Cognitive Science based Software Engineering

This presentation uses OpenDyslexic font

What is Cognitive Science?

"Cognitive science is the study of the human mind and brain, focusing on how the mind represents and manipulates knowledge and how mental representations and processes are realized in the brain. Conceiving of the mind as an abstract computing device instantiated in the brain, cognitive scientists endeavor to understand the mental computations underlying cognitive functioning and how these computations are implemented by neural tissue." [1]

[1] Johns Hopkins University

What is Cognitive Science?

Cognitive science explores how brains process information

Memory

- Short Term Memory (STM)
- Working Memory (WM)
- Long Term Memory (LTM)

Short Term Memory

Capacity:

- 7 (±2) items [1]
- 4 items (±1) [2]
- number depends on their category

Whatever the number is, it is **limited**

[1] The magical number seven, plus or minus two, by G. A. Miller
[2] The magical number 4 in short term memory, by N. Cowan

Short Term Memory

Duration

- 15-30 seconds [1]

Association and rehearsal make it become Long Term Memory

[1] The control process of short-term memory by R.C. Atkinsons and R.M. Shiffrin

The hippocampus are the parts of the brain in charge for consolidating information from **STM** to **LTM**

Study of **H. M.** [1]

[1] Permanent Present Tense, by S. Corkin

Short Term Memory

Try to memorize this sequence of numbers

9 5 8 6 5 5 3 6

Short Term Memory

Try to memorize this sequence of numbers

9 5 8 6 5 5 3 6

Can work better in patterns

9 5 8 6 5 5 3 6

Working Memory

- very close to STM
- it's STM applied to processing

Working Memory

- very close to STM
- it's STM applied to processing

16 x 9 = ?

Working Memory

- very close to STM
- it's STM applied to processing

16 x 9 = 144

23 x 68 = ?

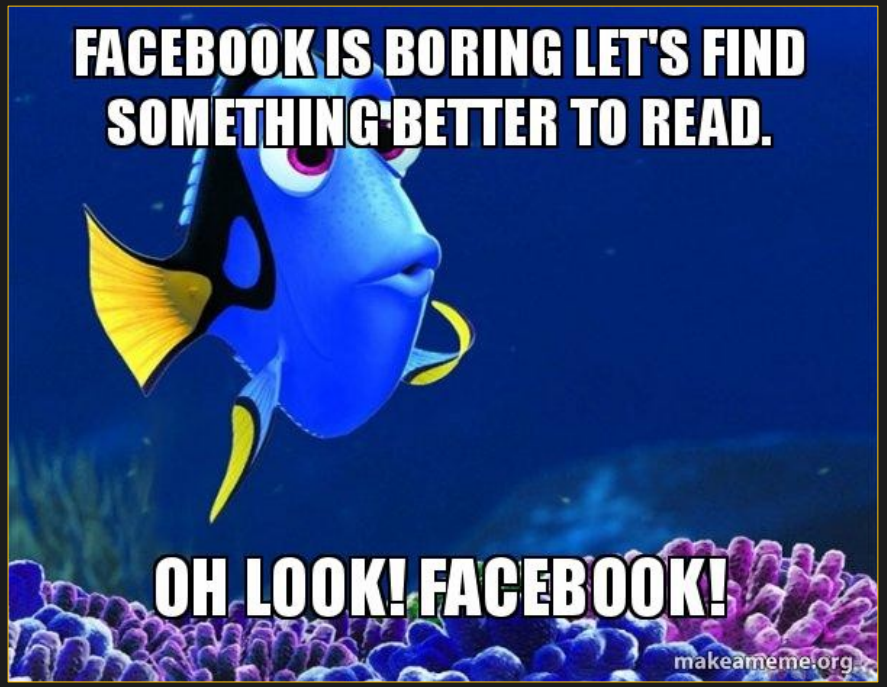
Working Memory

- very close to STM
- it's STM applied to processing

16 x 9 = 144

23 x 68 = 1,564

Long Term Memory



Long Term Memory

Procedural / Implicit

Declarative / Explicit

Episodic

Semantic

Long Term Memory

Procedural / Implicit

- Touch Typing
- Vim users: ESC : q !
- Close a tab: Ctrl + w

Declarative / Explicit

Episodic

- Coding interview
- When you broke production

Semantic

- ArrayList class
- Binary Search
- Singleton

Chunking

Study on chess showed that masters think in patterns [1]

Chunking allows to overcome some limits of STM / WM using LTM

[1] Perception in chess, by W. Chase and H. Simon

Do you remember the number?

Do you remember the number?

9 5 8 6 5 5 3 6

Do you remember the number?

9 5 8 6 5 5 3 6

9.58” 65,536

100 meters WR Max val of a word: 2^{16}

Cognitive load [1]

Cognitive load measures the amount of mental processing required for performing a task

The higher the cognitive load, the higher the fatigue for your brain

[1] Cognitive Load During Problem Solving: Effects on Learning, by J. Sweller

Cognitive load types

- intrinsic
- extraneous
- germane

Intrinsic Cognitive Load

```
int abs(int a) {

    if (a < 0) {
        return -a;
    }

    return a;
}
```

Intrinsic Cognitive Load

```
int abs(int a) {

    if (a < 0) {
        return -a;
    }

    return a;
}
```

```
int gcd(int a, int b) {

    if (b == 0) {
        return a;
    }

    return gcd(b, a % b);
}
```

Intrinsic Cognitive Load

```
int abs(int a) {

    if (a < 0) {
        return -a;
    }

    return a;
}
```

Low

```
int gcd(int a, int b) {

    if (b == 0) {
        return a;
    }

    return gcd(b, a % b);
}
```

High

Extraneous Cognitive Load

```
int binSearch(int[]arr, int x) {int
   l=0; int r=arr.length -1; while
      (l<=r) {int mid=l+(r-l) /2;
         if(arr[mid]==x){return
           mid;}if(arr[mid]>x)
            {r=mid-1;}else{
               l= mid+1;}
                return
                 -1 ;
                  }
```

Extraneous Cognitive Load

```
int binSearch(int[]arr, int x) {int
l=0; int r=arr.length -1; while
(l<=r) {int mid=l+(r-l) /2;
if(arr[mid]==x){return
mid;}if(arr[mid]>x)
{r=mid-1;}else{
l= mid+1;}
return
-1 ;
}
```

```
int binSearch(int[] arr, int x){
    int l = 0;
    int r = arr.length - 1;

    while (l <= r) {
        int mid = l + (r - l) / 2;
        if (arr[mid] == x) {
            return mid;
        }

        if (arr[mid] > x) {
            r = mid - 1;
        } else {
            l = mid + 1;
        }

    return -1;
}
```

Extraneous Cognitive Load

```
int binSearch(int[]arr, int x) {int
   l=0; int r=arr.length -1; while
      (l<=r) {int mid=l+(r-l) /2;
         if(arr[mid]==x){return
           mid;}if(arr[mid]>x)
            {r=mid-1;}else{
               l= mid+1;}
                return
                 -1 ;
                  }
```

High

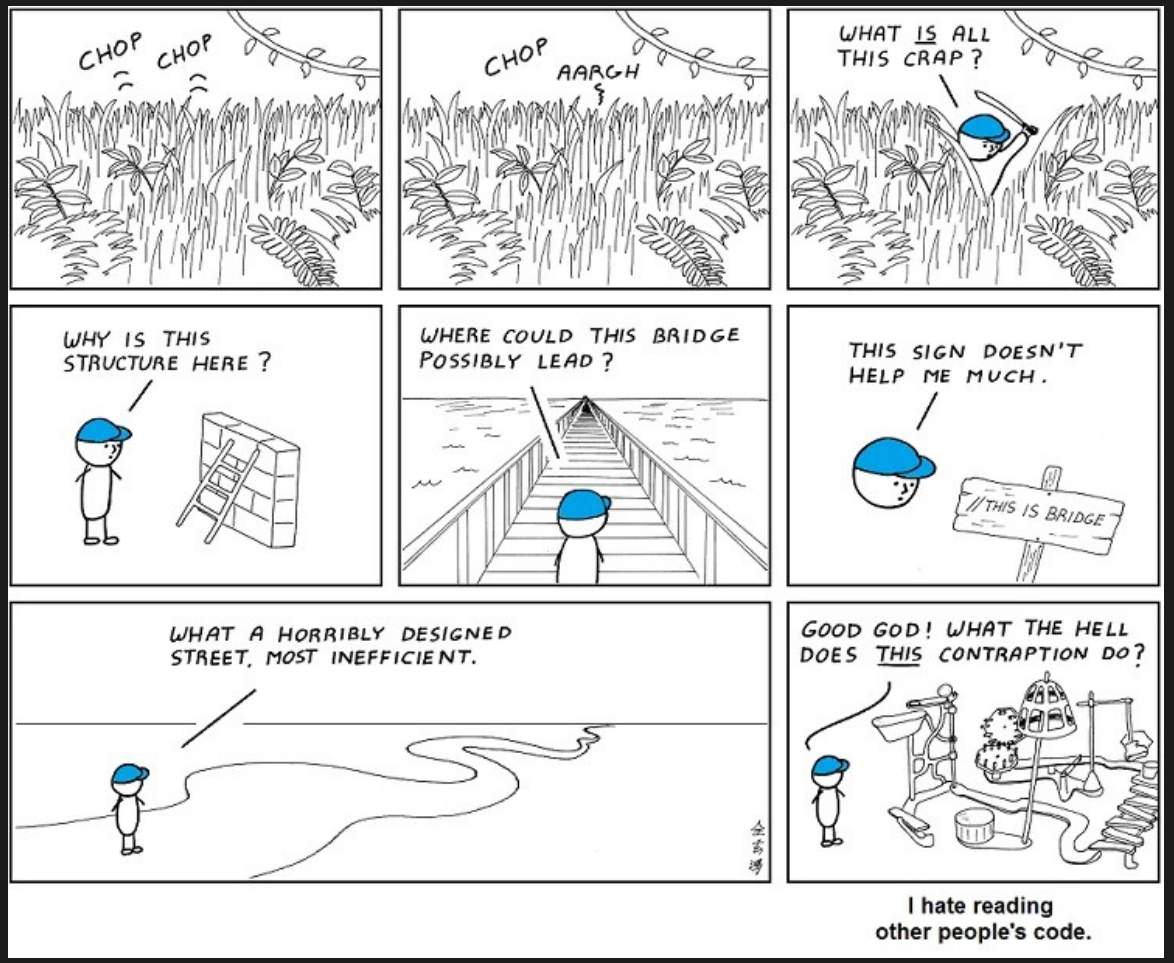
```
int binSearch(int[] arr, int x){
    int l = 0;
    int r = arr.length - 1;

    while (l <= r) {
        int mid = l + (r - l) / 2;
        if (arr[mid] == x) {
            return mid;
        }

        if (arr[mid] > x) {
            r = mid - 1;
        } else {
            l = mid + 1;
        }

    return -1;
}
```

Low

Germane Cognitive Load

The effort we make transitioning
from STM to LTM

Reading Code



I hate reading other people's code.

Reading Code

We spend 58% of our time reading code

rather than writing it [1]

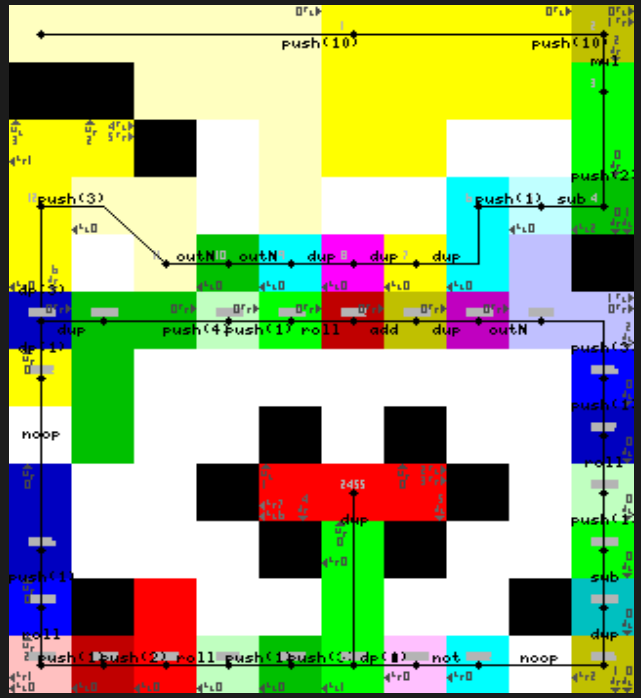
[1] Measuring Program Comprehension by Xin Xia et al., 2017

What does this program computes?

What does this program computes?

It computes a Fibonacci number

What does this program computes?



Hue Change	Lightness Change		
	No change	1 darker	2 darker
No change	N/A	Push	Pop
1 step	Add	Subtract	Multiply
2 steps	Divide	Modulo	Not
3 steps	Greater	Pointer	Switch
4 steps	Duplicate	Roll	Input num
5 steps	Input char	Output num	Output char

Light red (#FFC0C0)	Light yellow (#FFFFC0)	Light green (#C0FFC0)	Light cyan (#C0FFFF)	Light blue (#C0C0FF)	Light magenta (#FFC0FF)
Red (#FF0000)	Yellow (#FFFF00)	Green (#00FF00)	Cyan (#00FFFF)	Blue (#0000FF)	Magenta (#FF00FF)
Dark red (#C00000)	Dark yellow (#C0C000)	Dark green (#00C000)	Dark cyan (#00C0C0)	Dark blue (#0000C0)	Dark magenta (#C000C0)

Reading Code

This time for real

Reading Code

```java
public int lengthOfLongestSubstring(String s) {
    Set<Character> set = new HashSet<>();
    int n = s.length();
    int maxLength = 0;
    int left = 0;

    for (int right = 0; right < n; right++) {
        while (set.contains(s.charAt(right))) {
            set.remove(s.charAt(left));
            left++;
        }
        set.add(s.charAt(right));
        maxLength = Math.max(maxLength, right - left + 1);
    }
    return maxLength;
}
```

Reading Code

```python
def length_of_longest_substring(s: str) -> int:
    char_set = set()
    max_length = 0
    left = 0

    for right in range(len(s)):
        while s[right] in char_set:
            char_set.remove(s[left])
            left += 1
        char_set.add(s[right])
        max_length = max(max_length, right - left + 1)

    return max_length
```

Roles of a variable [1]

Fixed value	its value is not changed in run-time (after initialization)
Stepper	goes through a succession of values in some systematic way
Most-recent holder	its value is not changed in run-time after initialization
Most-wanted holder	is the "best" or most-wanted value out of the values gone through so far
Gatherer	accumulates all the values gone through so far
Follower	gets the old value of another known variable as its new value
One-way flag	is a Boolean variable which once changed cannot get its original value anymore
Temporary	its value is always needed only for a very short period
Organizer	is used for reorganizing its elements after initialization
Container	is a data structure where elements can be added and removed
Walker	traverses a data structure

[1] Roles of Variables in Experts' Programming Knowledge, by J. Sajaniemi and R. Navarro Prieto

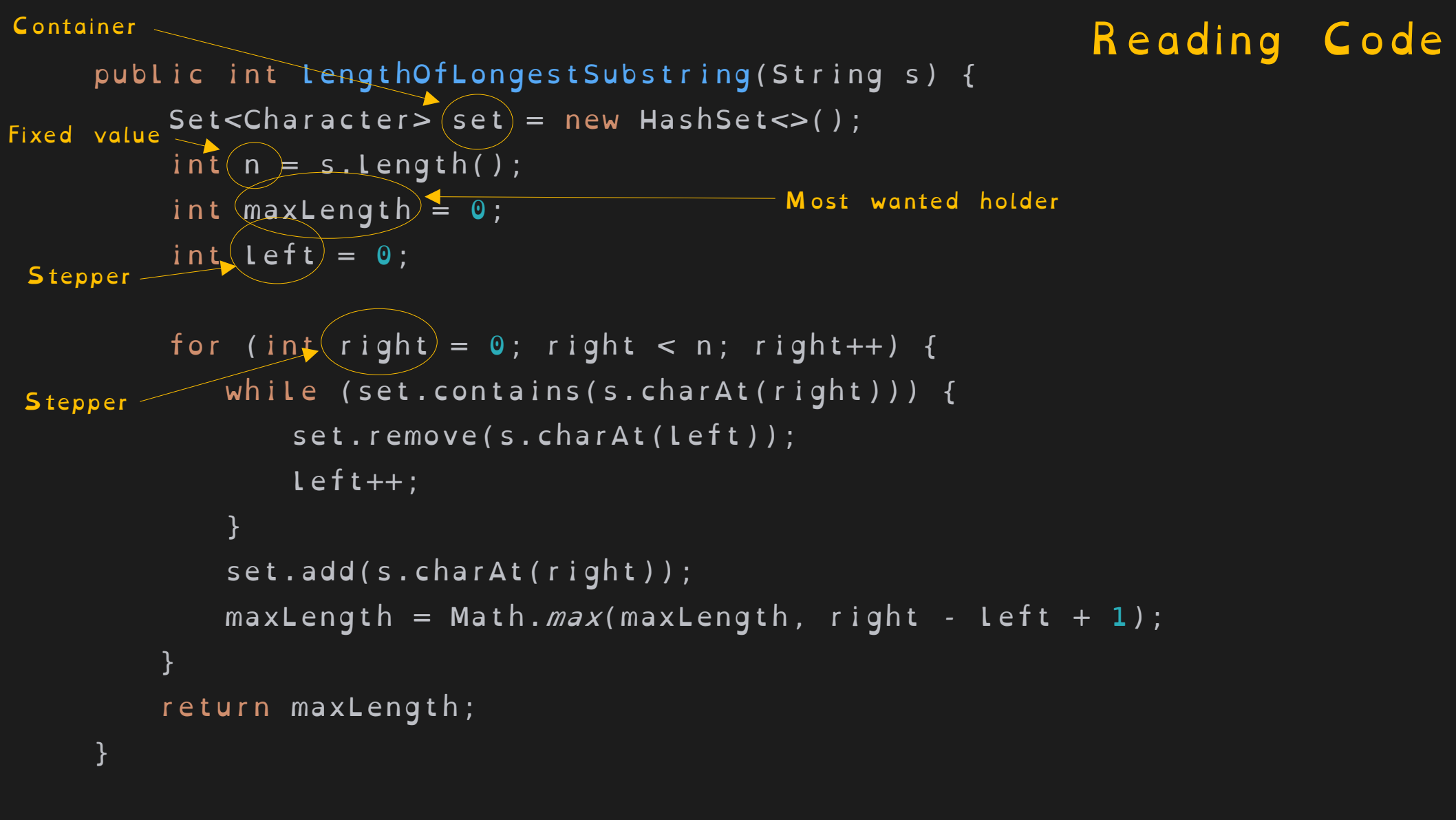
Reading Code

```java
public int lengthOfLongestSubstring(String s) {
    Set<Character> set = new HashSet<>();
    int n = s.length();
    int maxLength = 0;
    int left = 0;

    for (int right = 0; right < n; right++) {
        while (set.contains(s.charAt(right))) {
            set.remove(s.charAt(left));
            left++;
        }
        set.add(s.charAt(right));
        maxLength = Math.max(maxLength, right - left + 1);
    }
    return maxLength;
}
```

Reading Code

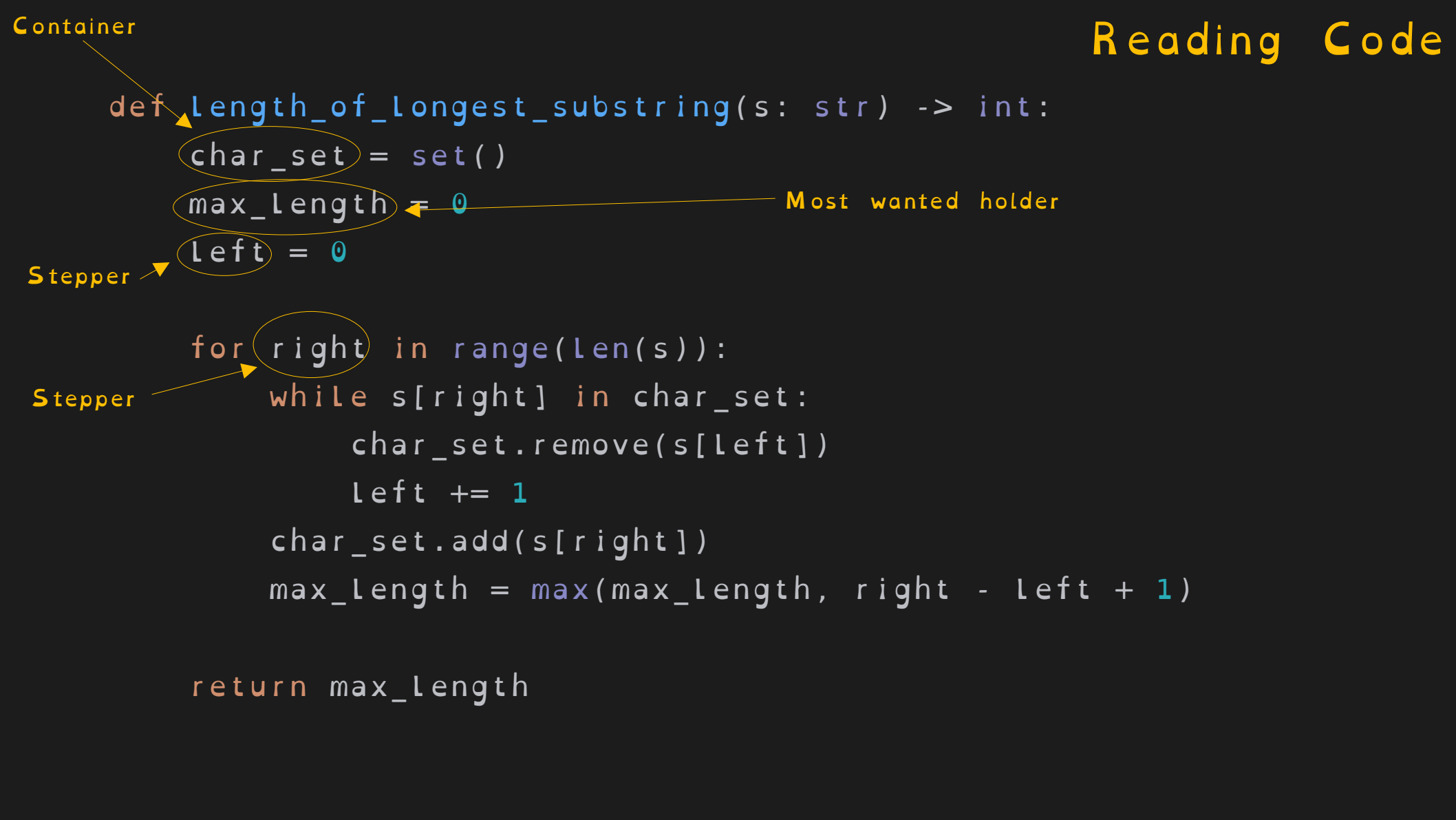
```python
def length_of_longest_substring(s: str) -> int:
    char_set = set()
    max_length = 0
    left = 0

    for right in range(len(s)):
        while s[right] in char_set:
            char_set.remove(s[left])
            left += 1
        char_set.add(s[right])
        max_length = max(max_length, right - left + 1)

    return max_length
```

Reading Code




```java
public int lengthOfLongestSubstring(String s) {
    Set<Character> set = new HashSet<>();
    int n = s.length();
    int maxLength = 0;
    int left = 0;

    for (int right = 0; right < n; right++) {
        while (set.contains(s.charAt(right))) {
            set.remove(s.charAt(left));
            left++;
        }
        set.add(s.charAt(right));
        maxLength = Math.max(maxLength, right - left + 1);
    }
    return maxLength;
}
```

Reading Code




```python
def length_of_longest_substring(s: str) -> int:
    char_set = set()
    max_length = 0
    left = 0

    for right in range(len(s)):
        while s[right] in char_set:
            char_set.remove(s[left])
            left += 1
        char_set.add(s[right])
        max_length = max(max_length, right - left + 1)

    return max_length
```

Reading Code

We create a mental model of the algorithm and reason about it:

- edge cases
- expected behavior
- memory consumption

Reading Code

```
public class MovingAverageCalculator {
    private int count = 0;
    private double average = 0.0;
    private double sum = 0.0;

    public void add(double number) {
        sum += number;
        count++;
        average = sum / count;
    }

    public double getAverage() {
        return average;
    }
}
```

Reading Code

```
public class MovingAverageCalculator {
    private int count = 0;
    private double average = 0.0;
    private double sum = 0.0;

    public void add(double number) {
        sum += number;
        count++;
        average = sum / count;
    }

    public double getAverage() {
        return average;
    }
}
```

Now read this code knowing that an instance of this class is accessed by *multiple* threads

Mental Models

- Algorithms
- System architectures
- Business Domains

Mental Models

Tennis Rules [1]

Scoring:

- Points: 0, 15, 30, 40, Game
- Win a game: Score 4 points and be 2 points ahead
- Win a set: Win at least 6 games, leading by 2
- Win the match: Win 3 out of 5 sets

Given an array of scored points [p1, p2, p2, p1, ...]

find the winner

[1] Claude 3.5 Sonnet

Mental Models

Diverse teams can help in approaching problems from different angles

Range by David Epstein, 2020

Reading Code

Seniority leverages chunking

Writing code

A lot of good practices lower cognitive load

- split long methods/functions
- single responsibility principle
- early optimization

Writing code

Side effects

```
public void setStatus(Status status) {
    this.status = status;
    if (status == Status.SHIPPED) {
        sendEmailToCustomer();
    }
}
```

You use chunking, but it's wrong

Naming Conventions

Name	Description	Example of flawed identifier(s)
Capitalisation Anomaly	Identifiers should be appropriately capitalised.	HTMLEditorKit.pagecounter
Consecutive Underscores	Consecutive underscores should not be used in identifier names.	foo__bar
Dictionary Words	Identifier names should be composed of words in dictionary and abbreviations, and acronyms, that are more commonly used than the unabbreviated form.	strlen
Excessive Words	Identifier names should be composed of no more than four words or abbreviations.	floatToRawIntBits()
Enumeration Identifier Declaration Order	Unless there are compelling and obvious reasons otherwise, enumeration constants should be declared in alphabetical order.	enum Card {ACE, EIGHT, FIVE,FOUR, JACK, KING ...}
External Underscores	Identifiers should not have either leading or trailing underscores.	_foo_
Identifier Encoding	Type information should not be encoded in identifier names using Hungarian notation or similar.	Int iCount;
Long Identifier Name	Long identifier names should be avoided where possible.	getPolicyQualifiersRejectedNaming
Convention Anomaly	Identifiers should not consist of non-standard mixes of upper and lower case characters.	FOO_bar
Number of Words	Identifiers should be composed of between two andfour words.	ArrayOutOfBoundsException.name
Numeric Identifier Name	Identifiers should not be composed of numeric words or nums.	FORTY_TWO
Short Identifier Name	Identifiers should not consist of fewer than eight characters, with the exception of: c, d, e, g,i, in, j, k, m, n, o, out,t, x, y, z	name

Source: Relating identifier naming flaws and code quality by S. Butler et al, 2009

Comments

Comments are another tool for lowering the cognitive load

9 types of comments [1]

- Function comments
- Design comments
- Why comments
- Teacher comments
- Checklist comments
- Guide comments
- Trivial comments
- Debt comments
- Backup comments

[1] Writing system software: code comments, by S. Sanfilippo

Comments

```
if (idle > server.repl_backlog_time_limit) {
    /* When we free the backlog, we always use a new
     * replication ID and clear the ID2. This is needed
     * because when there is no backlog, the master_repl_offset
     * is not updated, but we would still retain our replication
     * ID, leading to the following problem:
     *
     * 1. We are a master instance.
     * 2. Our replica is promoted to master. It's repl-id-2 will
     *    be the same as our repl-id.
     * 3. We, yet as master, receive some updates, that will not
     *    increment the master_repl_offset.
     * 4. Later we are turned into a replica, connect to the new
     *    master that will accept our PSYNC request by second
     *    replication ID, but there will be data inconsistency
     *    because we received writes. */
    changeReplicationId();
    clearReplicationId2();
    freeReplicationBacklog();
    serverLog(LL_NOTICE,
        "Replication backlog freed after %d seconds "
        "without connected replicas.",
        (int) server.repl_backlog_time_limit);
}
```

Source: register.c

Comments

```
// check if the restaurant actually exists
if (restaurant == null) {
      throw new RestaurantNotFoundException(booking.getRestaurantId());
}

// check if the number of diners in the booking is more than the number of seats in the restaurant
if (restaurant.capacity() < booking.getNumberOfDiners()) {
      throw new NoAvailableCapacityException("Number of diners exceeds available restaurant capacity");
}

// check the restaurant is open on the day of the booking
if (!restaurant.openingDays().contains(booking.getDate().getDayOfWeek())) {
      throw new RestaurantClosedException("Restaurant is not open on: " + booking.getDate());
}

// find all the bookings for that day and check that with all the booked diners the restaurant still has space for the new
booking diners
List allByRestaurantIdAndDate = repository.findAllByRestaurantIdAndDate(booking.getRestaurantId(),
                  booking.getDate());
int totalDinersOnThisDay = allByRestaurantIdAndDate.stream().mapToInt(Booking::getNumberOfDiners).sum();
if (totalDinersOnThisDay + booking.getNumberOfDiners() > restaurant.capacity()) {
      throw new NoAvailableCapacityException("Restaurant all booked up!");
}

// if we got this far, the booking is valid and we can save it
return repository.save(booking);
```

Source: https://trishagee.com/2021/05/14/do-we-need-comments-in-our-code/

Documentation

Like comments, documentation is another tool for lowering the cognitive load

Consistency also helps

Writing code

What happens when the task we're working on is not too easy or tedious and not too complex, without any external distraction?

The flow [1]

Hard to reach, easy to lose

Pure focus, where time flies, happyness and satisfaction

Takes 15 minutes to reach [2]

It doesn't come out from the blue: you have to plan for it

[1] Flow – The psycology of optimal experience by M. Cziksentmihalyi, 1990
[2] Peopleware: productive projects and teams by T. DeMarco & T. Lister, 1999

The flow

How to keep being in it

Organize your time:

- arrange with the team for reserved coding time
- early bird or night owl?
- setup no meeting days

The flow

How to keep being in it

Avoid distractions:

- signal you're concentrated (status on slack/teams/etc)
- shut down notification (yes, on your mobile too)
- in office: hoodies / hat / earphones / signs
- use fullscreen / no distraction mode
- maintain focus with pomodoro technique

The flow

How to keep being in it

Reduce tedious tasks:

- Use IDE / GenAI for writing trivial code

Testing

We want immediate feedback

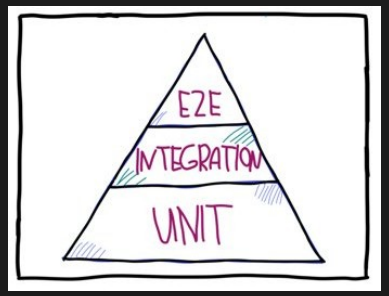
Ideally, when writing code you have:

- unit testing below 10 secs (modularize)
- integration testing below 30 seconds

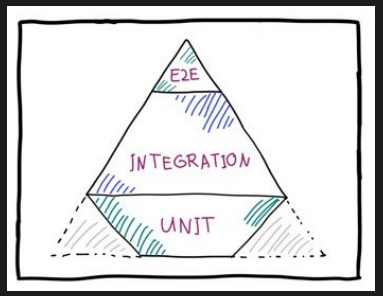
At a later stage you can have:

- E2E testing on CI/CD pipelines
- Smoke testing after deploy

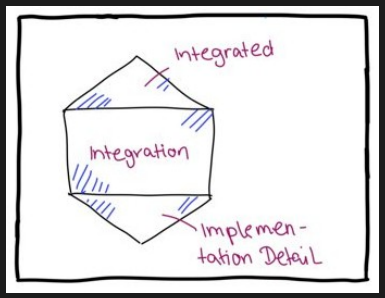
Testing Strategies



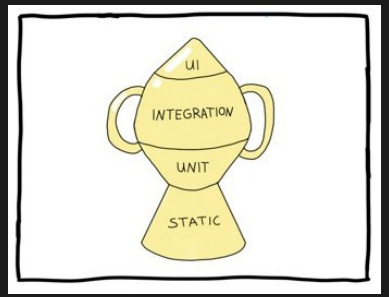
Pyramid



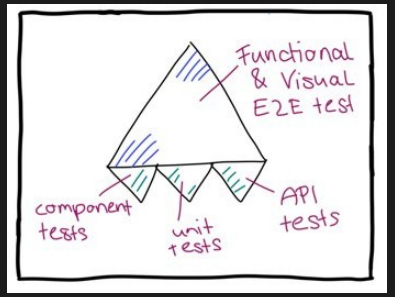
Diamond



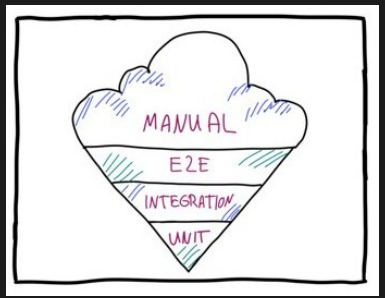
Honeycomb



Trophy



Crab



Ice Cone

Image source: https://web.dev/articles/ta-strategies

We are all different

What works for me, may not work for you.

Experiment with your team (and do it whenever a new hire joins or a team member leaves)!

But, most of all, the brain needs

Why We Sleep: The New Science of Sleep and Dreams by M. Walker, 2018

By the way, do you remember the number?

By the way, do you remember the number?

9 5 8 6 5 5 3 6

Questions?

Download this presentation here: