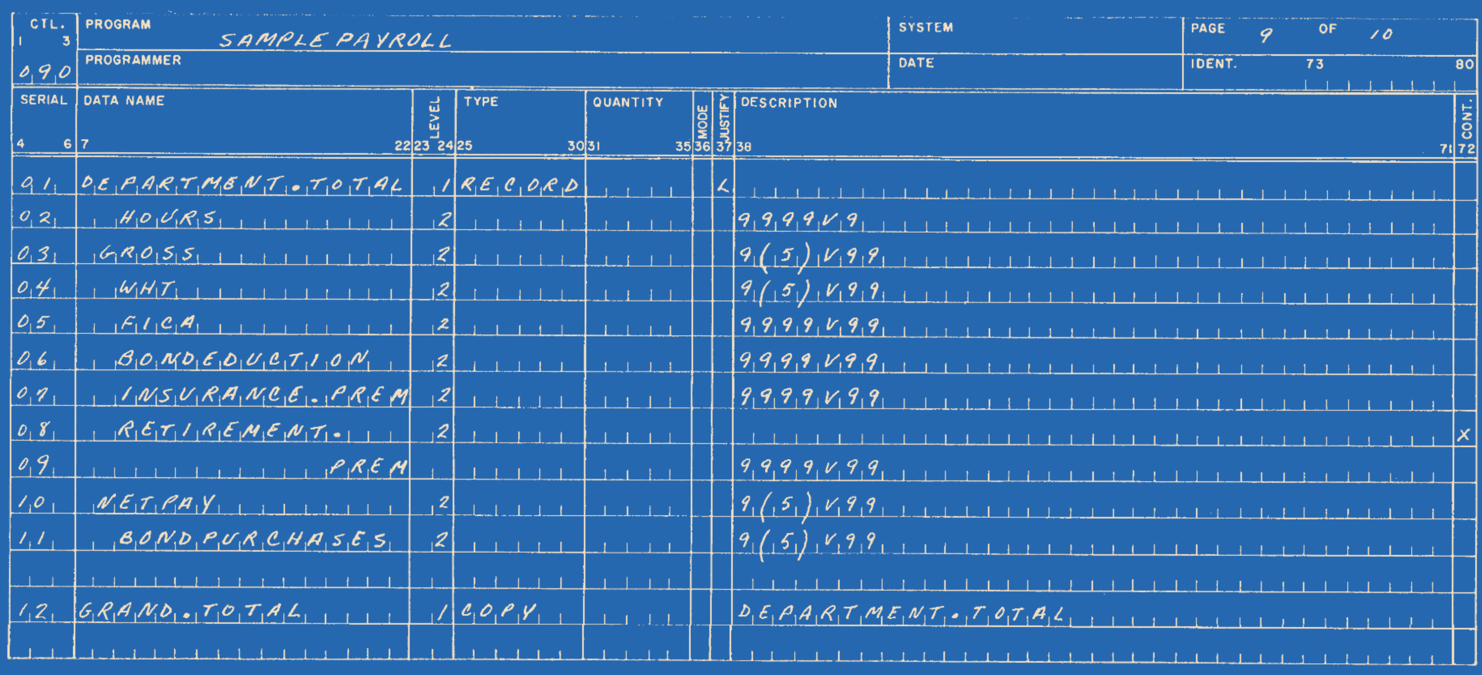
Programming with types

Many programming languages use types to make programs easier and safer, but they differ widely in their understanding of the notion of a type. Are types formal mathematical entities, a clever trick, or a software engineering tool?

1903: Mathematical origins

Bertrand Russell proposes a "Doctrine of Types" to avoid paradoxes such as the one which arises when considering the set of all sets that do not contain themselves as elements.

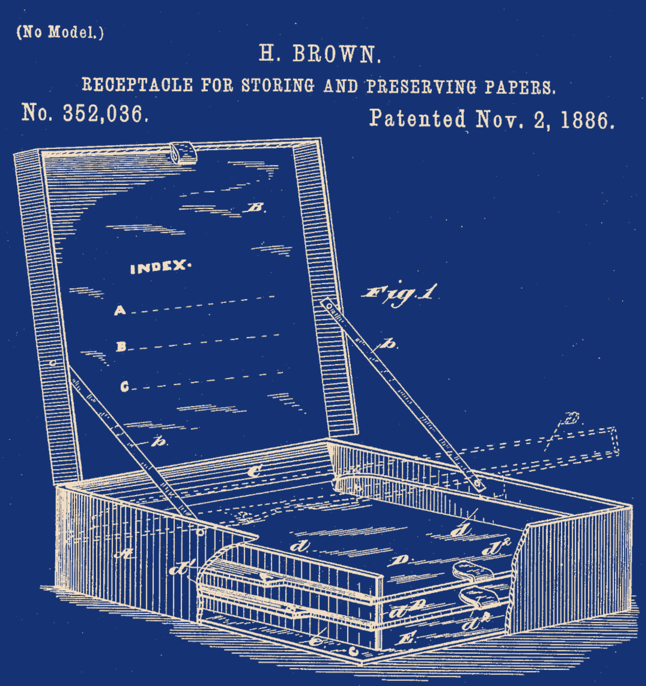


1956: Hacker origins

FORTRAN variables can be in two modes: fixed-point and floating-point, but in 1956 the term "type" is not yet used.

1957: Managerial origins

Inspired by the filing cabinet, COMTRAN & FLOW-MATIC use data description cards to specify the structure of data using files, records and fields.



```
let   t = "X + (X * ( ¬ X))"  ;;
let  s1 = ss(map (AXIOM `BA`)[`andinv` ; `oride`] );;
let  t1,th1 = simpterm  s1  t ;;
let  s2 = ss  (map(AXIOM `BA`) [`ordist` ; `orinv`; `andide`] );;
let  t2,th2 = simpterm  s2  t ;;
let  th = TRANS(SYM th1, th2) ;;
```

1974: Engineering origins

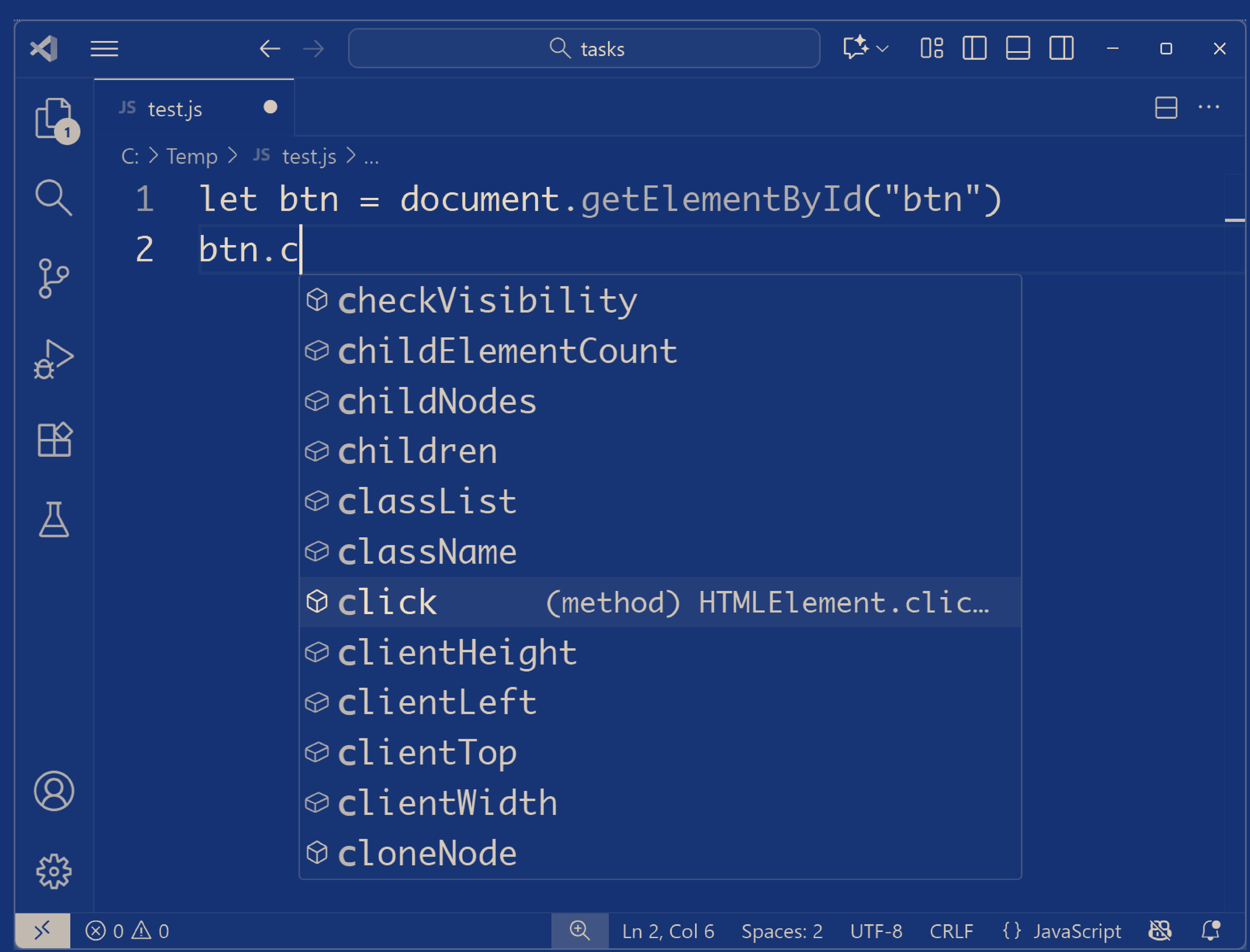
The CLU language created by Barbara Liskov introduces abstract data types to hide the underlying representation of data. Types do not describe the structure of data. Instead, they are used as a checking mechanism.

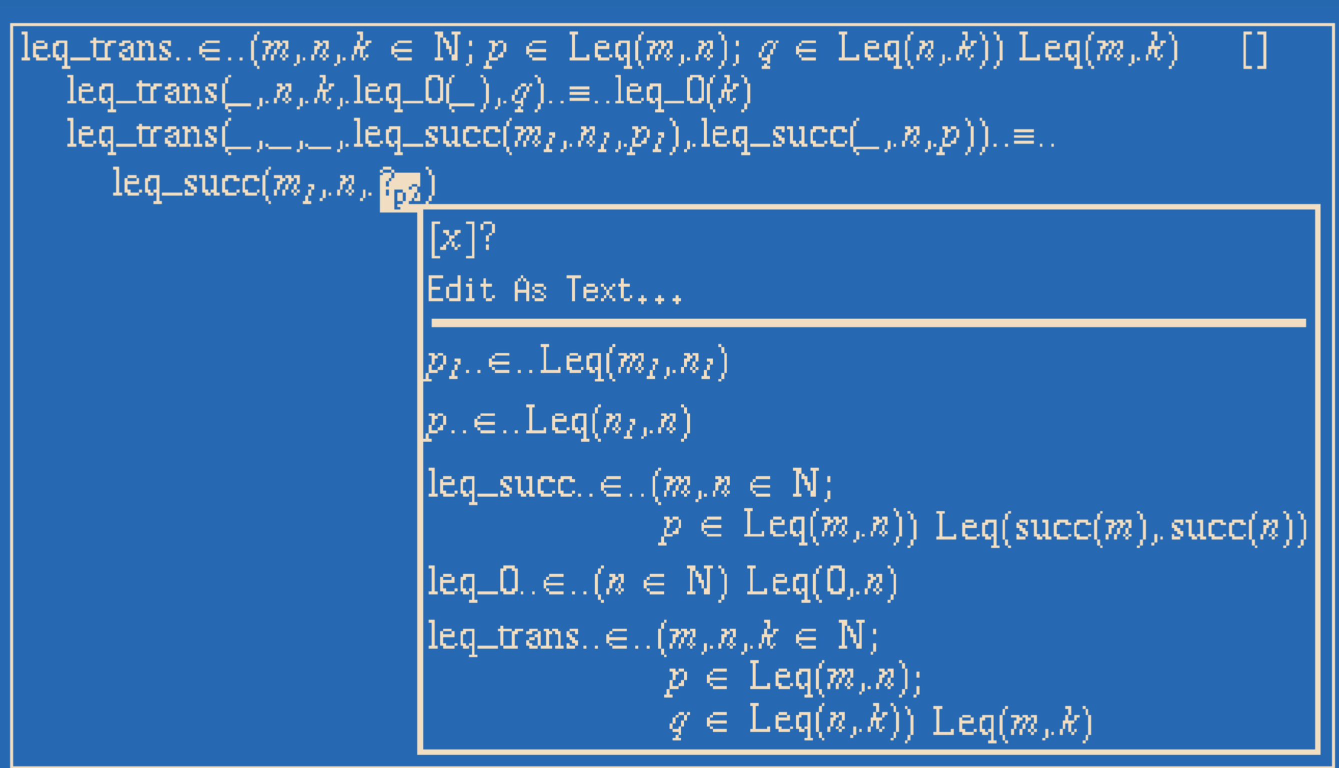
1978: Meeting of ideas

The interactive theorem prover Edinburgh LCF introduces a metalanguage (ML) for writing programs that construct proofs. ML later combines the ideas of multiple cultures, using types for both data structuring and type checking.

1989: New mathematical directions

Theorem provers like ALF and Rocq use programming with types to prove formal mathematical theorems. Type checking is used to certify the correctness of proofs.





2012: New engineering directions

Types in TypeScript are used for better developer tools. They are checked, but the type system is unsound and cannot make any formal correctness guarantees.