

BY BERTRAND RUSSELL.

Programming with types

The following theory of symbolic logic has been added to the first instance by its ability to solve certain contradictions, of which the one best known to mathematicians is Burali's. The theory in question seems not wholly dependent on this subject recommendation; it has also, if I am not mistaken, a certain philosophical sense which makes it inherently of a kind which much stress should be laid; for common sense is far more fallible than it likes to believe. I shall then show how the theory of logical types effects their solution.

- (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 him were certainly true. Is he a liar? The simplest form of this contradiction is also set by the man who says, "I am lying" if he is lying, he is speaking the truth, and vice versa.
- (2) Let x 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 ". Hence " x is a w " is equivalent to " w is not a w ".
- (3) Let T be the relation which subsists between two relations R and S whenever R does not have the relation R 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 relation R to S ".

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.

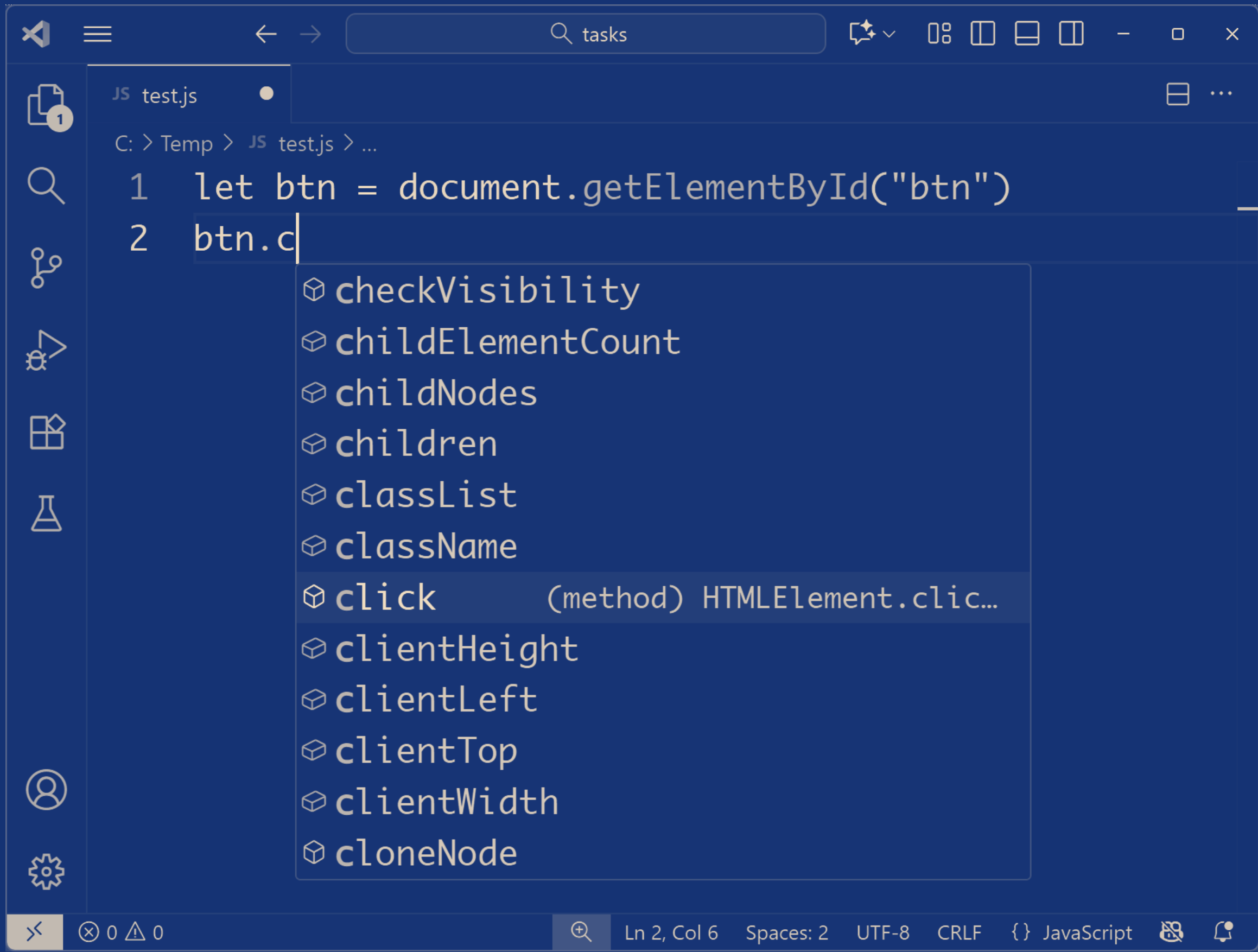
1956: Hacker origins

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.



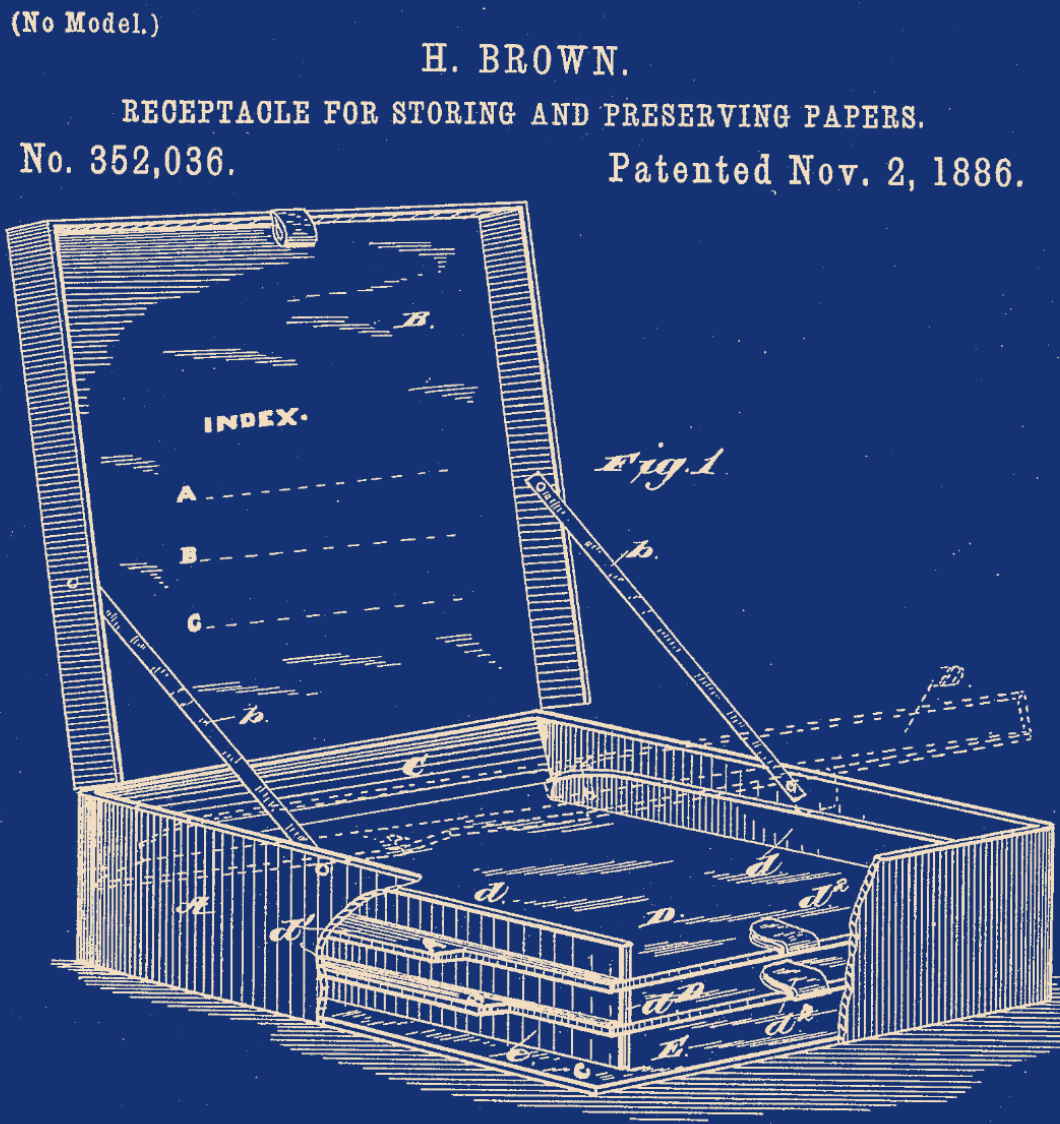
2012: New engineering directions

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.

CTL		PROGRAM	SYSTEM		PAGE	OF	
1	3	PROGRAMMER	DATE	IDENT.	72	80	
SAMPLE PAYROLL							
SERIAL	DATA NAME	UNIT	TYPE	QUANTITY	PRICE	DESCRIPTION	REMARKS
4	5	7					
0.1	DEPARTMENT-TOTAL	1	RECORD				
0.2	MONTH	2				9,999.99	
0.3	MONTHS	2				9(5).999	
0.4	MONTH	2				9(5).999	
0.5	CITY	2				9,999.99	
0.6	MONTHS-OUT-OF	2				9,999.99	
0.7	MONTHS-OUT-OF	2				9,999.99	
0.8	RECEIPTS	2					
0.9	PAY	2				9,999.99	
1.0	MONTH	2				9(5).999	
1.1	MONTHS	2				9(5).999	
1.2	GRAND-TOTAL	1	DOCP			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 = simp term s1 t ;;
let s2 = ss(map (AXIOM `BA`) [¬ ordist` ; ¬ orinv` ; ¬ andide`]) ;;
let t2, th2 = simp term 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_l, n_l, p_l), leq_succ(_ , n, p))..≡..
  leq_succ(m_l, n, p_l)
[x]?
Edit As Text...
p_l..∈..Leq(m_l, n_l)
p..∈..Leq(n_l, 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)
```