

BY BERTRAND RUSSELL.

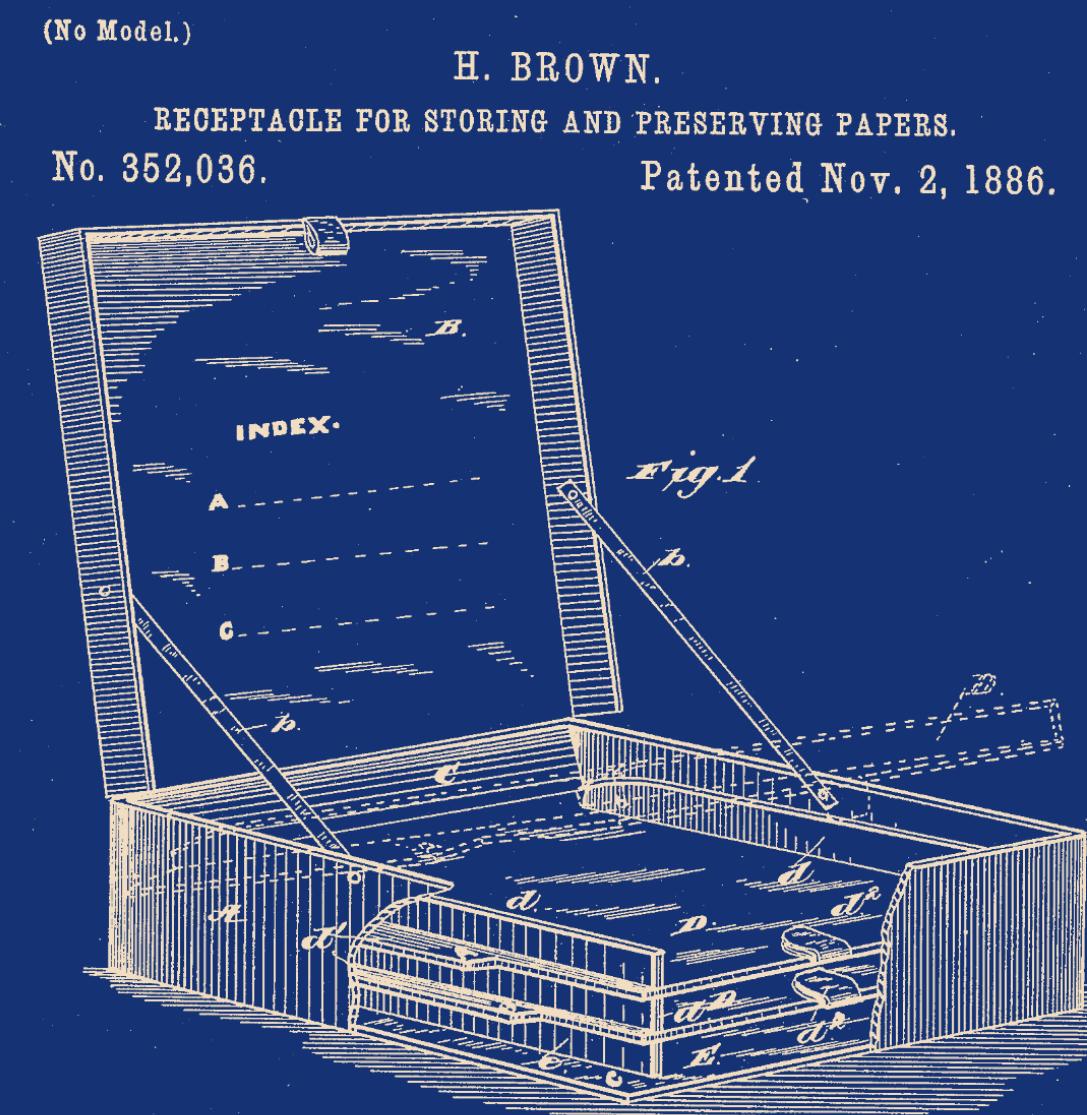
Programming with types

The following theory of symbolic logic record, intended to form the basis of a new instance by its ability to solve certain contradictions, of which the one best known to mathematicians is Buridan's paradox. The theory in question seems not wholly dependent on this indirect record, and the theory in question seems not wholly dependent on this indirect record. It has also, if I am not mistaken, a certain consonance with common sense which makes it inherently correct. Much stress should be laid; for common sense is far more fallible than it likes to believe. I shall then find a logical solution to the paradox, and shall then be solved, and shall then show how the theory of logical types effects their solution.

COL. 3	PROGRAM	SAMPLE PAYROLL	PROGRAMMER	SYSTEM	PAGE 9 OF 10
SERIAL	DATA NAME	LEVEL	TYPE	QUANTITY	MODE
4	6.1. DEPARTMENT-TOTAL	1	RECORD	1	35 36 37 38
0.2.	HOURS	2			
0.3.	GROSS	2			
0.4.	WHT.	2			
0.5.	FICA	2			
0.6.	BONDEDUCTION	2			
0.7.	INSURANCE-PREM	2			
0.8.	RETIREMENT	2			
0.9.	PREM	1			
1.0.	NETPAY	2			
1.1.	BOND PURCHASES	2			
1.2.	GRAND-TOTAL	1	COPY	1	DEPARTMENT-TOTAL

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);;
```

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.

```
leq_trans..∈..(m, n, k ∈ N; p ∈ Leq(m, n); q ∈ Leq(n, k)) Leq(m, k) []
leq_trans(_, n, k, leq_0(_), q)..≡..leq_0(k)
leq_trans(_, _, _, leq_succ(m_i, n_i, p_i), leq_succ(_, n, p))..≡..
leq_succ(m_i, n_i, p_i)
```

[x]?
Edit As Text...

p_i..∈..Leq(m_i, n_i)
p..∈..Leq(n_i, n)
leq_succ..∈..(m, n ∈ N;
p ∈ Leq(m, n)) Leq(succ(m), succ(n))
leq_0..∈..(n ∈ N) Leq(0, n)
leq_trans..∈..(m, n, k ∈ N;
p ∈ Leq(m, n);
q ∈ Leq(n, k)) Leq(m, k)

(1) The oldest contradiction of the kind in question is the *Epimenides*. Epimenides the Cretan said that all Cretans were liars, and all other statements made by Cretans were true. Is he a lie? The simplest form of this contradiction is embodied by the man who says "I am lying" if he is lying, he is speaking the truth, and vice versa.

(2) Let w be the class of all those classes which are not members of themselves. Then, whatever class x may be, " x is a w " is equivalent to " x is not a w " is equivalent to " w is not a w " is equivalent to " w is not

(3) Let T be the relation which subsists between two relations R and S whenever R does not have the relation T to S . Then, whatever relations R and S may be, " R has the relation T to S " is equivalent to " R does not have the

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.

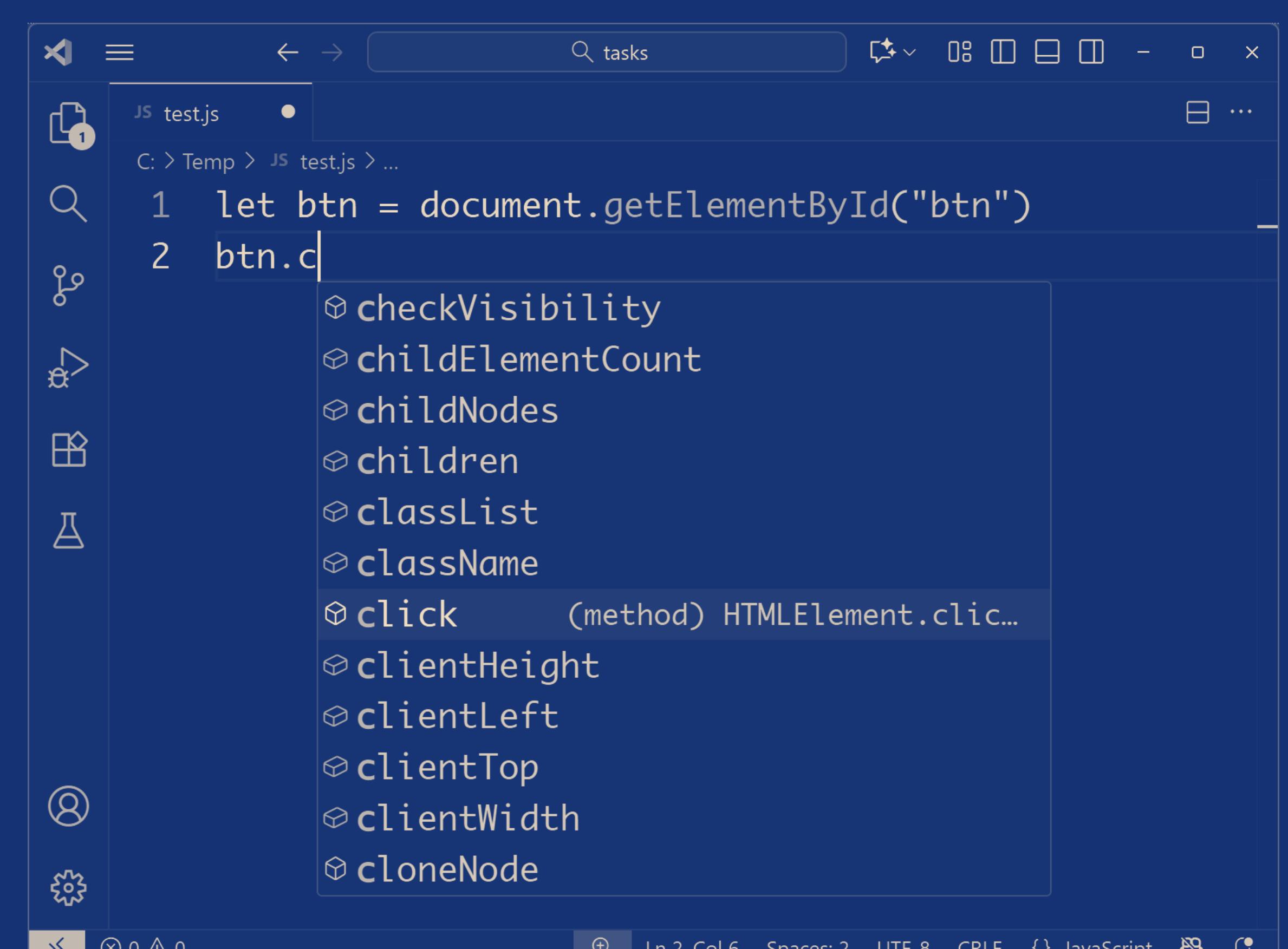
1956: Hacker origins

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



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.



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.