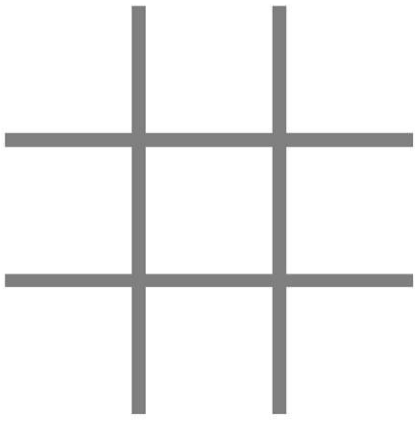
Iks-oks (Tic-Tac-Toe)

Primjer: Iks-oks (Tic-Tac-Toe)

Implementirajmo igru Iks-oks (Tic-Tac-Toe). (finished)



Iks-oks plan

1. Svaki klik na prazno polje popunjava to polje sa "X" tako što se dodaje slika slova "x" u prazan `<div>`
2. Poslije našeg poteza (tj. popunavanja polja sa X), računar postavlja "O" u proizvoljno prazno polje
3. Ako postoje 3 X ili 3 O u redu, koloni ili dijagonali, proglasimo pobjednika

CodePen starter code

Prazno polje -> X

Prvi korak: postavimo da je moguće kliknuti na svaki svi elementi `div` koji je dijete `#grid`

```
<body>
  <h1>Tic-Tac-Toe</h1>
  <div id="grid">
    <div></div>
    <div></div>
    <div></div>

    <div></div>
    <div></div>
    <div></div>

    <div></div>
    <div></div>
    <div></div>
  </div>
</body>
```

Prazno polje -> X

```
<body>
  <h1>Tic-Tac-Toe</h1>
  <div id="grid">
    <div></div>
    <div></div>
    <div></div>

    <div></div>
    <div></div>
    <div></div>

    <div></div>
    <div></div>
    <div></div>
  </div>
</body>
```

```
function changeToX(event) {
  // ...
}

const boxes = document.querySelectorAll('#grid div');
for (const box of boxes) {
  box.addEventListener('click', changeToX);
}
```

U funkciji changeToX, postavljamo <img> tag u kliknuti element…

Kako?

Prazno polje -> X

```
<body>
  <h1>Tic-Tac-Toe</h1>
  <div id="grid">
    <div></div>
    <div></div>
    <div></div>

    <div></div>
    <div></div>
    <div></div>

    <div></div>
    <div></div>
    <div></div>
  </div>
</body>
```

```
function changeToX(event) {
  const container = event.currentTarget;
  const image = document.createElement('img');
  image.src = X_IMAGE_URL;
  container.appendChild(image);
  container.removeEventListener('click', changeToX);
}

const boxes = document.querySelectorAll('#grid div');
for (const box of boxes) {
  box.addEventListener('click', changeToX);
}
```

Završen korak 1: CodePen

Iks-oks plan

1. ~~Svaki klik na prazno polje popunjava to polje sa "X" tako što se dodaje slika slova "x" u prazan `<div>`~~
2. **Poslije našeg poteza (tj. popunavanja polja sa X), računar postavlja "O" u proizvoljno prazno polje**
3. Ako postoje 3 X ili 3 O u redu, koloni ili dijagonali, proglasimo pobjednika

Slučajni brojevi u JS

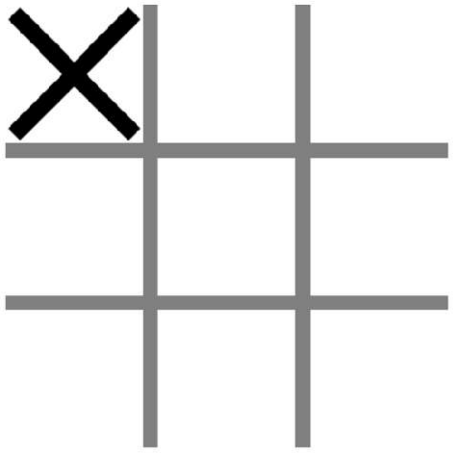
JavaScript ima samo jedan generator slučajnih brojeva (ako zanemarimo crypto biblioteke) : `Math.random()`

- `Math.random()` vraća slučajan floating point broj između [0, 1) (0 inclusive, 1 exclusive)

Da bi dobili slučajan cio broj od 0 do `max` (tj. interval je [0.max)):

```
Math.floor(Math.random() * max);
```

*aside from crypto libraries



Kako da otkrijemo koje je polje prazno?

Prazno polje: DOM pristup

Jedan način:

- For each `#grid div`
  - Provjeriti da li postoji dijete `img`

`querySelector` se može upotrebiti i na elementu a na samo na `document`:

```
const sectionElement = document.querySelector('section');
// All h1s that are children of sectionElement:
const headers = sectionElement.querySelector('h1');
```

```
function computerChooseO() {
  const allBoxes  = document.querySelectorAll('#grid div');
  const freeBoxes = [];
  for (const box of allBoxes) {
    let imageChild = box.querySelector('img');
    if (!imageChild) {
      freeBoxes.push(box);
    }
  }
  const index = Math.floor(Math.random() * freeBoxes.length);
  const freeSpace = freeBoxes[index];
  const image = document.createElement('img');
  image.src = O_IMAGE_URL;
  freeSpace.removeEventListener('click', changeToX);
  freeSpace.appendChild(image);
}
```

Što je loše sa ovim pristupom?
(CodePen)

Pitamo UI za stanje

Stanje igre određujemo na osnovu korisnikig interfejsa.

Nije dobra praksa kreiranja softvera :

- Prožimanje tzv. "view" i "model"
- Može dovesti do grešaka koje je teško otkriti :
  - Što kao kasnije odlučimo da prikazujemo X i O pomoću background-image umjesto taga `<img>`?
- Kod nije čitljiv
  - Kakv je veza tagova "`img`" sa praznim poljima?

Bolje je čuvati stanja odvojeno od UI!

Bolji (?) pristup: globalna promjenljiva

U globalnoj pomjenljivoj čuvamo stanje igre:

```
const freeBoxes = [];
const boxes = document.querySelectorAll('#grid div');
for (const box of boxes) {
  box.addEventListener('click', changeToX);
  freeBoxes.push(box);
}
```

`freeBoxes` je niz koji sadrži prazna polja

Bolji (?) pristup: globalna promjenljiva

```
function changeToX(event) {
  const container = event.currentTarget;
  const image = document.createElement('img');
  image.src = X_IMAGE_URL;
  container.appendChild(image);
  container.removeEventListener('click', changeToX);

  // Also remove |container| from |freeBoxes|
  const indexToRemove = freeBoxes.indexOf(container);
  freeBoxes.splice(indexToRemove, 1);
  computerChooseO();
}
```

Zatim mijenjamo niz `freeBoxes` kada dodamo X...

Bolji (?) pristup: globalna promjenljiva

```
function computerChooseO() {
  const allBoxes  = document.querySelectorAll('#grid div');
  const index = Math.floor(Math.random() * freeBoxes.length);
  const freeSpace = freeBoxes[index];
  // Remove the chosen box from freeBoxes.
  freeBoxes.splice(index, 1);
  const image = document.createElement('img');
  image.src = O_IMAGE_URL;
  freeSpace.removeEventListener('click', changeToX);
  freeSpace.appendChild(image);
}
```

...i kada računar dodaje O. (CodePen)

Da li je stvarno bolji pristup?

Što je loše kod ovog pristupa?

- I dalje su stanja igre uvezanasa UI
  - Čuvamo reference na UI elemente u nizu `freeBoxes`
- Globalne promjneljive nisu dobar pristup
  - U svim jezicima se izbjegava kreiranje globalnih promjenljivih...

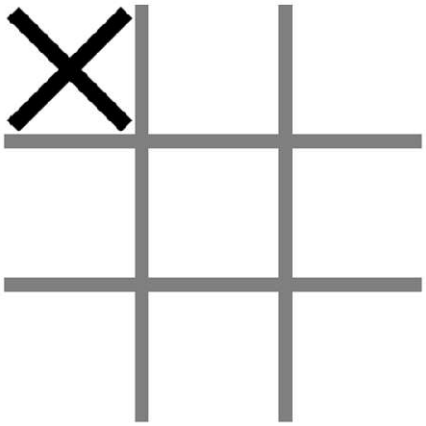
(U suštini nam trebaju klase)

(Stižu uskoro)

Iks-oks plan

1. ~~Svaki klik na prazno polje popunjava to polje sa "X" tako što se dodaje slika slova "x" u prazan `<div>`~~
2. ~~Poslije našeg poteza (tj. popunavanja polja sa X), računar postavlja "O" u proizvoljno prazno polje~~
3. **Ako postoje 3 X ili 3 O u redu, koloni ili dijagonali, proglasimo pobjednika**

Kako razlikovati polja



Isti event handler se poziva za svaki elementis.

Kako da razlikujemo elemente?

Užasna ideja: 9 event handler-a

```
<body>
  <h1>Tic-Tac-Toe</h1>
  <div id="grid">
    <div id="one"></div>
    <div id="two"></div>
    <div id="three"></div>

    <div id="four"></div>
    <div id="five"></div>
    <div id="six"></div>

    <div id="seven"></div>
    <div id="eight"></div>
    <div id="nine"></div>
  </div>
</body>
```

```
function changeToXRow1Column1(event) {
  //...
}
function changeToXRow1Column2(event) {
  //...
}
function changeToXRow1Column3(event) {
  //...
}
```

```
const first = document.querySelector('#one');
first.addEventListener('click', changeToXRow1Column1);
const second = document.querySelector('#two');
second.addEventListener('click', changeToXRow1Column2);
```

Jedinstvena identifikacija polja

```
<body>
  <h1>Tic-Tac-Toe</h1>
  <div id="grid">
    <div id="one"></div>
    <div id="two"></div>
    <div id="three"></div>

    <div id="four"></div>
    <div id="five"></div>
    <div id="six"></div>

    <div id="seven"></div>
    <div id="eight"></div>
    <div id="nine"></div>
  </div>
</body>
```

Ideja da se svakom polju zada jedinstveni identifikator je dobra!

Rješenje

```
const freeBoxes = [];
// Map of box number -> 'x' or 'o'
const takenBoxes = {};
const boxes = document.querySelectorAll('#grid div');
for (const box of boxes) {
  box.addEventListener('click', changeToX);
  freeBoxes.push(box);
}
```

Dodajemo još jednu promjenljivu `takenBoxes` koja preslikava broj polja u vlasnika polja

```
function changeToX(event) {
  assignSpace(event.currentTarget, 'x');
}
```

```
function computerChooseO() {
  const allBoxes = document.querySelectorAll('#grid di
  const index = Math.floor(Math.random() * freeBoxes.le
  const freeSpace = freeBoxes[index];

  assignSpace(freeSpace, 'o');
}
```

```
function assignSpace(space, owner) {
  const image = document.createElement('img');
  image.src = owner === 'x' ? X_IMAGE_URL : O_IMAGE_URL;
  space.appendChild(image);

  takenBoxes[space.id] = owner;
  const indexToRemove = freeBoxes.indexOf(space);
  freeBoxes.splice(indexToRemove, 1);
  space.removeEventListener('click', changeToX);
}
```

Mijenajmo `takenBoxes` svaki put kada se zauzme polje.

```
// Returns 'x', 'o', or null for no winner yet.
function getWinner() {
  // Check rows
  let rowResult = checkBoxes('one', 'two', 'three') ||
      checkBoxes('four', 'five', 'six') ||
      checkBoxes('seven', 'eight', 'nine');

  // Check columns
  let colResult = checkBoxes('one', 'four', 'seven') ||
      checkBoxes('two', 'five', 'eight') ||
      checkBoxes('three', 'six', 'nine');

  // Check diagonal
  let diagonalResult = checkBoxes('one', 'five', 'nine') ||
      checkBoxes('three', 'five', 'seven');
  return rowResult || colResult || diagonalResult;
}
```

```
function checkBoxes(one, two, three) {
  if (takenBoxes[one] !== undefined &&
      takenBoxes[one] === takenBoxes[two] &&
      takenBoxes[two] === takenBoxes[three]) {
    return takenBoxes[one];
  }
  return null;
}
```

Određujemo pobjednika provjerom redova, kolona i dijagonala

(CodePen)

```
  </div>
  <div id="results"></div>
</body>
```

```
function displayWinner() {
  const winner = getWinner();

  const resultContainer = document.querySelector('#results');
  const header = document.createElement('h1');
  if (winner === 'x') {
    header.textContent = 'You win!';
  } else if (winner === 'o'){
    header.textContent = 'Computer wins';
  } else {
    header.textContent = 'Tie';
  }
  resultContainer.appendChild(header);
```

Kreiramo element div za rezultat CodePen)

Zakačimo "data" za elemente div?

```
<body>
  <h1>Tic-Tac-Toe</h1>
  <div id="grid">
    <div id="one"></div>
    <div id="two"></div>
    <div id="three"></div>

    <div id="four"></div>
    <div id="five"></div>
    <div id="six"></div>

    <div id="seven"></div>
    <div id="eight"></div>
    <div id="nine"></div>
  </div>
</body>
```

Bilo bi bolje kada bi mogli koristiti brojeve umjesto stringova za id?

Postoji li način da se zakače dodatni podaci na element?

Data atributi

Mogu se dodati posebni data-* attributes HTML elementima – ascorianje dodatnih podataka sa elementima.

`data-your-name="Your Value"`

```
<article
  id="electriccars"
  data-columns="3"
  data-index-number="12314"
  data-parent="cars">
...
</article>
```

Data atributi u jeziku JavaScript

Možete pristupiti vašim zakačenim podacima primjenom objekta `dataset` na DOM objektu:

```
var article = document.getElementById('electriccars');

article.dataset.columns // "3"
article.dataset.indexNumber // "12314"
article.dataset.parent // "cars"
```

- Riječi koje se razdvojene crticama postaju imena u camel-notaciji npr. `data-index-number` u HTML-u je `dataset.indexNumber` u jeziku JS
- Data atributi se vraćaju kao stringovi ali se mogu konvertovati u Number pomoću `parseInt`

Data atributi u CSS-u

Stil data atributa pomoću CSS-a:

`[data-`***variable-name***`]` ili
`[data-`***variable-name***`='`***value***`']` ili
element`[data-`***variable-name***`]` itd.

```
article[data-columns='3'] {
  width: 400px;
}
article[data-columns='4'] {
  width: 600px;
}
```

```
<body>
  <h1>Tic-Tac-Toe</h1>
  <div id="grid">
    <div data-index="0"></div>
    <div data-index="1"></div>
    <div data-index="2"></div>

    <div data-index="3"></div>
    <div data-index="4"></div>
    <div data-index="5"></div>

    <div data-index="6"></div>
    <div data-index="7"></div>
    <div data-index="8"></div>
  </div>
  <div id="results"></div>
</body>
```

Final Solution
CodePen

```
const index = parseInt(space.dataset.index);
takenBoxes[index] = owner;
```