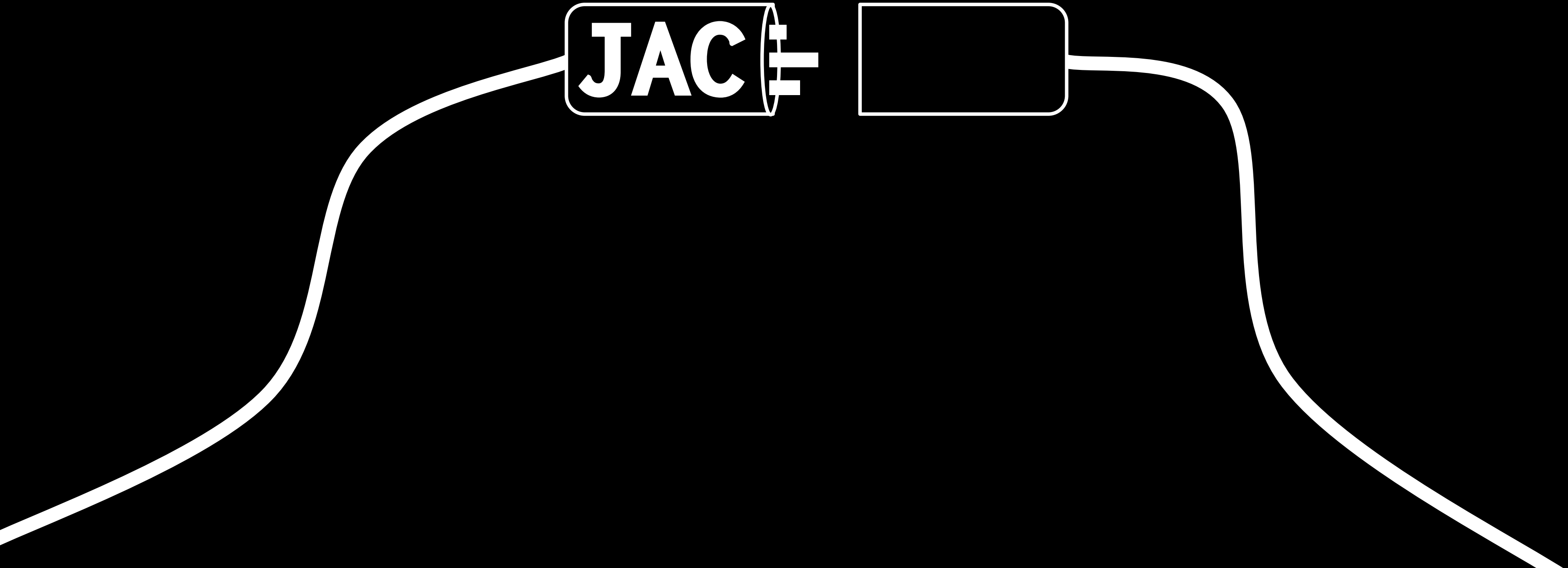

JAC

Run Once

Edit Fast

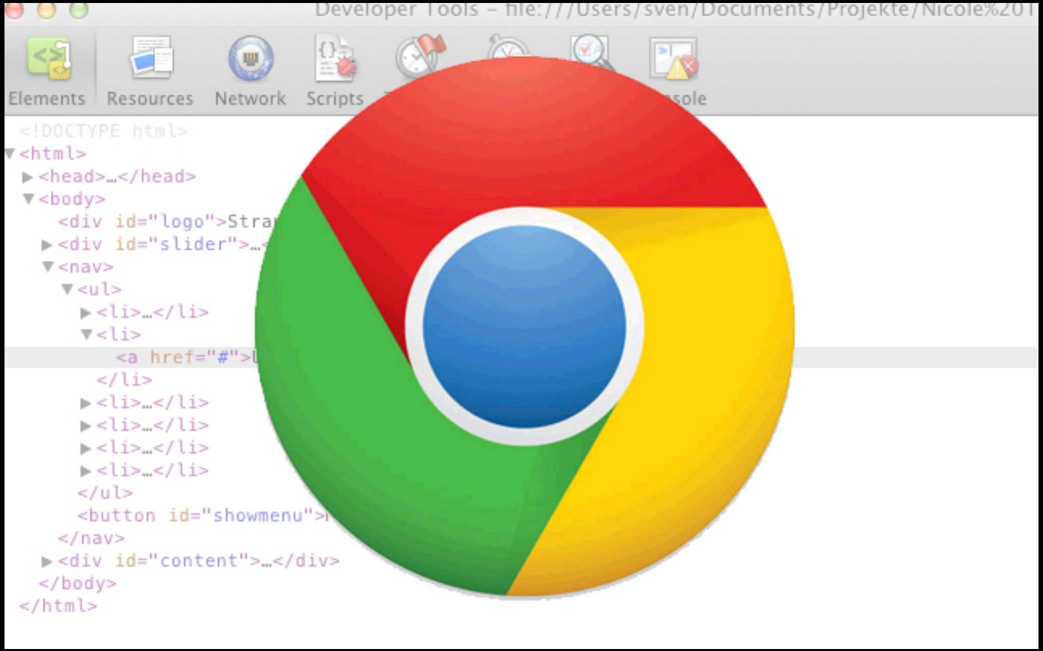
to make developing easy

Inspiration

LIGHT TABLE

the next generation code editor

Microsoft® Visual Studio®

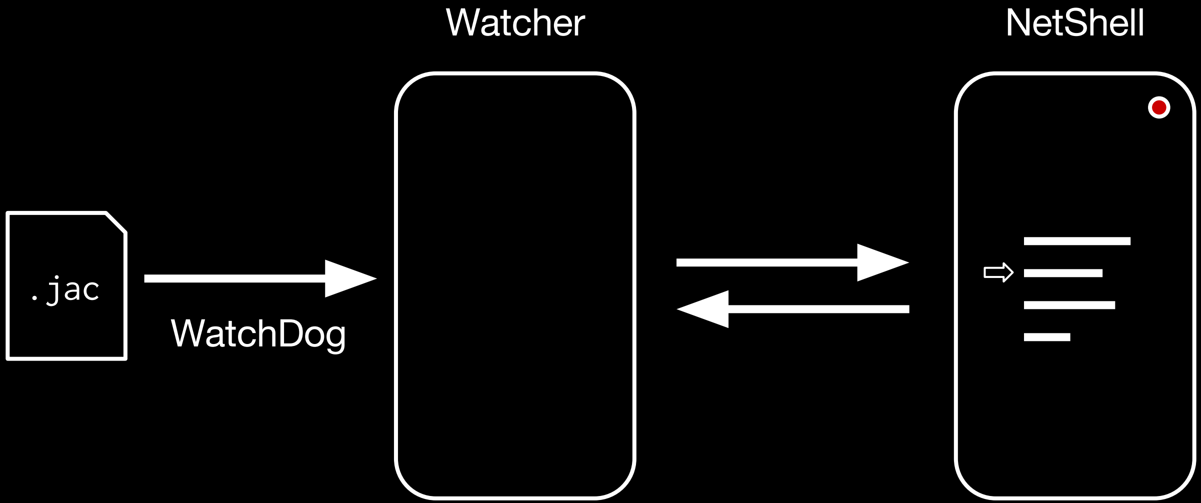

Developer Tools – file:///Users/sven/Documents/Projekte/Nicole%201
Elements
Resources
Network
Scripts
sole

```
<!DOCTYPE html>
▼<html>
  ▶<head>…</head>
  ▼<body>
      <div id="logo">Stra
    ▶<div id="slider">…<
    ▼<nav>
      ▼<ul>
        ▶<li>…</li>
        ▼<li>
            <a href="#">U
          </li>
        ▶<li>…</li>
        ▶<li>…</li>
        ▶<li>…</li>
        ▶<li>…</li>
        </ul>
        <button id="showmenu">
      </nav>
    ▶<div id="content">…</div>
    </body>
</html>
```

Demo

How?

Extension of func


Watcher
NetShell
.jac
WatchDog

Code watcher

Updates internal representation via socket

Watcher

```
class CodeChangedEventHandler(FileSystemEventHandler):
    def __init__(self, address, port):
        self.clientsocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.clientsocket.connect((address,port))

    def on_modified(self, event):
        file_name, file_extension = os.path.splitext(event.src_path)
        if file_extension != ".jac" or event.is_directory: return

        with open (event.src_path, "r") as myfile:
            data = myfile.read().replace('\n', ' ') + '\n'
            self.clientsocket.sendall(data)
```

Big Step vs Small Step

expr → val

expr → expr → val

Small Step Evaluation

```
main_expr = MExpr of expr * (string * main_expr) list
          | MTerm of value
```

NetShell

```
eval expr → new_expr
message = socket.read()
if message = "\n": continue
if message = code:
    find diff
    update global env
```

diff

Compare internal representations

```
fun irDiff (I.MTerm t1) (I.MTerm t2) currentFunc =
        if (valueEquals t1 t2) then [] else [currentFunc]
   | irDiff (I.MExpr (e1,_)) (I.MExpr (e2,_)) currentFunc = (case (e1, e2) of
        (I.EIf (e1, f1, g1), I.EIf (e2, f2, g2)) =>
          (irDiff e1 e2 currentFunc)@(irDiff f1 f2 currentFunc)@(irDiff g1 g2 currentFunc)
      | (I.EIdent name1, I.EIdent name2) =>
          if (name1 = name2) then [] else [currentFunc]
      | (I.ELet (name1, e1, body1), I.ELet (name2, e2, body2)) =>
          (if (name1 = name2)
            (* let x = e1 in body1 => let x = e2 in body2
               look at e1, e2 and body1, body2 to find diffs lower down *)
           then (irDiff e1 e2 currentFunc) @ (irDiff body1 body2 currentFunc)
            (* let x = ... has changed to let y = ... so current func has changed.
               stop looking for diffs lower down.  *)
           else [currentFunc, (name1, NONE)])
      | (I.ELetFun (name1, param1, functionBody1, body1), I.ELetFun (name2, param2, functionBody2, body2)) =>
          (if (name1 = name2)
           then
            let val newFunc = (name2, SOME (I.MTerm (I.VRecClosure (name2, param2, functionBody2, [])))) in
              (irDiff functionBody1 functionBody2 newFunc)
            end
           else [currentFunc,(name1,NONE)])@
          (if (param1 = param2) then [] else [currentFunc])@
          (irDiff body1 body2 currentFunc)
      | (I.EApp (e1_old, e2_old), I.EApp (e1_new, e2_new)) =>
          (irDiff e1_old e1_new currentFunc) @ (irDiff e2_old e2_new currentFunc)
      | (I.EFun (name1, body1), I.EFun (name2, body2)) =>
          (if (name1 = name2) then (irDiff body1 body2 currentFunc) else [currentFunc])
      | (_, _) => [currentFunc])
    | irDiff _ _ currentFunc = [currentFunc]
```

Old

$$\begin{tikzcd}
 & \texttt{let} \arrow[dl] \arrow[dr] & \\
\texttt{“foo”} & & \texttt{(x+1)}
\end{tikzcd}$$

New

$$\begin{tikzcd}
 & \texttt{let} \arrow[dl] \arrow[dr] & \\
\texttt{“foo”} & & \texttt{(x+2)}
\end{tikzcd}$$

Global Env

```
val globalEnv : ((string * ((I.main_expr))) list) ref = ref []

fun addToGlobal funcName value = ((globalEnv:=remDups ((funcName, value)::(!globalEnv))); ())

fun changeGlobal funcName newClosure =
  ((globalEnv :=
    (List.foldl (fn ((n, oldClosure),y) =>
                  if (funcName = n)
                  then (funcName, (changeClosureEnv oldClosure newClosure))::y
                  else (n, oldClosure)::y) [] (!globalEnv))); ())
fun removeGlobal name =
  ((globalEnv :=
    (List.foldl (fn ((n,v),y) =>
                  if (name = n)
                  then y
                  else (n, v)::y) [] (!globalEnv))); ())

fun updateGlobals ((name, NONE)::funcList) = (removeGlobal name; updateGlobals funcList)
  | updateGlobals ((name, SOME closure)::funcList) = (changeGlobal name closure; updateGlobals funcList)
  | updateGlobals [] = ()
```

Are you JAC’d in?