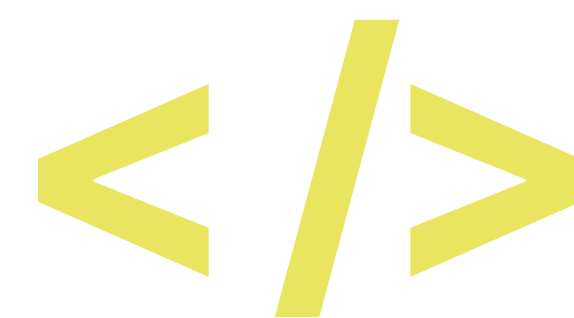


za<pro/>gramowani.com

</> webmaster



kontakt

E-mail: hello@filiposinski.com

Discord: <https://discord.gg/eyhAtuxJcv>

GitHub: github.com/zaprogramowaniFO

</> Zajęcia nr 6

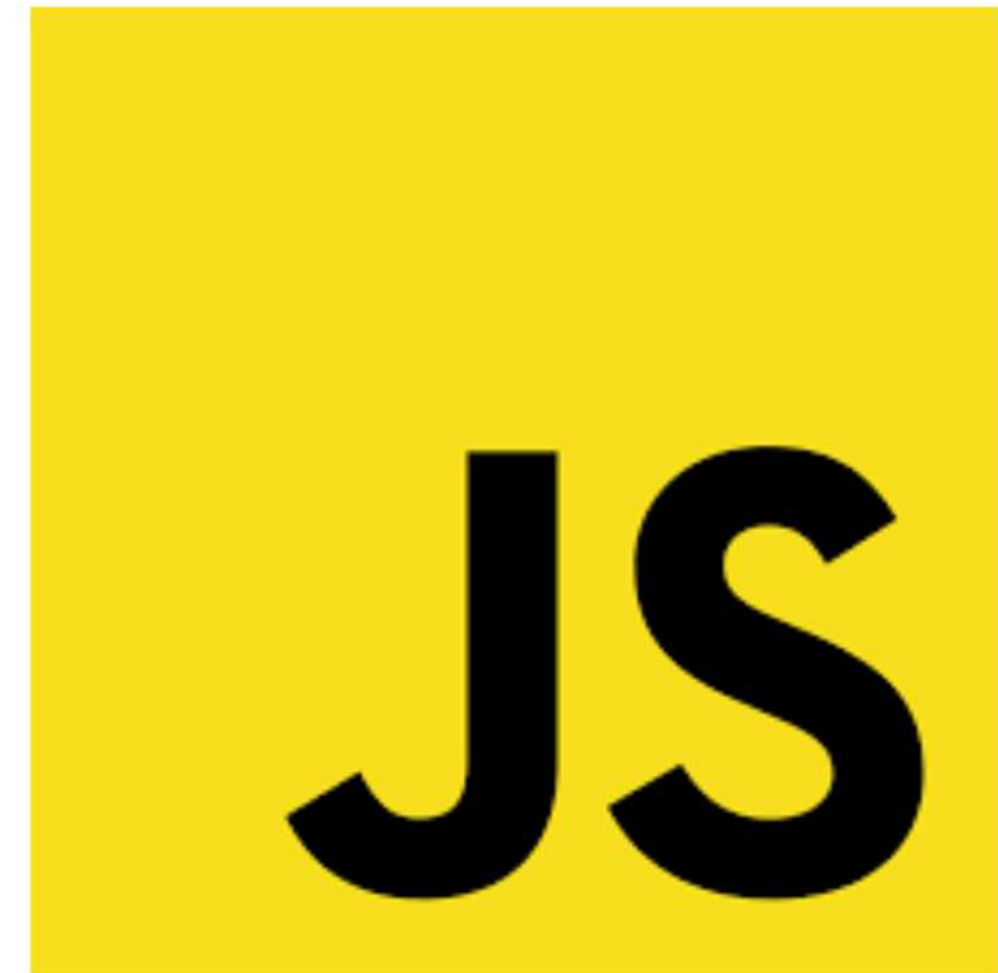
2023/11/28

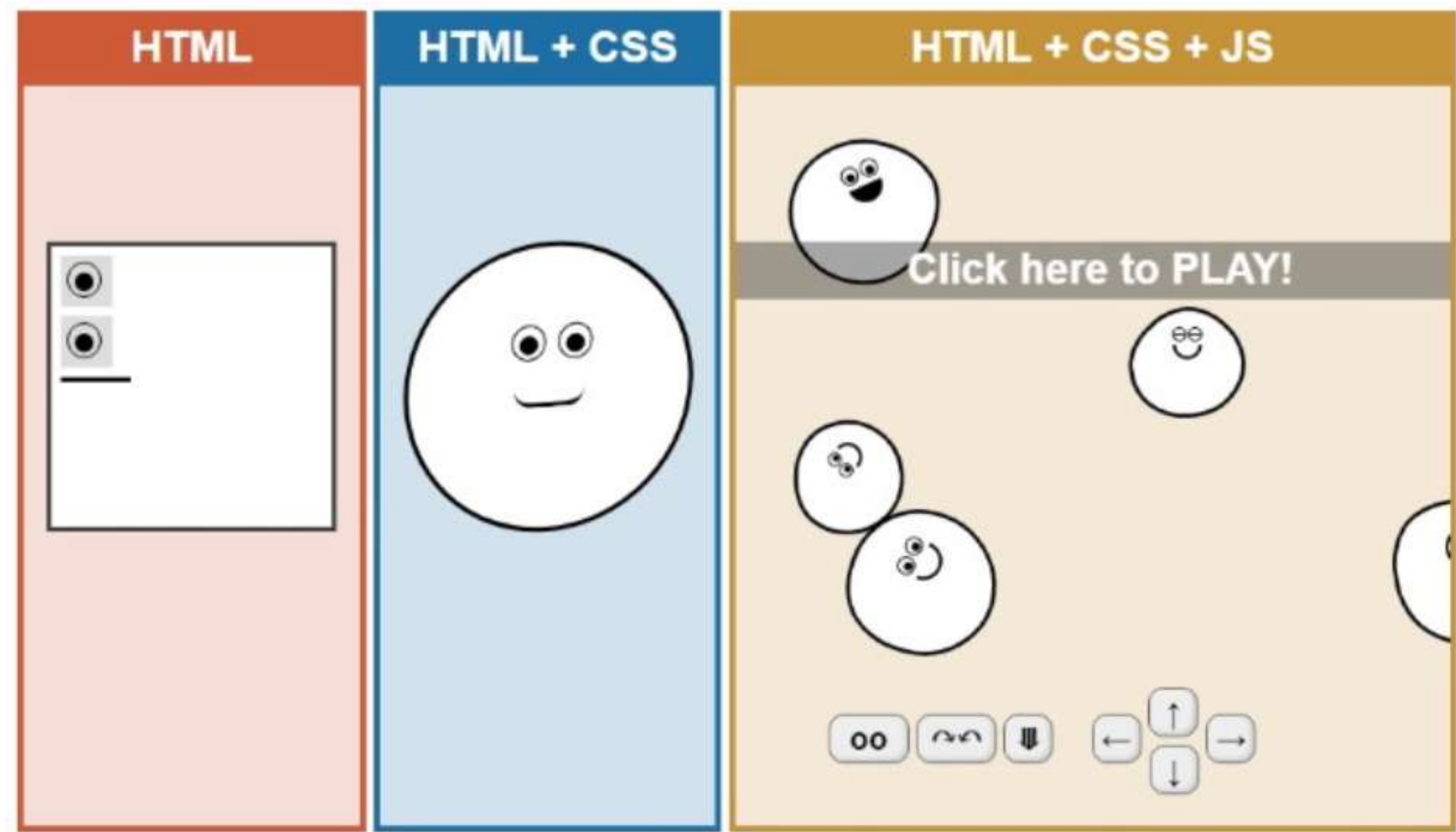
</> JAVASCRIPT

JavaScript

Jest to skryptowy język programowania.
Wykorzystywany podczas pisania stron internetowych.

Pozwala na stworzenie interakcji pomiędzy użytkownikiem, a stroną internetową.





< /> Java != JavaScript



Java

- Bardziej backend
- Różne aplikacje
- Do pisania potrzebujemy JDK
- Kod musi być skompilowany
- Kod jest ukryty po skompilowaniu
- Samodzielny język
- Język silnie typowany

JavaScript

- Bardziej frontend
- Głównie aplikacje webowe
- Do pisania wystarczy notatnik
- Kod od razu gotowy
- Kod jest dostępny dla każdego
- Do działania potrzebny HTML
- Język słabo typowany

Zasady tworzenia arkusza oraz przypięcia go do projektu

A dark-themed code editor window with three window control buttons (white, grey, red) in the top right corner. Inside the editor, the text `<script src="ścieżka pliku"></script>` is written in a pink/magenta color.

```
<script src="ścieżka pliku"></script>
```

Aby utworzyć arkusz **JS** i móc go używać należy:

1. Utworzyć nowy arkusz o rozszerzeniu **.js**
2. Arkusz należy “przypiąć” do pliku strony z rozszerzeniem **html**

Zrobimy to dodając odpowiednią linijkę do naszego kodu strony na samym końcu tuż przed znacznikiem **</body>**

Podobnie jak **CSS**, skrypt możemy pisać bezpośrednio w pliku **HTML**, pomiędzy znacznikami **<script>** **</script>**, ale nie jest to poprawna praktyka.



PODSTAWOWE FUNKCJE

Podstawowe funkcje

`alert('Hello World!');`

funkcja, która wywołuje okno alertu na stronie

`console.log('Hello World!');`

funkcja, która drukuje w konsoli treść zawartą w argumencie

Poniższe funkcje dziś już nie stosowane:

`prompt('tekst');`

funkcja, która wywołuje okno z możliwością wprowadzenia treści, np. spytać użytkownika o imię, i przypisać do zmiennej

`confirm('tekst');`

funkcja, która działa podobnie do alert, ale zwraca wartość logiczną, którą można przypisać do zmiennej i później wykorzystać



Zmienne w JavaScript

Zmienna to **“pudełko”**, które przechowuje wartości/informacje do niej przypisane.

Każda stworzona zmienna musi mieć swoją unikatową nazwę (identyfikator).

JavaScript to język słabo typowany.

Do zmiennej możemy przypisać **dowolny typ danych**, w zależności od potrzeb.

Mogą to być np. **liczby całkowite, zmiennoprzecinkowe, teksty, wartości logiczne.**

Typy zmiennych

//inicjowanie zmiennej

typ nazwa = wartość;

const - zmienna stała, nie można później zmienić wartości;

let - zmienna do których można przypisywać wielokrotnie różne wartości;

var - zmienna ogólna, dziś już nie używana ale spotykana, podobna do let;

Nazwa ma znaczenie

Wielkość liter ma znaczenie;

Nazwa zmiennej nie może zaczynać się od cyfry;

Nazwa zmiennej nie może zawierać spacji, kropki, przecinka, myślnika;

Nazwa zmiennej nie może być “słowem kluczowym” w JavaScript

Nazywaj zmienne tak aby nazwa zmiennej miała sens i odnosiła się do wartości
(np. zmienna liczbowa **“number”**, a nie **“aaaaa”**;

Używaj zasady **camelCase**;

Dobłą praktyką jest **nazywanie zmiennych po angielsku**;

Typy danych

```
let age = 20;  
let name = "Adam";  
let isTrue = true;  
let variable;  
let test = null;
```

Number - wartość liczbowa;

String - wartość znakowa;

Boolean - wartość logiczna (prawda lub fałsz);

Undefined - wartość domyślna;

Null - wartość bez wartości (typ wyświetli się jako obiekt);

Łączenie wyświetlania tekstu i wartości zmiennej

```
const name = 'Adam';  
let age = 20;  
console.log(`Mam na imię ${name} i mam ${age} lat!`);
```

Możemy w łatwy sposób wyświetlać zarówno tekst jak i wartości przypisane do zmiennej jednocześnie przy pomocy pojedynczej funkcji.

</> OPERATOR

Operatory

+ dodawanie

- odejmowanie

***** mnożenie

/ dzielenie

****** potęgowanie

% modulo

++ inkrementacja

-- dekrementacja



INSTRUKCJE WARUNKOWE

Instrukcja IF

//składnia

```
if(argument){  
    instrukcja;  
}
```

Instrukcja, która **wykona** zawartą w klamrach **instrukcję**, pod **warunkiem**, że **argument** jest prawdą.

Instrukcja IF ELSE

//składnia

```
if(argument){  
    instrukcja nr 1;  
} else {  
    instrukcja nr 2;  
}
```

Instrukcja, która **wykona** zawartą w klamrach **instrukcję nr 1**, pod **warunkiem**, że **argument** jest **prawdą**.

Jeżeli **argument** będzie **nieprawdą**, zostanie wykonana **instrukcja nr 2**.

Instrukcja IF ELSE IF

//składnia

```
if(argument nr 1){  
    instrukcja nr 1;  
} else if(argument nr 2){  
    instrukcja nr 2;  
} else if(argument nr 3){  
    instrukcja nr 3;  
} else if(argument nr N-1){  
    instrukcja nr N-1;  
} else {  
    instrukcja nr N;  
}
```

Instrukcję warunkową **IF ELSE** możemy rozbudowywać o kolejne **argumenty** do weryfikacji, tak jak na schemacie.

Instrukcja SWITCH

//składnia

```
switch(argument){  
  case 1:  
    instrukcja nr 1;  
    break;  
  case 2:  
    instrukcja nr 2;  
    break;  
  ...  
  default:  
    instrukcja nr N;  
    break;  
}
```

Instrukcja wykonuje to samo zadanie co poprzednia, jednakże różni się składnią, która jest bardziej przejrzysta.



Operatory

> większe od

< mniejsze od

>= większe/równe od

<= mniejsze/równe od

== przyrównanie wartości

=== przyrównanie typu danych

!= różnica wartości

!== różnica typu danych

Logika w JavaScriptcie

&& (AND, czyli KONIUNKCJA) wszystkie warunki spełnione

|| (OR, czyli ALTERNATYWA) wszystkie warunki spełnione

! (NOT, czyli NEGACJA) wszystkie warunki spełnione



Pętla WHILE

//składnia

```
while(argument){  
    instrukcja;  
}
```

Pętla **WHILE** pozwala na wywołanie **instrukcji** tak długo, jak **argument** jest prawdą.

Pętla DO WHILE

//składnia

```
do{  
    instrukcja;  
} while(argument);
```

Pętla **DO WHILE** pozwala na wywołanie **instrukcji** tak długo, jak **argument** jest prawdą.

Różnicą jest to, że zadana **instrukcja** wykona się przynajmniej raz, przed pierwszym sprawdzeniem **argumentu**.

Pętla FOR

//składnia

```
for(iterator; warunek; inkrementacja){  
    instrukcja;  
}
```

Instrukcja pętli **FOR** pozwala na wywołanie danej **instrukcji**, określoną liczbę razy, dopóki **warunek** jest spełniony.

Dodatkowymi elementami w **argumencie** jest **iterator**, oraz **inkrementacja**.

Iterator jest elementem tymczasowym, żyjącym w obrębie **pętli**, a **inkrementacja** jest instrukcją, która steruje **iteratorem**, dopóki nie spełni postawionego **warunku**.