using vim at work

@danishprakash
danish prakash
software engineer @HackerRank
danishprakash.github.io
what led to this talk?
Mentor
Kindly change your editor by Monday or else it will be difficult for us to work together. See you on Monday!
outline

psychology

mouse & scroll

visual cues

extras

takeaways
psychology
classical conditioning

an associative learning process that occurs between novel and familiar stimuli
The mere-exposure effect is a psychological phenomenon by which people tend to develop a preference for things merely because they are familiar with them. Learning something new is easier when complemented with familiar elements.

---

from Wikipedia, the free encyclopedia

Reder et al
Carnegie Mellon University
mouse & scroll

“but vim promotes a mouse-less workflow!”
The mouse is ubiquitous in navigating websites, highlighting text, creating presentations, managing emails, modifying system settings, designing graphics, and exploring image galleries.
keyboard vs mouse

test subjects report that keyboarding is faster than mousing, whereas stopwatch proves mousing is faster than keyboarding.

Bruce Tognazzini, Apple Interface Design
:set mouse-=a
// ReferenceInfo holds information about reference to an identifier in Go source.

type ReferenceInfo struct {
    Name     string
    mappedRange
    ident    *ast.Ident
    obj      *types.Object
    pkg      Package
    isDeclaration bool
}

// References returns a list of references for a given identifier within the packages
// containing i.File. Declarations appear first in the result.

func (i *IdentifierInfo) References(ctx context.Context) ([]*ReferenceInfo, error) {
    ctx, done := trace.StartSpan(ctx, "source.References")
    defer done()

    references := []*ReferenceInfo{
    }

    // If the object declaration is nil, assume it is an import spec and do not look for references.
    if i.Declaration.obj == nil {
        return nil, errors.Errorf("no references for an import spec")
    }

    info := i.pkg.TypesInfo()
    if info == nil {
        return nil, errors.Errorf("package %s has no types info", i.pkg.PkgPath())
    }

    if i.Declaration.isImplicit {
        // The definition is implicit, so we must add it separately.
        // This occurs when the variable is declared in a type switch statement
        // or is an implicit package name. Both implicit are local to a file.
        references = append(references, &ReferenceInfo{
            Name:     i.Declaration.obj.Name(),
            mappedRange: 1.Declaration.mappedRange,
            obj:      i.Declaration.obj,
            pkg:      i.pkg,
            isDeclaration: true,
        })
    }

    for ident, info := range info.Defs {
        if obj == nil || !sameObj(obj, i.Declaration.obj) {
            continue
        }
        info := postToMappedRange(ctx, i.pkg, ident.Pos(), ident.End())
        if err != nil {
            continue
        }

        references = append(references, &ReferenceInfo{
            Name:     info.Name(),
            mappedRange: 1.mappedRange,
            obj:      info.Obj,
            pkg:      i.pkg,
            isDeclaration: true,
        })
    }

    return references, nil
}
term buffer
scroll
// ReferenceInfo holds information about reference to an identifier in Go source.

type ReferenceInfo struct {
  Name string
  mappedRange 
  ident   *ast.Ident
  obj     types.Object
  pkg     Package
  isDeclaration bool
}

// References returns a list of references for a given identifier within the packages containing i.File. Declarations appear first in the result.

func (i *IdentifierInfo) References(ctx context.Context) ([]*ReferenceInfo, error) {
  ctx, done := trace.StartSpan(ctx, *source.References)
  defer done()
  // references []*ReferenceInfo

  // If the object declaration is nil, assume it is an import spec and do not look for references.
  if i.Declaration.obj == nil {
    return nil, errors.Errorf("no references for an import spec")
  }
  info := i.pkg.TypesInfo()
  if info == nil {
    return nil, errors.Errorf("package %s has no types info", i.pkg.PkgPath())
  }

  if i.Declaration.wasImplicit {
    // The definition is implicit, so we must add it separately.
    // This occurs when the variable is declared in a type switch statement
    // or is an implicit package name. Both implicit are local to a file.
    references = append(references, &ReferenceInfo{
      Name:          i.Declaration.obj.Name(),
      mappedRange:   i.Declaration.mappedRange,
      obj:           i.Declaration.obj,
      pkg:           i.pkg,
      isDeclaration: true,
    })
  }

  for _, obj := range info.Defs {
    if obj == nil || !sameObj(obj, i.Declaration.obj) {
      continue
    }
    ref, err := posToMappedRange(ctx, i.pkg, ident.Pos(), ident.End())
    if err != nil {
      // Handle error
    }
  }

  return references, nil
}
bottomline

:set mouse=a
even if you are a keyboarder

unexpected surprises
no more of them

pair programming
easy on the person next to you
visual cues

“visual what? and why?”
the human brain can process visual information 60,000 times faster than textual.

effective use of visuals can decrease learning time, improve comprehension, enhance retrieval, and increase retention.
familiar stimuli
directory

tree

netrw (:Vexplore)

NerdTree
func encodeString(s string) (encoding string, err error) {
    for _, r := range s {
        if r == '!' || r >= utf8.RuneSelf {
            // This should be disallowed by CheckPath, but diagnose anyway.
            return "", fmt.Errorf("internal error: inconsistency in EncodePath")
        }
    }
    var buf []byte
    for _, r := range s {
        if 'A' <= r && r <= 'Z' {
            buf = append(buf, ',1', byte(r-'A'))
        } else {
            buf = append(buf, byte(r))
        }
    }
    return string(buf), nil
}

func DecodePath(encoding string) (path string, err error) {
    path, ok := decodeString(encoding)
    if !ok {
        return "", fmt.Errorf("invalid module path encoding %v", encoding)
    }
    err := CheckPath(path); err != nil {
        return "", fmt.Errorf("invalid module path encoding %v: %v", encoding, err)
    }
    return path, nil
}

func GetVersion(encoding string) (version string, err error) {
    // DecodeVersion returns the version string for the given safe encoding.
    // It fails if the encoding is invalid or encodes an invalid version.
    // It returns an error if the version is invalid.
    // It returns nil if the encoding is not a valid version.
    // It returns the version string if the encoding is valid.
}
func EncodePath(path string) (encoding string, err error) {
    if err := CheckPath(path); err != nil {
        return "", err
    }

    return encodeString(path)
}

func EncodeVersion(v string) (encoding string, err error) {
    if err := checkElem(v, true); err != nil || strings.Contains(v, "!") {
        return "", fmt.Errorf("disallowed version string %q", v)
    }

    return encodeString(v)
}

func encodeString(s string) (encoding string, err error) {
    haveUpper := false
    for _, r := range s {
        if r == '!' || r >= utf8.RuneSelf {
            // This should be disallowed by CheckPath, but diagnose anyway.
            return "", fmt.Errorf("internal error: inconsistency in EncodePath")
        }

        if 'A' <= r && r <= 'Z' {
            haveUpper = true
        }
    }

    if !haveUpper {
        return s, nil
    }
}
extras
takeaways
use features you are already familiar with
it makes it easier to learn and get accustomed to new information

make the most out of visual cues
our brains are better at processing visuals, use it to your advantage

try not to be repulsive towards new stimuli
whether it’s a new plugin, pattern or a feature altogether
thank you
references

overview of classical conditioning
https://www.verywellmind.com/classical-conditioning-2794859

mere-exposure effect
https://en.wikipedia.org/wiki/Mere-exposure_effect

Building knowledge requires bricks, not sand
Lynne M. Reder, Xiaonan L. Liu, Alexander Keinath, and Vencislav Popov

mouse vs keyboard, AskTog
https://www.asktog.com/TOI/toi06KeyboardVMouse1.html

human brain and visual cues

effects of visual learning