A Concrete Syntax Tree (CST) is a tree representation of a source program's syntax that represents the entire syntactical structure of the code, including all the rules, tokens, and even some elements like parentheses and semicolons that are not strictly necessary for understanding the code's semantics.

Key Differences Between AST and CST

Property	Abstract Syntax Tree (AST)	Concrete Syntax Tree (CST)
Nodes	Logical constructs like statements, expressions, loops, etc.	Every element of the code, including tokens and syntax rules
Abstraction	High; ignores many syntax details for semantic understanding	Low; includes all syntax details, even if not semantically significant
Use-Cases	Compilation, code optimization, static analysis, etc.	Syntax highlighting, code formatting, initial parsing stages
Complexity	Lower; fewer nodes and less detail	Higher; more nodes representing even trivial syntax elements

Example in JavaScript

Consider the simple JavaScript code `a = 1 + 2;`

CST Representation

```
                    Program
                      |
                    Statement
                      |
                     =
                    / \
                   a   Expression
                      /     |     \
                     1      +      2
```

Here, even the semicolon and the individual integers are treated as separate nodes in the tree.

AST Representation

```
                     =
                    / \
                   a   +
                      / \
                     1   2
```

In the AST, only the essential elements for understanding the semantics are retained.

When to Use Which

- Use a CST when you need full information about the program's syntax, such as for code formatting or initial parsing stages.
- Use an AST for tasks like code analysis, optimization, or other applications where you are more concerned with the program's semantics rather than its exact syntax.

In summary, a Concrete Syntax Tree provides a more detailed, almost one-to-one mapping of the source code's syntax, while an Abstract Syntax Tree abstracts away many of these details to focus on the code's logical structure.