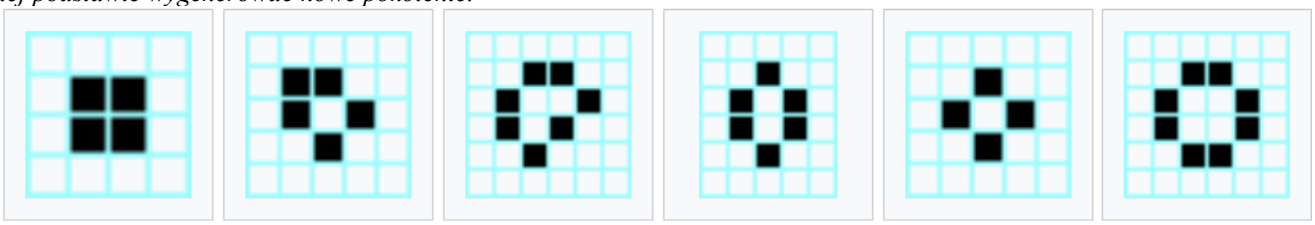
JS 08 - LIVE – automat komórkowy (10)

Gra w życie – jeden z pierwszych i najbardziej znanych przykładów automatu komórkowego, wymyślony w roku 1970 przez brytyjskiego matematyka Johna Conwaya. Od momentu publikacji wzbudzał duże zainteresowanie z powodu zaskakującego sposobu, w jaki struktury potrafią ewoluować. Ta ewolucja zainteresowała automatami komórkowymi ekonomistów, biologów i fizyków, którzy zwrócili uwagę na możliwości automatów w zakresie symulacji procesów życiowych - jak przy zastosowaniu kilku prostych reguł mogą powstawać skomplikowane struktury.
Gra toczy się w turach. Kolejne pokolenie pojawi się po wyliczeniu położenia nowego pokolenia robaczków i narysowaniu na nowo całej tablicy. Jak tworzy się nowe pokolenie? Badany jest teren dokoła każdego robaczka. Jeśli wokół pustego pola znajduje się dokładnie 3 robaczki, to rodzi się na tym polu nowy robaczek. Jeśli wokół pola z robaczkiem jest dokładnie 2 lub 3 robaczki – to ten robaczek przeżywa. W każdym innym przypadku, gdy wokół pola z robaczkiem jest 0,1,4,5,6,7,8 innych robaczków – robaczek ginie (samotność lub tłok). Wykonujemy kolejny ruch i pojawia się kolejne pokolenie robaczków - część z nich umiera lub rodzą się nowe. Wystarczy sprawdzić wszystkie komórki tablicy (ile wokół każdej jest robaczków) i na tej podstawie wygenerować nowe pokolenie.



https://pl.wikipedia.org/wiki/Gra_w_%C5%BCycie#cite_note-:0-1

Pamiętaj o tym, by zrzut ekranu DOKUMENTOWAŁ Twoją pracę

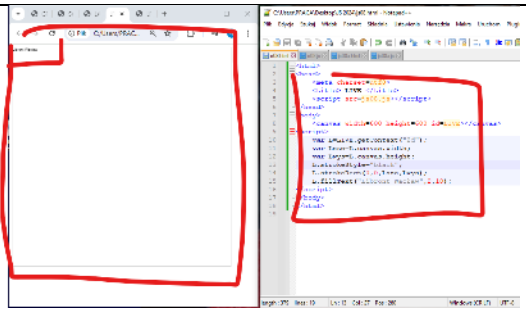
Pliki i canvas (1)

- W swoim folderze utwórz 2 nowe dokumenty: **js08.html** i **js08.js**
- Otwórz oba dokumenty w notatniku, a dokument HTML w przeglądarce
- Dokument **HTML,** wklej tekst z ramki

```
<html>
<head>
	<meta charset=utf8>
	<title> LIVE </title>
	<script src=js08.js></script>
</head>
<body>
	<canvas width=600 height=600 id=LIVE></canvas>
<script>
	var L=LIVE.getContext("2d");
	var Lsze=L.canvas.width;
	var Lwys=L.canvas.height;
	L.strokeStyle="black";
	L.strokeRect(0,0,Lsze,Lwys);
	L.fillText("Libront Wacław",2,10);
</script>
</body>
</html>
```

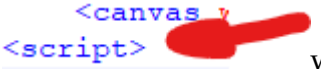
obszar canvas ma rozmiar 600x600
dostęp do obszaru za pomocą „uchwytu” o nazwie L
zadeklarowane dwie zmienne Lsze i Lwys, w których zapamiętujemy szerokość i wysokość obszaru canvas

- Zmień tytuł strony **LIVE** na swoje **inicjały**
- Wpisz swoje nazwisko i imię
- Zapisz dokumenty i odśwież przeglądarkę
  powinna pojawić się kwadratowa ramka o rozmiarze 600x600 pikseli
- Wklej do ramki zrzut ekranu (przeglądarka i notatnik)



Kwadrat – Suwak (1)

za pomocą suwaka ustawiamy bok kwadratu

- Dokument **HTML**, przed znacznik **<script>**  wpisz znaczniki

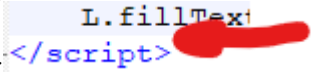
```
<br>
<input type=range min=2 max=100 value=25 id=idBOK onchange=FPlansza()>
BOK:
<label id=idBOK1></label>
```

idBOK *suwak, do którego „podczepiono” funkcję FPlansza rysującą kwadrat (na razie)*
idBOK1 *etykieta, do której wpisujemy wartość ustawioną na suwaku*

- Dokument **JS,** wklej tekst z ramki

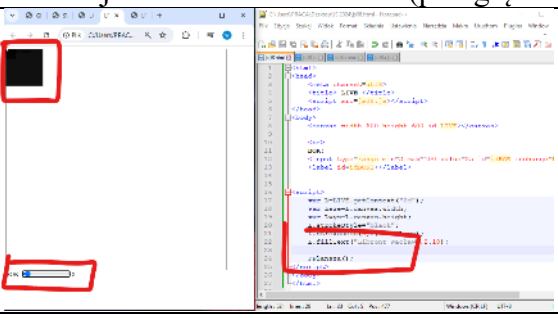
```
function FPlansza(){
      var bok=document.getElementById("idBOK").value;
      idBOK1.innerHTML=bok;
      L.strokeRect(0,0,bok,bok);
}
```

Fkwadrat
funkcja pobiera wartość ustawioną na suwaku za pomocą getElementById
funkcja wstawia wartość do etykiety idBOK1
funkcja rysuje kwadrat o boku bok od punktu (0,0)

- Dokument **HTML**, przed znacznik **</script>**  wpisz wywołanie funkcji `FPlansza();`

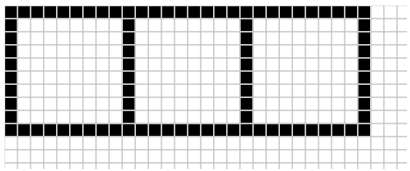
funkcja wywołana przy uruchamianiu strony i od razu narusuje się kwadrat o boku25

- Zapisz dokumenty i odśwież przeglądarkę
- Zmieniaj ustawienie suwaka i „narysuj” wszystkie kwadraty od 5 do 100, co 1 aż zapełnisz pole na czarno
  możesz posłużyć się klawiszami ze strzałkami
- Wklej do ramki zrzut ekranu (przeglądarka i notatnik)



Pole gry (1) – szachownica z kwadratów

Gra toczy się na nieskończonej płaszczyźnie podzielonej na kwadratowe komórki. Nasz automat będzie miał maksymalne wymiary 100x100 komórek. Na ekranie będziemy wyświetlać planszę, której szerokość i liczbę komórek ustawimy za pomocą suwaków. Kwadraty zachodzą na siebie jednym bokiem, jak pokazuje to rysunek, dla kwadratów o boku 9 pikseli. (1 kwadrat zaczyna się w (0,0), drugi (9,0), trzeci (18,0), itd.



- Dokument **HTML**, przed znacznik **<script>** <script> wpisz znaczniki z ramki

```
<br>
<input type=range min=10 max=100 value=25 id=idPOL onchange=FPlansza()>
ILE PÓL:
<label id=idPOL1></label>
```

nowy suwak do ustawiani liczby pól z funkcją Fplansza – oba suwaki mają przypisaną taką samą funkcję

- Dokument **JS**, funkcja **Fplansza()** wklej instrukcje z ramki

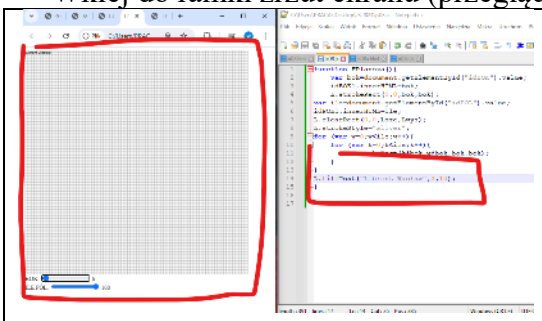
```
var ile=document.getElementById("idPOL").value;
idPOL1.innerHTML=ile;
L.clearRect(0,0,Lsze,Lwys);
L.strokeStyle="silver";
for (var w=0;w<ile;w++){
       for (var k=0;k<ile;k++){
              L.strokeRect(k*bok,w*bok,bok,bok);
       }
}
```

do zmiennej ile pobieramy wartość ustawioną na suwaku i wstawiamy wartość do etykiety
czyścimy obszar canvas – każda zmiana suwaka rysuje nową planszę
w dwóch pętlach FOR rysowana jest szachownica kwadratów
wewnętrzna rysuje kwadraty w jednym wierszu (k – od 0 do ile-1)
zewnętrzna ustawia kolejny wiersz (w od 0 do ile-1)
rysujemy kwadrat, którego położenie wyliczamy z k, w i boku

- Przenieś polecenie `L.fillText("Libront Wacław",2,10);`

z dokumentu HTML do dokumentu JS, na koniec funkcji **Fplansza**
nazwisko i imię jest cały czas widoczne

- Zapisz dokumenty i odśwież przeglądarkę
- Ustaw na suwakach planszę, która ma **100 pól o szerokości 6**
dowolna liczba pól lub dowolna szerokość pola
- Wklej do ramki zrzut ekranu (przeglądarka i notatnik)



Pole gry (1) - automatycznie

Wygodniej będzie, jeżeli na jednym suwaku ustawiamy liczbę pól, z których wyliczymy długość boku, tak, aby plansza zajmowała cały obszar canvas.Matematycznie dokonujemy tego za pomocą dzielenia całkowitego DIV (ile całych dzielników mieści się w dzielnej), którego oczywiście JS nie potrafi.

- Dokument **JS** wpisz tekst funkcji

```
function div(a, b){
    return (Math.round(a/b - 0.5));
}
```

funkcja DIV wykonuje operację dzielenia całkowitego, za pomocą zaokrąglania w dół

- Dokument **JS**, w funkcji **Fplansza()**

- przed pętlami **for** wpisz

```
var bok=div(Lsze,ile);
```

obliczamy bok dzieląc całkowicie szerokość canvas przez liczbę pól planszy
możesz usunąć z funkcji pobierani boku z suwaka

- usuń instrukcję `var bok=document.getElementById("idBOK").value;`
pobiera wartość zmiennej bok z suwaka

- przesuń dwie instrukcje `idBOK1.innerHTML=bok;` `L.strokeRect(0,0,bok,bok);` w nowe miejsce

```
var bok=div(Lsze,ile);

for (var w=0;w<ile;w++){
    for (var k=0;k<ile;k++){
```

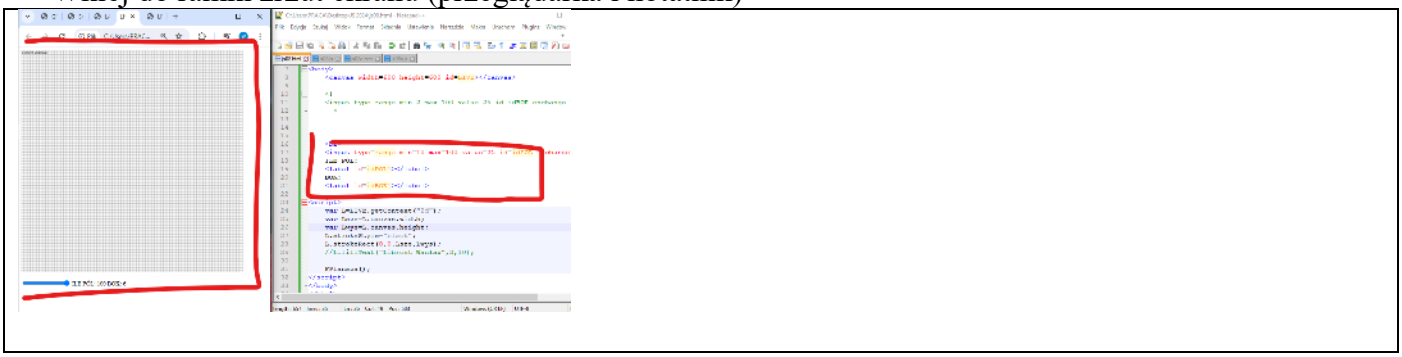
- Po zmianach funkcja **FPlansza()** powinna wyglądać jak obrazek w ramce

```
function FPlansza(){
    var ile=document.getElementById("idPOL").value;
    idPOL1.innerHTML=ile;
    L.clearRect(0,0,Lsze,Lwys);
    L.strokeStyle="silver";
    var bok=div(Lsze,ile);
    idBOK1.innerHTML=bok;
    L.strokeRect(0,0,bok,bok);
    for (var w=0;w<ile;w++){
        for (var k=0;k<ile;k++){
            L.strokeRect(k*bok,w*bok,bok,bok);
        }
    }
    L.fillText("Libront Wacław",2,10);
}
```

- W dokumencie **HTML**
- usuń suwak do zmiany boku `<input type=range min=2 max=100 value=25 id=idBOK onchange=FPlansza()>`
`BOK:`
- przestaw instrukcje `<label id=idBOK1></label>` aby wyglądały jak na załączonym obrazku

```
<br>
<input type=range min=10 max=100 value=2
ILE PÓL:
<label id=idPOL1></label>
BOK:
<label id=idBOK1></label>
```

- Zapisz dokumenty i odśwież przeglądarkę
- Ustaw planszę o 100 polach
jednym suwakiem ustawiamy planszę, za suwakiem dwa pola pokazujące liczbę pól i szerokość pola
- Wklej do ramki zrzut ekranu (przeglądarka i notatnik)



Myszka (1)

Początkowy układ komórek ustawiamy za pomocą myszki. Po kliknięciu w białe pole, zamieni się na czarne (komórka żyje), a jeżeli jest czarne, to zamieni się na białe (komórka martwa). Aby wypełnić odpowiedni pole na czarno (lub biało) musimy znać współrzędne lewego górnego rogu pola i kolumnę i wiersz pola, w które kliknięto.
W programie będziemy się wielokrotnie posługiwać ilością pól na szachownicy i bokiem pola, dlatego warto jest zadeklarować zmienne globalne, które będą widoczne w całym programie. Obliczymy je po każdej zmianie suwaka i nie trzeba będzie liczyć na nowo w innych funkcjach

```
<label id
<script>
```

- Dokument **HTML**, przed znacznik **<script>** wpisz znaczniki

```
<br>
X,Y: <label id=XY>()</label>
K,W: <label id=KW>()</label>
L,G: <label id=LG>()</label>
```

3 ramki, do których wpiszemy

X,Y	*współrzędne piksela, w który kliknięto myszką*
K,W	*kolumnę i wiersz pola na planszy – wyliczone z X,Y*
L,G	*współrzędne lewego, górnego rogu pola, w które kliknięto*

- Dokument **HTML**, przed funkcję **FPlansza** `FPlansza();` wpisz deklaracje zmiennych

```
var ILE=0;
var BOK=0;
```

dwie zmienne globalne: liczba pól w poziomie (pionie) na planszy i długość boku pola

```
L.strokeRect(0,0,bok,bok);

for (var w=0;w<ile;w++){
    for (var k=0;k<ile;k++){
```

- Dokument **JS**, funkcja **FPlansza**, wpisz instrukcje

```
BOK=bok;
ILE=ile;
```

JS rozróżnia nazwy pisane dużymi i małymi literami
zmienne lokalne ile i bok istnieją tylko w czasie działania funkcji i potem giną
zmienne globalne ILE i BOK istnieją cały czas w trakcie działania programu

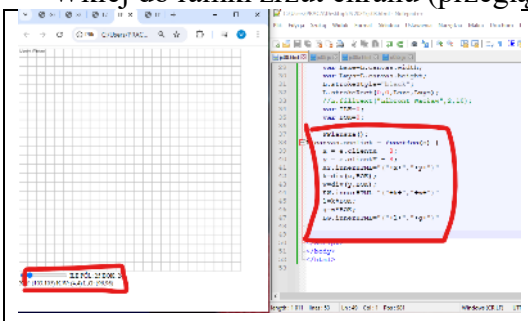
```
    FPlansza();
</script>
```

- Dokument **HTML**, przed znacznik **</script>** wklej obsługę myszki

```
L.canvas.onclick = function(e) {
        x = e.offsetX;
        y = e.offsetY;
        XY.innerHTML="("+x+","+y+")"
        k=div(x,BOK);
        w=div(y,BOK);
        KW.innerHTML="("+k+","+w+")"
        l=k*BOK;
        g=w*BOK;
        LG.innerHTML="("+l+","+g+")"
}
```

do zmiennych x i y pobieramy współrzędne kliknięcia myszką i wpisujemy do etykiety id=XY
kolumnę i wiersz liczymy podobnie jak w czasie rysowania planszy, za pomocą DIV i wpisujemy do etykiety id=KW
lewy, górny róg wyliczamy mnożąc kolumnę i wiersz przez długość boku i wpisujemy do etykiety id=LG

- Zapisz dokumenty i odśwież przeglądarkę
- Spróbuj kliknąć w punkt o współrzędnych **(100,100)**
- Wklej do ramki zrzut ekranu (przeglądarka i notatnik)



Robaczek (1)

Po kliknięciu myszką w pole planszy powinno się zaczernić (robaczek żyje)

- Dokument **JS,** wpisz nową funkcję

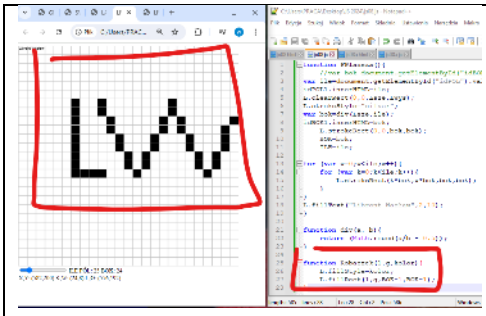
```
function Robaczek(l,g,kolor){
    L.fillStyle=kolor;
    L.fillRect(l,g,BOK-1,BOK-1);
}
```

ustawiamy wypełnienie i rysujemy kwadrat w kolorze podanym jako parametr
BOK zmniejszamy o jeden, aby widać było na planszy szare ramki

- Dokument **HTML**, funkcja `L.canvas.onclick = function(e)` wpisz jako ostatnią instrukcję

```
Robaczek(l,g,"black");
```

- Zapisz dokumenty i odśwież przeglądarkę
- Klikaj w pola i ustaw **na polach planszy swoje inicjały**
- Wklej do ramki zrzut ekranu (przeglądarka i notatnik)



Zapalanie i gaszenie robaczków (1)

Jak „zabić” (ustawić na biało) robaczka? Sprawdzać kolor piksela? W planszowych grach komputerowych, stan poszczególnych pól zapisujemy w tablicach – w naszym przypadku w tablicy dwuwymiarowej, które w JS definiuje się bardziej skomplikowanie niż w innych językach.
Jeżeli pole jest białe, to zmieniamy je na czarne, a jeżeli czarne, to na białe. Stan każdego pola zapisujemy w tablicy ROB: 1-czarne, 0-białe.

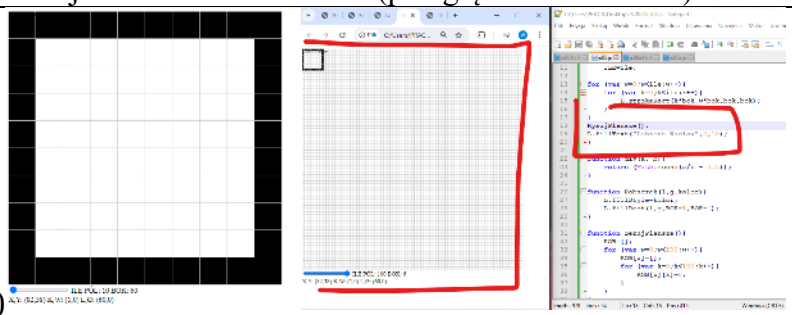
- Dokument **HTML**, funkcja **Fplansza()**
```
var BOK=0;
FPlansza();
```
wpisz instrukcje
```
var ROB=[];
ZerujPlansze();
```
ROB *pusta tablica na stan każdego z robaczków: 1 – czarny(żyje), 0 – biały (brak robaczka)*
ZerujPlansze *funkcja tworząca tablicę dwuwymiarową i wpisująca do każdej z 10000 komórek zero*
- Dokument **HTML** popraw funkcję `L.canvas.onclick = function(e)`, zastępując instrukcję `Robaczek(l,g,"black");` instrukcją warunkową
```
if (ROB[w][k]==1){
    Robaczek(l,g,"white");
    ROB[w][k]=0;
}else{
    Robaczek(l,g,"black");
    ROB[w][k]=1;
}
```
jeżeli w tablicy ROB komórka ustawione na 1 (robaczek żyje), to należy wstawić tam białe pole i do komórki wpisać zero w przeciwnym razie pole na czarno i do komórki jeden
- Dokument **JS**, wklej nowe funkcje z ramki

```
function ZerujPlansze(){
      ROB=[];
      for (var w=0;w<100;w++){
            ROB[w]=[];
            for (var k=0;k<100;k++){
                  ROB[w][k]=0;
            }
      }
}

function RysujPlansze(){
      for (var w=0;w<100;w++){
            for (var k=0;k<100;k++){
                  jak=ROB[w][k];
                  kolor="white";
                  if (jak==1) kolor="black";
                  l=k*BOK;
                  g=w*BOK;
                  Robaczek(l,g,kolor);
            }
      }
      L.fillStyle="black";
}
```

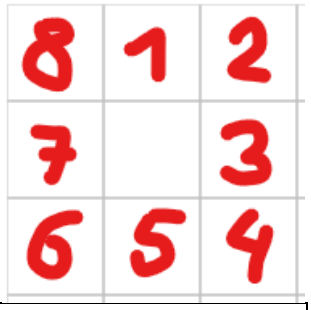
ZerujPlansze *funkcja ustawia dwuwymiarową tablicę ROB i wstawia do wszystkich komórek zera*
RysujPlansze *funkcja sprawdza każdą komórkę tablicy ROB i gdy jest równa 1 to rysujemy robaczka czarnego, w przeciwnym razie białego*
aby narysować robaczka, należy wyliczyć współrzędne lewego, górnego rogu

- Dokument **JS**, funkcja **FPlansza()** wpisz instrukcję
```
RysujPlansze();
```
po każdej zmianie stanu suwaka plansza jest przerysowana i odtwarzają się ustawione robaczki
- Zapisz dokumenty i odśwież przeglądarkę
- Ustaw suwakiem **10 pól**
- **Ustaw robaczki** na każdym brzegu planszy - kwadrat 10x10
- Ustaw suwakiem **100 pól**
robaczki nie giną po przeskalowaniu planszy
- Wklej do ramki zrzut ekranu (przeglądarka i notatnik)

(1) 

Zliczanie sąsiadów (1)

Każde pole planszy ma dokładnie 8 sąsiadów (prócz brzegowych). Gra toczy się w turach, tzn. nowe położenie robaczków obliczamy na podstawie aktualnego stany, po czym wyświetlamy planszę z nowymi ustawieniami.
Nowy robaczek „rodzi się", gdy wokół białego pola (w tablicy ROB zero) jest dokładnie 3 robaczki. Nie wnikamy w ich skomplikowany sposób rozmnażania się.
Robaczek umiera, gdy ma zbyt mało sąsiadów (0 lub 1) albo zbyt dużo sąsiadów (4, 5, 6, 7, 8)
Gdy robaczek ma 2 lub 3 sąsiadów nic się nie dzieje – żyje nadal
Aby ustalić, co będzie z naszym robaczkiem, należy nauczyć go zliczać swoich sąsiadów.



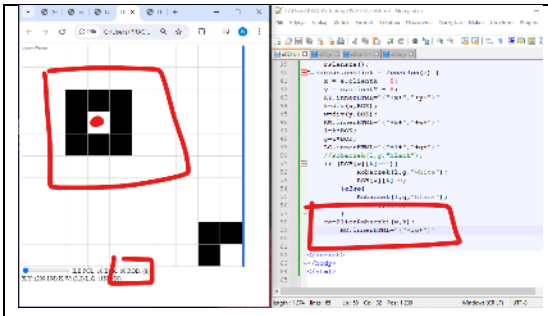
- Dokument **JS,** wklej nową funkcję z ramki

```
function ZliczRobaczki(w,k){
	var i=0;
	if (k<ILE-1)		i=i+ROB[w+0][k+1]; // 3
	if (k<ILE-1 && w<ILE-1)	i=i+ROB[w+1][k+1]; // 4
	if (w<ILE-1)		i=i+ROB[w+1][k+0]; // 5
	if (k>0 && w<ILE-1)	i=i+ROB[w+1][k-1]; // 6
	if (k>0)			i=i+ROB[w+0][k-1]; // 7
	if (k>0 && w>0)		i=i+ROB[w-1][k-1]; // 8
	if (w>0)			i=i+ROB[w-1][k+0]; // 1
	if (k<ILE-1 && w>0)	i=i+ROB[w-1][k+1]; // 2
	return i;
}
```

zliczamy w zmiennej i, którą na początku zerujemy
komórki w tablicy ponumerowane są od 0 do 99, dlatego brzeg ILE-1
w komentarzach zapisano, którą komórkę sprawdzamy i sumujemy ich zawartość (zero nie zmienia sumy)

- Dokument **HTML** wstaw etykietę pokazującą liczbę robaczków
```
BOK: <label id=idBOK1></label><br>
ROB: <label id=RO>()</label>
```

- Dokument **HTML**, funkcja `L.canvas.onclick = function(e)` wpisz instrukcje
```
ro=ZliczRobaczki(w,k);
RO.innerHTML="("+ro+")"
```
za pomocą funkcji ZliczRobaczki obliczamy liczbę sąsiadów i wstawiamy do etykiety RO
- Zapisz dokumenty i odśwież przeglądarkę
- Ustaw robaczki, jak pokazuje rysunek i kliknij w środek
*licznik środkowego robaczka pokazuje **8 sąsiadów***
- Wklej do ramki zrzut ekranu (przeglądarka i notatnik)



Nowe pokolenie (1)

Na podstawie zliczonych robaczków decydujemy o tym, kto przeżyje i gdzie się urodzi nowy. Stan po przeliczeniach też należy zapisać w tablicy – i nie może być to tablica ROB, bo to w niej prowadzimy zliczanie.

- Dokument **HTML**, przed znacznik **<script>** wpisz znaczniki

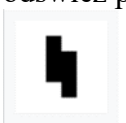
```
<br>
<input type=button value="NOWE POKOLENIE" onClick=NowePokolenie()>
```

przycisk z funkcją NowePokolenie()

- Dokument **JS**, wklej nową funkcję z ramki

```
function NowePokolenie(){
	//zerowanie tablicy
	POK=[];
	for (var w=0;w<100;w++){
		POK[w]=[];
		for (var k=0;k<100;k++){
			POK[w][k]=0;
		}
	}
	//ustawianie nowego pokolenia
	for (var w=0;w<100;w++){
		for (var k=0;k<100;k++){
			ile=ZliczRobaczki(w,k);
			if (ROB[w][k]==0 && ile==3){POK[w][k]=1}
			if (ROB[w][k]==1 && (ile==2 || ile==3)){POK[w][k]=1}
			if (ROB[w][k]==1 && (ile<2 || ile>3)){POK[w][k]=0}
		}
	}
	//przepisywanie nowego stanu do głównej tablicy
	for (var w=0;w<100;w++){
		for (var k=0;k<100;k++){
			ROB[w][k]=POK[w][k];
		}
	}
	RysujPlansze();
}
```

NowePokolenie() funkcja wyliczająca nowy stan robaczków
zerujemy dwuwymiarową tablicę POK, w której będziemy ustawiać stan nowego pokolenia
w dwóch pętlach FOR sprawdzamy stan każdego pola – ilu ma sąsiadów
jeżeli pole jest puste i 3 sąsiadów, to w tablicy POK ustawiamy 1 – rodzi się nowy – w tablicy POK ustawiamy 1
jeżeli w polu jest robaczek i ma 2 lub 3 sąsiadów, to przeżywa – w tablicy POK ustawiamy 1
jeżeli w polu jest robaczek i ma mniej niż 2 albo więcej niż 3 sąsiadów, to umiera – w tablicy POK ustawiamy 0
Na końcu przepisuje wszystkie komórki z tablicy POK do komórek tablicy ROB
I rysujemy planszę od nowa

- Zapisz dokumenty i odśwież przeglądarkę
- Ustaw tzw. „żabkę”  i sprawdź, jak wyglądają kolejne pokolenia
  wciskaj przycisk NOWE POKOLENIE
- Ustaw tzw. „szybowiec”  i sprawdź, jak wyglądają kolejne pokolenia
- Wklej do ramki zrzut ekranu (przeglądarka i notatnik)

Automat (1)

No przecież nie będziemy klikać tysiąc razy w przycisk

- Dokument **HTML**, przed znacznik **<script>** wpisz znaczniki

```
<input type=button value="START" onClick=FStart()>
<input type=button value="STOP" onClick=FStop()>
<input type=button value="ZERUJ" onClick=FOdNowa()>
```

trzy nowe przyciski

- Dokument **HTML**, przed funkcję Fplansza wpisz definicję zmiennej `var automat;`

zmienna jest odpowiedzialna za wywołanie funkcji SetTimeout używaną podczas animacji

- Dokument **JS** wklej tekst z ramki

```
function FOdNowa(){
	ZerujPlansze();
	RysujPlansze();
}

function FStart(){
	NowePokolenie();
	clearTimeout(automat);
	automat = setTimeout(FStart, 100);
}
function FStop(){
	clearTimeout(automat);
}
```

funkcja FodNowa — *zeruje tablicę ROB i przerysowuje planszę*
funkcja Fstart — *działa podobnie jak podczas animacji – rekurencyjnie wywołuje samą siebie, co 100 milisekund*
funkcja Fstop — *zatrzymuje wykonywanie funkcji setTimeout*

- Zapisz dokumenty i odśwież przeglądarkę
- Przygotuj dowolny układ robaczków
- Wciśnij przycisk **START** i obserwuj ich żywot

W każdej chwili możesz kliknąć w planszę i pobawić się w „boga" dodając robaczka

- Wklej do ramki zrzut ekranu (przeglądarka i notatnik)