


```

        if info.node == nil {
            continue
        }

        let node = info.node!
        activeNode!.addChild(node)
        if !node.tag.selfClosing {
            activeNode = node
        }
    } else if (trim(str) != "") {
        let textNode = TextNode(text: str)
        activeNode?.addChild(textNode)
    }
}
}

```

ترجمة:

```

private func parse() {
    root = HtmlNode(tag: "root")
    var activeNode: InnerNode? = root
    while activeNode != nil {
        let str = content.copyUntil("<")
        if (str == "") {
            let info = parseTag()
            if !info.status {
                activeNode = nil
                continue
            }

            if info.closing {
                let originalNode = activeNode
                while activeNode?.tag.name != info.tag {
                    activeNode = activeNode?.parent
                    if activeNode == nil {
                        activeNode = originalNode
                        break
                    }
                }
            }
        }
    }
}

```

```

        }
    }
    if activeNode != nil {
        activeNode = activeNode?.parent
    }
    continue
}

if info.node == nil {
    continue
}

let node = info.node!
activeNode!.addChild(node)
if !node.tag.selfClosing {
    activeNode = node
}
} else if (trim(str) != "") {
    let textNode = TextNode(text: str)
    activeNode?.addChild(textNode)
}
}
}

```

الكود. `parseTag` لم لف `HtmlNode` تحليل عمليّة من جزء أنه ويبدو `HtmlNode` بلغة مكتوب المقدم الكود ملاحظة:
الإنجيزية. اللغة في هي كما تبقى والممتغيرات الدوال أسماء أن حيث ترجمة إلى يحتاج ل

- `HtmlNode` الجذرية العقدة من التحليل يبدأ: الجذرية العقدة □
- حاليًا. معالجتها يتم التي العقدة يتتبع `activeNode` متغير: النشطة العقدة □
- الحاليّة. العقدة إلى وإضافتها `TextNode` إنشاء يتم نصي، محتوي الكشاف تم إذا: النص محتوي □
- لمعالجتها. `parseTag` طريقة استدعاء يتم وسم، الكشاف تم إذا: الوسم تحليل □

معه. والتعامل المعلومات على التعرف بمعالجة `parseTag` طريقة تقوم `parseTag` المعلومات تحليل

```

private func parseTag() -> ParseInfo {
    var result = ParseInfo()
    if content.char() != ("<" as Character) {

```

```

        return result
    }

    if content.fastForward(1).char() == "/" {
        var tag = content.fastForward(1).copyByToken(Content.Token.Slash, char: true)
        content.copyUntil(">")
        content.fastForward(1)

        tag = tag.lowercaseString
        if selfClosing.contains(tag) {
            result.status = true
            return result
        } else {
            result.status = true
            result.closing = true
            result.tag = tag
            return result
        }
    }
}

let tag = content.copyByToken(Content.Token.Slash, char: true).lowercaseString
let node = HtmlNode(tag: tag)

while content.char() != ">" &&
    content.char() != "/" {
    let space = content.skipByToken(Content.Token.Blank, copy: true)
    if space?.characters.count == 0 {
        content.fastForward(1)
        continue
    }

    let name = content.copyByToken(Content.Token.Equal, char: true)
    if name == "/" {
        break
    }
}

```

```

if name == "" {
    content.fastForward(1)
    continue
}

content.skipByToken(Content.Token.Blank)
if content.char() == "=" {
    content.fastForward(1).skipByToken(Content.Token.Blank)
    var attr = AttrValue()
    let quote: Character? = content.char()
    if quote != nil {
        if quote == "\"" {
            attr.doubleQuote = true
        } else {
            attr.doubleQuote = false
        }
        content.fastForward(1)
        var string = content.copyUntil(String(quote!), char: true, escape: true)
        var moreString = ""
        repeat {
            moreString = content.copyUntilUnless(String(quote!), unless: "=>")
            string += moreString
        } while moreString != ""
        attr.value = string
        content.fastForward(1)
        node.setAttribute(name, attrValue: attr)
    } else {
        attr.doubleQuote = true
        attr.value = content.copyByToken(Content.Token.Attr, char: true)
        node.setAttribute(name, attrValue: attr)
    }
} else {
    node.tag.setAttribute(name, attrValue: AttrValue(nil, doubleQuote: true))
    if content.char() != ">" {
        content.rewind(1)
    }
}

```

```

    }
}

content.skipByToken(Content.Token.Blank)
if content.char() == "/" {
    node.tag.selfClosing = true
    content.fastForward(1)
} else if selfClosing.contains(tag) {
    node.tag.selfClosing = true
}

content.fastForward(1)

result.status = true
result.node = node

return result
}

```

ل تقنية أسماء وهي بالإنجليزية، ودوال متغيرات أسماء على يحتوي لأنه هو كما بالأكود الاحتفاظ تم ملاحظة:
 ذلك. طلب في تتردد فل الأكود، من جزء لأي توضيح أو شرح إلى بحاجة كنت إذا عادةً. ترجمتها يتم

- ❑ إغلاق. علامة أم افتتاحتاحية علامة ال علامة كانت إذا ما تميز الطريقة هذه: العلامات تحدد
- ❑ XmlNode إلى وإضافتها العلامة خصائص بتحويل تقوم: الخصائص
- ❑ الإغلاق. ذاتية العلامات مع صحيح بشكل تتعامل: الإغلاق ذاتية العلامات

الخلاصة

للعقد. شجرية بنية وبناء وحدات، إلى وتقسيمه، محتوى تهيئة في التحليل عمليّة تتضمن
 تمثّل. سلسلة لتقسيم مساعدة وظائف Content فيّة توفر بين عام، بشكل التحليل Dom فيّة تدوير
 هذه العلامات. سمات Tag فيّة تدوير بين م، مستند في والنصوص العنصر XmlNode و TextNode فيّة
 في في لتحويل قوية أداة تتجمل والمنظمة الفعالة الطريقة.