1Score];
    return (player1Score === player2Score)
      ? score + '-All'
      : score + '-' + SCORE_NAMES[player2Score];
  } else {
    // … more code
  }
}
```

Extract

First, declare the variable

```
function calculateScore(player1Score, player2Score) {
  const appleSauce = player1Score < 4 &&
                     player2Score < 4 &&
                     !(player1Score + player2Score === 6);

  if (player1Score < 4 && player2Score < 4 && !(player1Score + player2Score === 6)) {
    const score = SCORE_NAMES[player1Score];
    return (player1Score === player2Score)
      ? score + '-All'
      : score + '-' + SCORE_NAMES[player2Score];
  } else {
    // … more code
  }
}
```

Extract

First, declare the variable

Not sure what the name should be? Use a nonsense-name for now!

```
function calculateScore(player1Score, player2Score) {
  const appleSauce = player1Score < 4 &&
                     player2Score < 4 &&
                     !(player1Score + player2Score === 6);

  if (player1Score < 4 && player2Score < 4 && !(player1Score + player2Score === 6)) {
    const score = SCORE_NAMES[player1Score];
    return (player1Score === player2Score)
      ? score + '-All'
      : score + '-' + SCORE_NAMES[player2Score];
  } else {
    // … more code
  }
}
```

Extract

Then, replace the occurrence(s)

```
function calculateScore(player1Score, player2Score) {
  const appleSauce = player1Score < 4 &&
                     player2Score < 4 &&
                     !(player1Score + player2Score === 6);

  if (appleSauce) {
    const score = SCORE_NAMES[player1Score];
    return (player1Score === player2Score)
      ? score + '-All'
      : score + '-' + SCORE_NAMES[player2Score];
  } else {
    // … more code
  }
}
```

It's condition for the early game!

Extract

```
function calculateScore(player1Score, player2Score) {
  const isEarlyGame = player1Score < 4 &&
                      player2Score < 4 &&
                      !(player1Score + player2Score === 6);

  if (isEarlyGame) {
    const score = SCORE_NAMES[player1Score];
    return (player1Score === player2Score)
      ? score + '-All'
      : score + '-' + SCORE_NAMES[player2Score];
  } else {
    // … more code
  }
}
```

It's condition for the early game!

Rename

Extract

Use-case 2: Duplicated logic

```
if (item.name == "Aged Brie") {
  if (item.quality < 50) {
    item.quality = item.quality + 1;
  }

  item.sellIn = item.sellIn - 1;

  if (item.sellIn < 0) {
    if (item.quality < 50) {
      item.quality = item.quality + 1;
    }
  }
} else if (item.name == "Backstage passes to a TAFKAL80ETC concert") {
  if (item.quality < 50) {
    item.quality = item.quality + 1;
  }

  if (item.sellIn < 11) {
    if (item.quality < 50) {
      item.quality = item.quality + 1;
    }
  }

  // … more code
}
```

Extract

Use-case 2: Duplicated logic

```
if (item.name == "Aged Brie") {
  if (item.quality < 50) {
    item.quality = item.quality + 1;
  }

  item.sellIn = item.sellIn - 1;

  if (item.sellIn < 0) {
    if (item.quality < 50) {
      item.quality = item.quality + 1;
    }
  }
} else if (item.name == "Backstage passes to a TAFKAL80ETC concert") {
  if (item.quality < 50) {
    item.quality = item.quality + 1;
  }

  if (item.sellIn < 11) {
    if (item.quality < 50) {
      item.quality = item.quality + 1;
    }
  }

  // … more code
}
```

Extract

Let's extract this chunk

```
if (item.name == "Aged Brie") {
  if (item.quality < 50) {
    item.quality = item.quality + 1;
  }

  item.sellIn = item.sellIn - 1;

  if (item.sellIn < 0) {
    if (item.quality < 50) {
      item.quality = item.quality + 1;
    }
  }
} else if (item.name == "Backstage passes to a TAFKAL80ETC concert") {
  if (item.quality < 50) {
    item.quality = item.quality + 1;
  }

  if (item.sellIn < 11) {
    if (item.quality < 50) {
      item.quality = item.quality + 1;
    }
  }

  // … more code
}
```

Extract

```
if (item.name == "Aged Brie") {
  if (item.quality < 50) {
    item.quality = item.quality + 1;
  }

  item.sellIn = item.sellIn - 1;

  if (item.sellIn < 0) {
    if (item.quality < 50) {
      item.quality = item.quality + 1;
    }
  }
} else if (item.name == "Backstage passes to a TAFKAL80ETC concert") {
  if (item.quality < 50) {
    item.quality = item.quality + 1;
  }

  if (item.sellIn < 11) {
    if (item.quality < 50) {
      item.quality = item.quality + 1;
    }
  }

  // … more code
}
```

```
function incrementQuality() {
}
```

First, declare the new function

Extract

```
if (item.name == "Aged Brie") {
  if (item.quality < 50) {
    item.quality = item.quality + 1;
  }

  item.sellIn = item.sellIn - 1;

  if (item.sellIn < 0) {
    if (item.quality < 50) {
      item.quality = item.quality + 1;
    }
  }
} else if (item.name == "Backstage passes to a TAFKAL80ETC concert") {
  if (item.quality < 50) {
    item.quality = item.quality + 1;
  }

  if (item.sellIn < 11) {
    if (item.quality < 50) {
      item.quality = item.quality + 1;
    }
  }

  // … more code
}
```

```
function incrementQuality() {
}
```

Then, copy the code

Extract

Then, figure out what needs to be passed as params

```
if (item.name == "Aged Brie") {
  if (item.quality < 50) {
    item.quality = item.quality + 1;
  }

  item.sellIn = item.sellIn - 1;

  if (item.sellIn < 0) {
    if (item.quality < 50) {
      item.quality = item.quality + 1;
    }
  }
} else if (item.name == "Backstage passes to a TAFKAL80ETC concert") {
  if (item.quality < 50) {
    item.quality = item.quality + 1;
  }

  if (item.sellIn < 11) {
    if (item.quality < 50) {
      item.quality = item.quality + 1;
    }
  }

  // … more code
}
```

```
function incrementQuality() {
  if (item.quality < 50) {
    item.quality = item.quality + 1;
  }
}
```

Extract

```
if (item.name == "Aged Brie") {
  if (item.quality < 50) {
    item.quality = item.quality + 1;
  }

  item.sellIn = item.sellIn - 1;

  if (item.sellIn < 0) {
    if (item.quality < 50) {
      item.quality = item.quality + 1;
    }
  }
} else if (item.name == "Backstage passes to a TAFKAL80ETC concert") {
  if (item.quality < 50) {
    item.quality = item.quality + 1;
  }

  if (item.sellIn < 11) {
    if (item.quality < 50) {
      item.quality = item.quality + 1;
    }
  }

  // … more code
}
```

Now you can progressively replace occurrences

```
function incrementQuality(item) {
  if (item.quality < 50) {
    item.quality = item.quality + 1;
  }
}
```

Extract

```
if (item.name == "Aged Brie") {
  incrementQuality(item);

  item.sellIn = item.sellIn - 1;

  if (item.sellIn < 0) {
    if (item.quality < 50) {
      item.quality = item.quality + 1;
    }
  }
} else if (item.name == "Backstage passes to a TAFKAL80ETC concert") {
  if (item.quality < 50) {
    item.quality = item.quality + 1;
  }

  if (item.sellIn < 11) {
    if (item.quality < 50) {
      item.quality = item.quality + 1;
    }
  }

  // … more code
}
```

```
function incrementQuality(item) {
  if (item.quality < 50) {
    item.quality = item.quality + 1;
  }
}
```

Extract

```
if (item.name == "Aged Brie") {
  incrementQuality(item);

  item.sellIn = item.sellIn - 1;

  if (item.sellIn < 0) {
    incrementQuality(item);
  }
} else if (item.name == "Backstage passes to a TAFKAL80ETC concert") {
  if (item.quality < 50) {
    item.quality = item.quality + 1;
  }

  if (item.sellIn < 11) {
    if (item.quality < 50) {
      item.quality = item.quality + 1;
    }
  }

  // … more code
}
```

You can stop anytime, code is not broken

```
function incrementQuality(item) {
  if (item.quality < 50) {
    item.quality = item.quality + 1;
  }
}
```

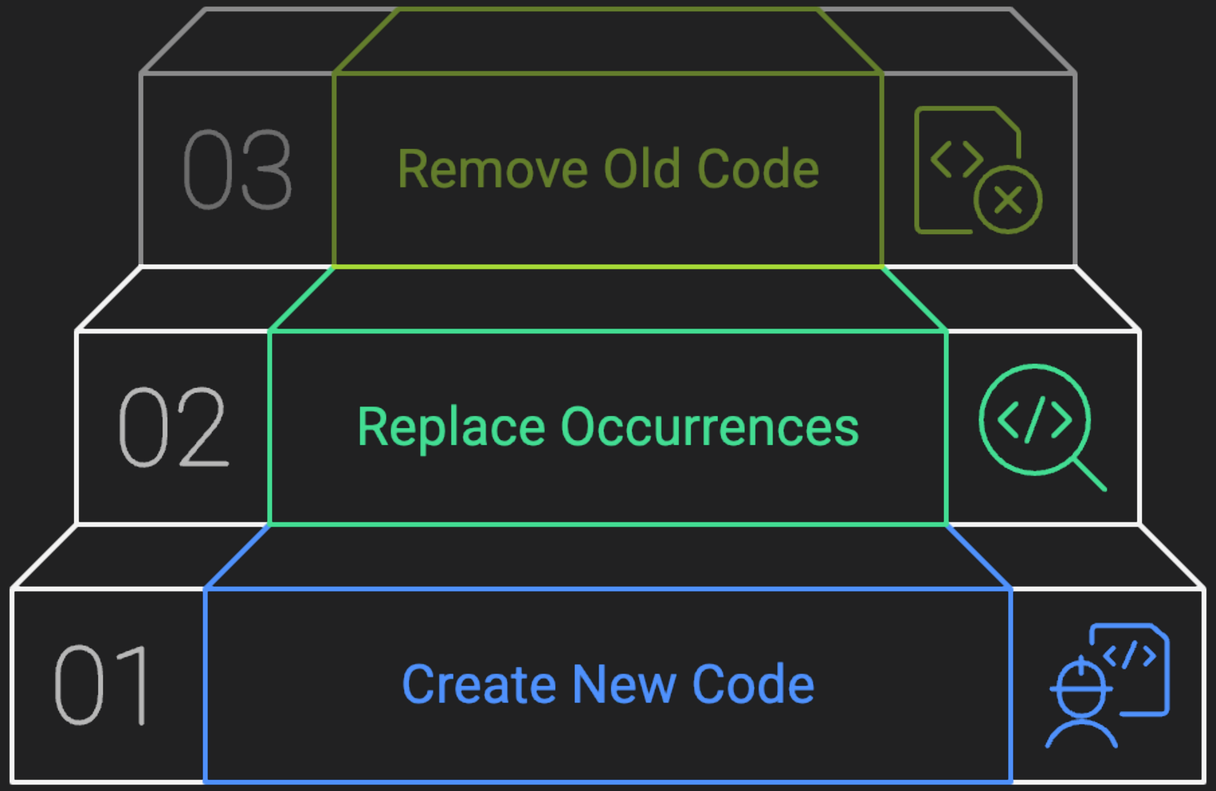
Extract

```
if (item.name == "Aged Brie") {
  incrementQuality(item);

  item.sellIn = item.sellIn - 1;

  if (item.sellIn < 0) {
    incrementQuality(item);
  }
} else if (item.name == "Backstage passes to a TAFKAL80ETC concert") {
  incrementQuality(item);

  if (item.sellIn < 11) {
    incrementQuality(item);
  }

  // … more code
}
```

```
function incrementQuality(item) {
  if (item.quality < 50) {
    item.quality = item.quality + 1;
  }
}
```

Extract

Create useful abstractions



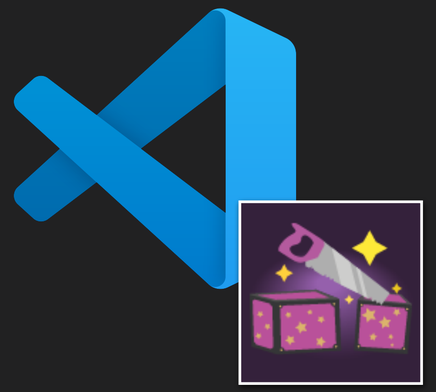
A consistent recipe

Extract

Create useful abstractions

Ctrl + Alt + V

⌘ ⌥ V



+ Abracadabra extension

6 Refactorings for Legacy Code

Inline

Extract

Distill & Extract

Rename

Change Signature

Extend & Override

Change
Signature

What if it's used at MANY places?
Tackling all changes at once can be tricky

```
export async function addReservation(customerId, date) {
  const customer = await findCustomer(customerId);
  if (!customer) {
    throw new Error(`Customer not found: ${customerId}`);
  }

  // … more code to create the reservation
}
```

Change
Signature

```
export async function addReservation(customerId, date) {
  const customer = await findCustomer(customerId);
  if (!customer) {
    throw new Error(`Customer not found: ${customerId}`);
  }

  // … more code to create the reservation
}
```

First, "Extract Function" on the body

Change
Signature

We now have a single proxy

```
export async function addReservation(customerId, date) {
  return await addReservation__NEW(customerId, date);
}


export async function addReservation__NEW(customerId, date) {
  const customer = await findCustomer(customerId);
  if (!customer) {
    throw new Error(`Customer not found: ${customerId}`);
  }

  // … more code to create the reservation
}
```

Change
Signature

Then, "Inline Function" on every call site
(we can stop anytime if needed)

```
export async function addReservation(customerId, date) {
  return await addReservation__NEW(customerId, date, false);
}


export async function addReservation__NEW(customerId, date, isVIP) {
  const customer = await findCustomer(customerId);
  if (!customer) {
    throw new Error(`Customer not found: ${customerId}`);
  }

  // … more code to create the reservation
}
```

Change
Signature

When unused, we can remove

```
export async function addReservation(customerId, date) {
  return await addReservation__NEW(customerId, date, false);
}


export async function addReservation__NEW(customerId, date, isVIP) {
  const customer = await findCustomer(customerId);
  if (!customer) {
    throw new Error(`Customer not found: ${customerId}`);
  }

  // … more code to create the reservation
}
```

Change
Signature

Finally, "Rename Function"
(automated, or search & replace)

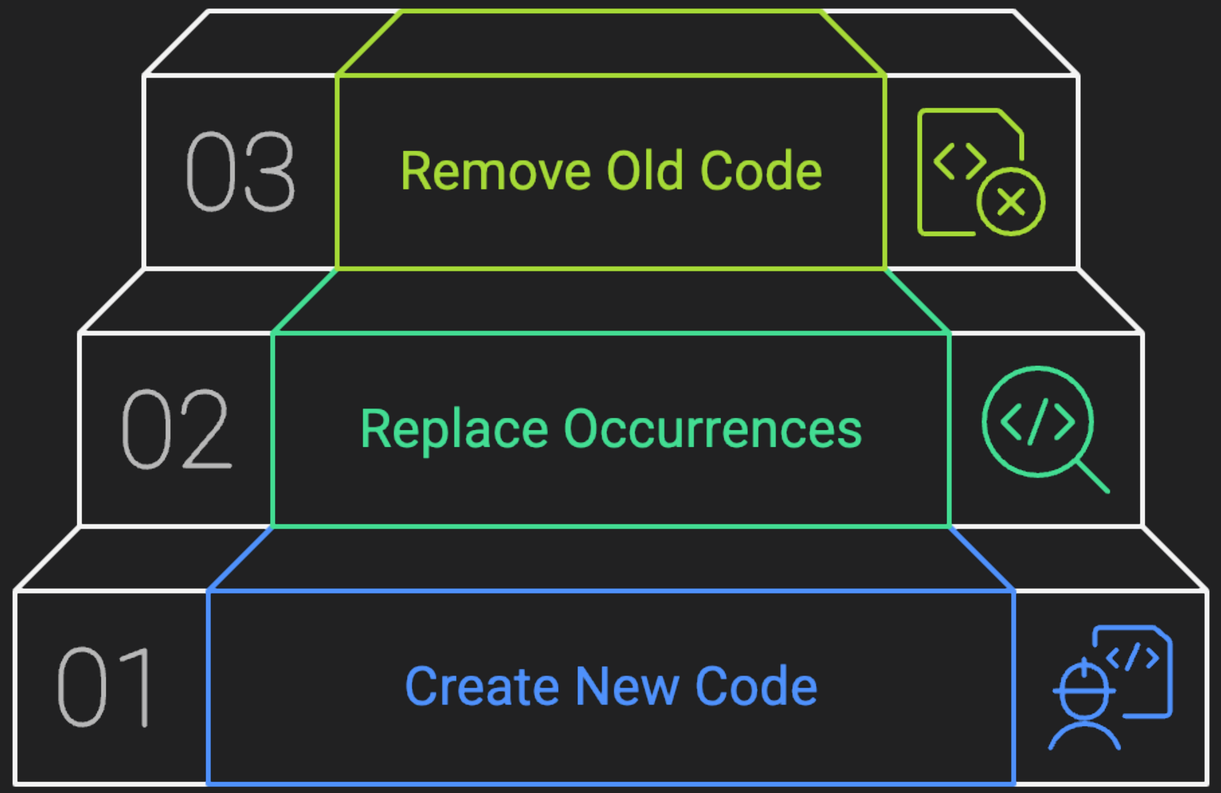
```
export async function addReservation__NEW(customerId, date, isVIP) {
  const customer = await findCustomer(customerId);
  if (!customer) {
    throw new Error(`Customer not found: ${customerId}`);
  }

  // … more code to create the reservation
}
```

Change
Signature

```
export async function addReservation(customerId, date, isVIP) {
  const customer = await findCustomer(customerId);
  if (!customer) {
    throw new Error(`Customer not found: ${customerId}`);
  }

  // … more code to create the reservation
}
```

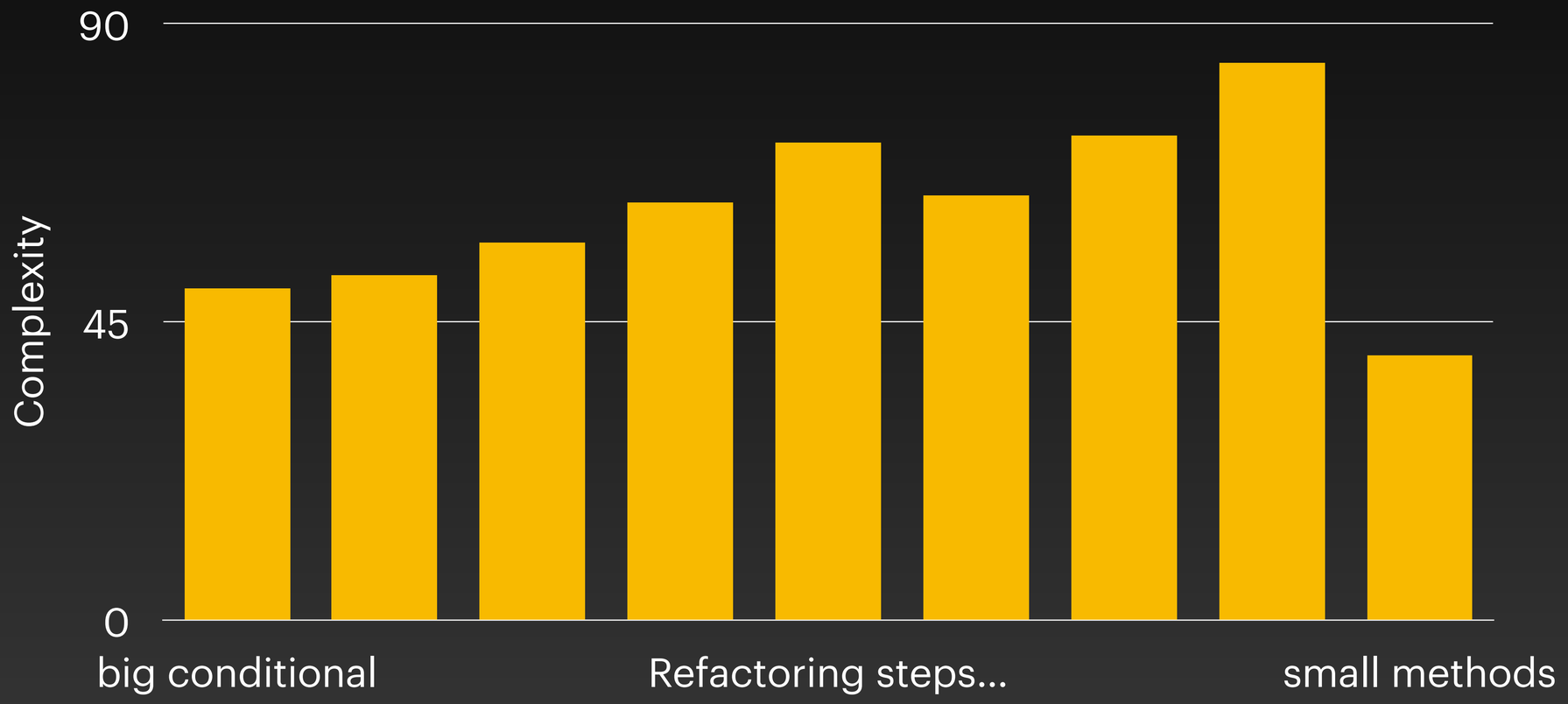
Change Signature

A larger, essential refactoring



A consistent recipe

Code gets more complex before it becomes simple



Sandi Metz, *All the Little Things* (2014)

Change Signature

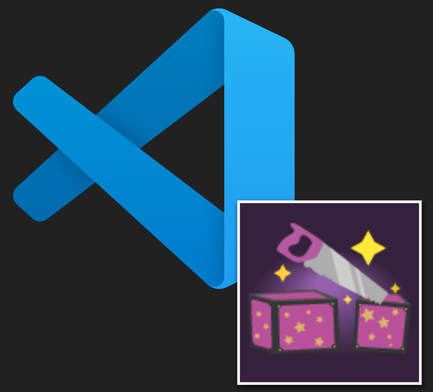
A larger, essential refactoring

Ctrl + F6

⌘ F6

Alt + ↵

⌥ ↵



+ Abracadabra extension

6 Refactorings for Legacy Code

Inline

Extract

Distill & Extract

Rename

Change Signature

Extend & Override

Distill &
Extract

What makes this code painful to test and re-use?

```
app.get("/prices", (req: Request, res: Response) => {
  if (req.query.age < 6) {
    res.json({ cost: 0 });
  } else {
    if (req.query.type !== "night") {
      if (req.query.age < 15) {
        res.json({ cost: Math.ceil(BASE_COST * 0.7) });
      } else {
        const reduction = isMonday(req.query.date) ? 35 : 0;
        res.json({ cost: Math.ceil(BASE_COST * (1 - reduction / 100)) });
      }
    } else {
      if (req.query.age > 64) {
        res.json({ cost: Math.ceil(BASE_COST * 0.4) });
      } else {
        res.json({ cost: 0 });
      }
    }
  }
});
```

Things outside of our control

```
app.get("/prices", (req: Request, res: Response) => {
  if (req.query.age < 6) {
    res.json({ cost: 0 });
  } else {
    if (req.query.type !== "night") {
      if (req.query.age < 15) {
        res.json({ cost: Math.ceil(BASE_COST * 0.7) });
      } else {
        const reduction = isMonday(req.query.date) ? 35 : 0;
        res.json({ cost: Math.ceil(BASE_COST * (1 - reduction / 100)) });
      }
    } else {
      if (req.query.age > 64) {
        res.json({ cost: Math.ceil(BASE_COST * 0.4) });
      } else {
        res.json({ cost: 0 });
      }
    }
  }
});
```

Request comes from the framework

```
app.get("/prices", (req: Request, res: Response) => {
  if (req.query.age < 6) {
    res.json({ cost: 0 });
  } else {
    if (req.query.type !== "night") {
      if (req.query.age < 15) {
        res.json({ cost: Math.ceil(BASE_COST * 0.7) });
      } else {
        const reduction = isMonday(req.query.date) ? 35 : 0;
        res.json({ cost: Math.ceil(BASE_COST * (1 - reduction / 100)) });
      }
    } else {
      if (req.query.age > 64) {
        res.json({ cost: Math.ceil(BASE_COST * 0.4) });
      } else {
        res.json({ cost: 0 });
      }
    }
  }
});
```

Distill &
Extract

… and spreads everywhere!

```
app.get("/prices", (req: Request, res: Response) => {
  if (req.query.age < 6) {
    res.json({ cost: 0 });
  } else {
    if (req.query.type !== "night") {
      if (req.query.age < 15) {
        res.json({ cost: Math.ceil(BASE_COST * 0.7) });
      } else {
        const reduction = isMonday(req.query.date) ? 35 : 0;
        res.json({ cost: Math.ceil(BASE_COST * (1 - reduction / 100)) });
      }
    } else {
      if (req.query.age > 64) {
        res.json({ cost: Math.ceil(BASE_COST * 0.4) });
      } else {
        res.json({ cost: 0 });
      }
    }
  }
});
```

Distill &
Extract

Same for Response

```
app.get("/prices", (req: Request, res: Response) => {
  if (req.query.age < 6) {
    res.json({ cost: 0 });
  } else {
    if (req.query.type !== "night") {
      if (req.query.age < 15) {
        res.json({ cost: Math.ceil(BASE_COST * 0.7) });
      } else {
        const reduction = isMonday(req.query.date) ? 35 : 0;
        res.json({ cost: Math.ceil(BASE_COST * (1 - reduction / 100)) });
      }
    } else {
      if (req.query.age > 64) {
        res.json({ cost: Math.ceil(BASE_COST * 0.4) });
      } else {
        res.json({ cost: 0 });
      }
    }
  }
});
```

Distill &
Extract

Let's *distill* them above/below the rest

```
app.get("/prices", (req: Request, res: Response) => {
  if (req.query.age < 6) {
    res.json({ cost: 0 });
  } else {
    if (req.query.type !== "night") {
      if (req.query.age < 15) {
        res.json({ cost: Math.ceil(BASE_COST * 0.7) });
      } else {
        const reduction = isMonday(req.query.date) ? 35 : 0;
        res.json({ cost: Math.ceil(BASE_COST * (1 - reduction / 100)) });
      }
    } else {
      if (req.query.age > 64) {
        res.json({ cost: Math.ceil(BASE_COST * 0.4) });
      } else {
        res.json({ cost: 0 });
      }
    }
  }
});
```

Distill &
Extract

```
app.get("/prices", (req: Request, res: Response) => {
  const { age } = req.query;

  if (age < 6) {
    res.json({ cost: 0 });
  } else {
    if (req.query.type !== "night") {
      if (age < 15) {
        res.json({ cost: Math.ceil(BASE_COST * 0.7) });
      } else {
        const reduction = isMonday(req.query.date) ? 35 : 0;
        res.json({ cost: Math.ceil(BASE_COST * (1 - reduction / 100)) });
      }
    } else {
      if (age > 64) {
        res.json({ cost: Math.ceil(BASE_COST * 0.4) });
      } else {
        res.json({ cost: 0 });
      }
    }
  }
});
```

Extract Variable

Distill &
Extract

```
app.get("/prices", (req: Request, res: Response) => {
  const { age, type } = req.query;

  if (age < 6) {
    res.json({ cost: 0 });
  } else {
    if (type !== "night") {
      if (age < 15) {
        res.json({ cost: Math.ceil(BASE_COST * 0.7) });
      } else {
        const reduction = isMonday(req.query.date) ? 35 : 0;
        res.json({ cost: Math.ceil(BASE_COST * (1 - reduction / 100)) });
      }
    } else {
      if (age > 64) {
        res.json({ cost: Math.ceil(BASE_COST * 0.4) });
      } else {
        res.json({ cost: 0 });
      }
    }
  }
});
```

Distill &
Extract

```
app.get("/prices", (req: Request, res: Response) => {
  const { age, type, date } = req.query;

  if (age < 6) {
    res.json({ cost: 0 });
  } else {
    if (type !== "night") {
      if (age < 15) {
        res.json({ cost: Math.ceil(BASE_COST * 0.7) });
      } else {
        const reduction = isMonday(date) ? 35 : 0;
        res.json({ cost: Math.ceil(BASE_COST * (1 - reduction / 100)) });
      }
    } else {
      if (age > 64) {
        res.json({ cost: Math.ceil(BASE_COST * 0.4) });
      } else {
        res.json({ cost: 0 });
      }
    }
  }
});
```

Distill &
Extract

```
app.get("/prices", (req: Request, res: Response) => {
  const { age, type, date } = req.query;
  const sendResponse = (...args) => res.json(...args);

  if (age < 6) {
    res.json({ cost: 0 });
  } else {
    if (type !== "night") {
      if (age < 15) {
        res.json({ cost: Math.ceil(BASE_COST * 0.7) });
      } else {
        const reduction = isMonday(date) ? 35 : 0;
        res.json({ cost: Math.ceil(BASE_COST * (1 - reduction / 100)) });
      }
    } else {
      if (age > 64) {
        res.json({ cost: Math.ceil(BASE_COST * 0.4) });
      } else {
        res.json({ cost: 0 });
      }
    }
  }
});
```

Extract Function

Distill &
Extract

```
app.get("/prices", (req: Request, res: Response) => {
  const { age, type, date } = req.query;
  const sendResponse = (...args) => res.json(...args);

  if (age < 6) {
    sendResponse({ cost: 0 });
  } else {
    if (type !== "night") {
      if (age < 15) {
        sendResponse({ cost: Math.ceil(BASE_COST * 0.7) });
      } else {
        const reduction = isMonday(date) ? 35 : 0;
        sendResponse({ cost: Math.ceil(BASE_COST * (1 - reduction / 100)) });
      }
    } else {
      if (age > 64) {
        sendResponse({ cost: Math.ceil(BASE_COST * 0.4) });
      } else {
        sendResponse({ cost: 0 });
      }
    }
  }
});
```

Distill & Extract

```
app.get("/prices", (req: Request, res: Response) => {
  const { age, type, date } = req.query;
  const sendResponse = (...args) => res.json(...args);

  if (age < 6) {
    sendResponse({ cost: 0 });
  } else {
    if (type !== "night") {
      if (age < 15) {
        sendResponse({ cost: Math.ceil(BASE_COST * 0.7) });
      } else {
        const reduction = isMonday(date) ? 35 : 0;
        sendResponse({ cost: Math.ceil(BASE_COST * (1 - reduction / 100)) });
      }
    } else {
      if (age > 64) {
        sendResponse({ cost: Math.ceil(BASE_COST * 0.4) });
      } else {
        sendResponse({ cost: 0 });
      }
    }
  }
});
```

Pure business logic

Distill & Extract

```
app.get("/prices", (req: Request, res: Response) => {
  const { age, type, date } = req.query;
  const sendResponse = (...args) => res.json(...args);

  computeAndSendCost(sendResponse, { age, type, date });
});

function computeAndSendCost(sendResponse, { age, type, date }) {
  if (age < 6) {
    sendResponse({ cost: 0 });
  } else {
    if (type !== "night") {
      if (age < 15) {
        sendResponse({ cost: Math.ceil(BASE_COST * 0.7) });
      } else {
        const reduction = isMonday(date) ? 35 : 0;
        sendResponse({ cost: Math.ceil(BASE_COST * (1 - reduction / 100)) });
      }
    } else {
      if (age > 64) {
        sendResponse({ cost: Math.ceil(BASE_COST * 0.4) });
      } else {
        sendResponse({ cost: 0 });
      }
    }
  }
```

Extract Function

Distill & Extract

```
app.get("/prices", (req: Request, res: Response) => {
  const { age, type, date } = req.query;
  const sendResponse = (...args) => res.json(...args);

  computeAndSendCost(sendResponse, { age, type, date });
});
```

Pure logic you can test, then refactor

```
function computeAndSendCost(sendResponse, { age, type, date }) {
  if (age < 6) {
    sendResponse({ cost: 0 });
  } else {
    if (type !== "night") {
      if (age < 15) {
        sendResponse({ cost: Math.ceil(BASE_COST * 0.7) });
      } else {
        const reduction = isMonday(date) ? 35 : 0;
        sendResponse({ cost: Math.ceil(BASE_COST * (1 - reduction / 100)) });
      }
    } else {
      if (age > 64) {
        sendResponse({ cost: Math.ceil(BASE_COST * 0.4) });
      } else {
        sendResponse({ cost: 0 });
      }
    }
  }
```

Distill & Extract

Framework code

```
app.get("/prices", (req: Request, res: Response) => {
  const { age, type, date } = req.query;
  const sendResponse = (...args) => res.json(...args);

  computeAndSendCost(sendResponse, { age, type, date });
});

function computeAndSendCost(sendResponse, { age, type, date }) {
  if (age < 6) {
    sendResponse({ cost: 0 });
  } else {
    if (type !== "night") {
      if (age < 15) {
        sendResponse({ cost: Math.ceil(BASE_COST * 0.7) });
      } else {
        const reduction = isMonday(date) ? 35 : 0;
        sendResponse({ cost: Math.ceil(BASE_COST * (1 - reduction / 100)) });
      }
    } else {
      if (age > 64) {
        sendResponse({ cost: Math.ceil(BASE_COST * 0.4) });
      } else {
        sendResponse({ cost: 0 });
      }
    }
  }
```

Logic is hard to test

It's sprinkled with objects we don't control

```
app.get("/prices", (req: Request, res: Response) => {
  if (req.query.age < 6) {
    res.json({ cost: 0 });
  } else {
    if (req.query.type !== "night") {
      if (req.query.age < 15) {
        res.json({ cost: Math.ceil(BASE_COST * 0.7) });
      } else {
        const reduction = isMonday(req.query.date) ? 35 : 0;
        res.json({ cost: Math.ceil(BASE_COST * (1 - reduction / 100)) });
      }
    } else {
      if (req.query.age > 64) {
        res.json({ cost: Math.ceil(BASE_COST * 0.4) });
      } else {
        res.json({ cost: 0 });
      }
    }
  }
});
```

Distill & Extract

Extract business, so we can test and refactor that part

```
app.get("/prices", (req: Request, res: Response) => {
  const { age, type, date } = req.query;
  const sendResponse = (...args) => res.json(...args);

  computeAndSendCost(sendResponse, { age, type, date });
});

function computeAndSendCost(sendResponse, { age, type, date }) {
  if (age < 6) {
    sendResponse({ cost: 0 });
  } else {
    if (type !== "night") {
      if (age < 15) {
        sendResponse({ cost: Math.ceil(BASE_COST * 0.7) });
      } else {
        const reduction = isMonday(date) ? 35 : 0;
        sendResponse({ cost: Math.ceil(BASE_COST * (1 - reduction / 100)) });
      }
    } else {
      if (age > 64) {
        sendResponse({ cost: Math.ceil(BASE_COST * 0.4) });
      } else {
        sendResponse({ cost: 0 });
      }
    }
  }
```

6 Refactorings for Legacy Code

Inline

Extract

Distill & Extract

Rename

Change Signature

Extend & Override

Extend &
Override

What makes this code annoying to test?

```
export class Game {
  add(name) {
    this.players.push(name);
    this.places[this.howManyPlayers() - 1] = 0;
    this.purses[this.howManyPlayers() - 1] = 0;
    this.inPenaltyBox[this.howManyPlayers() - 1] = false;

    console.log(name + " was added");
    console.log("They are player number " + this.players.length);

    return true;
  }

  // … more code
}
```

Things outside of our control (again)

```
export class Game {
  add(name) {
    this.players.push(name);
    this.places[this.howManyPlayers() - 1] = 0;
    this.purses[this.howManyPlayers() - 1] = 0;
    this.inPenaltyBox[this.howManyPlayers() - 1] = false;

    console.log(name + " was added");
    console.log("They are player number " + this.players.length);

    return true;
  }

  // … more code
}
```

Extend &
Override

```
export class Game {
  add(name) {
    this.players.push(name);
    this.places[this.howManyPlayers() - 1] = 0;
    this.purses[this.howManyPlayers() - 1] = 0;
    this.inPenaltyBox[this.howManyPlayers() - 1] = false;

    console.log(name + " was added");
    console.log("They are player number " + this.players.length);

    return true;
  }

  // … more code
}
```

Extract Variable

Extend &
Override

```
export class Game {
  add(name) {
    this.players.push(name);
    this.places[this.howManyPlayers() - 1] = 0;
    this.purses[this.howManyPlayers() - 1] = 0;
    this.inPenaltyBox[this.howManyPlayers() - 1] = false;

    const appleSauce = name + " was added";
    console.log(appleSauce);
    console.log("They are player number " + this.players.length);

    return true;
  }

  // … more code
}
```

Extract
Function

Extend &
Override

```
export class Game {
  add(name) {
    this.players.push(name);
    this.places[this.howManyPlayers() - 1] = 0;
    this.purses[this.howManyPlayers() - 1] = 0;
    this.inPenaltyBox[this.howManyPlayers() - 1] = false;

    const appleSauce = name + " was added";
    this.log(appleSauce);
    console.log("They are player number " + this.players.length);

    return true;
  }

  log(message) {
    console.log(message);
  }

  // … more code
}
```

Inline
Variable

Extend & Override

```
export class Game {
  add(name) {
    this.players.push(name);
    this.places[this.howManyPlayers() - 1] = 0;
    this.purses[this.howManyPlayers() - 1] = 0;
    this.inPenaltyBox[this.howManyPlayers() - 1] = false;

    this.log(name + " was added");
    console.log("They are player number " + this.players.length);

    return true;
  }

  log(message) {
    console.log(message);
  }

  // … more code
}
```

Replace this statement too

Extend & Override

```
export class Game {
  add(name) {
    this.players.push(name);
    this.places[this.howManyPlayers() - 1] = 0;
    this.purses[this.howManyPlayers() - 1] = 0;
    this.inPenaltyBox[this.howManyPlayers() - 1] = false;

    this.log(name + " was added");
    this.log("They are player number " + this.players.length);

    return true;
  }

  log(message) {
    console.log(message);
  }

  // … more code
}
```

Extend &
Override

We have isolated the annoying part

```
export class Game {
  add(name) {
    this.players.push(name);
    this.places[this.howManyPlayers() - 1] = 0;
    this.purses[this.howManyPlayers() - 1] = 0;
    this.inPenaltyBox[this.howManyPlayers() - 1] = false;

    this.log(name + " was added");
    this.log("They are player number " + this.players.length);

    return true;
  }

  log(message) {
    console.log(message);
  }

  // … more code
}
```

Extend &
Override

We can Extend the code

```
export class Game {
  add(name) {
    // … code to test
  }

  log(message) {
    console.log(message);
  }

  // … more code
}
```



```
export class TestableGame extends Game {

}
```

Extend &
Override

Then, Override the implementation

```
export class Game {
  add(name) {
    // … code to test
  }

  log(message) {
    console.log(message);
  }

  // … more code
}
```



```
export class TestableGame extends Game {
  log(message) {
    // custom implementation
  }
}
```

Extend &
Override

Then, Override the implementation

```
export class Game {
  add(name) {
    // … code to test
  }

  log(message) {
    console.log(message);
  }

  // … more code
}
```



```
export class TestableGame extends Game {
  loggedMessages = []

  log(message) {
    this.loggedMessages.push(message)
  }
}
```

Refactoring is not over...

```
export class TestableGame extends Game {
  loggedMessages = []

  log(message) {
    this.loggedMessages.push(message)
  }
}
```

But now, you can write tests!

Extend &
Override

But what if my code doesn't use classes?

```
export async function handleSlackWebhook(event) {
  console.log("Received Slack webhook", event.data);

  const teamId = event.data.context.teamId;
  const matchingAccount = await findSlackMatchingAccount(teamId);
  const tenantId = matchingAccount?.tenantId;
  if (!tenantId) {
    throw new Error(`No Tenant ID found for team "${teamId}"`);
  }

  // … more code

  console.log("Successfully posted Slack webhook");
  return "ok";
}
```

Extend &
Override

Same mindset: lean on small refactorings, create an Override

```
export async function handleSlackWebhook(event) {
  console.log("Received Slack webhook", event.data);

  const teamId = event.data.context.teamId;
  const matchingAccount = await findSlackMatchingAccount(teamId);
  const tenantId = matchingAccount?.tenantId;
  if (!tenantId) {
    throw new Error(`No Tenant ID found for team "${teamId}"`);
  }

  // … more code

  console.log("Successfully posted Slack webhook");
  return "ok";
}
```

Extend &
Override

Extract
Function

```
export async function handleSlackWebhook(event) {
  console.log("Received Slack webhook", event.data);

  const teamId = event.data.context.teamId;
  const matchingAccount = await findSlackMatchingAccount(teamId);
  const tenantId = matchingAccount?.tenantId;
  if (!tenantId) {
    throw new Error(`No Tenant ID found for team "${teamId}"`);
  }

  // … more code

  console.log("Successfully posted Slack webhook");
  return "ok";
}
```

Extend &
Override

Change
Signature

```
export async function handleSlackWebhook(event) {
  log("Received Slack webhook", event.data);

  const teamId = event.data.context.teamId;
  const matchingAccount = await findSlackMatchingAccount(teamId);
  const tenantId = matchingAccount?.tenantId;
  if (!tenantId) {
    throw new Error(`No Tenant ID found for team "${teamId}"`);
  }

  // … more code

  console.log("Successfully posted Slack webhook");
  return "ok";
}

function log(...args) {
  console.log(...args);
}
```

Extend &
Override

```
export async function handleSlackWebhook(event, log = log) {
  log("Received Slack webhook", event.data);

  const teamId = event.data.context.teamId;
  const matchingAccount = await findSlackMatchingAccount(teamId);
  const tenantId = matchingAccount?.tenantId;
  if (!tenantId) {
    throw new Error(`No Tenant ID found for team "${teamId}"`);
  }

  // … more code

  console.log("Successfully posted Slack webhook");
  return "ok";
}

function log(...args) {
  console.log(...args);
}
```

Repeat

Extend & Override

```
export async function handleSlackWebhook(event, log = log) {
  log("Received Slack webhook", event.data);

  const teamId = event.data.context.teamId;
  const matchingAccount = await findSlackMatchingAccount(teamId);
  const tenantId = matchingAccount?.tenantId;
  if (!tenantId) {
    throw new Error(`No Tenant ID found for team "${teamId}"`);
  }

  // … more code

  log("Successfully posted Slack webhook");
  return "ok";
}

function log(...args) {
  console.log(...args);
}
```

Extend &
Override

```
export async function handleSlackWebhook(
  event,
  log = log
) {
  log("Received Slack webhook", event.data);

  const teamId = event.data.context.teamId;
  const matchingAccount = await findSlackMatchingAccount(teamId);
  const tenantId = matchingAccount?.tenantId;
  if (!tenantId) {
    throw new Error(`No Tenant ID found for team "${teamId}"`);
  }

  // … more code

  log("Successfully posted Slack webhook");
  return "ok";
}

function log(...args) {
  console.log(...args);
}
```

Extend &
Override

```
export async function handleSlackWebhook(
  event,
  log = log,
  findSlackMatchingAccount = findSlackMatchingAccount
) {
  log("Received Slack webhook", event.data);

  const teamId = event.data.context.teamId;
  const matchingAccount = await findSlackMatchingAccount(teamId);
  const tenantId = matchingAccount?.tenantId;
  if (!tenantId) {
    throw new Error(`No Tenant ID found for team "${teamId}"`);
  }

  // … more code

  log("Successfully posted Slack webhook");
  return "ok";
}

function log(...args) {
  console.log(...args);
}
```

Refactoring is not over...

```
export async function handleSlackWebhook(
  event,
  log = log,
  findSlackMatchingAccount = findSlackMatchingAccount
) {
  // … implementation
}
```

But now, you can write tests!

6 Refactorings for Legacy Code

Inline

Extract

Distill & Extract

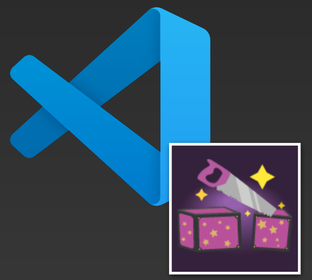
Rename

Change Signature

Extend & Override

Can't afford to stop?
Refactor while you develop!

Lean on automated refactorings

	Inline	Rename	Extract	Change Signature
	Ctrl Alt N	⇧ F6	Ctrl Alt V	Ctrl F6 ⌘ F6
	⌘ ⌥ N	F2	⌘ ⌥ V	Alt ↵ ⌥ ↵

To go further

- 📕 Martin Fowler's "Refactoring"
- 🪚 Refactoring Guru
- 📙 Legacy Code: First Aid Kit
- 🔮 Abracadabra, VS Code extension

Thanks!

Nicolas Carlo

@nicoespeon

Can AI do the job?
Bonus

Not yet, because:

- How can you tell if it got it right?
- Legacy Code has implicit surprises that will trick AI = guaranteed regressions

However:

✓ AI assistant can help you grok code faster

✓ AI can help you change code faster

✓ Combine AI with static analysis to improve reliability (see CodeScene)