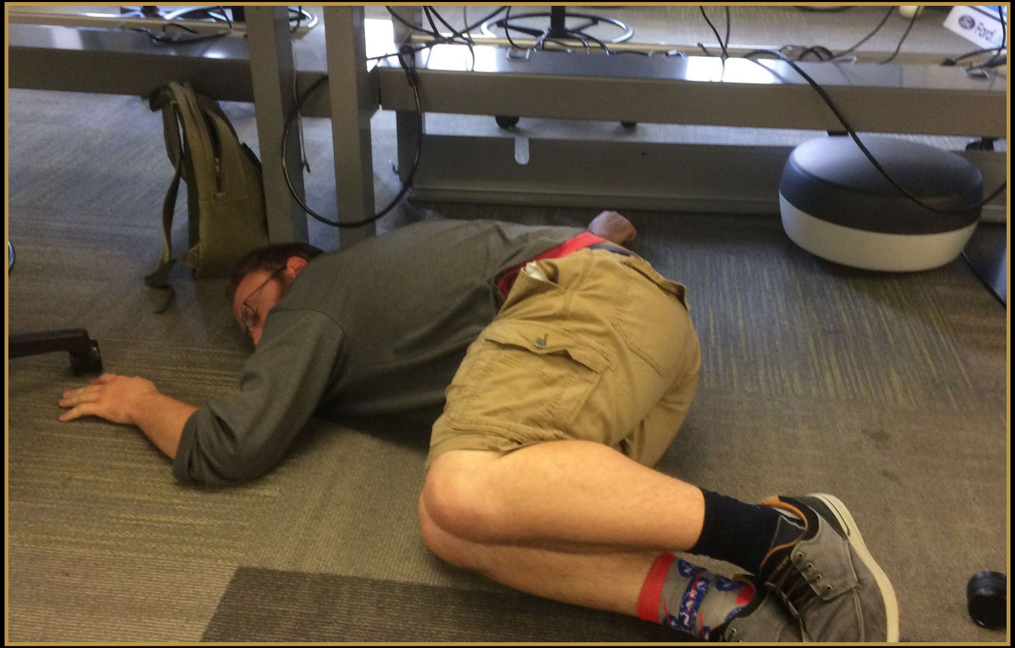
Introduction to Programming

Part 0 of N

Who is Chris Boyer?



SUCCESS

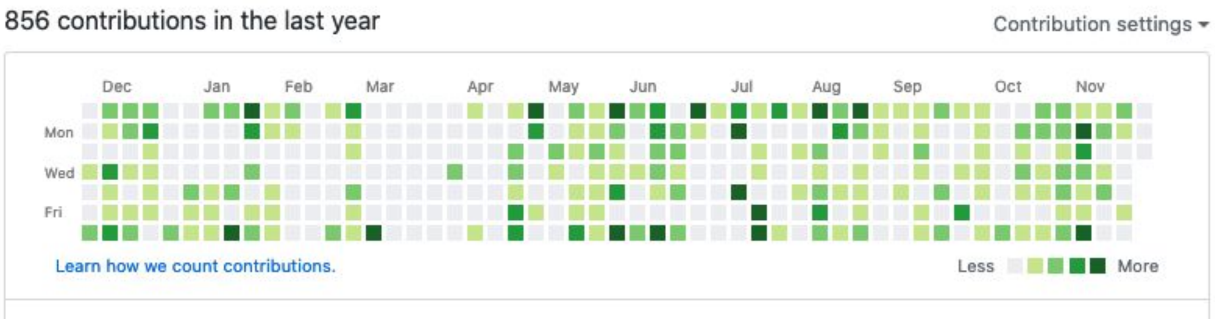
Begins with succ.

Fun Facts

My first time programming was in my Introduction to Programming in Java course as a First-Year student

Fun Facts

Programming is what I do for a job, but also what I do for a hobby



856 contributions in the last year

Learn how we count contributions.

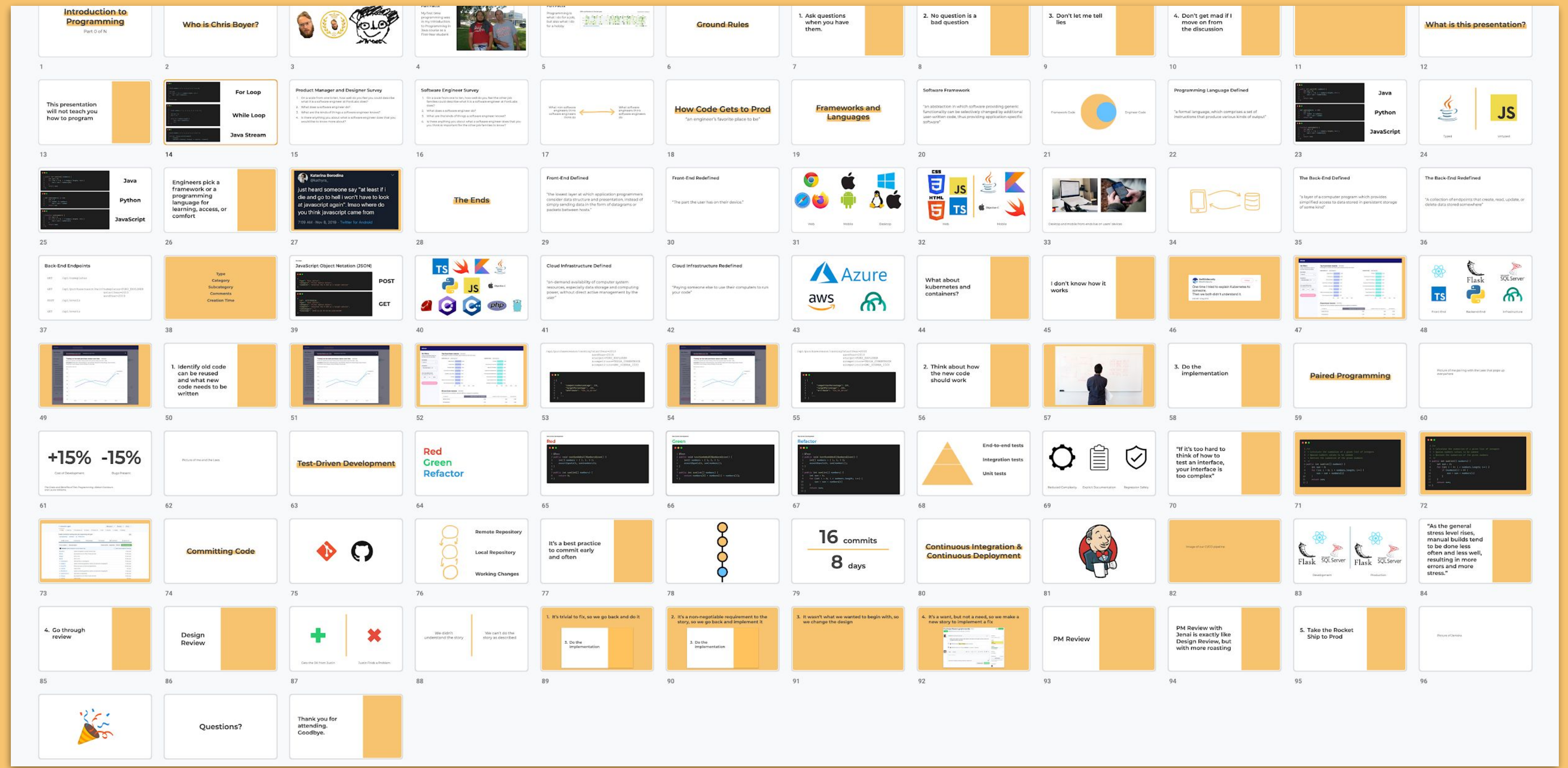
Ground Rules

1. Ask questions when you have them.

2. No question is a bad question

3. Don’t let me tell lies

4. Don’t get mad if I move on from the discussion



Introduction to Programming
Part 0 of N
1

Who is Chris Boyer?
2

3

4

5

Ground Rules
6

1. Ask questions when you have them.
7

2. No question is a bad question
8

3. Don't let me tell lies
9

4. Don't get mad if I move on from the discussion
10

11

What is this presentation?
12

This presentation will not teach you how to program
13

For Loop
While Loop
Java Stream
14

Product Manager and Designer Survey
15

Software Engineer Survey
16

17

How Code Gets to Prod
"an engineer's favorite place to be"
18

Frameworks and Languages
19

Software Framework
20

21

Programming Language Defined
"a formal language, which comprises a set of instructions that produce various kinds of output"
22

Java
Python
JavaScript
23

JS
Typed
Untyped
24

Java
Python
JavaScript
25

Engineers pick a framework or a programming language for learning, access, or comfort
26

Katerina Borodina
just heard someone say "at least if i die and go to hell i won't have to look at javascript again". lmao where do you think javascript came from
7:09 AM · Nov 8, 2019 · Twitter for Android
27

The Ends
28

Front-End Defined
29

Front-End Redefined
"The part the user has on their device."
30

Web
Mobile
Desktop
31

JS
TS
Web
Mobile
32

33

34

The Back-End Defined
35

The Back-End Redefined
36

Back-End Endpoints
37

Type
Category
Subcategory
Comments
Creation Time
38

JavaScript Object Notation (JSON)
POST
GET
39

TS
JS
php
40

Cloud Infrastructure Defined
41

Cloud Infrastructure Redefined
"Paying someone else to use their computers to run your code"
42

Azure
aws
43

What about kubernetes and containers?
44

I don't know how it works
45

46

47

Flask
SQL Server
TS
Front-End
Backend-End
Infrastructure
48

49

1. Identify old code can be reused and what new code needs to be written
50

51

52

53

54

55

2. Think about how the new code should work
56

57

3. Do the implementation
58

Paired Programming
59

60

+15%
Cost of Development
-15%
Bugs Present
61

62

Test-Driven Development
63

Red
Green
Refactor
64

Red
65

Green
66

Refactor
67

End-to-end tests
Integration tests
Unit tests
68

Reduced Complexity
Explicit Documentation
Regression Safety
69

"If it's too hard to think of how to test an interface, your interface is too complex"
70

71

72

73

Committing Code
74

75

Remote Repository
Local Repository
Working Changes
76

It's a best practice to commit early and often
77

78

16 commits
8 days
79

Continuous Integration & Continuous Deployment
80

81

82

Flask
SQL Server
Development
Production
83

"As the general stress level rises, manual builds tend to be done less often and less well, resulting in more errors and more stress."
84

4. Go through review
85

Design Review
86

87

We didn't understand the story
We can't do the story as described
88

1. It's trivial to fix, so we go back and do it
3. Do the implementation
89

2. It's a non-negotiable requirement to the story, so we go back and implement it
3. Do the implementation
90

3. It wasn't what we wanted to begin with, so we change the design
91

4. It's a want, but not a need, so we make a new story to implement a fix
92

PM Review
93

PM Review with Jenai is exactly like Design Review, but with more roasting
94

5. Take the Rocket Ship to Prod
95

96

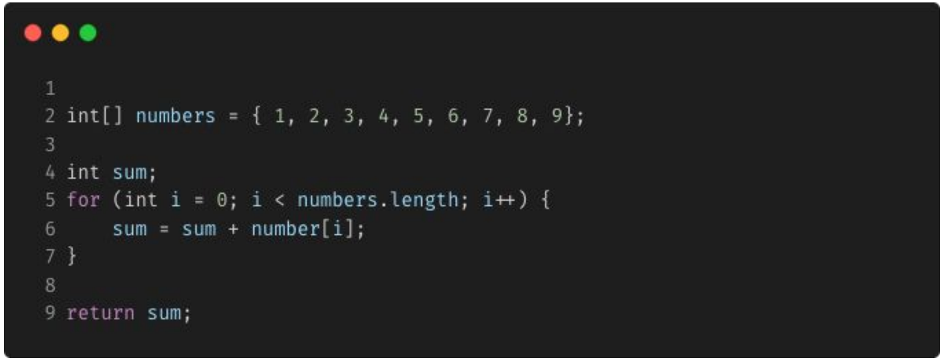
97

Questions?
98

Thank you for attending. Goodbye.
99

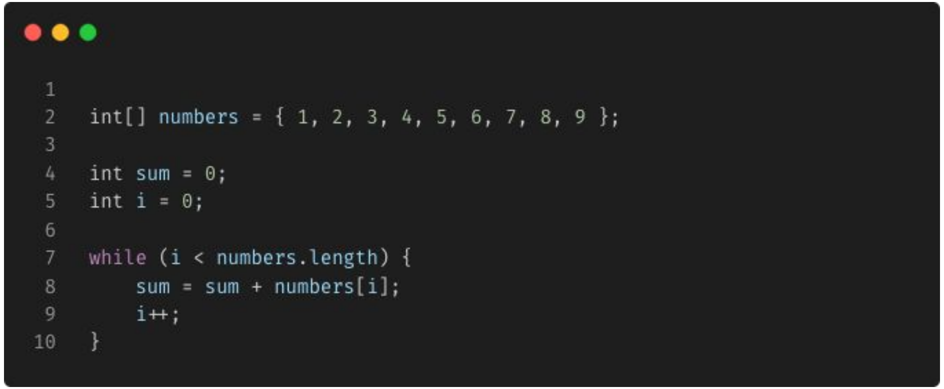
What is this presentation?

This presentation will not teach you how to program



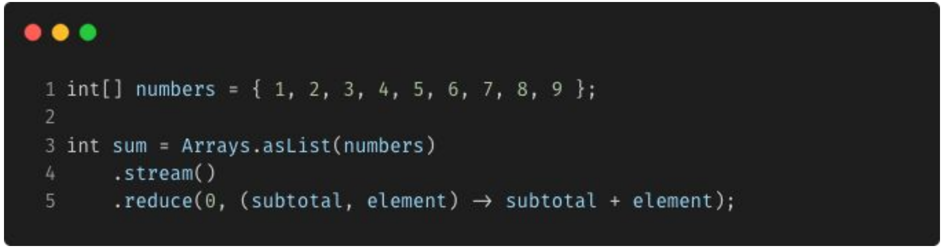
```java

int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9};

int sum;
for (int i = 0; i < numbers.length; i++) {
    sum = sum + number[i];
}

return sum;
```

For Loop



```java

int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

int sum = 0;
int i = 0;

while (i < numbers.length) {
    sum = sum + numbers[i];
    i++;
}
```

While Loop



```java
int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

int sum = Arrays.asList(numbers)
    .stream()
    .reduce(0, (subtotal, element) -> subtotal + element);
```

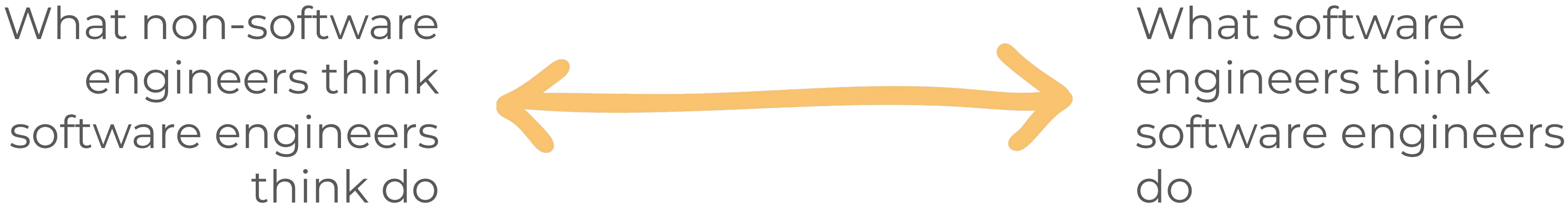
Java Stream

Product Manager and Designer Survey

1. On a scale from one to ten, how well do you feel you could describe what it is a software engineer at FordLabs does?
2. What does a software engineer do?
3. What are the kinds of things a software engineer knows?
4. Is there anything you about what a software engineer does that you would like to know more about?

Software Engineer Survey

1. On a scale from one to ten, how well do you feel the other job families could describe what it is a software engineer at FordLabs does?
2. What does a software engineer do?
3. What are the kinds of things a software engineer knows?
4. Is there anything you about what a software engineer does that you you think is important for the other job families to know?



What non-software engineers think software engineers think do

What software engineers think software engineers do

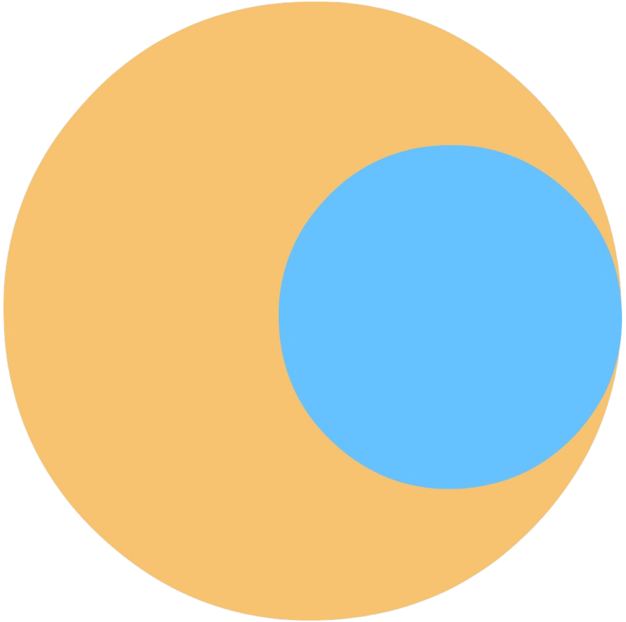
How Code Gets to Prod

“an engineer’s favorite place to be”

Frameworks and Languages

Software Framework

“an abstraction in which software providing generic functionality can be selectively changed by additional user-written code, thus providing application-specific software”

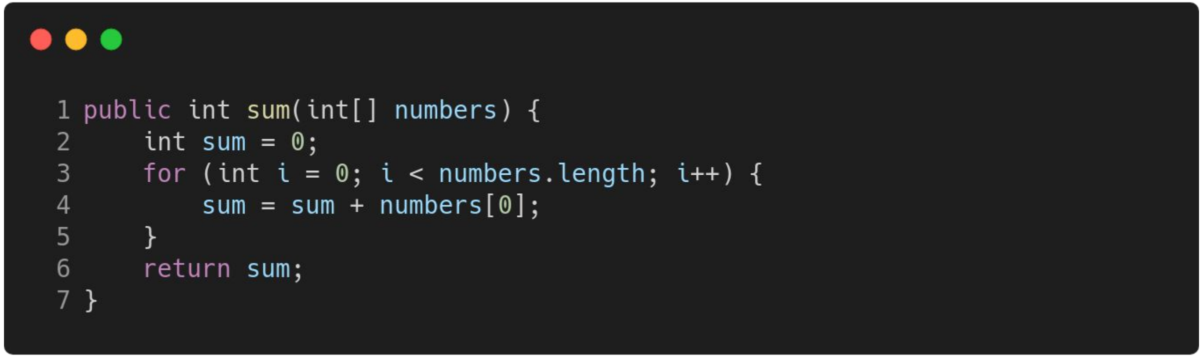


Framework Code

Engineer Code

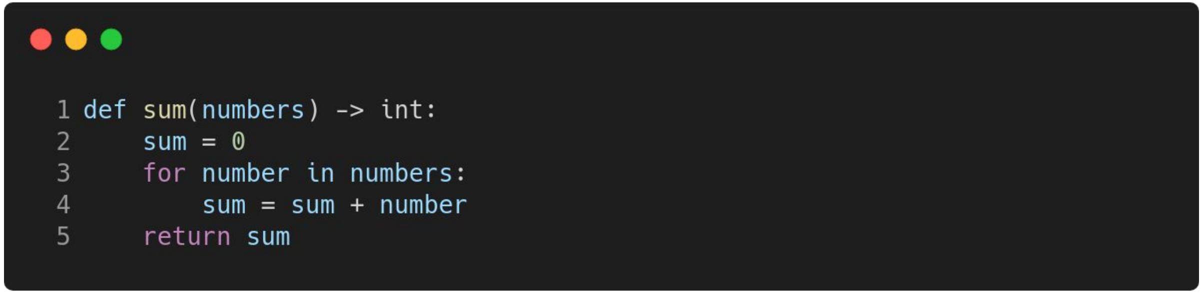
Programming Language Defined

“a formal language, which comprises a set of instructions that produce various kinds of output”



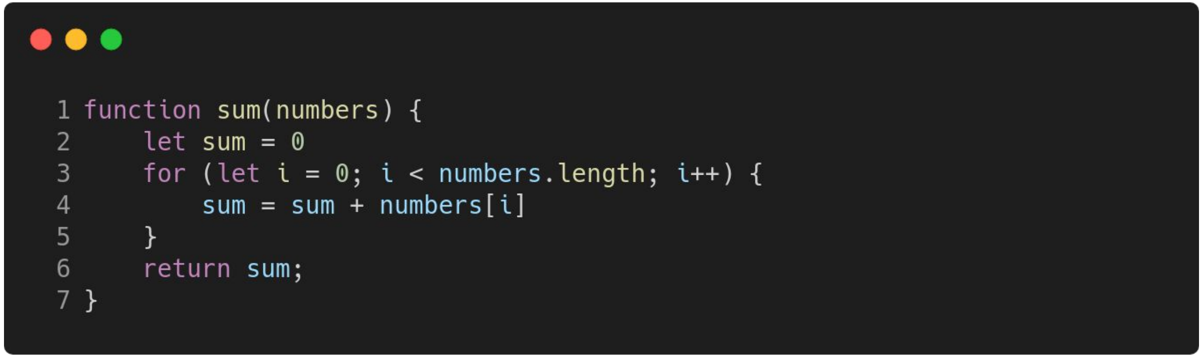
```java
public int sum(int[] numbers) {
    int sum = 0;
    for (int i = 0; i < numbers.length; i++) {
        sum = sum + numbers[0];
    }
    return sum;
}
```

Java



```python
def sum(numbers) -> int:
    sum = 0
    for number in numbers:
        sum = sum + number
    return sum
```

Python

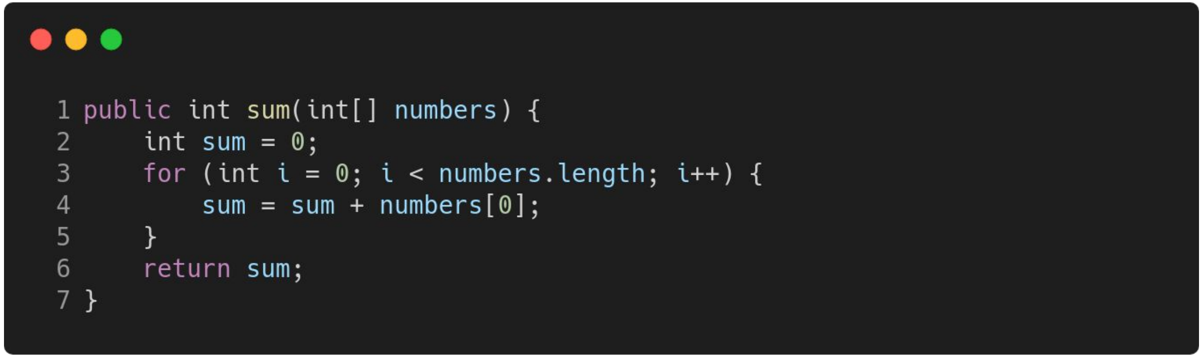


```javascript
function sum(numbers) {
    let sum = 0
    for (let i = 0; i < numbers.length; i++) {
        sum = sum + numbers[i]
    }
    return sum;
}
```

JavaScript

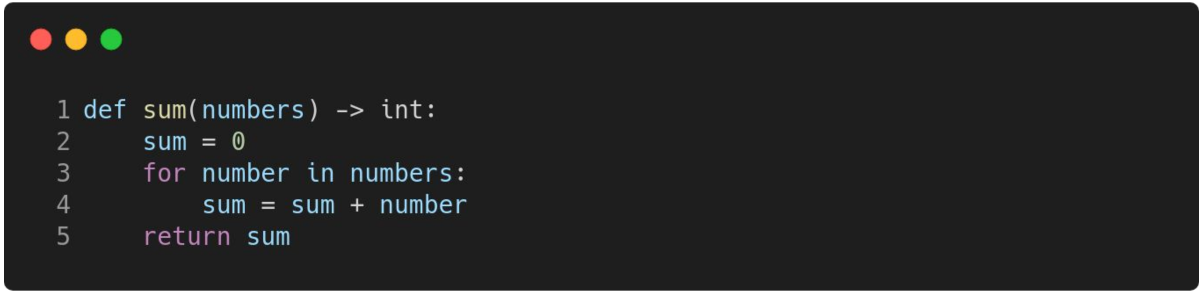
Typed

Untyped



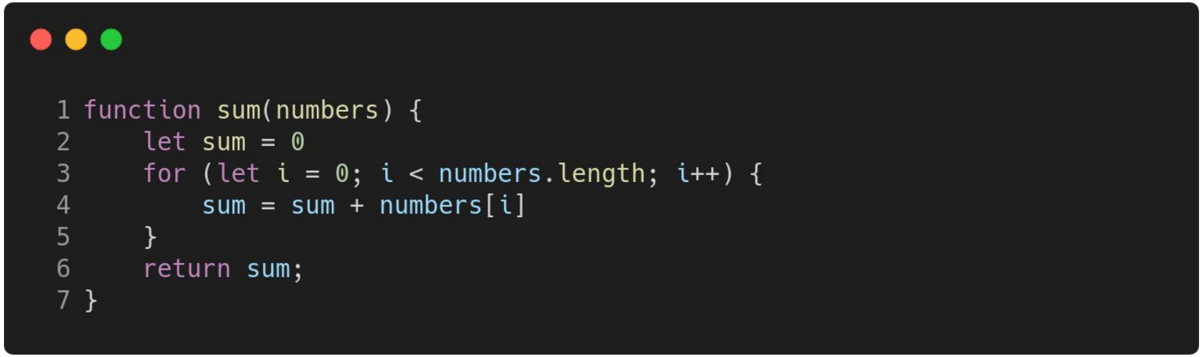
```java
public int sum(int[] numbers) {
    int sum = 0;
    for (int i = 0; i < numbers.length; i++) {
        sum = sum + numbers[0];
    }
    return sum;
}
```

Java



```python
def sum(numbers) -> int:
    sum = 0
    for number in numbers:
        sum = sum + number
    return sum
```

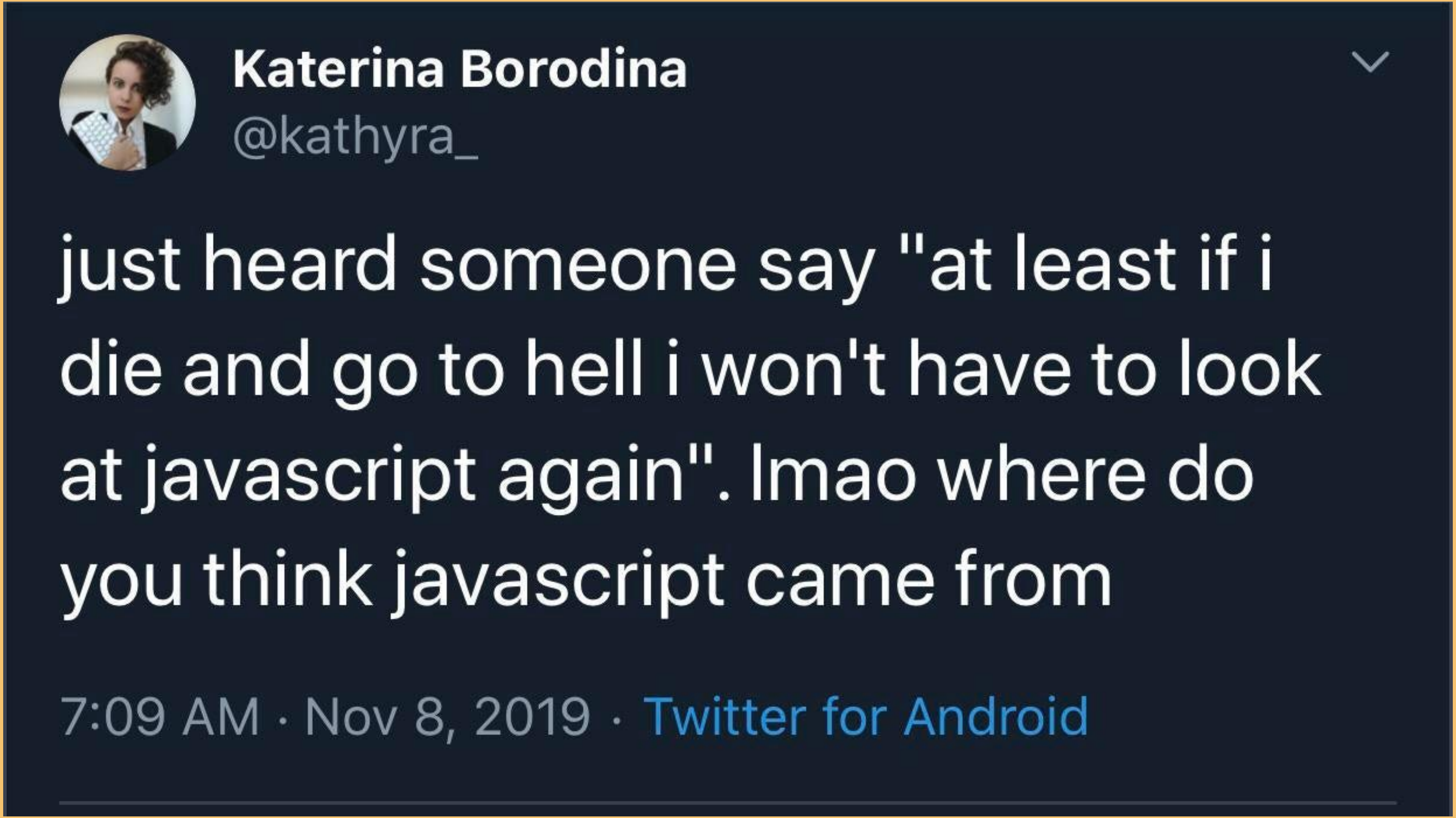
Python



```javascript
function sum(numbers) {
    let sum = 0
    for (let i = 0; i < numbers.length; i++) {
        sum = sum + numbers[i]
    }
    return sum;
}
```

JavaScript

Engineers pick a framework or a programming language for learning, access, or comfort



Katerina Borodina
@kathyra_

just heard someone say "at least if i die and go to hell i won't have to look at javascript again". lmao where do you think javascript came from

7:09 AM · Nov 8, 2019 · Twitter for Android

The Ends

Front-End Defined

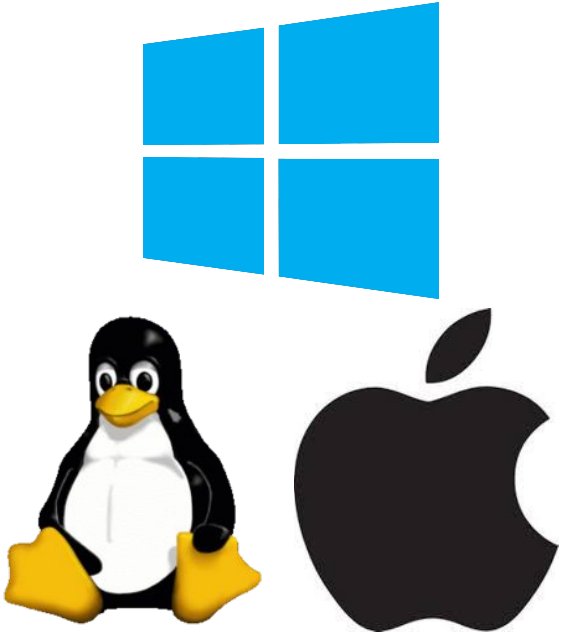
“the lowest layer at which application programmers consider data structure and presentation, instead of simply sending data in the form of datagrams or packets between hosts.”

Front-End Redefined

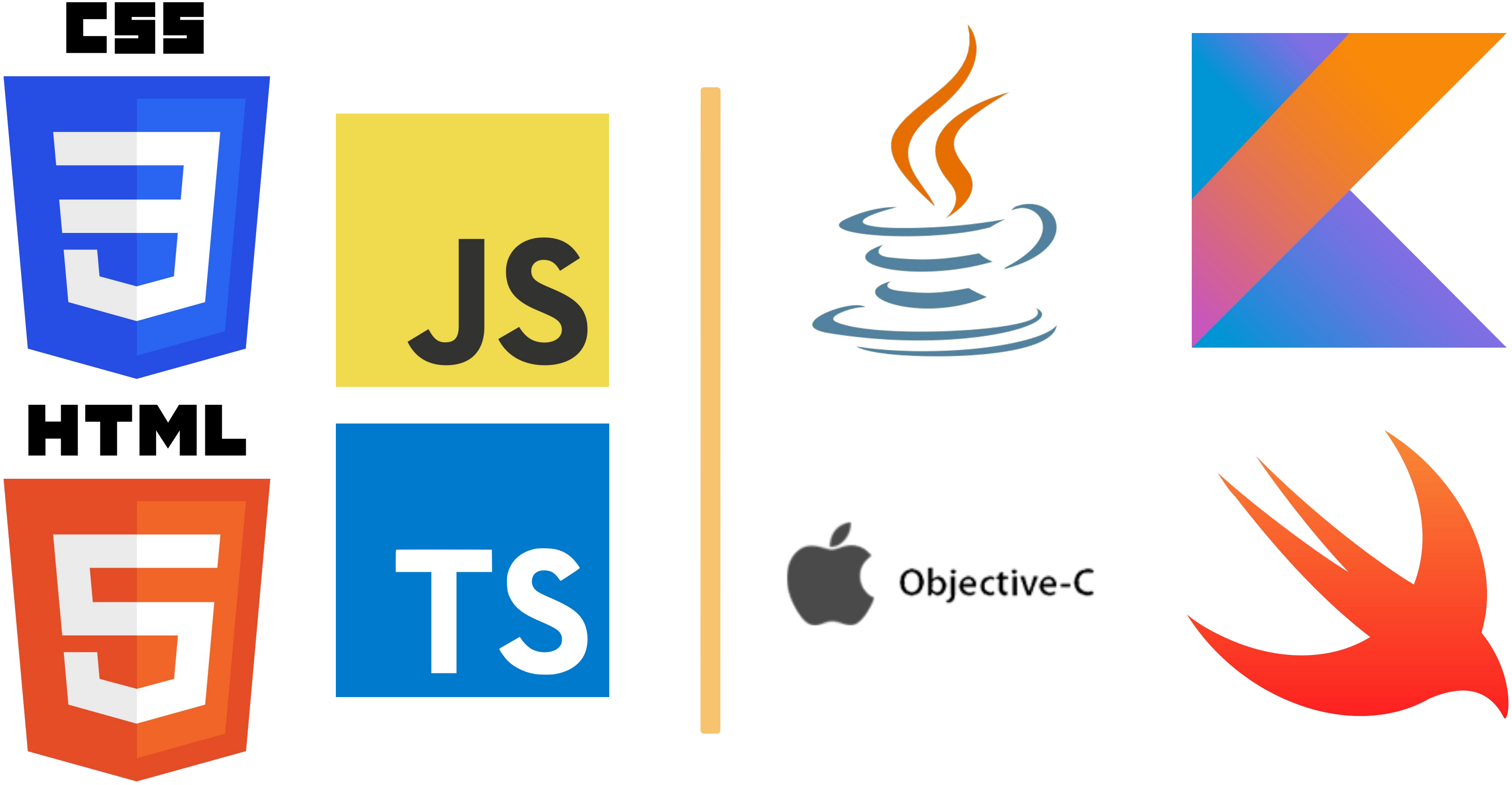
“The part the user has on their device.”

Web

Mobile

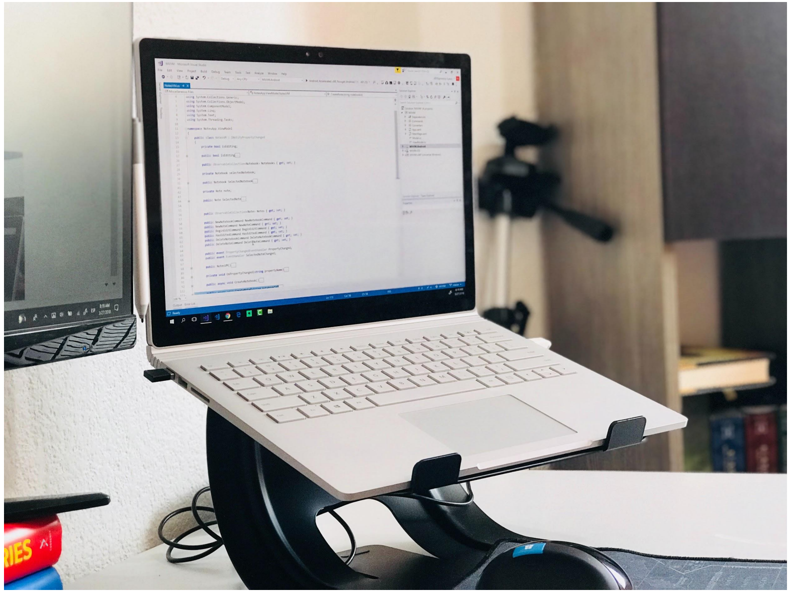


Desktop

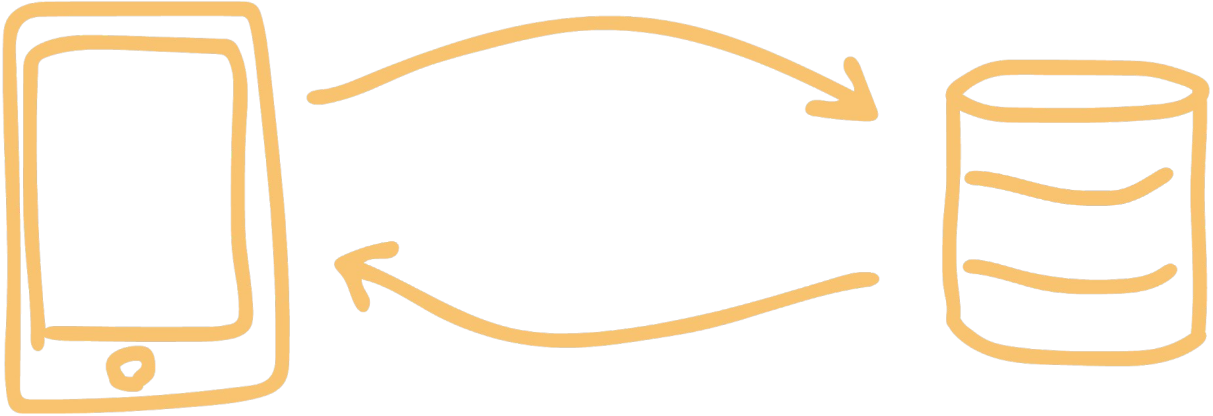


Web

Mobile



Desktop and mobile front-ends live on users’ devices



The Back-End Defined

“a layer of a computer program which provides simplified access to data stored in persistent storage of some kind”

The Back-End Redefined

“A collection of endpoints that create, read, update, or delete data stored somewhere”

Back-End Endpoints

```
GET      /api/nameplates


GET      /api/purchasereason/main?nameplates=STALLION
                                  &startYear=2010
                                  &endYear=2019

POST     /api/events


GET      /api/events
```

Type

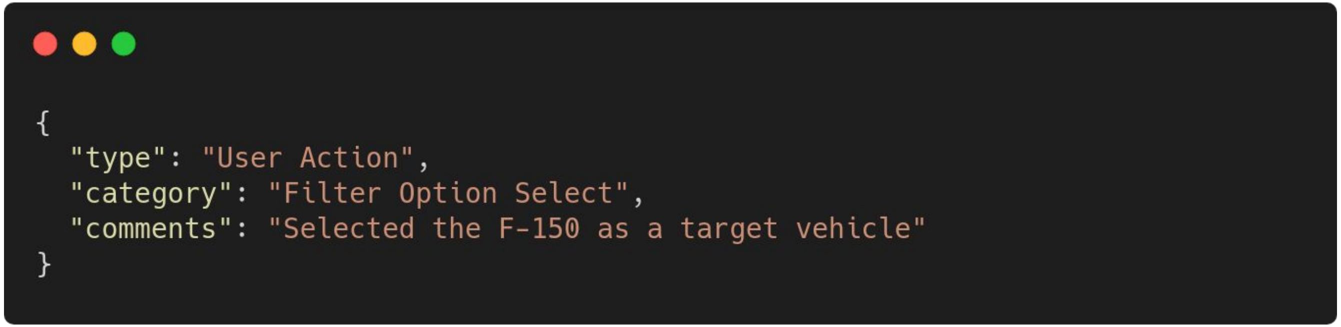
Category

Subcategory

Comments

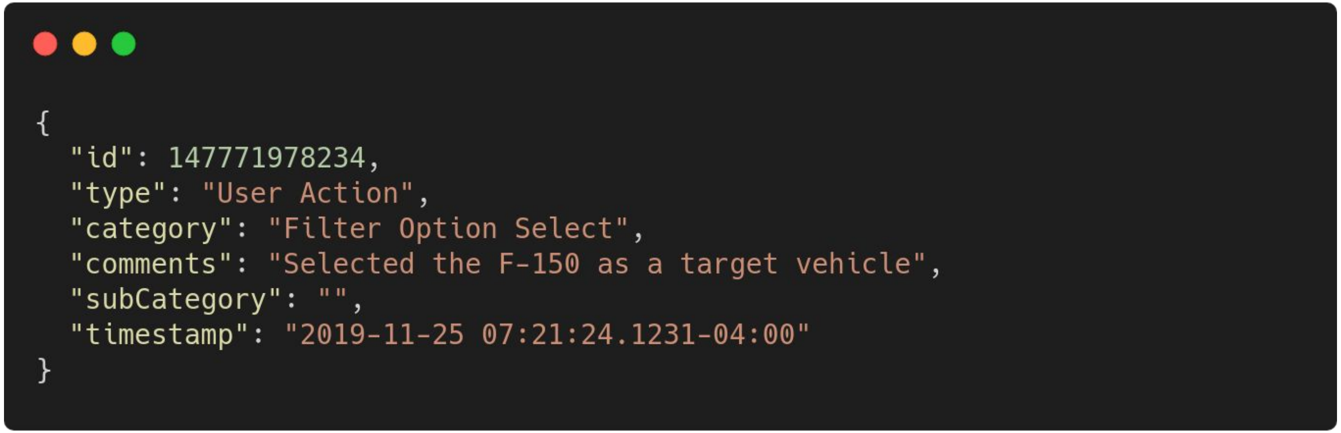
Creation Time

JavaScript Object Notation (JSON)



```
{
  "type": "User Action",
  "category": "Filter Option Select",
  "comments": "Selected the F-150 as a target vehicle"
}
```

POST



```
{
  "id": 147771978234,
  "type": "User Action",
  "category": "Filter Option Select",
  "comments": "Selected the F-150 as a target vehicle",
  "subCategory": "",
  "timestamp": "2019-11-25 07:21:24.1231-04:00"
}
```

GET

TS
JS
Objective-C
C#
C++
php

Cloud Infrastructure Defined

“on-demand availability of computer system resources, especially data storage and computing power, without direct active management by the user”

Cloud Infrastructure Redefined

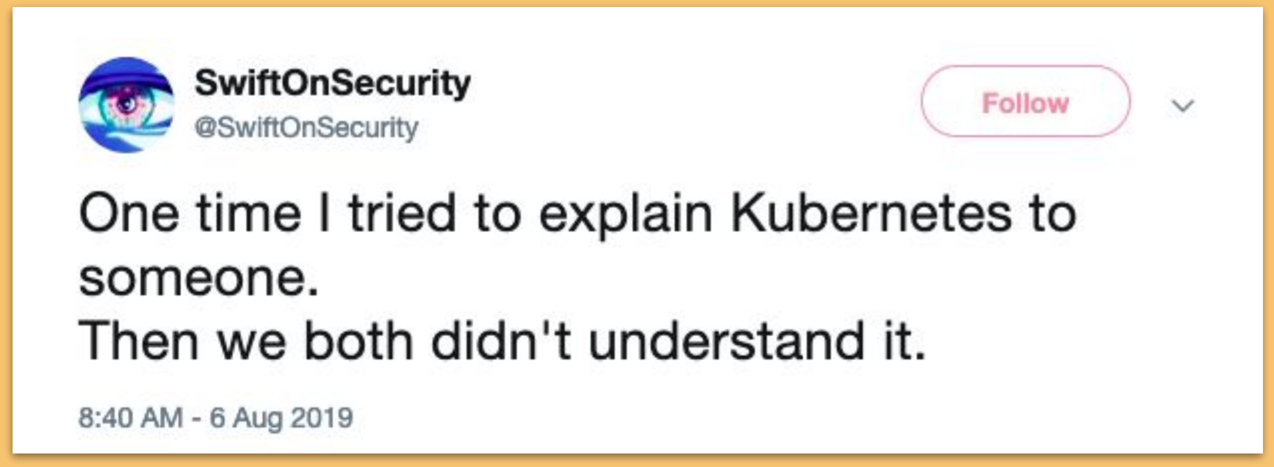
“Paying someone else to use their computers to run your code”

Azure

aws

What about kubernetes and containers?

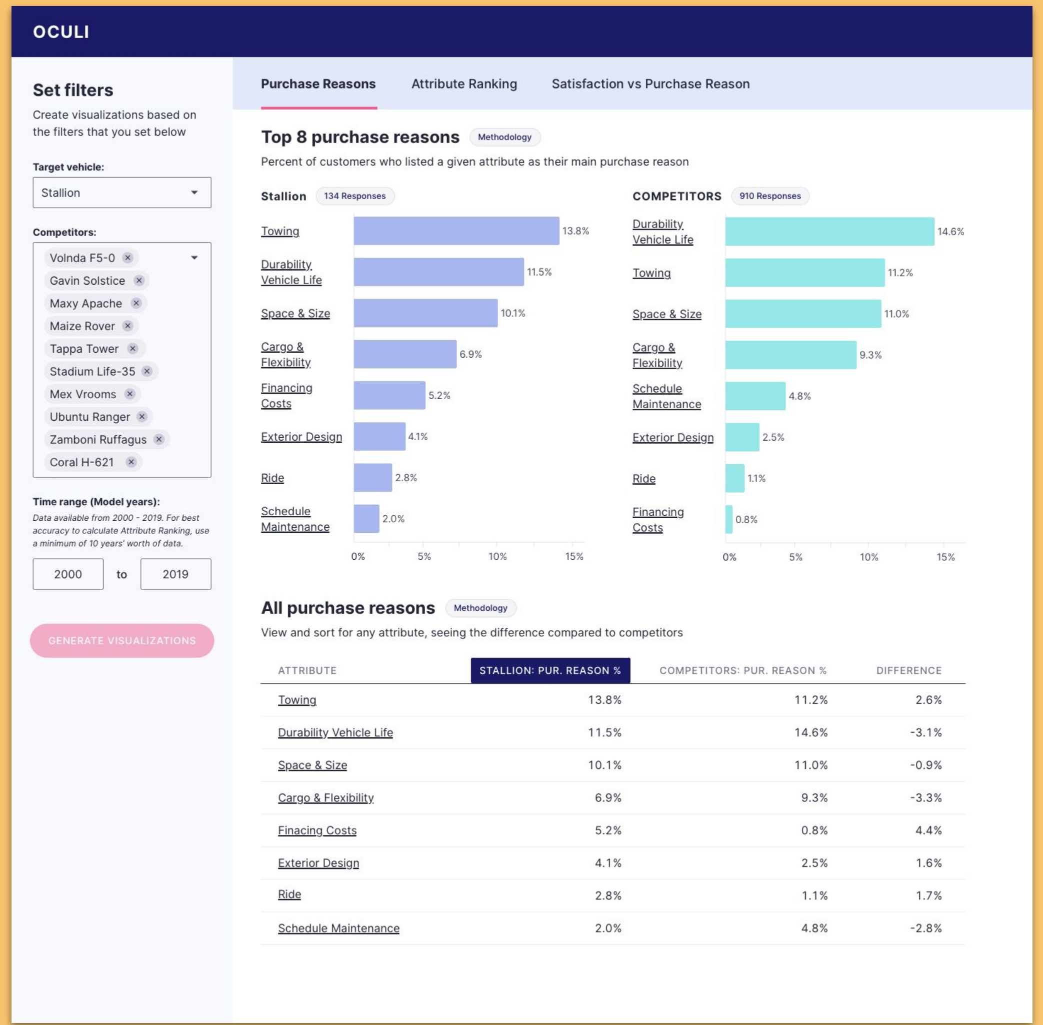
I don’t know how it works



SwiftOnSecurity
@SwiftOnSecurity

One time I tried to explain Kubernetes to someone.
Then we both didn't understand it.

8:40 AM - 6 Aug 2019



Set filters

Create visualizations based on the filters that you set below

Target vehicle:

Stallion

Competitors:

- Volnda F5-0 ×
- Gavin Solstice ×
- Maxy Apache ×
- Maize Rover ×
- Tappa Tower ×
- Stadium Life-35 ×
- Mex Vrooms ×
- Ubuntu Ranger ×
- Zamboni Ruffagus ×
- Coral H-621 ×

Time range (Model years):

Data available from 2000 - 2019. For best accuracy to calculate Attribute Ranking, use a minimum of 10 years' worth of data.

2000 to 2019

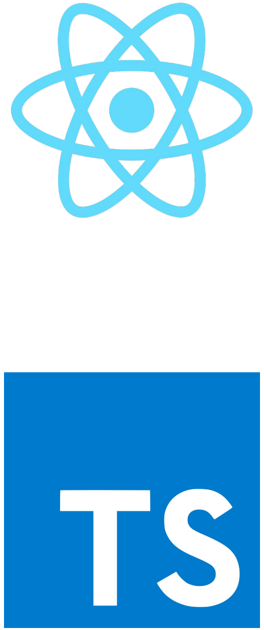
Top 8 purchase reasons

Percent of customers who listed a given attribute as their main purchase reason

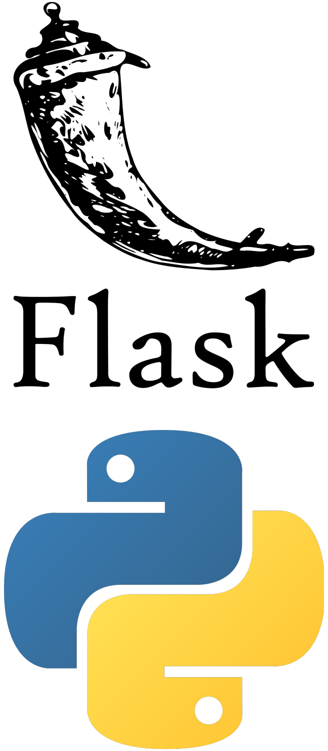
All purchase reasons

View and sort for any attribute, seeing the difference compared to competitors

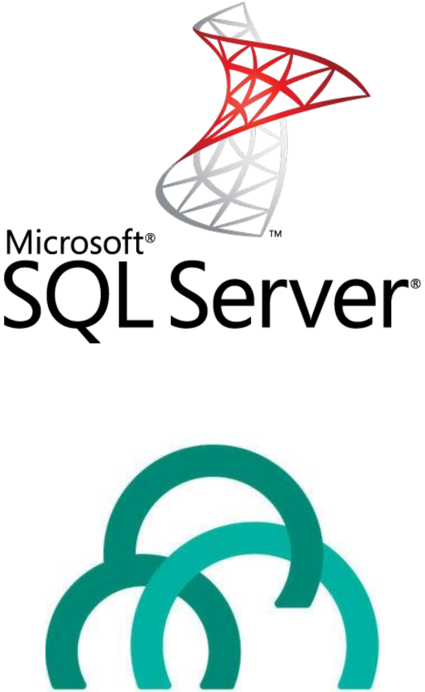
ATTRIBUTE	STALLION: PUR. REASON %	COMPETITORS: PUR. REASON %	DIFFERENCE
Towing	13.8%	11.2%	2.6%
Durability Vehicle Life	11.5%	14.6%	-3.1%
Space & Size	10.1%	11.0%	-0.9%
Cargo & Flexibility	6.9%	9.3%	-3.3%
Finacing Costs	5.2%	0.8%	4.4%
Exterior Design	4.1%	2.5%	1.6%
Ride	2.8%	1.1%	1.7%
Schedule Maintenance	2.0%	4.8%	-2.8%



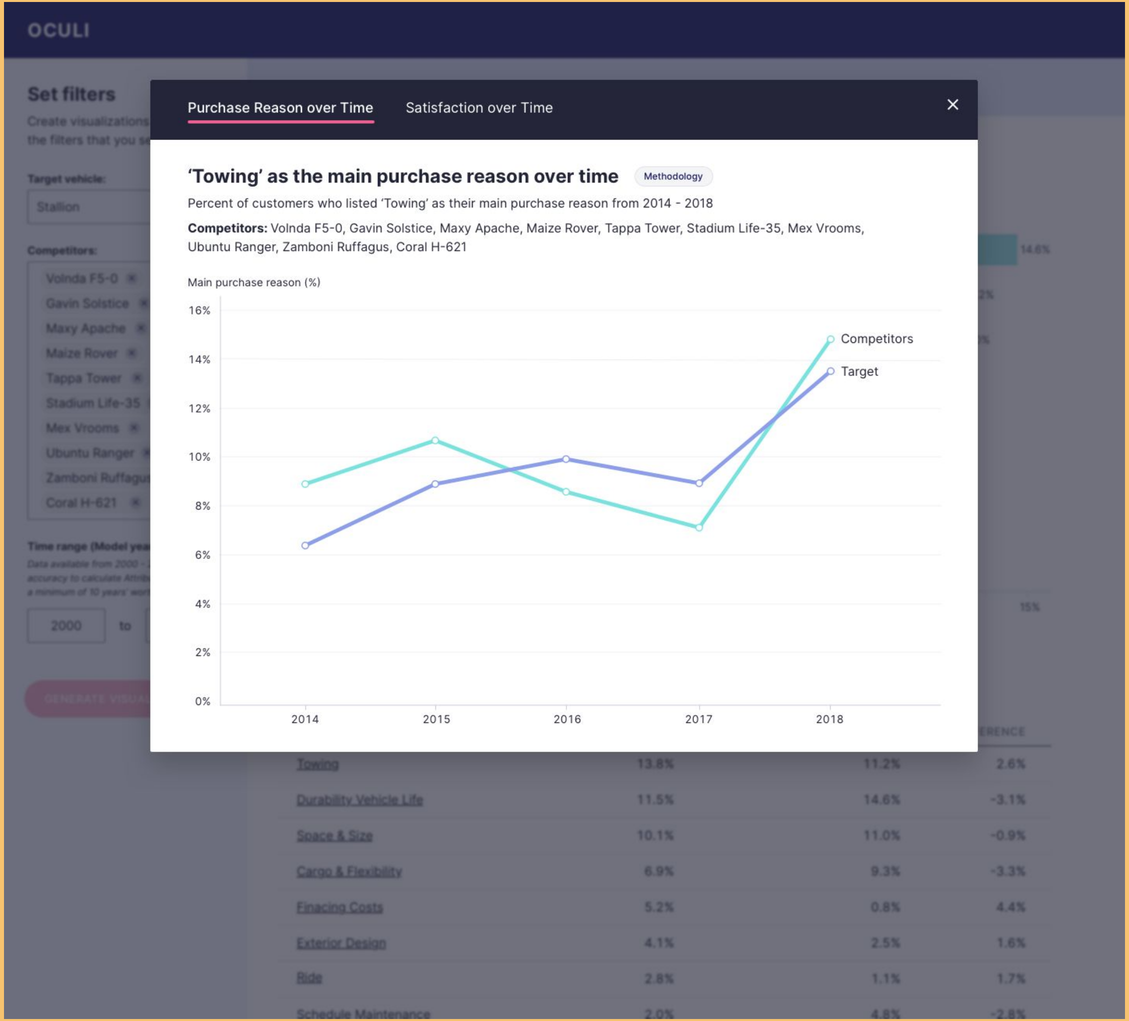
Front-End



Backend-End



Infrastructure



OCULI

Set filters

Create visualizations
the filters that you se

Target vehicle:

Stallion

Competitors:

Volnda F5-0
Gavin Solstice
Maxy Apache
Maize Rover
Tappa Tower
Stadium Life-35
Mex Vrooms
Ubuntu Ranger
Zamboni Ruffagus
Coral H-621

Time range (Model yea

Data available from 2000 -
accuracy to calculate Attrib
a minimum of 10 years' wor

2000 to

GENERATE VISUA

Purchase Reason over Time | Satisfaction over Time | ✕

'Towing' as the main purchase reason over time

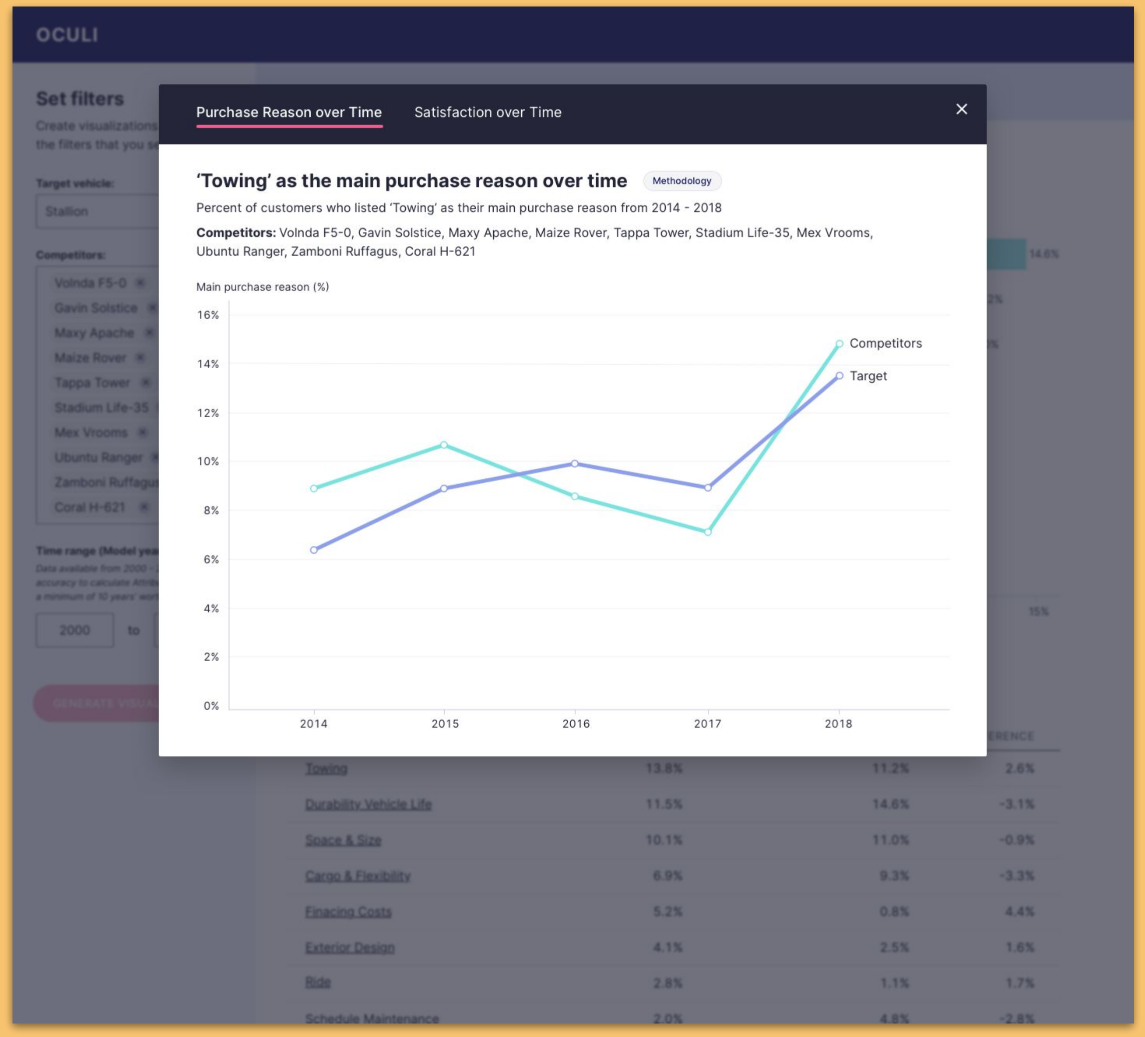
Methodology

Percent of customers who listed 'Towing' as their main purchase reason from 2014 - 2018

Competitors: Volnda F5-0, Gavin Solstice, Maxy Apache, Maize Rover, Tappa Tower, Stadium Life-35, Mex Vrooms, Ubuntu Ranger, Zamboni Ruffagus, Coral H-621

			ERENCE
Towing	13.8%	11.2%	2.6%
Durability Vehicle Life	11.5%	14.6%	-3.1%
Space & Size	10.1%	11.0%	-0.9%
Cargo & Flexibility	6.9%	9.3%	-3.3%
Finacing Costs	5.2%	0.8%	4.4%
Exterior Design	4.1%	2.5%	1.6%
Ride	2.8%	1.1%	1.7%
Schedule Maintenance	2.0%	4.8%	-2.8%

1. Identify old code can be reused and what new code needs to be written



OCULI

Set filters

Create visualizations
the filters that you se

Target vehicle:

Stallion

Competitors:

Volnda F5-0 ×
Gavin Solstice ×
Maxy Apache ×
Maize Rover ×
Tappa Tower ×
Stadium Life-35
Mex Vrooms ×
Ubuntu Ranger ×
Zamboni Ruffagus
Coral H-621 ×

Time range (Model yea

2000 to

Purchase Reason over Time | Satisfaction over Time ×

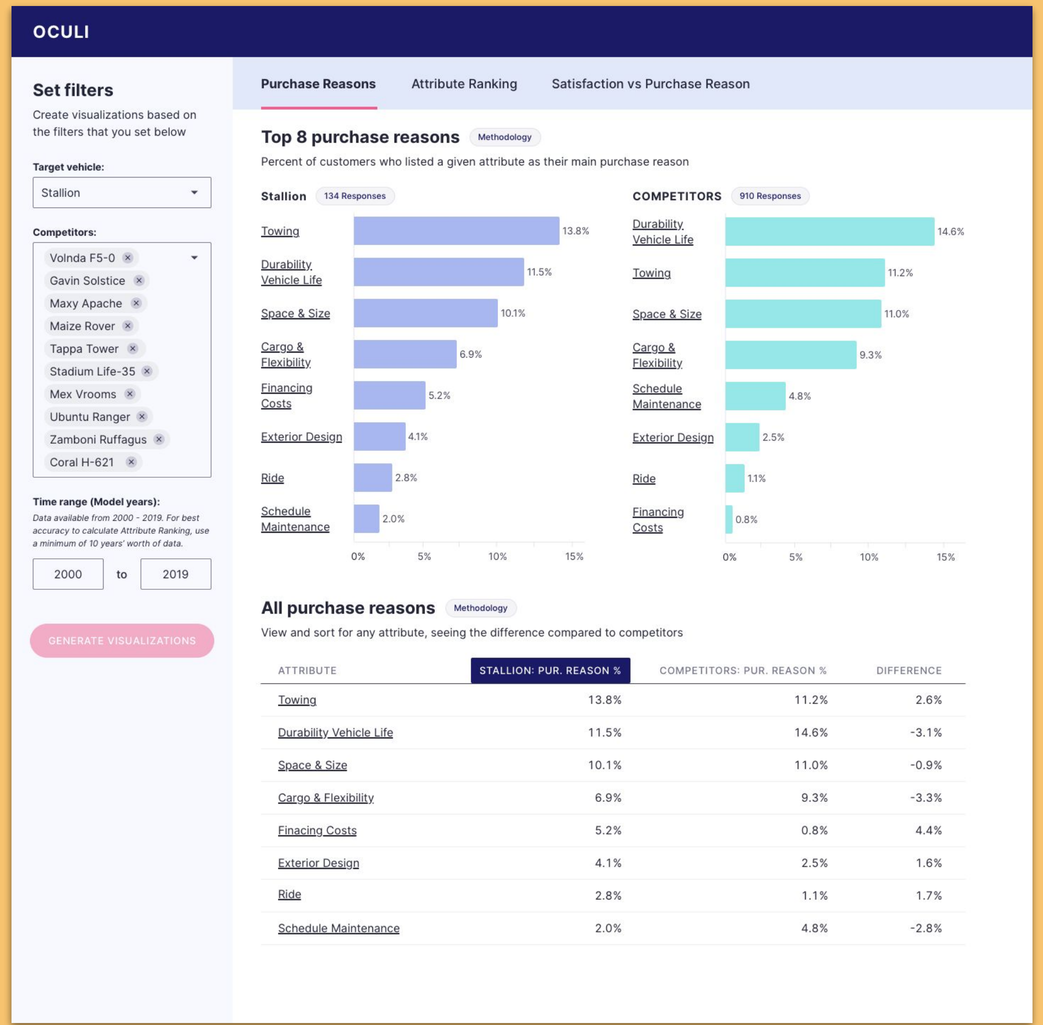
‘Towing’ as the main purchase reason over time

Methodology

Percent of customers who listed ‘Towing’ as their main purchase reason from 2014 - 2018

Competitors: Volnda F5-0, Gavin Solstice, Maxy Apache, Maize Rover, Tappa Tower, Stadium Life-35, Mex Vrooms, Ubuntu Ranger, Zamboni Ruffagus, Coral H-621

Towing	13.8%	11.2%	2.6%
Durability Vehicle Life	11.5%	14.6%	-3.1%
Space & Size	10.1%	11.0%	-0.9%
Cargo & Flexibility	6.9%	9.3%	-3.3%
Finacing Costs	5.2%	0.8%	4.4%
Exterior Design	4.1%	2.5%	1.6%
Ride	2.8%	1.1%	1.7%
Schedule Maintenance	2.0%	4.8%	-2.8%



Set filters

Create visualizations based on the filters that you set below

Target vehicle: Stallion

Competitors:

- Volnda F5-0 ×
- Gavin Solstice ×
- Maxy Apache ×
- Maize Rover ×
- Tappa Tower ×
- Stadium Life-35 ×
- Mex Vrooms ×
- Ubuntu Ranger ×
- Zamboni Ruffagus ×
- Coral H-621 ×

Time range (Model years):

Data available from 2000 - 2019. For best accuracy to calculate Attribute Ranking, use a minimum of 10 years' worth of data.

2000 to 2019

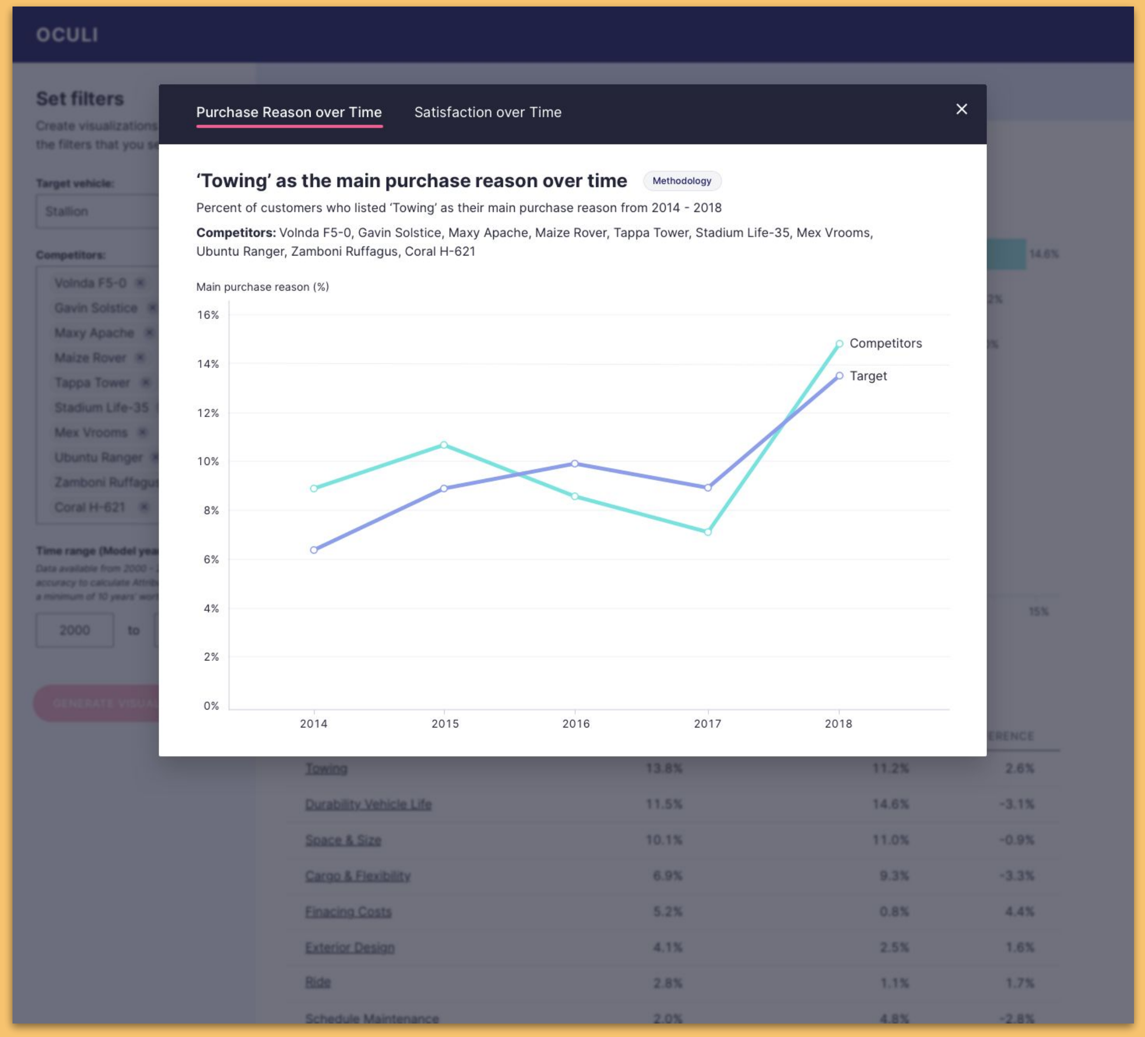
Top 8 purchase reasons

Percent of customers who listed a given attribute as their main purchase reason

All purchase reasons

View and sort for any attribute, seeing the difference compared to competitors

ATTRIBUTE	STALLION: PUR. REASON %	COMPETITORS: PUR. REASON %	DIFFERENCE
Towing	13.8%	11.2%	2.6%
Durability Vehicle Life	11.5%	14.6%	-3.1%
Space & Size	10.1%	11.0%	-0.9%
Cargo & Flexibility	6.9%	9.3%	-3.3%
Finacing Costs	5.2%	0.8%	4.4%
Exterior Design	4.1%	2.5%	1.6%
Ride	2.8%	1.1%	1.7%
Schedule Maintenance	2.0%	4.8%	-2.8%



Set filters

Create visualizations
the filters that you s

Target vehicle:

Stallion

Competitors:

- Volnda F5-0
- Gavin Solstice
- Maxy Apache
- Maize Rover
- Tappa Tower
- Stadium Life-35
- Mex Vrooms
- Ubuntu Ranger
- Zamboni Ruffagus
- Coral H-621

Time range (Model yea

Data available from 2000 -
accuracy to calculate Attrib
a minimum of 10 years' wor

2000 to

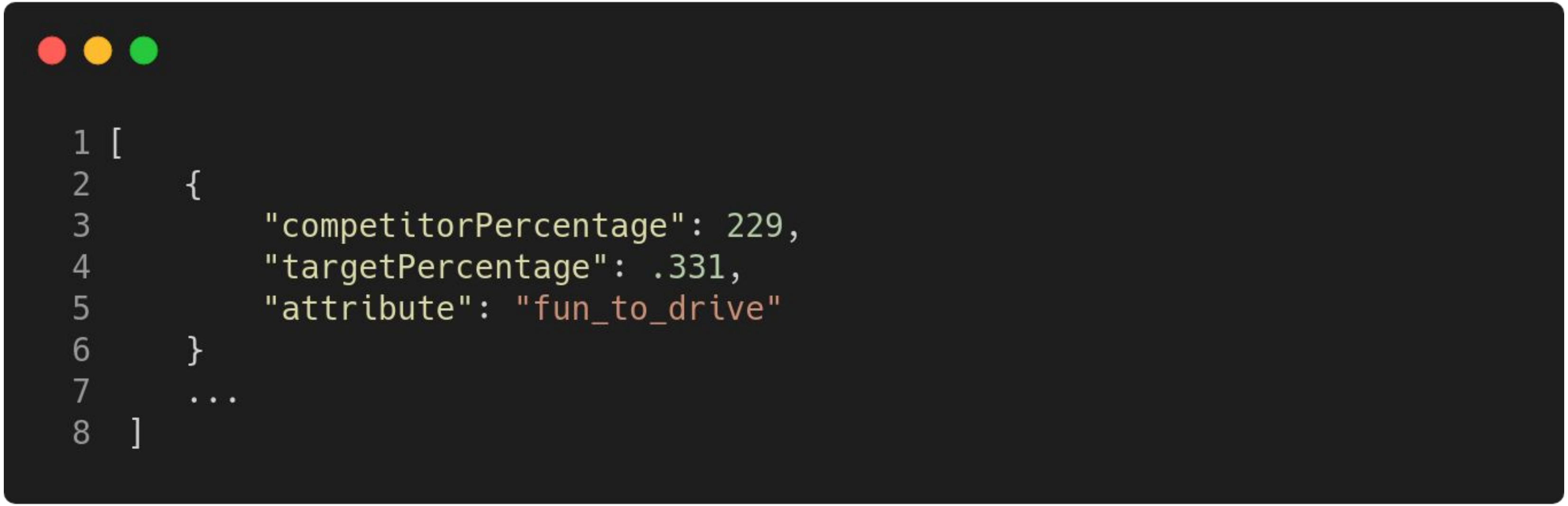
‘Towing’ as the main purchase reason over time

Percent of customers who listed ‘Towing’ as their main purchase reason from 2014 - 2018

Competitors: Volnda F5-0, Gavin Solstice, Maxy Apache, Maize Rover, Tappa Tower, Stadium Life-35, Mex Vrooms, Ubuntu Ranger, Zamboni Ruffagus, Coral H-621

			ERENCE
Towing	13.8%	11.2%	2.6%
Durability Vehicle Life	11.5%	14.6%	-3.1%
Space & Size	10.1%	11.0%	-0.9%
Cargo & Flexibility	6.9%	9.3%	-3.3%
Finacing Costs	5.2%	0.8%	4.4%
Exterior Design	4.1%	2.5%	1.6%
Ride	2.8%	1.1%	1.7%
Schedule Maintenance	2.0%	4.8%	-2.8%

```
/api/purchasereason/ranking?startYear=2010
                            &endYear=2019
                            &target=STALLION
                            &competitors=UBUNTU_RANGER
                            &competitors=ZAMBONI_RUFFAGUS
```



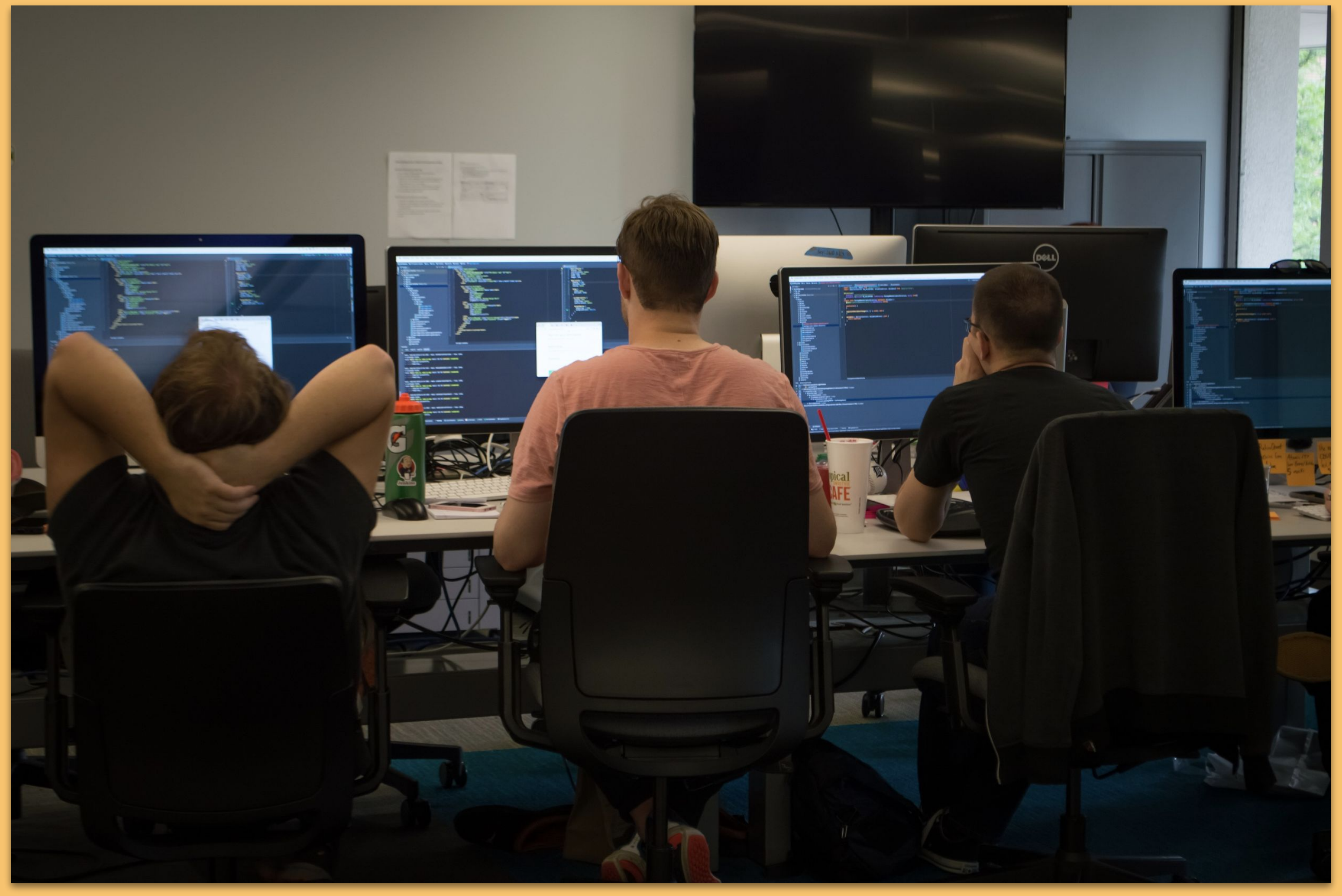
```
[
    {
        "competitorPercentage": 229,
        "targetPercentage": .331,
        "attribute": "fun_to_drive"
    }
    ...
 ]
```

2. Think about how the new code should work

HOME
Who We Are
What We Do
Contact
Sign Up
Sign In
Dashboard
PLANS
Budget
Select Plans
LEARN
PLAN
MOBILE

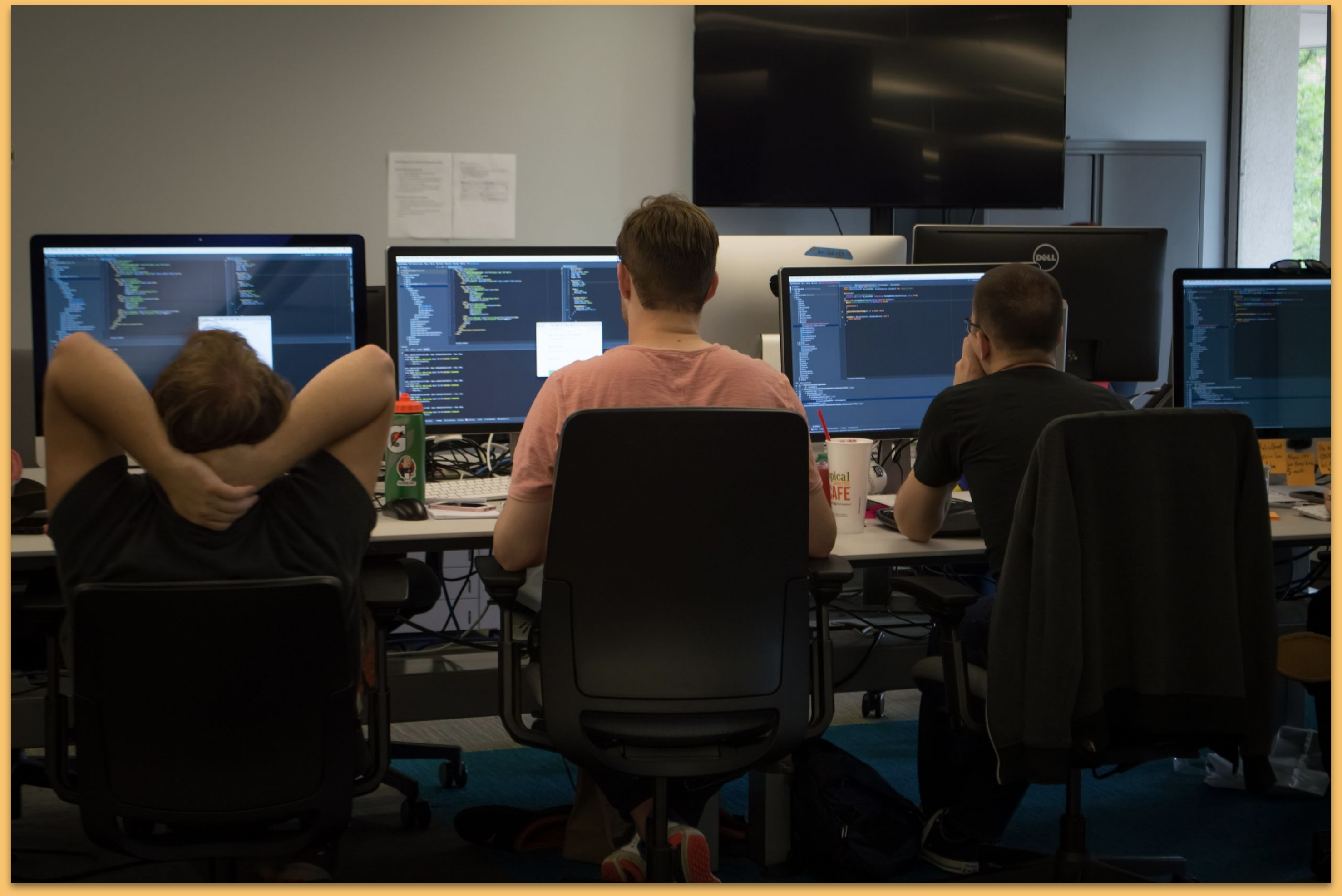
3. Do the implementation

Paired Programming


DELL
pical
AFE

+15%	-15%
Cost of Development	Bugs Present

The Costs and Benefits of Pair Programming. Alistair Cockburn and Laurie Williams


DELL
pical
AFE

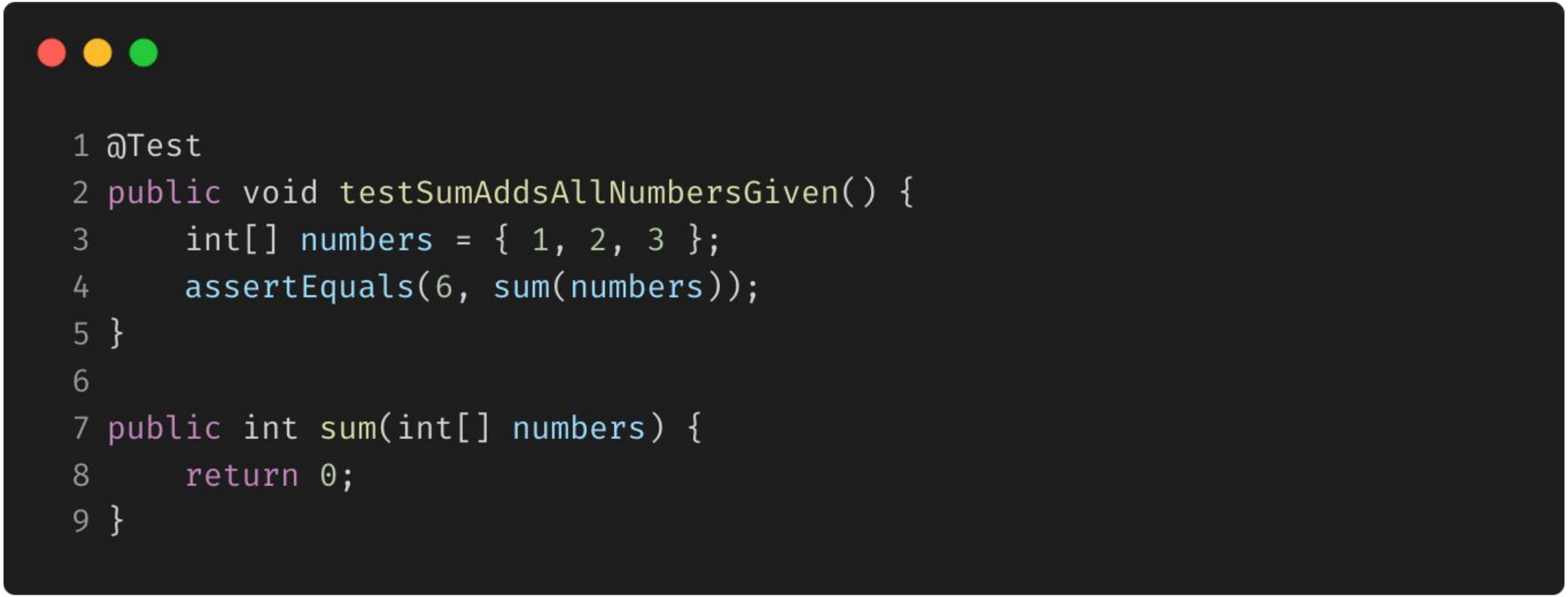
Test-Driven Development

Red

Green

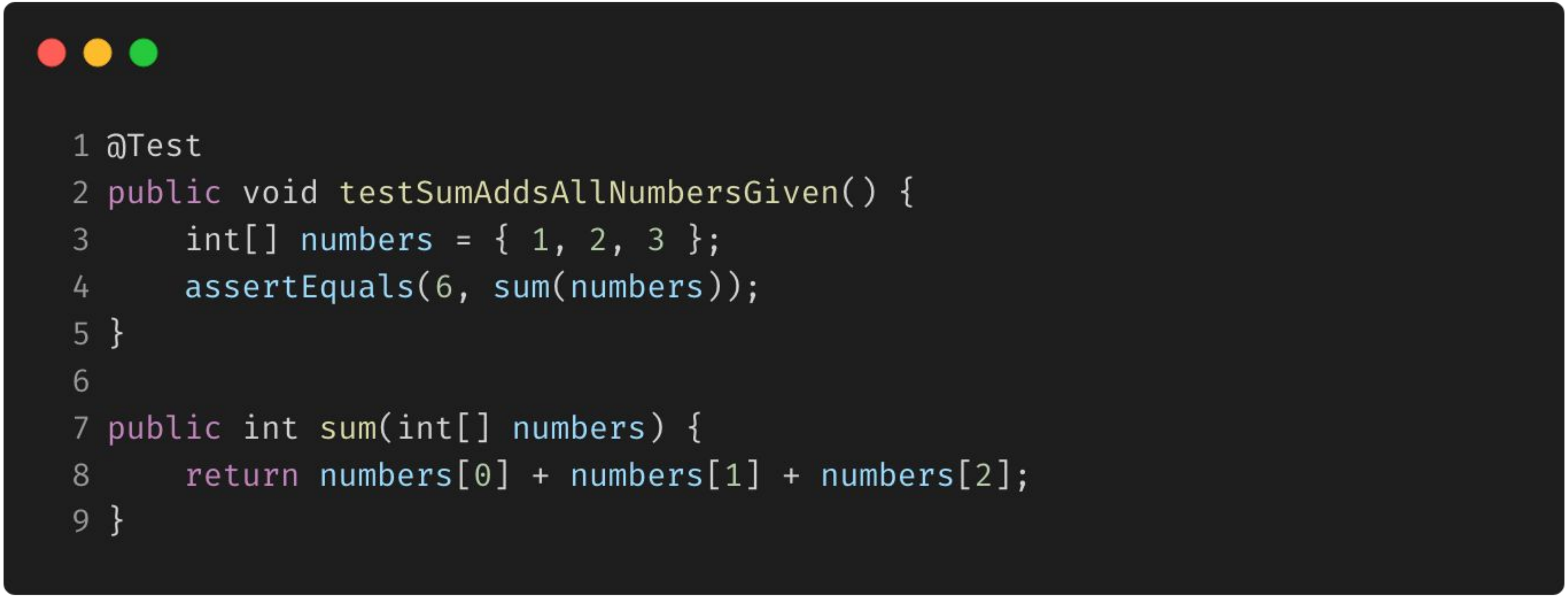
Refactor

Red



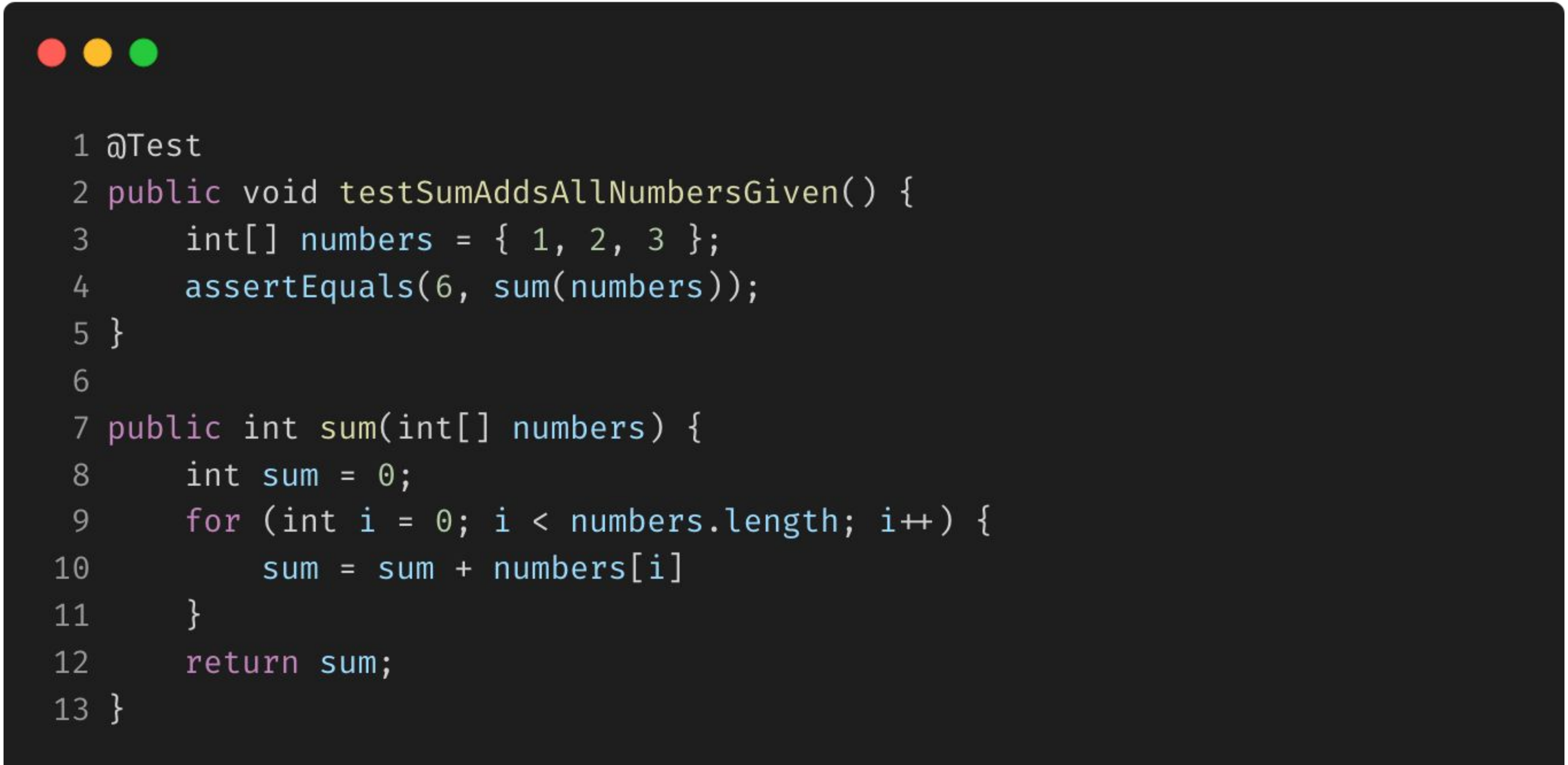
```
@Test
public void testSumAddsAllNumbersGiven() {
    int[] numbers = { 1, 2, 3 };
    assertEquals(6, sum(numbers));
}

public int sum(int[] numbers) {
    return 0;
}
```

Green

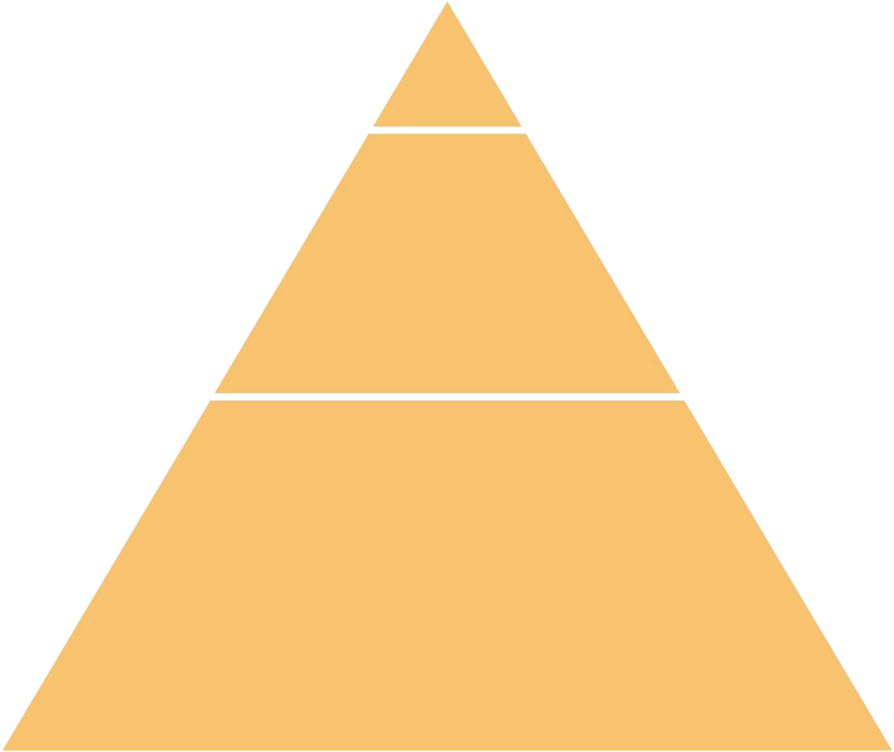


```
@Test
public void testSumAddsAllNumbersGiven() {
    int[] numbers = { 1, 2, 3 };
    assertEquals(6, sum(numbers));
}

public int sum(int[] numbers) {
    return numbers[0] + numbers[1] + numbers[2];
}
```

Refactor



```java
@Test
public void testSumAddsAllNumbersGiven() {
    int[] numbers = { 1, 2, 3 };
    assertEquals(6, sum(numbers));
}

public int sum(int[] numbers) {
    int sum = 0;
    for (int i = 0; i < numbers.length; i++) {
        sum = sum + numbers[i]
    }
    return sum;
}
```



End-to-end tests

Integration tests

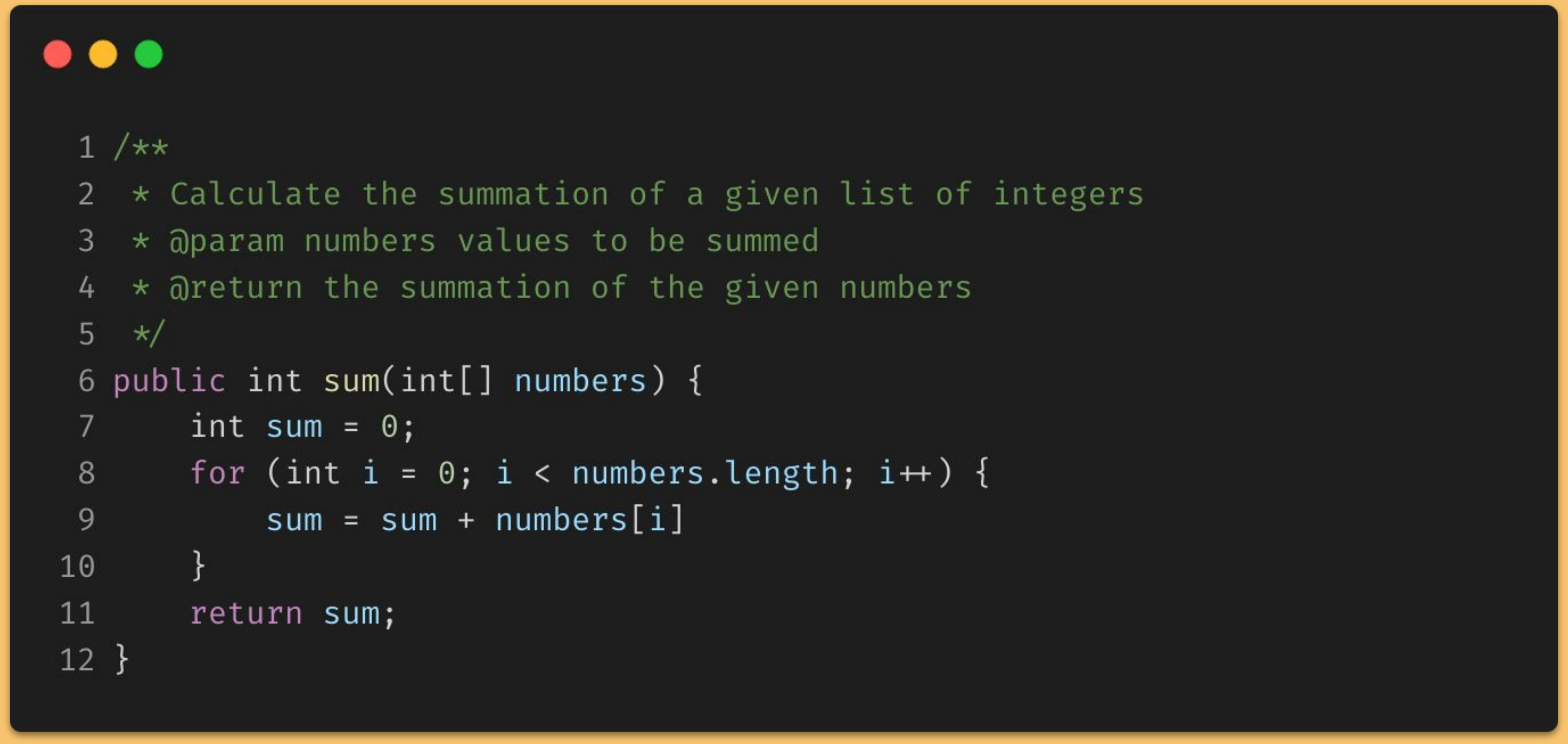
Unit tests

Reduced Complexity

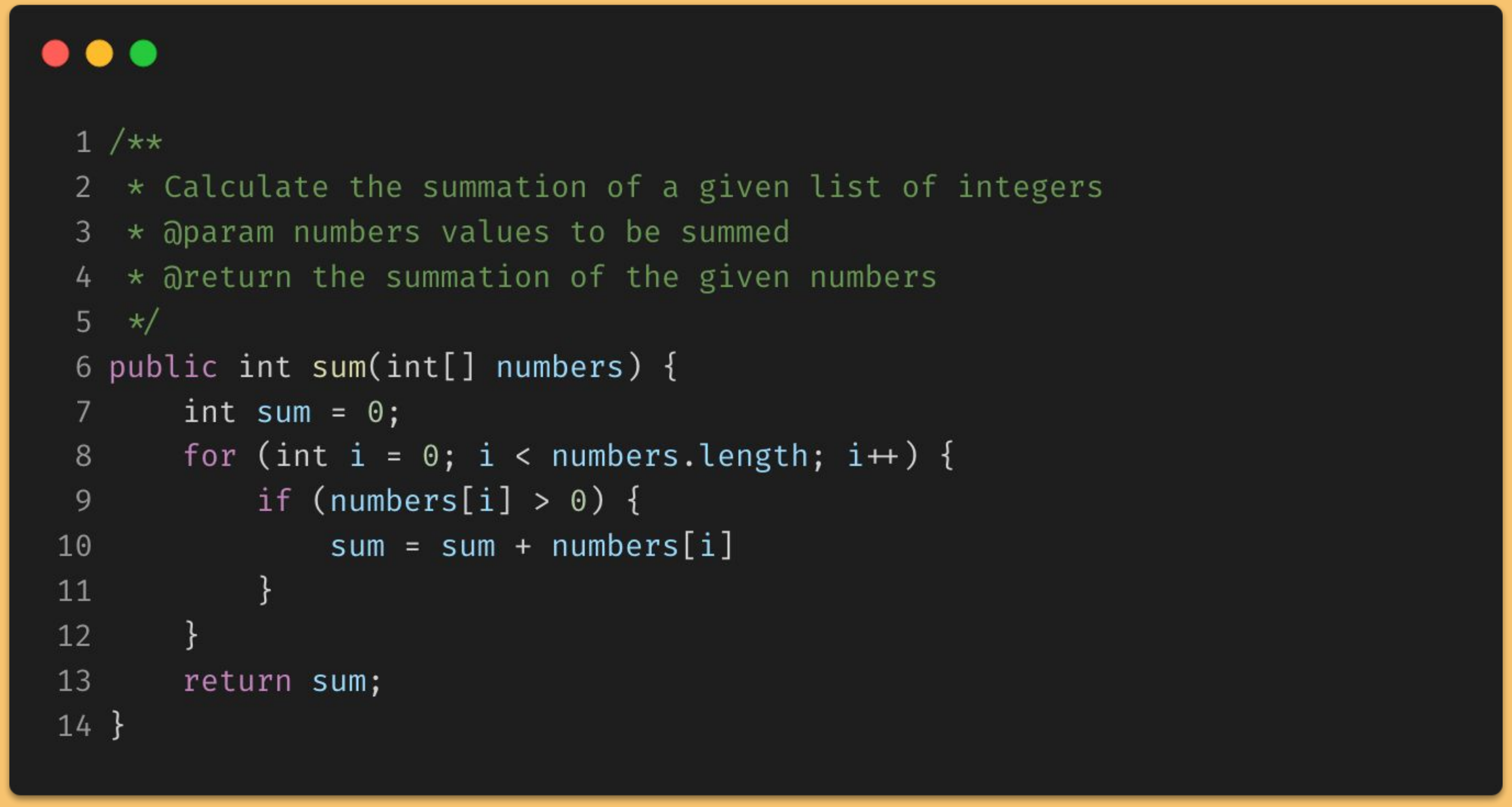
Explicit Documentation

Regression Safety

“If it’s too hard to think of how to test an interface, your interface is too complex”

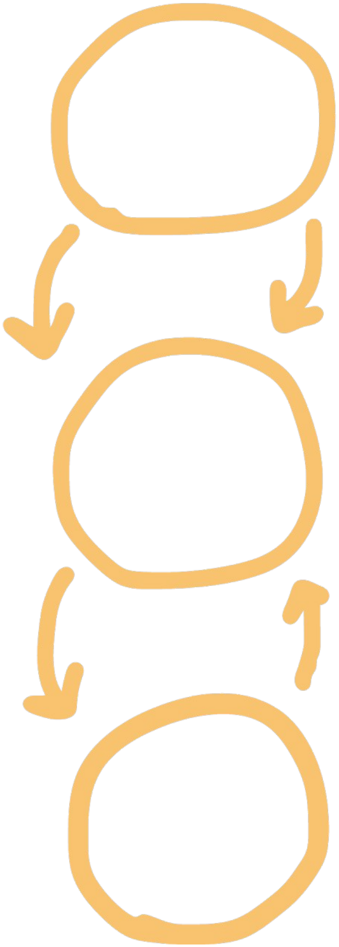


```java
/**
 * Calculate the summation of a given list of integers
 * @param numbers values to be summed
 * @return the summation of the given numbers
 */
public int sum(int[] numbers) {
    int sum = 0;
    for (int i = 0; i < numbers.length; i++) {
        sum = sum + numbers[i]
    }
    return sum;
}
```



```java
/**
 * Calculate the summation of a given list of integers
 * @param numbers values to be summed
 * @return the summation of the given numbers
 */
public int sum(int[] numbers) {
    int sum = 0;
    for (int i = 0; i < numbers.length; i++) {
        if (numbers[i] > 0) {
            sum = sum + numbers[i]
        }
    }
    return sum;
}
```

Committing Code

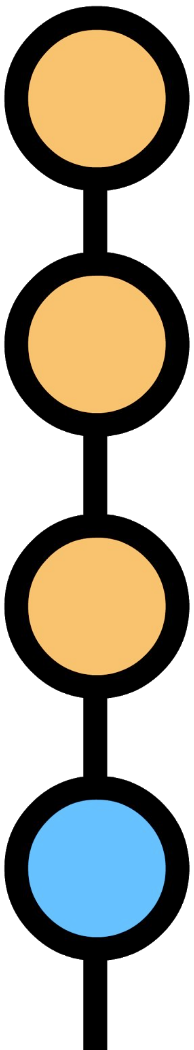


Remote Repository

Local Repository

Working Changes

It’s a best practice to commit early and often

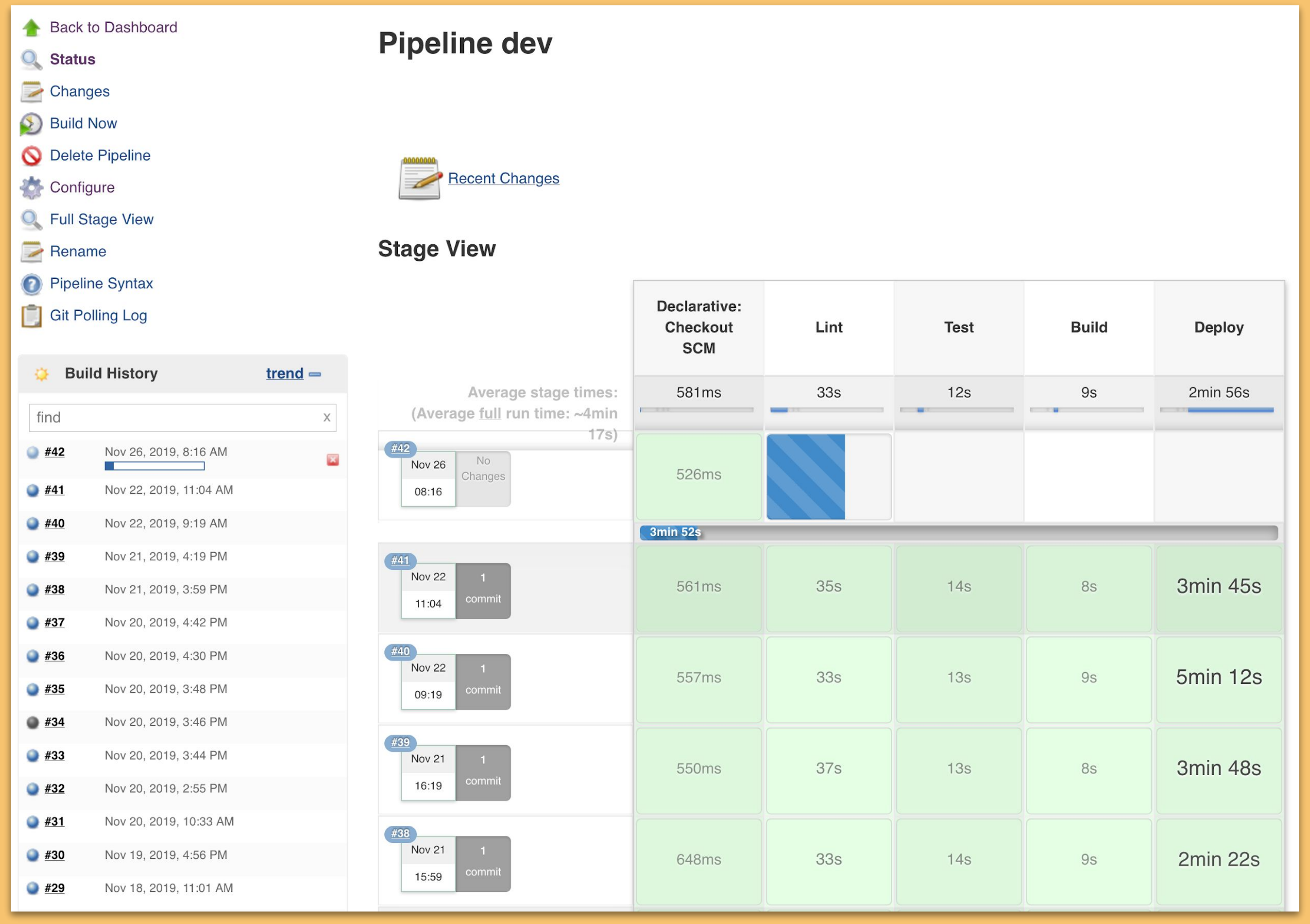


16 commits

---

8 days

Continuous Integration & Continuous Deployment



- Back to Dashboard
- Status
- Changes
- Build Now
- Delete Pipeline
- Configure
- Full Stage View
- Rename
- Pipeline Syntax
- Git Polling Log

Pipeline dev

Recent Changes

Stage View

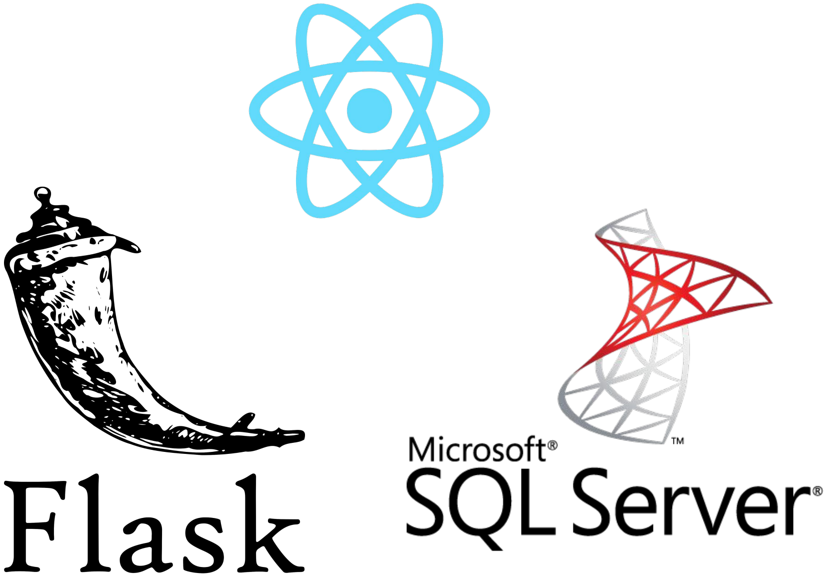
	Declarative: Checkout SCM	Lint	Test	Build	Deploy
Average stage times: (Average full run time: ~4min 17s)	581ms	33s	12s	9s	2min 56s
#42 Nov 26 08:16 No Changes	526ms				
3min 52s					
#41 Nov 22 11:04 1 commit	561ms	35s	14s	8s	3min 45s
#40 Nov 22 09:19 1 commit	557ms	33s	13s	9s	5min 12s
#39 Nov 21 16:19 1 commit	550ms	37s	13s	8s	3min 48s
#38 Nov 21 15:59 1 commit	648ms	33s	14s	9s	2min 22s

Build History trend

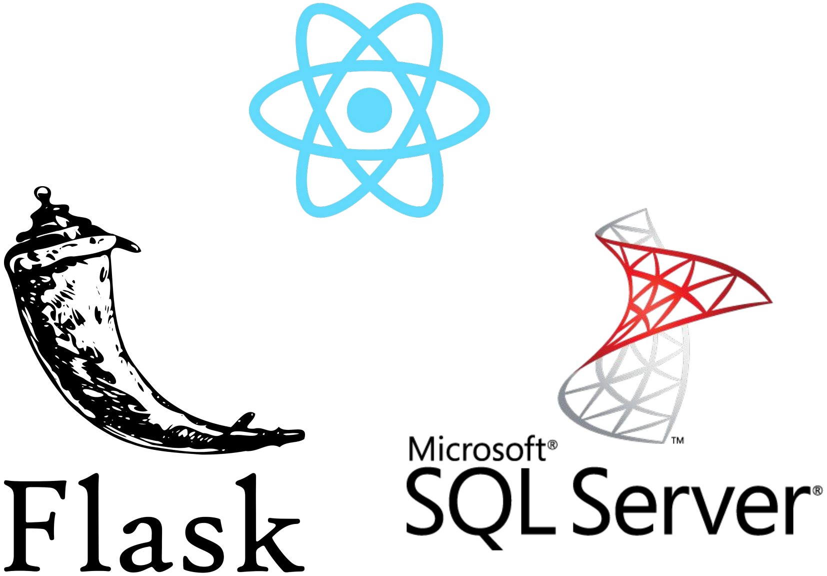
find x

#42	Nov 26, 2019, 8:16 AM
#41	Nov 22, 2019, 11:04 AM
#40	Nov 22, 2019, 9:19 AM
#39	Nov 21, 2019, 4:19 PM
#38	Nov 21, 2019, 3:59 PM
#37	Nov 20, 2019, 4:42 PM
#36	Nov 20, 2019, 4:30 PM
#35	Nov 20, 2019, 3:48 PM
#34	Nov 20, 2019, 3:46 PM
#33	Nov 20, 2019, 3:44 PM
#32	Nov 20, 2019, 2:55 PM
#31	Nov 20, 2019, 10:33 AM
#30	Nov 19, 2019, 4:56 PM
#29	Nov 18, 2019, 11:01 AM

“As the general stress level rises, manual builds tend to be done less often and less well, resulting in more errors and more stress.”



Development



Production

4. Go through review

Design Review

✚

Gets the OK from Justin

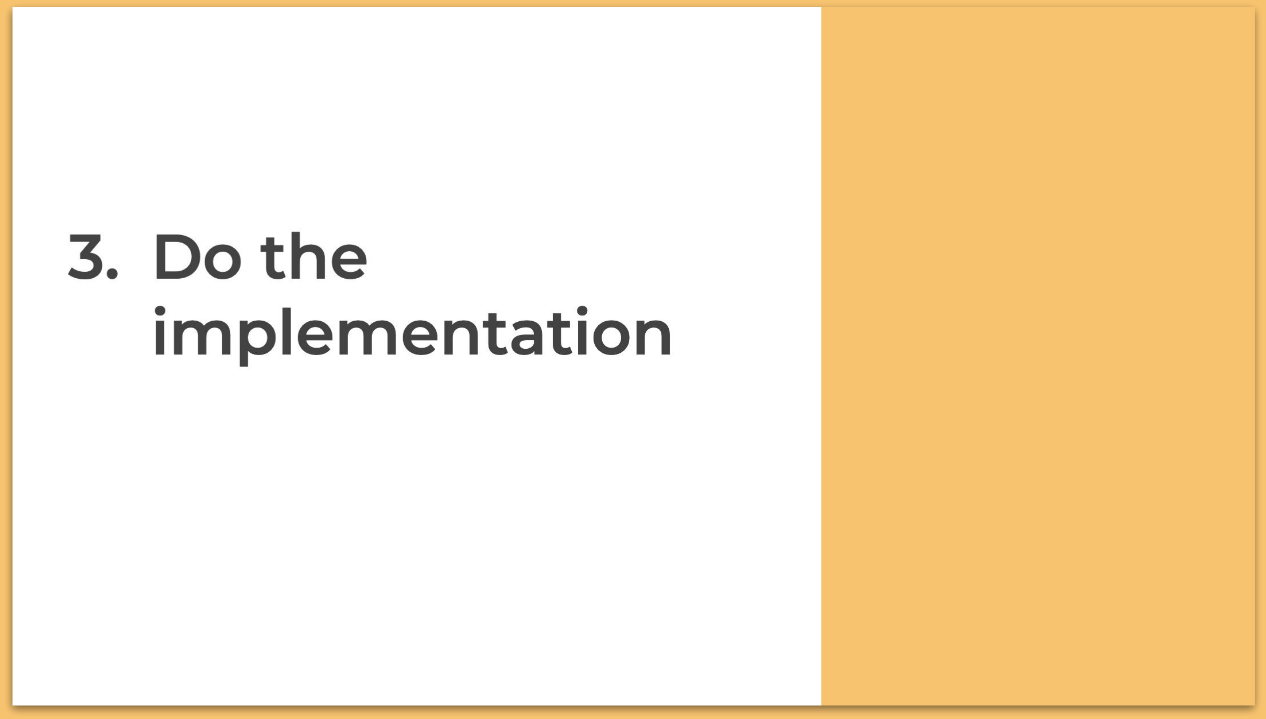
✖

Justin Finds a Problem

We didn’t
understand the story

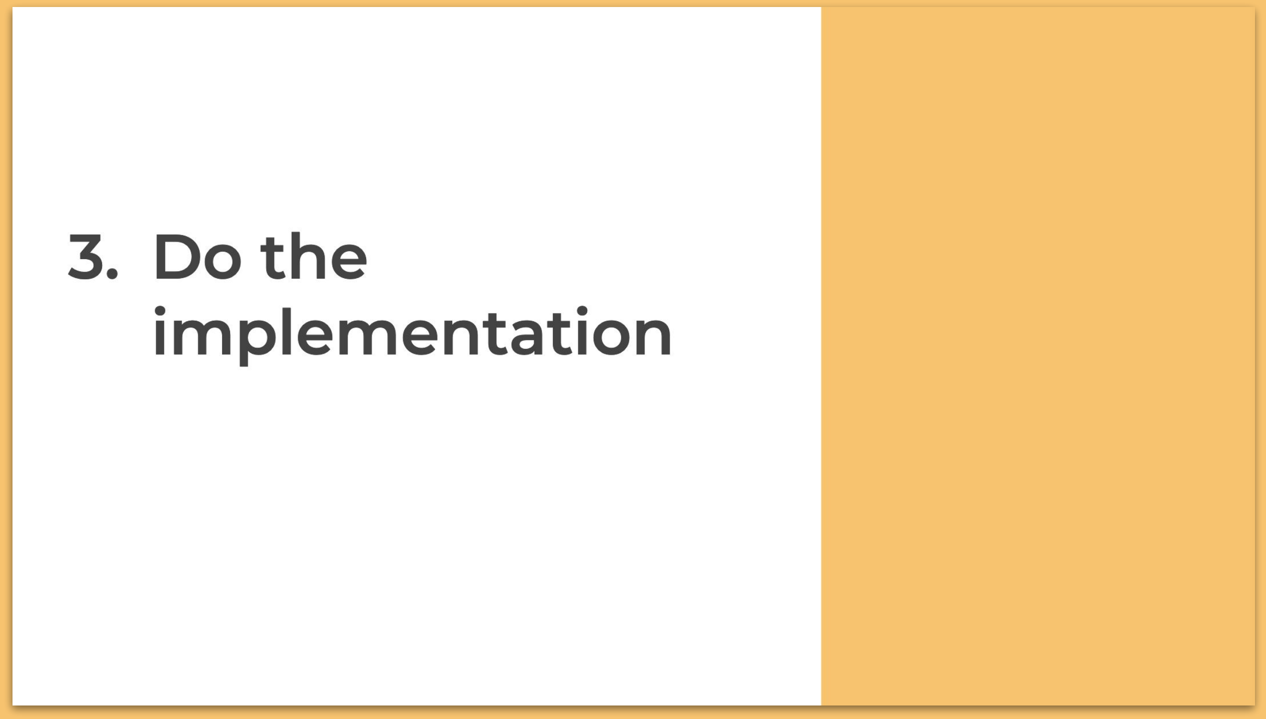
We can’t do the story
as described

1. It’s trivial to fix, so we go back and do it



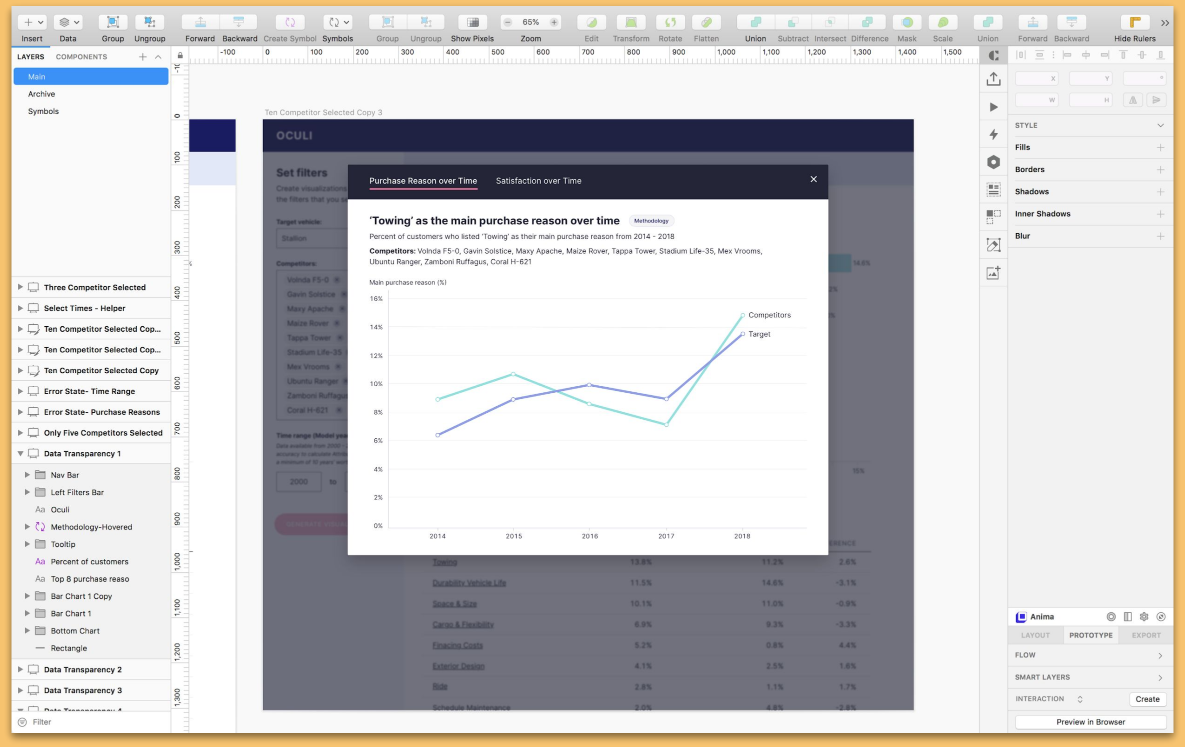
3. Do the implementation

2. It’s a non-negotiable requirement to the story, so we go back and implement it



3. Do the implementation

3. It wasn’t what we wanted to begin with, so we change the design



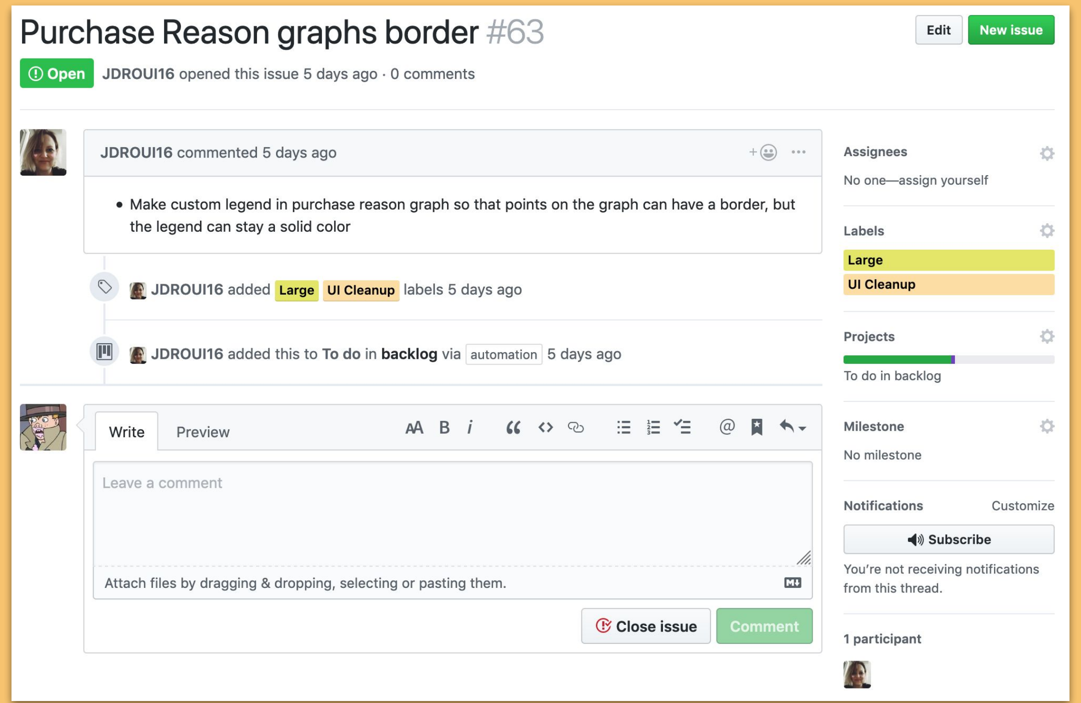
Purchase Reason over Time | Satisfaction over Time

‘Towing’ as the main purchase reason over time

Percent of customers who listed ‘Towing’ as their main purchase reason from 2014 - 2018

Competitors: Volnda F5-0, Gavin Solstice, Maxy Apache, Maize Rover, Tappa Tower, Stadium Life-35, Mex Vrooms, Ubuntu Ranger, Zamboni Ruffagus, Coral H-621

4. It’s not required for the feature to deliver value, so we defer it to a new story



Purchase Reason graphs border #63

JDROUI16 opened this issue 5 days ago · 0 comments

JDROUI16 commented 5 days ago

- Make custom legend in purchase reason graph so that points on the graph can have a border, but the legend can stay a solid color

JDROUI16 added Large UI Cleanup labels 5 days ago

JDROUI16 added this to To do in backlog via automation 5 days ago

Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting or pasting them.
Close issue
Comment

Assignees

No one—assign yourself

Labels

Large

UI Cleanup

Projects

To do in backlog

Milestone

No milestone

Notifications Customize

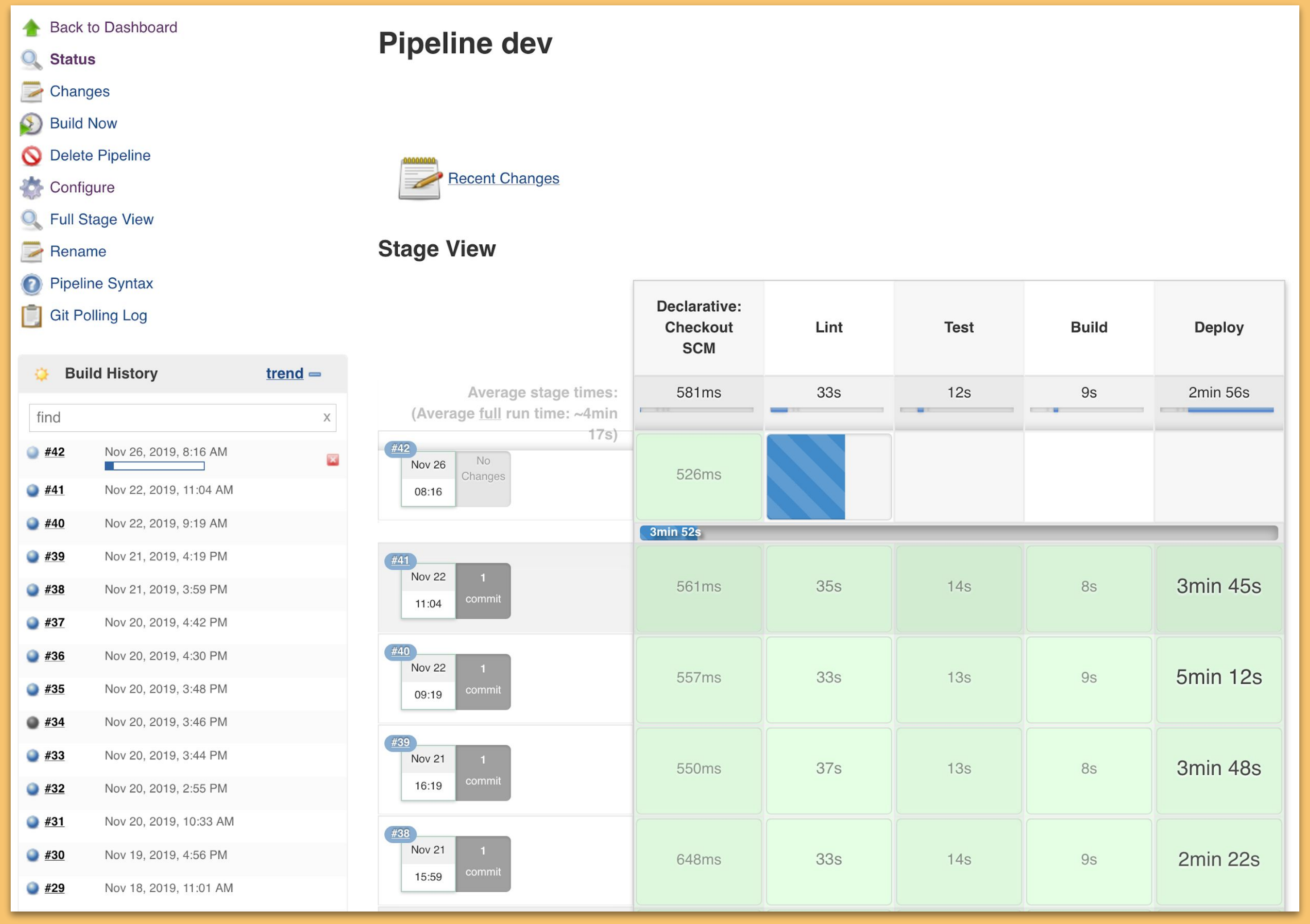
You’re not receiving notifications from this thread.

1 participant

PM Review

PM Review with Jenai is exactly like Design Review, but with more roasting

5. Take the Rocket Ship to Prod



Pipeline dev

Build History trend

find x

#42	Nov 26, 2019, 8:16 AM
#41	Nov 22, 2019, 11:04 AM
#40	Nov 22, 2019, 9:19 AM
#39	Nov 21, 2019, 4:19 PM
#38	Nov 21, 2019, 3:59 PM
#37	Nov 20, 2019, 4:42 PM
#36	Nov 20, 2019, 4:30 PM
#35	Nov 20, 2019, 3:48 PM
#34	Nov 20, 2019, 3:46 PM
#33	Nov 20, 2019, 3:44 PM
#32	Nov 20, 2019, 2:55 PM
#31	Nov 20, 2019, 10:33 AM
#30	Nov 19, 2019, 4:56 PM
#29	Nov 18, 2019, 11:01 AM

Stage View

	Declarative: Checkout SCM	Lint	Test	Build	Deploy
Average stage times: (Average full run time: ~4min 17s)	581ms	33s	12s	9s	2min 56s
#42 Nov 26 08:16 No Changes	526ms				
	3min 52s				
#41 Nov 22 11:04 1 commit	561ms	35s	14s	8s	3min 45s
#40 Nov 22 09:19 1 commit	557ms	33s	13s	9s	5min 12s
#39 Nov 21 16:19 1 commit	550ms	37s	13s	8s	3min 48s
#38 Nov 21 15:59 1 commit	648ms	33s	14s	9s	2min 22s

Questions?

Thank you for attending. Goodbye.