**9.7. Wspólnie napiszmy grę w JavaScripcie**

W tym rozdziale stworzysz swoją pierwszą prostą grę. Będzie to klasyczna gra w kamień, papier i nożyce - wszyscy ją doskonale znamy. Jest ona przy tym idealnym przykładem, w którym możemy wykorzystać poznane wcześniej możliwości.

Do budowy interfejsu skorzystamy ze znanych nam rozwiązań, czyli Font Awesome oraz Twitter Bootstrap.

**Założenia gry**

Założenia są takie, jak w klasycznej grze "Kamień, papier i nożyce":

* Gra ma się składać z dowolnej liczby rund. Koniec rozgrywki następuje, gdy jeden z zawodników zdobędzie 10 punktów. Remis nie daje punktów, a wygrana daje 1 punkt dla zwycięzcy danej rundy.
* Tuż po uruchomieniu strony gracz ma zobaczyć tytuł gry oraz przycisk 'New Game'.
* Po kliknięciu przycisku 'New Game', użytkownik zostanie poproszony o podanie swojego imienia.
* Następnie gra zostanie uruchomiona, a na ekranie zamiast przycisku 'New Game' pojawią się przyciski 'Rock', 'Paper', 'Scissors' oraz tabela przedstawiająca wybór graczy (kto co wybrał), a także rezultat danej rundy.
* W tabeli mają zostać wyświetlone imię gracza oraz przeciwnika, w tym wypadku ma to być to 'Computer', oraz uzyskany wynik.
* Po zdobyciu przez któregoś gracza 10 punktów zamiast przycisków 'Rock', 'Paper', 'Scissors' ma pojawić się napis "The winner is [nazwaGracza]", a poniżej przycisk 'Play again' o takiej samej funkcjonalności jak przycisk 'New Game'.

**Mechanika gry**

Zbiór elementów, czyli tabela graczy, dostępne przyciski i wynik rezultatu, jest stały.

Wszystkie elementy, które będą wyświetlane na stronie, będą istnieć przez cały czas rozgrywki, jednak będą ukrywane lub wyświetlane zależnie od potrzeb. Dlatego HTML można zakodować od razu.

W czasie rozgrywki będziemy nanosić aktualny wynik gry oraz pobierać wybór gracza za pomocą zdarzeń oraz modyfikacji elementów .

Wybór komputera będzie losowany za pomocą wbudowanej funkcji 'random' z puli możliwych wyborów.

Zamiast dodawać dwa osobne przyciski o tej samej funkcjonalności, po prostu zmienimy wyświetlany napis w tym samym elemencie.

**Struktura HTML**

Na początek stwórz nowy branch od brancha 'master'. Pamiętaj, żeby zadanie wykonywać jako osobny projekt.

**Dołączenie zewnętrznych arkuszy stylów**

W pliku 'index.html' w sekcji '' załącz potrzebne pliki stylów Bootstrapa oraz FontAwesome.

<head>

<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" rel="stylesheet" />

<link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.6.3/css/font-awesome.min.css" rel="stylesheet" />

</head>

Copy

Kolej na sekcję <body>.

**Tytuł gry**

<div class="container">

<h3 class="text-center">

Welcome to the game

<span class="label label-default">Rock</span>

<span class="label label-info">Paper</span>

<span class="label label-danger">Scissors</span>

</h3>

</div>

Copy

**Przycisk zainicjowania nowej gry**

To kluczowy element - służy on do uruchamiania rozgrywki.

Całemu elementowi zostaje nadane id js-newGameElement, aby móc odwołać się do niego z poziomu JavaScriptu.

Jest to ważne, ponieważ chcemy ukrywać ten element po rozpoczęciu rozgrywki. Dodatkowo, nasz przycisk ma wykonywać pewne czynności po jego kliknięciu(zdarzenie click), dlatego podpinamy do niego zdarzenie click. Lepiej jest to robić w kodzie Javascript niż poprzez atrybut onclick.

Wartość napisu będzie się zmieniać, dlatego potrzebujemy tutaj mieć również id, do którego odwołamy się przez JavaScript.

<div id="js-newGameElement" class="container">

<div class="row text-center">

<button id="js-newGameButton" class="btn btn-default">

New game

</button>

</div>

</div>

Copy

Kod Javascript:

var newGameBtn = document.getElementById('js-newGameButton');

newGameBtn.addEventListener('click', newGame);

Copy

**Wybór gracza**

Ten element zawiera przyciski wyboru gracza. Po kliknięciu gracza musimy przekazać informację, co zostało wybrane.

Każdy z elementów będzie miał podpięty listener click, który wywoła funkcję playerPick z odpowiednim parametrem.

Cały element również posiada unikalne id js-playerPickElement, aby można było go łatwo znaleźć za pomocą DOM API JavaScriptu.

<div id="js-playerPickElement" class="container">

<div class="row text-center">

<button id="js-playerPick\_rock" class="btn">

<span class="badge">

<i class="fa fa-hand-rock-o" aria-hidden="true"></i>

</span>

Rock

</button>

<button id="js-playerPick\_paper" class="btn">

<span class="badge">

<i class="fa fa-hand-paper-o" aria-hidden="true"></i>

</span>

Paper

</button>

<button id="js-playerPick\_scissors" class="btn">

<span class="badge">

<i class="fa fa-hand-scissors-o" aria-hidden="true"></i>

</span>

Scissors

</button>

</div>

</div>

Copy

Każdy z przycisków wyboru gracza posiada unikalną klasę. Należy jeszcze wskazać, co będzie działo się po ich kliknięciu.

var pickRock = document.getElementById('js-playerPick\_rock'),

pickPaper = document.getElementById('js-playerPick\_paper'),

pickScissors = document.getElementById('js-playerPick\_scissors');

pickRock.addEventListener('click', function() { playerPick('rock') });

pickPaper.addEventListener('click', function() { playerPick('paper') });

pickScissors.addEventListener('click', function() { playerPick('scissors') });

Copy

W powyższym kodzie, ustawimy listenery na każdy z przycisków. Po jego kliknięciu zostanie wywołana funkcja playerPick z parametrem reprezentującym wybór gracza.

**Tabela wyników**

W tym elemencie wyświetlamy aktualny stan gry - zarówno wybór gracza, jak i punktację oraz wynik pojedynczej rundy.

Tutaj będziemy aktualizować poszczególne elementy tabeli wyników, dlatego każdy z elementów, który będziemy aktualizować, otrzymał unikalne id.

<div id="js-resultsTableElement" class="container">

<div class="row">

<div class="col-xs-5 text-center">

<span id="js-playerPoints" class="badge">

0

</span>

<span id="js-playerName">

Player Name

</span>

</div>

<div class="col-xs-2 text-center">

VS.

</div>

<div class="col-xs-5 text-center">

<span id="js-computerPoints" class="badge">

0

</span>

Computer

</div>

</div>

<div class="row">

<div class="col-xs-5 text-center">

<span id="js-playerPick">

Player selection

</span>

</div>

<div class="col-xs-2"></div>

<div class="col-xs-5 text-center">

<span id="js-computerPick">

Computer selection

</span>

</div>

</div>

<div class="row">

<div class="col-xs-5 text-center">

<span id="js-playerResult">

Player Score

</span>

</div>

<div class="col-xs-2"></div>

<div class="col-xs-5 text-center">

<span id="js-computerResult">

Computer Score

</span>

</div>

</div>

</div>

Copy

**Czego się teraz spodziewać**

Na razie mamy stworzoną strukturę HTML + CSS (dzięki dołączonym stylom), ale nie napisaliśmy jeszcze funkcjonalności.

Chciałbym, żebyśmy zadbali jeszcze o odstępy pomiędzy kontenerami. Dlatego na kontenery nadaj odpowiednie style (np. padding).

U mnie wygląda to tak:

[](https://kodilla.com/static/bootcamp/8-13-rock-paper.png)

**Logika gry**

Po pierwsze, aktualny wynik gry oraz jej stan należy gdzieś przechowywać.

**Wartości początkowe**

Najpierw zainicjujmy wartości, których będziemy używać w grze i nadajmy im wartości początkowe.

var gameState = 'notStarted', //started // ended

player = {

name: '',

score: 0

},

computer = {

score: 0

};

Copy

Ustalamy stan gry na notStarted. Stan gry będzie nam służył do ustalenia, które kontenery należy w danej chwili wyświetlać.

Poza tym przechowujemy obiekt *player*, który będzie trzymał nazwę gracza oraz aktualny wynik, a także obiekt *computer*. Jednak ten posiada tylko właściwość *score* do trzymania wyniku, bo jego nazwa jest stała.

**Wyświetlanie elementów gry**

Każdemu z kontenerów gry nadane zostało unikalne id - czas, aby na coś się przydało :)

var newGameElem = document.getElementById('js-newGameElement'),

pickElem = document.getElementById('js-playerPickElement'),

resultsElem = document.getElementById('js-resultsTableElement');

Copy

W ten sposób tworzymy zmienne, które będą wskazywać na elementy gry, a konkretnie jej poszczególne części.

Wcześniej zdefiniowaliśmy zmienną gameState. Decydujemy, że może ona przyjąć kilka wartości - zależnie od tego, czy gra nie została jeszcze rozpoczęta, jest w trakcie czy została zakończona chcemy wyświetlić różne elementy na stronie.

Dlaczego nie stworzyć na to funkcji?

function setGameElements() {

switch(gameState) {

case 'started':

newGameElem.style.display = 'none';

pickElem.style.display = 'block';

resultsElem.style.display = 'block';

break;

case 'ended':

newGameBtn.innerText = 'Jeszcze raz';

case 'notStarted':

default:

newGameElem.style.display = 'block';

pickElem.style.display = 'none';

resultsElem.style.display = 'none';

}

}

Copy

Zależnie od stanu zmiennej gameState zostaną wykonane różne polecenia.

Kiedy gra jest "started", nie chcemy widzieć kontenera do rozpoczęcia gry, a jedynie możliwość wyboru przedmiotu i tabele rezultatów.

Kiedy gra będzie zakończona, nasz przycisk rozpoczynania gry ma zmienić napis na "Play again", a następnie być wyświetlony i ukryć pozostałe elementy gry (zauważ brak instrukcji break;).

W przypadku kiedy gra jest "notStarted" (dzieje się tak tylko przed rozpoczęciem gry), przycisk ma zachować stan "Nowa gra" i być widoczny, a pozostałe elementy ukryte.

Skoro funkcja już jest, pozostaje nam tylko ją wywołać.

setGameElements();

Copy

Sprawdź, jak zachowuje się gra, kiedy nadasz zmiennej gameState różne wartości.

**Rozpoczęcie gry**

Teraz należy zdefiniować funkcję newGame, która będzie odpowiadać za rozpoczęcie każdej gry.

var playerPointsElem = document.getElementById('js-playerPoints'),

playerNameElem = document.getElementById('js-playerName'),

computerPointsElem = document.getElementById('js-computerPoints');

Copy

Na początek zdefiniujmy zmienne odnoszące się do tych elementów na stronie, które będziemy aktualizować przed rozpoczęciem rozgrywki. Po kolei jest to punktacja gracza, imię gracza wyświetlane na stronie oraz punktacja komputera.

Dalej definiujemy funkcję, która będzie uruchamiana po wciśnięciu przycisku "New Game" / "Play Again"

function newGame() {

player.name = prompt('Please enter your name', 'imię gracza');

if (player.name) {

player.score = computer.score = 0;

gameState = 'started';

setGameElements();

playerNameElem.innerHTML = player.name;

// setGamePoints(); // This function has not been created yet

}

}

Copy

Najpierw pobieramy imię gracza i zapisujemy je w naszym obiekcie.

Następnie sprawdzamy, czy użytkownik podał imię. Jeśli pole było puste lub zostało kliknięte "Anuluj", instrukcja warunkowa nie zostanie wykonana, a zatem gra się nie rozpocznie.

Jeśli gracz prawidłowo podał imię i kliknął OK, wyniki zostają wyzerowane, status gry zmieniony, a następnie wywołana funkcja, która ustala, co w tym momencie powinno być widoczne na ekranie gry.

Ostatnia instrukcja służy do wpisania imienia gracza w tablicy wyników na stronie.

Zdefiniowaliśmy także funkcję do ustalania wyniku, jednak na razie pozostawmy ją zakomentowaną. Punkty w obiektach zostały wyzerowane, jednak będziemy musieli zadbać, aby tak stało się również w elementach strony.

**Wybór gracza**

Zdefiniujmy teraz funkcję, która odpowiada za pobranie wyboru gracza.

function playerPick(playerPick) {

console.log(playerPick);

}

Copy

Na razie wybór wyświetlamy jedynie w konsoli. Funkcja playerPick wywołana zostanie za każdym razem, kiedy użytkownik kliknie przycisk, a więc połowa za nami.

**Losowanie wyboru komputera**

Kiedy określiliśmy już, co wybrał gracz, trzeba napisać funkcję losującą wybór komputera.

Jak tego dokonać? Przy użyciu obiektu Math. Zawiera on w sobie, jak sama nazwa wskazuje, metody matematyczne. Pierwsza, jaką poznamy to Math.random.

Jeżeli przypiszemy ją do zmiennej, metoda ta zwraca nam losową liczbę zmiennoprzecinkową:

var x = Math.random();

Copy

to x może wynosić przykładowo:

0.9200883881643283

Copy

Jednak to tylko liczba, a musimy wybrać losowo jedną z trzech opcji.

By to zrobić, nakażemy komputerowi wylosować jedną liczbę, a następnie pomnożymy ją przez 3. Potem skorzystamy z kolejnej metody matematycznej: Math.floor, która zaokrągla liczby zmiennoprzecinkowe w dół, do liczby całkowitej.

Poniższy zapis zwróci nam losową liczbę całkowitą z przedziału od 0 do 2 (czyli 0, 1 lub 2):

Math.floor(Math.random()\*3)

Copy

Jest to bardzo popularna metoda na generowanie losowych wyborów.

Teraz każdej z liczb przypiszemy odpowiednio kamień, papier lub nożyce. Ciekawym rozwiązaniem będzie zwrócenie wybranego elementu tablicy.

function getComputerPick() {

var possiblePicks = ['rock', 'paper', 'scissors'];

return possiblePicks[Math.floor(Math.random()\*3)];

}

Copy

Mamy wybór gracza oraz komputera - skoro już potrafimy otrzymać te dwie rzeczy, umieśćmy ten wybór na stronie.

var playerPickElem = document.getElementById('js-playerPick'),

computerPickElem = document.getElementById('js-computerPick'),

playerResultElem = document.getElementById('js-playerResult'),

computerResultElem = document.getElementById('js-computerResult');

Copy

function playerPick(playerPick) {

var computerPick = getComputerPick();

playerPickElem.innerHTML = playerPick;

computerPickElem.innerHTML = computerPick;

}

Copy

Jeśli w tej chwili uruchomisz grę, powinieneś móc ją rozpocząć, a po wybraniu przycisku rock, paper lub scissors zobaczyć w tabeli wybór gracza oraz wybór komputera.

**Logika gry i przyznawanie punktów**

Kamień pokonuje nożyce, nożyce papier, a papier kamień. Remis to oczywiście remis :)

Moglibyśmy wyszczególnić tutaj wszystkie możliwe sytuacje za pomocą instrukcji warunkowych. Co jeśli gracz wybrał kamień, a komputer nożyce lub papier? A może jest remis, bo obaj gracze dokonali tego samego wyboru?

Bądźmy *spryciarzami*.

Na początek załóżmy, że to my wygraliśmy, a potem sprawdźmy czy to prawda.

function checkRoundWinner(playerPick, computerPick) {

playerResultElem.innerHTML = computerResultElem.innerHTML = '';

var winnerIs = 'player';

if (playerPick == computerPick) {

winnerIs = 'noone'; // remis

} else if (

(computerPick == 'rock' && playerPick == 'scissors') ||

(computerPick == 'scissors' && playerPick == 'paper') ||

(computerPick == 'paper' && playerPick == 'rock')) {

winnerIs = 'computer';

}

if (winnerIs == 'player') {

playerResultElem.innerHTML = "Win!";

player.score++;

} else if (winnerIs == 'computer') {

computerResultElem.innerHTML = "Win!";

computer.score++;

}

}

Copy

Na początku usuwamy wyświetlany tekst o wygranej któregoś z graczy. Jeszcze nie wiadomo, kto wygrał, będziemy to dopiero sprawdzać.

Funkcja otrzymuje wybory obu graczy. Na początek zakładamy, że to my wygraliśmy rundę - dzięki temu nie musimy tego sprawdzać, a jedynie potwierdzić.

Jeśli był remis, bo obaj gracze wybrali to samo, to wartość winnerIs zostanie nadpisana i zwrócona.

Jeśli nie było remisu, to sprawdzamy, czy komputer trafił w którąś z możliwości jego wygranej. Jeśli tak, to wartość winnerIs zostanie nadpisana i zwrócona.

Jeśli te sytuacje nie miały miejsca, to znaczy, że wygrana musi być nasza.

Na koniec funkcji dodajemy punkty zwycięzcy i stosowną informację w tabeli wyników.

Nasza funkcja powinna wywoływać się za każdym wyborem gracza. W tej chwili powinna wyglądać następująco.

function playerPick(playerPick) {

var computerPick = getComputerPick();

playerPickElem.innerHTML = playerPick;

computerPickElem.innerHTML = computerPick;

checkRoundWinner(playerPick, computerPick);

}

Copy

**Aktualizacja wyniku**

Teraz dodajemy wartości. Wynik gry znajduje się wewnątrz obiektów player oraz computer - brakuje tylko wyświetlania go na stronie.

Stwórzmy funkcję, która będzie za to odpowiedzialna.

function setGamePoints() {

playerPointsElem.innerHTML = player.score;

computerPointsElem.innerHTML = computer.score;

}

Copy

Ta krótka i prosta funkcja aktualizuje stan wyniku na elementach strony, pamiętasz? Dodawaliśmy ją w funkcji newGame() - proszę, odkomentuj ją teraz.

**Zakończenie rozgrywki**

Przyda nam się jeszcze funkcja, która sprawdzi, czy któryś z graczy zdobył 10 punktów.

Jednak to zostawiam już w pełni Tobie. Powodzenia!

**Potencjalne problemy**

Jeśli napotkasz jakiś problem, to po pierwsze, sprawdź, czy masz jakieś błędy w konsoli.

Jeśli chcesz sprawdzić, czy jakaś funkcja się uruchamia, wrzuć tam console.log - być może zdarzyło Ci się zapomnieć o jej wywołaniu.

Gra nie działa tak, jak powinna? Ponownie użyj console.log i sprawdź wartości, jakie w danej chwili posiadają zmienne.

**Podsumowanie rozdziału**

Uff, udało Ci się! To solidny kawał wiedzy. Z pewnością zanim będziesz mógł pisać skrypty z pamięci, przyda Ci się jeszcze sporo praktyki - na to potrzeba czasu.

Jednak nie ma wątpliwości, że stworzyłeś swoją pierwszą grę - jestem z Ciebie dumny! To zadanie z pewnością przyda Ci się w portfolio, szczególnie jeśli będziesz umiał powiedzieć, co się dzieje w poszczególnych fazach gry.

**Gratulacje!**

**Zadanie: Kamień, papier, nożyce!**

Dopisz kod odpowiadający za zakończenie rozgrywki. Wyświetl informację o zwycięzcy.

Stwórz nowe repozytorium i wyślij tam swoje zadanie, a link do niego wklej poniżej. Pamiętaj o udostępnieniu repozytorium swojemu mentorowi.