Tokenization

1. A state machine of tokenization

— Each state consumes one or more characters of the input stream and updates the next state according to those characters.

—The decision is influenced by the current tokenization state and by the tree construction state.

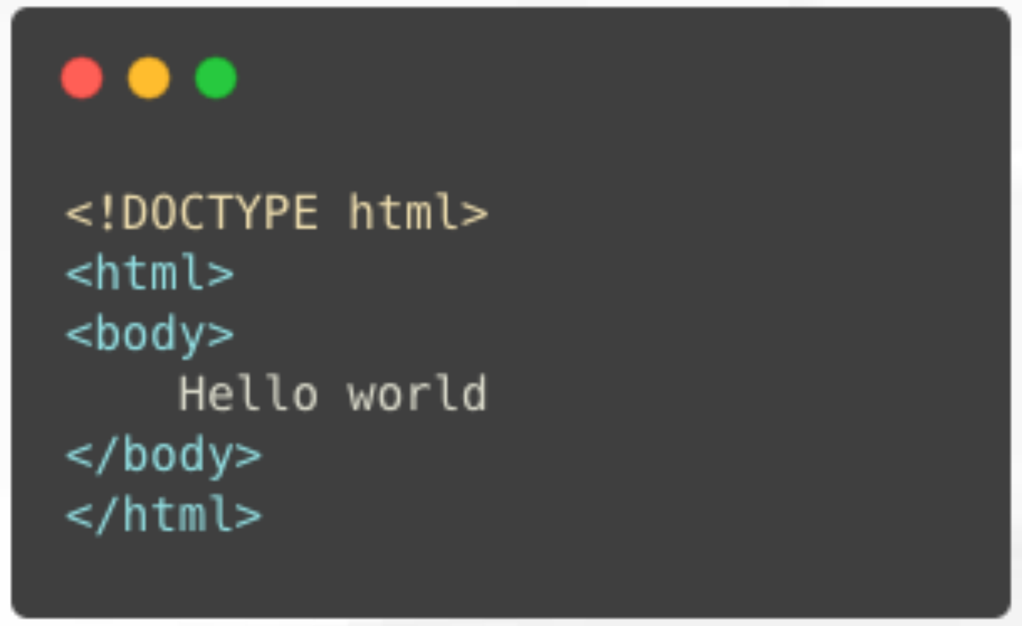
— The same consumed character will yield different results for the correct next state, depending on the current state.

2. An example of tokenization

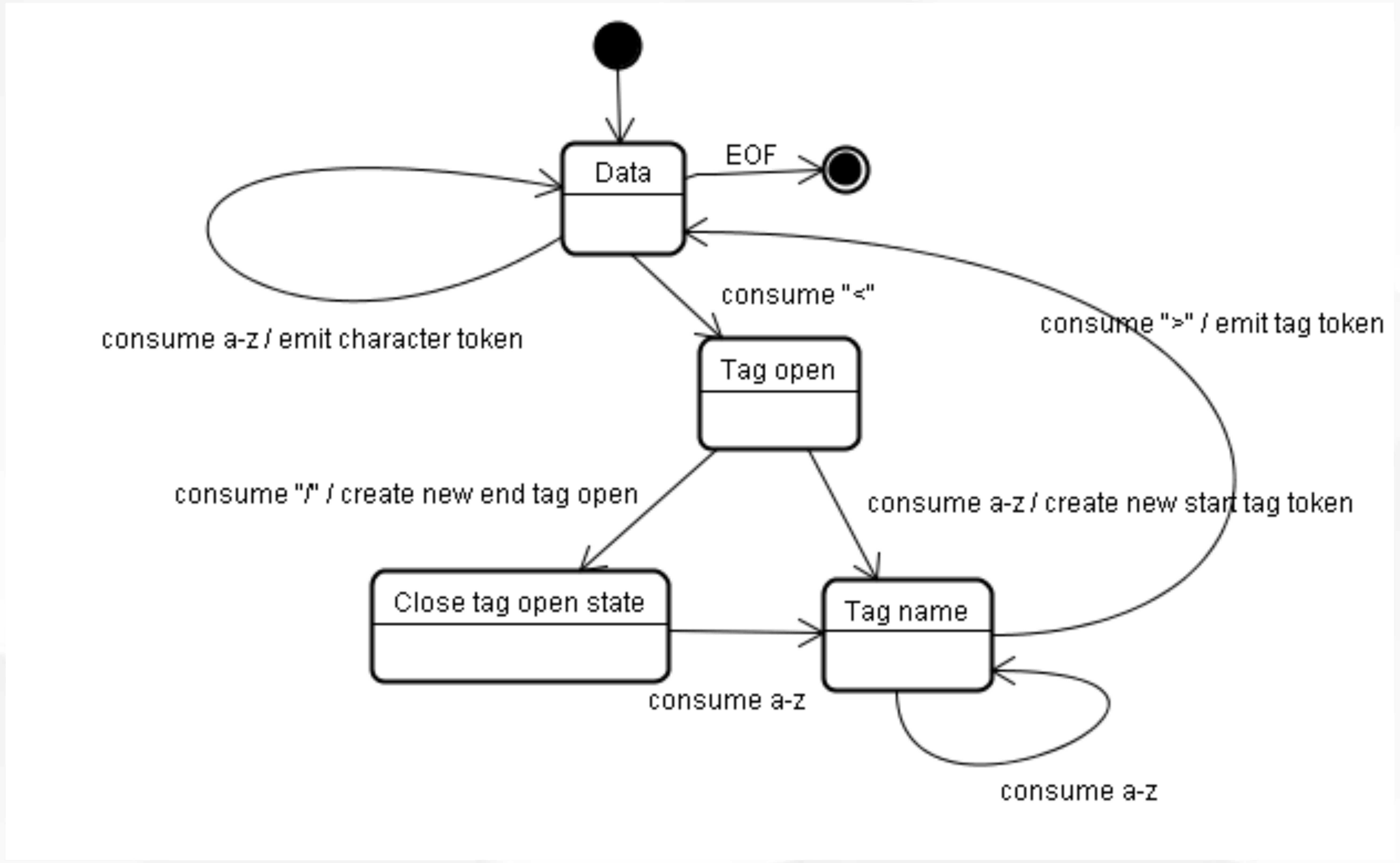
— Tag open state

— close tag open state

— data state



```
<!DOCTYPE html>
<html>
<body>
    Hello world
</body>
</html>
```



Data
EOF
consume a-z / emit character token
consume "<"
consume ">" / emit tag token
Tag open
consume "/" / create new end tag open
consume a-z / create new start tag token
Close tag open state
Tag name
consume a-z
consume a-z

STATE MACHINE OF TOKENIZATION

Tree construction

1. **Attach to Document object**

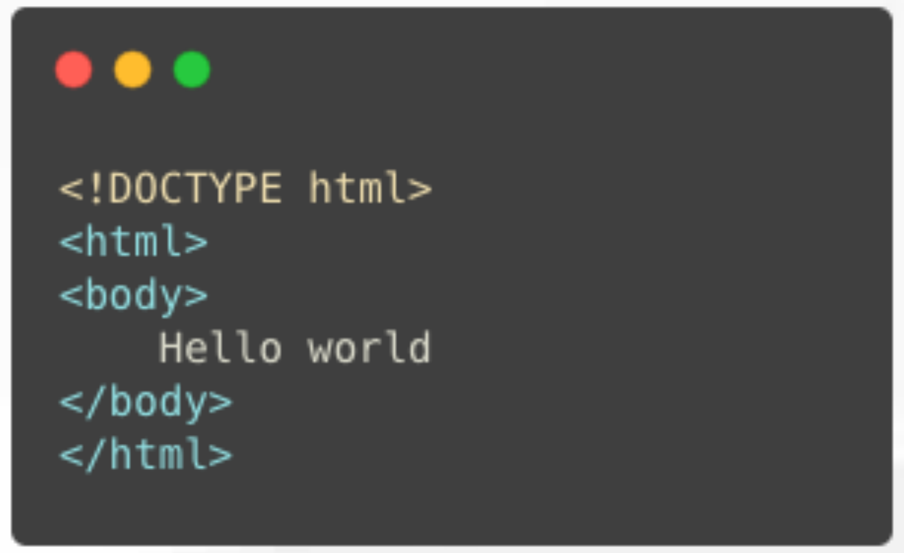
   — When the parser is created the Document object is created.

   — During the tree construction stage the DOM tree with the Document in its root will be modified and elements will be added to it.

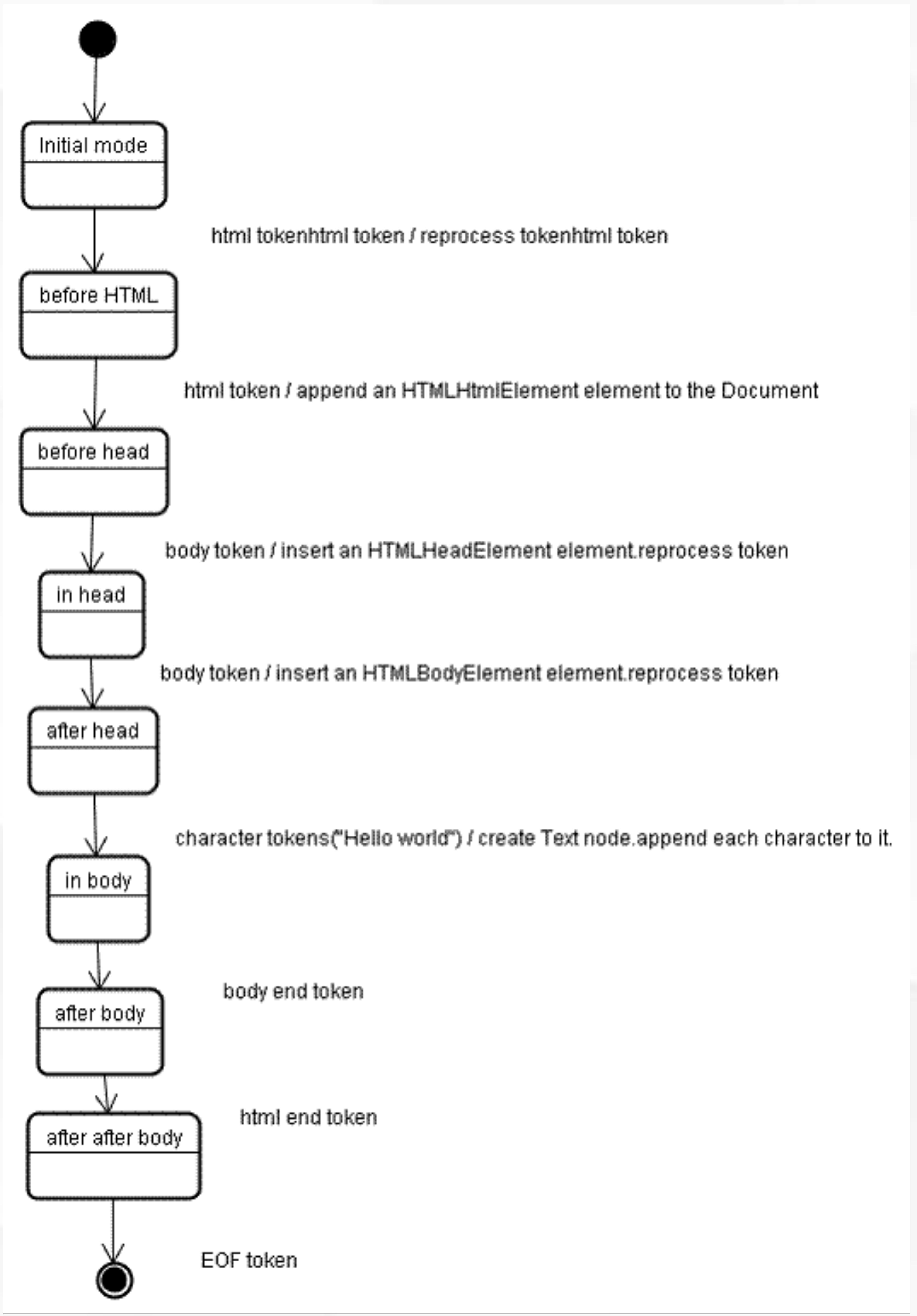
   — Each node emitted by the tokenizer will be processed by the tree constructor. For each token the specification defines which DOM element is relevant to it and will be created for this token. The element is added to the DOM tree, and also the stack of open elements.

   — This stack is used to correct nesting mismatches and unclosed tags. The algorithm is also described as a state machine. The states are called "insertion modes".

2. **Tree construction process for the example input**



```
<!DOCTYPE html>
<html>
<body>
    Hello world
</body>
</html>
```


Initial mode
html tokenhtml token / reprocess tokenhtml token
before HTML
html token / append an HTMLHtmlElement element to the Document
before head
body token / insert an HTMLHeadElement element.reprocess token
in head
body token / insert an HTMLBodyElement element.reprocess token
after head
character tokens("Hello world") / create Text node.append each character to it.
in body
body end token
after body
html end token
after after body
EOF token