astea conference

{practical.magic}

astea conference

Магията на минимализма

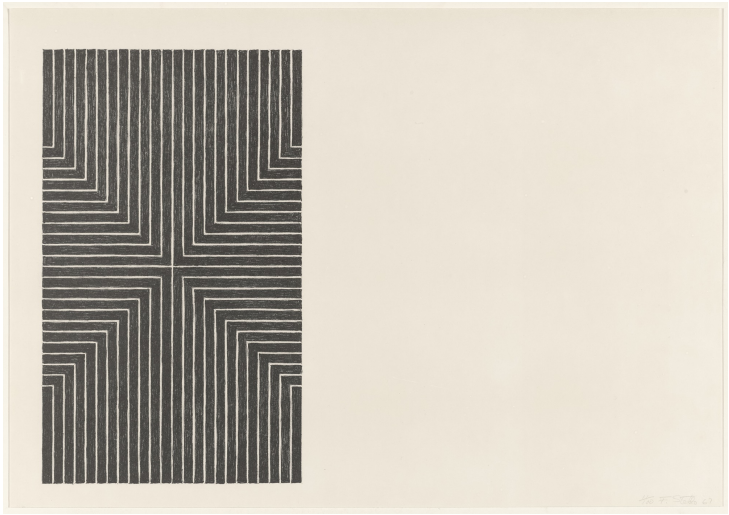
Янис Василев

astea conference

Jackson Pollock - Circle [10]

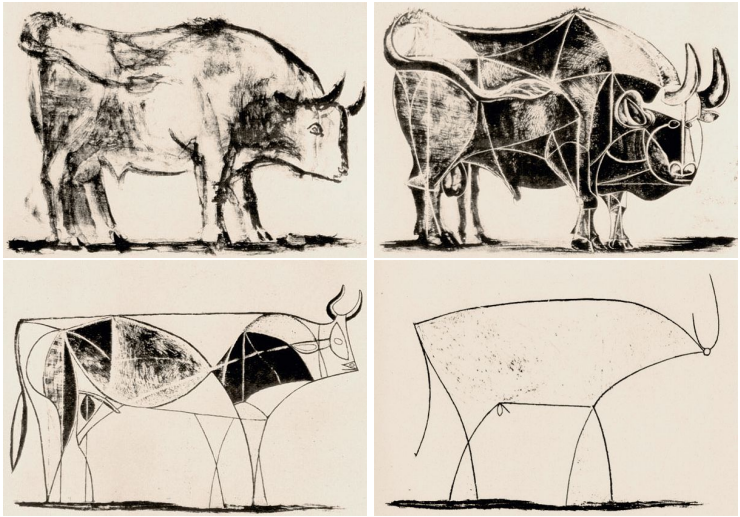
astea conference

Frank Stella - Die Fahne Hoch! [12]



astea conference

Pablo Picasso - Le Taurus [9]



Анонимен автор - анонимна функция [2]

```
function (array) {
  return array.filter(val => array > val ? 0 : [array=val])
}
```

astea conference

Психически дискомфорт

Видове знание

Можем да разгледаме **явното (explicit)** и **неявното (tacit)** знание като диаметрално противоположни според това това колко лесно се споделят с други хора.

Явно определения, декларации на функции

Неявно идеи зад тези определения и декларации

Инертното знание (inert knowledge) е усвоено, но несвързано с друго вече усвоено знание.

Понятие, след усвояването на което се променя възприятието на вече известни понятия, наричаме **прагово (threshold concept)** [8].

Облаци от понятия

Абстрактна линейна алгебра

Нека U и V са линейни пространства над полето F. Функцията $f : U \mapsto V$ ще наричаме **линейна функция** или **линеен оператор**, ако f е хомоморфизъм от U във V, т.е.

1. f е хомоморфизъм на адитивната група $(U, +)$ във $(V, +)$.
2. f е хомогенна относно умножение на аргумента си със скалари от F.

Множеството от линейните оператори от U във V означаваме с $\hom(U, V)$.

Абстрактна линейна алгебра

Пространството $\hom(U, V)$ е линейно относно операциите

Събиране $(f + g)(x) := f(x) + g(x)$

Композиция $(f \circ g)(x) := f(g(x))$

Доказателство.

Очевидно е, че линейните оператори наследяват събирането от съответните линейни пространства U и V.

За композицията имаме

$$(f \circ g)(\lambda x) = f(g(\lambda x)) = f(\lambda g(x)) = \lambda f(g(x)) = \lambda (f(g(x))). \qquad \square$$

Абстрактна конкретна линейна алгебра

За произволен вектор-стълб $\begin{bmatrix} x \\ y \end{bmatrix}$ дефинираме функциите

$$f\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) := \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$g\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) := \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Тогава композицията на f с g се задава чрез

$$f\left(g\left(\begin{bmatrix} x \\ y \end{bmatrix}\right)\right) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \left(\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}\right) = \left(\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}\right) \begin{bmatrix} x \\ y \end{bmatrix}$$

Modern mathematics tends to obliterate history: each new school rewrites the foundations of its subject in its own language, which makes for fine logic but poor pedagogy.

Robin Hartshorne, „Algebraic Geometry“, 1977 [5]

astea
conference

Когнитивни склонности
(Cognitive biases)

Когнитивна склонност

Когнитивна склонност е инстинктивно ирационално поведение, чрез което човек избягва психически дискомфорт.

Всички сме уязвими. Какво да правим:

- Да внимаваме да се не държим ирационално
- Да се стремим да не стигаме до ситуации, които предизвикват такова поведение

Долни лъжи

- ...
- От утре започвам диета
- Ще внимавам
- Има още много време
- ...

astea
conference

Склонност за потвърждаване
(Confirmation bias [15])

„Човек търси потвърждение и бяга от опровержение“

Склонност за потвърждаване

```
function min (a, b) {
  if (a < b || b !== b)
    return a
  return b
}
```

Защо b! == b?

Склонност за потвърждаване

```
function min (a, b) {
  if (a < b || b !== b)
    return a
  return b
}
```

Защо b! == b?

1. Случайно добавен по невнимание код

Склонност за потвърждаване

```
function min (a, b) {
  if (a < b || b !== b)
    return a
  return b
}
```

Защо `b !== b`?

1. Случайно добавен по невнимание код
2. Грешка поради неразбиране

Склонност за потвърждаване

```
function min (a, b) {
  if (a < b || b !== b)
    return a
  return b
}
```

Защо `b !== b`?

1. Случайно добавен по невнимание код
2. Грешка поради неразбиране
3. Авторът е искал да напише `a === b`

Склонност за потвърждаване

```
function min (a, b) {
  if (a < b || b !== b)
    return a
  return b
}
```

Защо $b !== b$?

1. Случайно добавен по невнимание код
2. Грешка поради неразбиране
3. Авторът е искал да напише $a === b$
4. Това е поредната странност на JavaScript

Склонност за потвърждаване

```
function min (a, b) {
  if (a < b || b !== b)
    return a
  return b
}
```

Защо `b !== b`?

1. Случайно добавен по невнимание код
2. Грешка поради неразбиране
3. Авторът е искал да напише `a === b`
4. Това е поредната странност на JavaScript
5. Това улавя следствие от спецификация на IEEE

Склонност за потвърждаване

```
function min (a, b) {
  if (a < b || b !== b)
    return a
  return b
}
```

Защо b! == b?

1. Случайно добавен по невнимание код
2. Грешка поради неразбиране
3. Авторът е искал да напише a === b
4. Това е поредната странност на JavaScript
5. Това улавя следствие от спецификация на IEEE
6. Друга причина

Склонност за потвърждаване

През 1960г. Питър Уасон установява тенденцията на хората да търсят потвърждения, но не и контрапримери, на подозренията си.

По думите на Уасон, хората имат склонност да търсят достатъчни, но не непременно необходими условия.

Склонност за потвърждаване

Той провежда един и същ експеримент с всеки от 29 произволно избрани студенти.
Дадени:

1. Тройка естествени числа (a, b, c)
2. Неизвестен предикат $\rho(x, y, z)$, който е удовлетворен от (a, b, c)

Студентът може неограничено да пита Уасон дали произволна тройка удовлетворява ρ. Задачата е да се намери предиката ρ.

Склонност за потвърждаване

Около 3/4 от студентите избързват да дадат отговор, след като се уверяват, че $\rho(x, y, z)$ е удовлетворен за няколко конкретни тройки.

Останалите студенти търсят контрапримери, т.е. тройки (a, b, c) от естествени числа, за които предиката не е удовлетворен. Така се възползват от метода на изключването, за да намерят ρ, вместо да дават първия отговор, който им хрумва.

astea conference

Проклятие на знанието
(Curse of knowledge [3])

„Човек не може да пренебрегне знанието си“

astea conference

Проклятие на знанието

През 1989г. група икономисти провеждат следния експеримент:

Една група от студенти получава данни за приходите на няколко компании за последните десетина години, с премахнати данни за последната година. Задачата на всеки един от тях е да екстраполира от данните приходите за последната година.

Проклятие на знанието

Друга група от студенти получава същите данни заедно с истински данни за последната година. Задачата им е да дадат оценка на това как някой от другата група би прогнозирал приходите.

Експериментът установява, че втората група студенти, които знаят данните за текущата година, използват съществено това знание в прогнозите си.

Една легенда за Какутани [6]

astea conference

Пълзящ детерминизъм
(Crippling determinism / hindsight bias [4])

„Всичко изглежда очевидно впоследствие“

Пълзящ детерминизъм

През 1975г. Барух Фишхоф провежда следния експеримент:

Не пет различни групи от студенти се дава кратък текст, описващ малко известно историческо събитие, съпътствано от четири възможни сценария за развитие на ситуацията.

Първата група студенти не получава никакви допълнителни сведения, докато на останалите четири групи той казва, че действителният изход от събитието е един от четирите посочени.

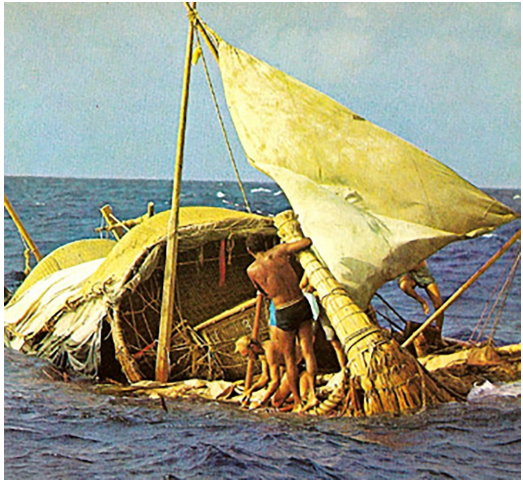
Пълзящ детерминизъм

С изключение на първата група студенти, които не знаят действителният изход, останалите посочват като най-вероятен изходът, който мислят за действителен.

Лодката Ра [11] [13]

През 1969г. етнографът Тур Хейердал решава да построи лодка от папирус и с група съмишленици и да прекоси Атлантическия океан.

Лодката Ра [7]



Лодката Ра 2 [7]

astea
conference

Програмиране

Де-факто стандарти

Следните функции са се наложили като де-факто стандартни:

Де-факто стандарти

Следните функции са се наложили като де-факто стандартни:

- *cdr*

Де-факто стандарти

Следните функции са се наложили като де-факто стандартни:

- *cdr*
- *sqrt*

Де-факто стандарти

Следните функции са се наложили като де-факто стандартни:

- *cdr*
- *sqrt*
- *atan2*

Де-факто стандарти

Следните функции са се наложили като де-факто стандартни:

- *cdr*
- *sqrt*
- *atan2*
- *lerp*

astea conference

Де-факто стандарти

Следните функции са се наложили като де-факто стандартни:

- *cdr*
- *sqrt*
- *atan2*
- *lerp*
- *strstr*

Де-факто стандарти

Следните функции са се наложили като де-факто стандартни:

- *cdr*
- *sqrt*
- *atan2*
- *lerp*
- *strstr*
- *fflush*

Де-факто стандарти

Следните функции са се наложили като де-факто стандартни:

- *cdr*
- *sqrt*
- *atan2*
- *lerp*
- *strstr*
- *fflush*
- *triu*

astea conference

Де-факто стандарти

Следните функции са се наложили като де-факто стандартни:

- *cdr*
- *sqrt*
- *atan2*
- *lerp*
- *strstr*
- *fflush*
- *triu*
- *lm*

Ясни спомени

Before:

```
function min (a, b) {
  if (a < b || b !== b)
    return a
  return b
}
```

After:

```
function min (a, b) {
  if (a < b || Number.isNaN(b))
    return a
  return b
}
```

Кое е по-просто?

```
function sumAgeV1 (people) {
  const result = 0;

  for (const person of people)
    result += person.age

  return result
}

function sumAgeV2 (people) {
  return people
    .map(person => person.age)
//  .sum()
    .reduce((total, individual) => total + individual, 0)
}
```

astea conference

Кое е по-просто?

```
function repeatV1 (value, count) {
  const result = [];

  for (let i = 0; i < count; i++)
    result.push(value)

  return result
}

function repeatV2 (value, count) {
  return Array.from({ length: count }).map(() => value)
}
```

Алгебрична затвореност

```
function employeeSummaryV1 (employees) {
  const result = {
    totalAge: 0,
    meanAge: 0
    minAge: 0
    maxAge: 0
  };

  for (const employee of employees) {
    ...
  }

  return result
}
```

Алгебрична затвореност

Къде е *employeeSummaryV2*?

Алгебрична затвореност

Къде е *employeeSummaryV2*?

> *Far better an approximate answer to the right question, which is often vague, than an exact answer to the wrong question, which can always be made precise.*
>
> John Tukey, „The future of data analysis“, 1962 [14]

Алгебрична затвореност

Къде е *employeeSummaryV2*?

> *Far better an approximate answer to the right question, which is often vague, than an exact answer to the wrong question, which can always be made precise.*
>
> John Tukey, „The future of data analysis“, 1962 [14]

На чист български: няма нужда от толкова специфична функция.

Алгебрична затвореност

Хората харесват алгебрични структури без да го осъзнават. Интересуваме се от алгебрична затвореност, т.е. доколко даден обект може да бъде съчетан със сродни обекти чрез някакви операции.

Положителни примери:

- Функции
- Структури
- Функтори (генерични/шаблонни типове)
  - Масиви
  - Асоциативни таблици
  - future-и
  - ...

Алгебрична затвореност

Хората харесват алгебрични структури без да го осъзнават. Интересуваме се от алгебрична затвореност, т.е. доколко даден обект може да бъде съчетан със сродни обекти чрез някакви операции.

Отрицателни примери:

- *Error*
- *fetchIcons*
- *createNestedBlockRule*
- *getStatesAtIndex*

Самолетопоклонничество (cargo cult)

- Абстрахиране на синтактично, а не семантично сходен код
- Преусложнени функционални конструкции
- Страх от *goto*
- Използване на чужд код без съображения за:
  - Лицензи
  - Съвместимост
  - Сигурност
  - Бъгове
  - Поддръжка

Самолетопоклонничество (cargo cult)

- Абстрахиране на синтактично, а не семантично сходен код
- Преусложнени функционални конструкции
- Страх от *goto*
- Използване на чужд код без съображения за:
  - Лицензи
  - Съвместимост
  - Сигурност
  - Бъгове
  - Поддръжка
- Точки и запетайки в JavaScript

Невалидно състояние

> *As is usual with programming disasters, nobody recognized it as such.*

Gerald Weinberg, „The Psychology of Computer Programming“, 1971 [16]

- Кодът, който е възможно да предизвика невалидното състояние, трябва да се изолира
- Колкото се може по-малко правила трябва да имат изключения

astea conference

Какво вижда автора [1]

astea conference

Какво виждат хората [1]

Един пример все пак...

```
class StopwatchV1 {
  constructor () {
    this.laps = []
    this.elapsed = 0
  }
  start () {
    this._interval = setInterval(() => { this.elapsed++ }, 1)
  }
  lap () {
    this.laps.push(this.elapsed)
  }
  stop () {
    this._interval && clearInterval(this._interval)
    this.elapsed = 0
  }
}
```

Един пример все пак...

Проблеми:

1. Функциите *setInterval* и *clearInterval* не са част от езика
2. Имаме „*private*“ поле *_interval*, което даже не знам от какъв тип е и какво прави
3. *laps* и *elapsed* са просто числа, но ние мислим за всяко едно от тях като за време

astea conference

Един пример все пак...

```
class IntervalRunner {
  constructor (callback, interval) {
    this.callback = callback
    this.interval = interval
  }
  start () {
    this._interval = setInterval(this.callback, this.interval)
  }
  stop () {
    if (this._interval)
      clearInterval(this._interval)
  }
}
```

astea conference

Един пример все пак...

```
// Преди
const stopwatch = new StopwatchV1()
stopwatch.start()
stopwatch.lap()
const lap1 = stopwatch.laps[1]
stopwatch.stop()
const elapsed = stopwatch.elapsed

// Сега
let elapsed = 0
const runner = new IntervalRunner(function () { elapsed += 1 }, 1)
runner.start()
const lap1 = elapsed
runner.stop()
```

astea conference

tl;dr

- Мисленето натоварва хората
- Хората искат да виждат това, с което са свикнали
- Хората не искат да научават нови неща
- Най-добрият код е този, който не трябва да се пише
- Сложният код е неизбежен, но можем да изберем да го изолираме
- Всяко отклонение от минималния работещ код изисква поне един коментар
- Ефективен начин да се постигне простота е да се стремим обектите ни да образуват алгебрични структури
- Трябва да позволяваме на абстракциите ни да бъдат гъвкави, но и да направим обичайното им използване просто

astea conference

Библиография I

- >DANGER | HD photo by Joey Banks on Unsplash. URL: https://unsplash.com/photos/F4oOrbkY3-8 (дата на посещ. 16.05.2019).
- Anonymous. Lossy Sorting (Implement Dropsort). 26 юни 2015. URL: https://codegolf.stackexchange.com/questions/61808/lossy-sorting-implement-dropsort (дата на посещ. 14.05.2019).
- Colin Camerer, Goerge Loewenstein и Martin Weber. „The Curse of Knowledge in Economic Settings“. В: Journal of Political Economy (1989). DOI: 10.1086/261651. URL: https://www.jstor.org/stable/1831894 (дата на посещ. 13.05.2019).

Библиография II

- Beruch Fischhoff. „Hindsight Is Not Equal to Foresight“. В: Journal of Experimental Psychology (1975). DOI: 10.1037/0096-1523.1.3.288. URL: https://psycnet.apa.org/record/1976-00159-001 (дата на посещ. 13.05.2019).
- Robin Hartshorne. Algebraic geometry. New York: Springer-Verlag, 1977. ISBN: 0-387-90244-9.
- Steven G. Krantz. Mathematical Apocrypha: Stories and Anecdotes of Mathematici The Mathematical Association of America, 2002. ISBN: 0883855399.

Библиография III

- Medice, cura te ipsum. Подвиг хирурга-полярника Леонида Рого URL: http://polzam.ru/index.php/istorii/item/949-medice-cura-te-ipsum-podvig-khirurga-polyarnika-leonida-rogozova (дата на посещ. 16.05.2019).
- Eric Meyer и Ray Land. On the Failure to Estimate Hypotheses in a Conceptual Task. 2003. URL: http://www.leeds.ac.uk/educol/documents/142206.pdf (дата на посещ. 15.05.2019).
- Pablo Picasso. Bull. URL: http://www.artyfactory.com/art_appreciation/animals_in_art/pablo_picasso.htm (дата на посещ. 12.05.2019).

Библиография IV

Jackson Pollock. Circle. URL: https://www.moma.org/collection/works/79675 (дата на посещ. 12.05.2019).

Ra Expeditions - The Kon-Tiki Museum. URL: https://www.kon-tiki.no/expeditions/ra-expeditions/ (дата на посещ. 12.05.2019).

Frank Stella. Die Fahne Hoch! 1967. URL: https://www.moma.org/collection/works/61215 (дата на посещ. 12.05.2019).

THEY CAME BEFORE COLUMBUS - Our Afrikan Heritage Magazine URL: http://www.afrikanheritage.com/they-came-before-columbus/ (дата на посещ. 16.05.2019).

Библиография V

- John W. Tukey. The Future of Data Analysis. 1962. DOI: 10.1214/aoms/1177704711. URL: https://projecteuclid.org/euclid.aoms/1177704711 (дата на посещ. 16.05.2019).
- Peter Wason. „On the Failure to Estimate Hypotheses in a Conceptual Task“. В: Quarterly Journal of Experimental Psychology (1960). DOI: 10.1080/17470216008416717. URL: https://www.tandfonline.com/doi/abs/10.1080/17470216008416717 (дата на посещ. 08.04.2013).
- Gerald Weinberg. The Psychology of Computer Programming. Litton Educational Publishing, Inc., 1971.

astea conference

Благодаря!

Янис Василев, ianis@ivasilev.net

Презентация: https://ivasilev.net/files/Slides