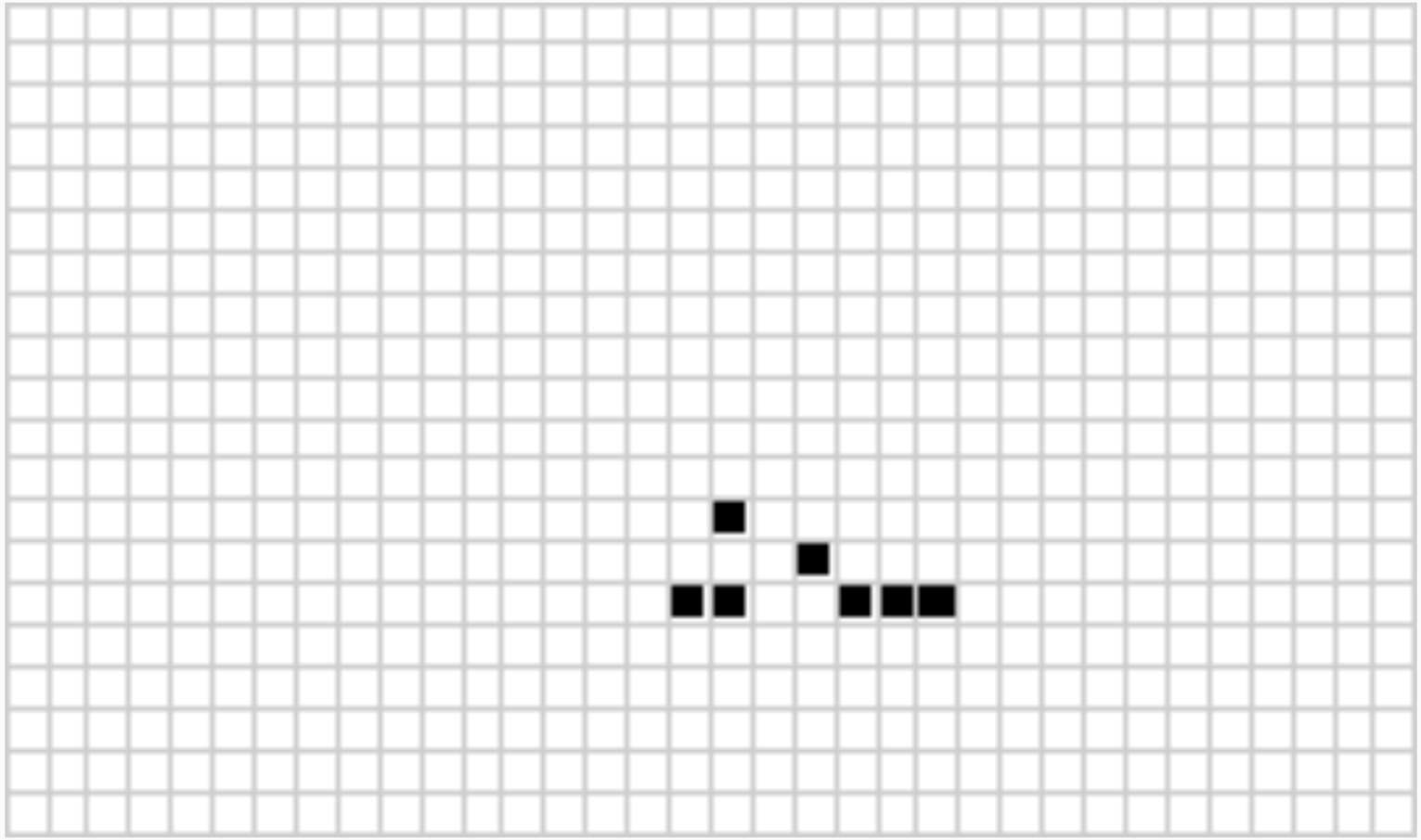
game of life



game of life

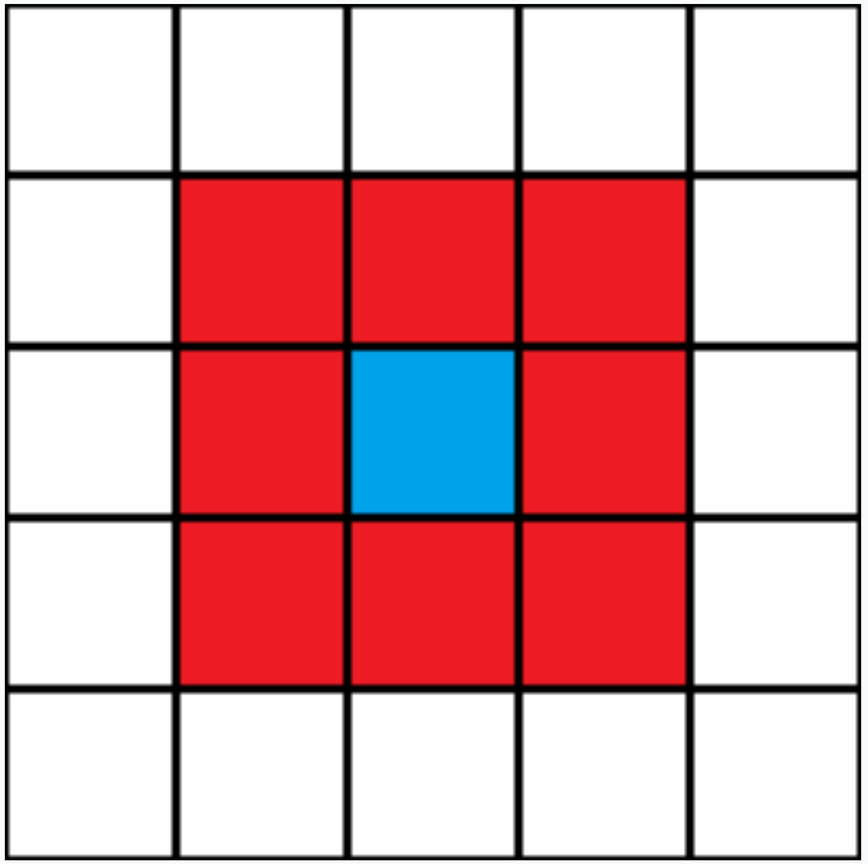
- “Zero-player” game (see *animation*)
- Rooted in Von Neumann’s quest for artificial/simulated life
- Created by Jon Conway in 1970
- Sparked niche field: **cellular automaton**
- Simple rules can produce complex behavior

HTTP://WWW.ERICWEISSTEIN.COM/ENCYCLOPEDIAS/LIFE/ACORN.HTML

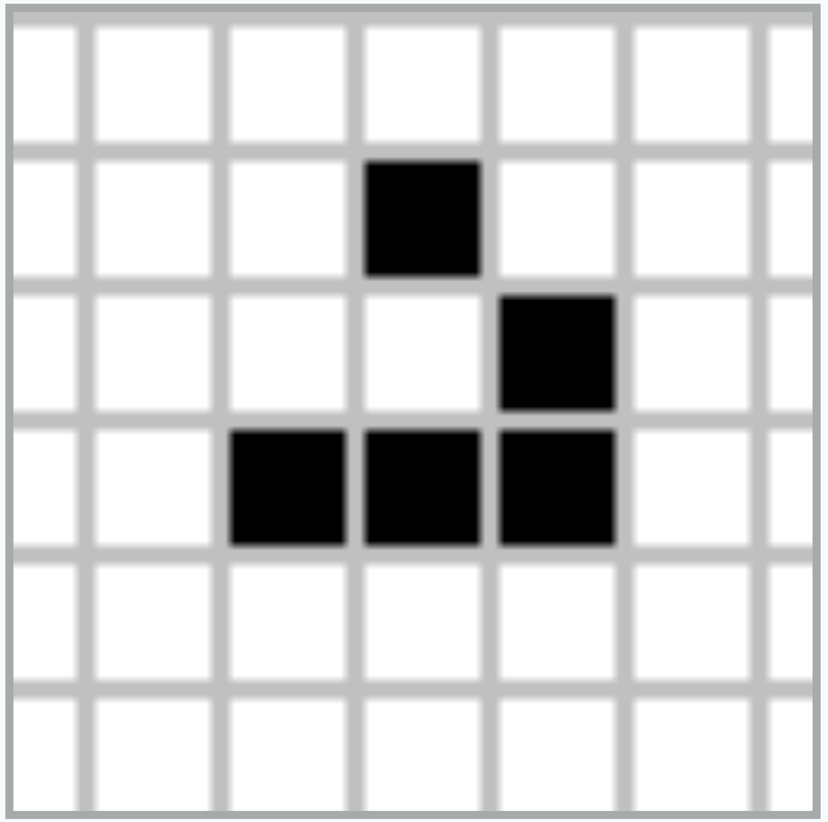
rules

- 2D grid of cells that are currently on or off (dead or alive)
- Each step, grid updates all-at-once
- Currently alive cell
  - “Underpopulation”: dies given fewer than 2 live neighbors
  - “Overcrowding”: dies given greater than 3 live neighbors
  - Otherwise, lives on
- Currently dead cell
  - “Birth”: comes to life given exactly 3 live neighbors
  - Otherwise, remains dead

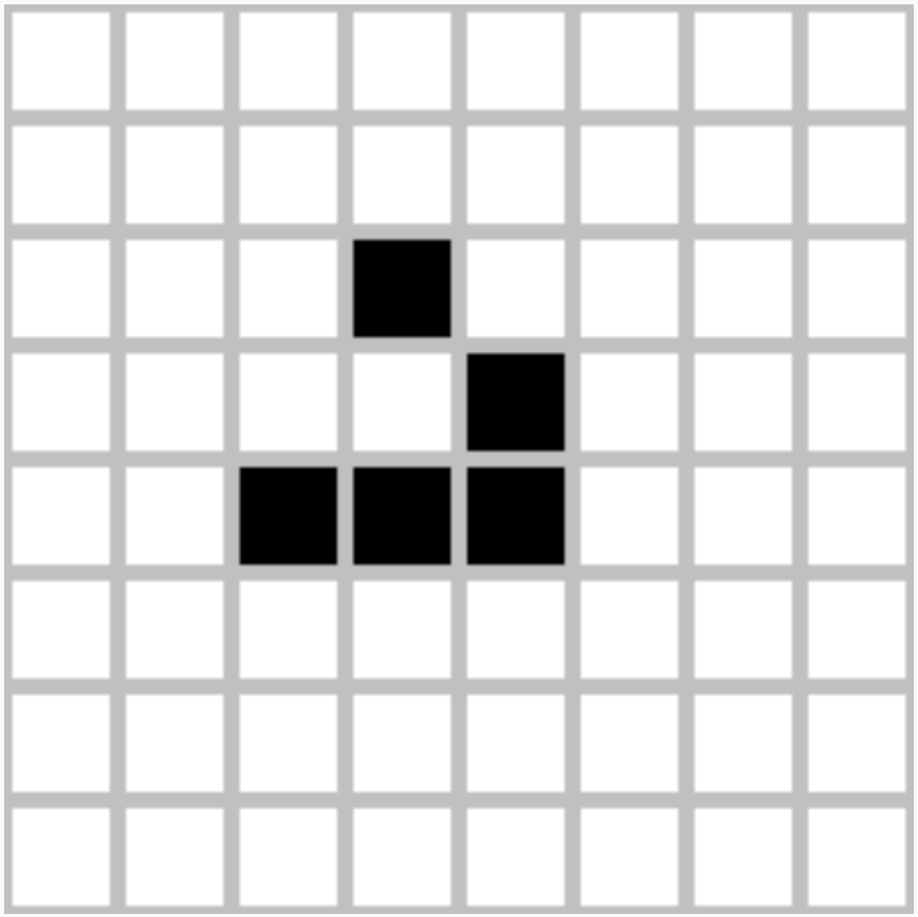
neighbors



game of life



game of life



`this` and the `.bind` method

`this`…

- …is the “context” for a function.
- …is determined when a function is *invoked*, not when it is defined.

To determine what `this` is for any function, take a look at its *call-site*.

types of context binding and call-site

- Default binding: `func();`
- Implicit binding: `obj.func();`
- Explicit binding: `func.call(obj);`
- `new` binding: `new func();`

the `.bind` method

- Requires one argument, a `thisArg`.
- Returns a new function whose `this` is always the thisArg.
- Does *not* invoke the function.

```
var boundFunc = oldFunc.bind(thisArg);
boundFunc(); //invoked with thisArg as `this`
```

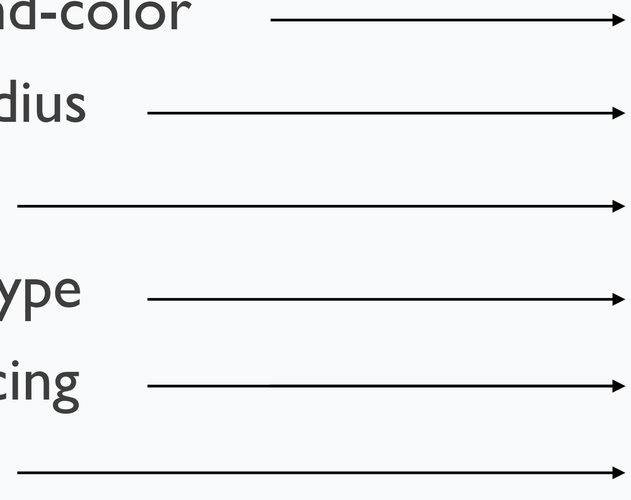
Manipulating the DOM

- Changing Attributes for Style
- Making Elements
- Putting them into the DOM
- Remove Elements
- innerHTML and the DOM HTML Reader

Changing style attributes

```
element.style.backgroundColor = “blue”;
```

- **CSS**
  - background-color
  - border-radius
  - font-size
  - list-style-type
  - word-spacing
  - z-index
- **JavaScript**
  - backgroundColor
  - borderRadius
  - fontSize
  - listStyleType
  - wordSpacing
  - zIndex



Changing CSS Classes

- *classList* is HTML5 way to modify which classes are on an Element

```
document.getElementById("MyElement").classList.add('class');

document.getElementById("MyElement").classList.remove('class');

if ( document.getElementById("MyElement").classList.contains('class') )

document.getElementById("MyElement").classList.toggle('class');
```

Responding to User Activity

- Event Handlers
- Default Events
- Bubbling and Propagation of Events

https://jsbin.com/sarohuyivu/1/edit?html,js,output

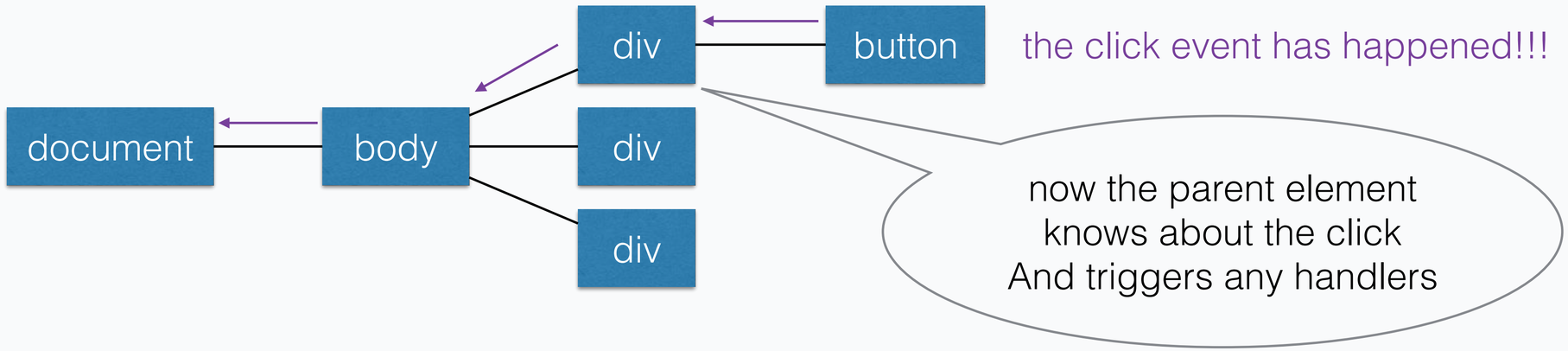
Event Handlers

```
element.addEventListener('click', function(event) {
    // Run this code on click
});
```

- JS that handles things that happen in the DOM
- Event examples:
  - click
  - (form) submit
  - hover
  - mouseover

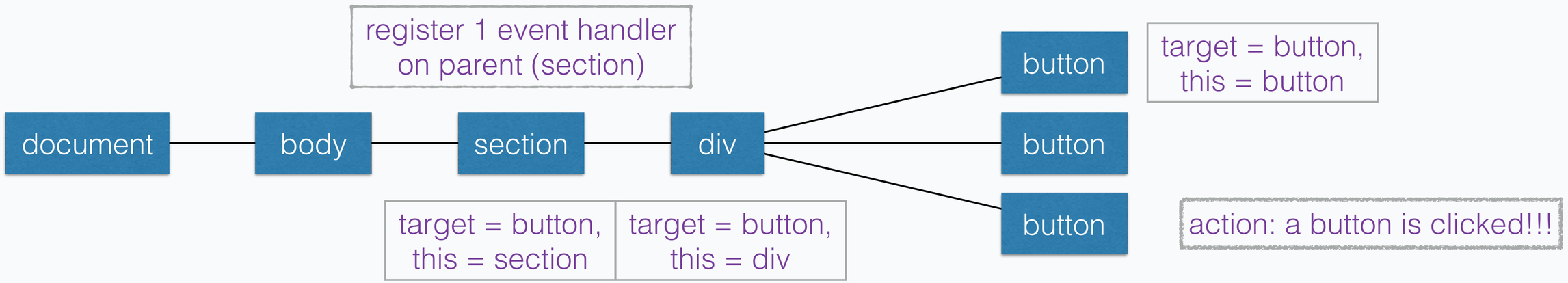
Event Propagation/Bubbling

- An event is directed to its intended target
  - If there is an event handler it is triggered
- From here, the **event** *bubbles* up to the containing elements
- This continues to the document element itself



Event Delegation

- The process of using event propagation to handle events at a higher level in the DOM
- Allows for a single event listener



workshop