strona 81:

Lekcja 1: Wprowadzenie do sekcji

W tej sekcji zapoznasz się z programowaniem asynchronicznym w JavaScript. Głównym celem jest zorientowanie się, w jaki sposób możemy komunikować się z serwerami za pomocą żądań HTTP, dzięki czemu nasze skrypty mogą pobierać i wysyłać dane.

Lekcja 2: Żądania HTTP z JavaScript

W tej lekcji dowiesz się, jak tworzyć żądania HTTP z kodu JavaScript po stronie klienta. Pozwoli nam to pobrać losowo wygenerowaną łamigłówkę do gry w kata.

XMLHttpRequest

Twoja przeglądarka wysyła żądania HTTP podczas odwiedzania adresu URL. Przeglądarka wysyła żądanie strony HTML i wszelkich zasobów, których potrzebuje strona. Te zasoby mogą obejmować obrazy, pliki JavaScript, pliki CSS i inne.  
Możesz również poprosić przeglądarkę, aby przygotowała dodatkowe żądania HTTP przy użyciu JavaScript. Odbywa się to za pomocą XMLHttpRequest. Chociaż możemy wysyłać i odbierać prawie wszystko, zwykle używamy XMLHttpRequest do wysyłania i odbierania danych JSON.  
Poniższy przykład wykorzystuje !! XMLHttpRequest, aby pobrać nową układankę do gry w kata. Zaczynasz od utworzenia nowego żądania i jego konfiguracji. W tym przypadku wysłaliśmy żądanie GET do http://puzzle.mead.io/puzzle, gdzie możemy uzyskać nową losową układankę.  
Po skonfigurowaniu żądania można go wysłać przy użyciu funkcji send i czekać na odpowiedź, nasłuchując zdarzenia "readystatechange °." Readystate ma wartość 4, gdy żądanie jest gotowe, a odpowiedź jest dostępna. Tutaj można przeanalizować JSON do obiektu JavaScript i zrobić coś z tymi informacjami.

strona 82:

// Making an HTTP request

const request = new XMLHttpRequest()

request.addEventListener('readystatechange', (e) => {

**if** (e.target.readyState === 4) {

const data = JSON.parse(e.target.responseText)

console.log(data) // Will print a new random puzzle

}

})

request.open('GET', 'http://puzzle.mead.io/puzzle')

request.send()

Lekcja 3: Nagłówki i błędy HTTP

Podczas tej lekcji zapoznasz się ze szczegółami żądania HTTP. Przeanalizujesz żądanie, aby zobaczyć, jak wymieniane są żądania i odpowiedzi.

Kody stanu HTTP

Każda odpowiedź HTTP zawiera kod statusu. Kod statusu to liczba, która oznacza, że dane żądanie zakończyło się powodzeniem. Jako przykład, kod stanu 200 reprezentuje sukces, podczas gdy kod stanu 400 oznacza żądanie HTTP zawierające błąd. Sprawdź poniższy link do dokumentacji, aby uzyskać pełną listę kodów statusu. Kod statusu dla żądań HTTP zależy od właściwości statusu. Poniższy kod sprawdza, czy status wynosił 200, zanim założy się, że żądanie zostało pomyślnie zakończone. Jeśli kod stanu wynosił 200, odpowiedź jest analizowana, a obiekt jest drukowany. Jeśli kod stanu nie miał 200, i komunikat o błędzie został wydrukowany.

// Making an HTTP request

const request = new XMLHttpRequest()

request.addEventListener('readystatechange', (e) => {

**if** (e.target.readyState === 4 && e.target.status === 200) {

const data = JSON.parse(e.target.responseText)

console.log(data)

} **else if** (e.target.readyState === 4) {

console.log('An error has taken place')

}

})

request.open('GET', 'http://puzzle.mead.io/puzzle?wordCount=3')

request.send()

Lekcja 3: Nagłówki i błędy HTTP

Podczas tej lekcji zapoznasz się ze szczegółami żądania HTTP. Przeanalizujesz żądanie, aby zobaczyć, jak wymieniane są żądania i odpowiedzi.

Kody stanu HTTP

Każda odpowiedź HTTP zawiera kod statusu. Kod statusu to liczba, która oznacza, że dane żądanie zakończyło się powodzeniem. Jako przykład, kod stanu 200 reprezentuje sukces, podczas gdy kod stanu 400 oznacza żądanie HTTP zawierające błąd. Sprawdź poniższy link do dokumentacji, aby uzyskać pełną listę kodów statusu. Kod statusu dla żądań HTTP zależy od właściwości statusu. Poniższy kod sprawdza, czy status wynosił 200, zanim założy się, że żądanie zostało pomyślnie zakończone. Jeśli kod stanu wynosił 200, odpowiedź jest analizowana, a obiekt jest drukowany. Jeśli kod stanu nie miał 200, a komunikat o błędzie został wydrukowany.