Accordion with animated height

Probably most of frontend developers are familiar with jQuery UI accordion component. It's been around for a while (close to two decades now) and it's concept is widely used. You can bump into it as a FAQ component on almost every website that introduces that section. We are talking about following markup.

```
<div id="accordion">
    <div class="row">
        <div class="question">question?</div>
        <div class="answer">answer</div>
    </div>
    <div class="row">
        <div class="question">question?</div>
        <div class="answer">answer</div>
    </div>
    <div class="row">
        <div class="question">question?</div>
        <div class="answer">answer</div>
    </div>
</div>
```

which renders into something like this:

```
+-------------------------+
| question?               |
+-------------------------+

+-------------------------+
| question?               |
+-------------------------+

+-------------------------+
| question?               |
+-------------------------+
```

and once you click on a question box you get this:

```
+-------------------------+
| question?               |
+-------------------------+
| answer                  |
+-------------------------+
+-------------------------+
| question?               |
+-------------------------+

+-------------------------+
| question?               |
+-------------------------+
```

The questionmark here is if there is a “modern” way how to code such component with no or very little Javascript and CSS animations. Is it possible reliably animate an element height without having set absolute maximum height? Is it possible to deal with inner padding in terms of animations? Will you need to chain animations and mess around with delays and stuff? Is there a way how to code this whole component in a way that hasn’t been available back then like flex or CSS grid? Well based on this answer it looks like it’s possible.

The whole solution is based on CSS grid The grid gives you the ability to split each Q&A row into 2 pieces. The first piece is left intact since you don’t want to manipulate it at all. The second piece - which it the one that all this story is about - needs to be foldable and animated. So how to do that? First let’s prepare the template:

```html
<div id="accordion">
    <div class="row">
        <div class="question">question?</div>
        <div class="answer">
            <div class="wrapper">
                <div>answer</div>
            </div>
        </div>
    </div>
    <div class="row">
        <div class="question">question?</div>
        <div class="answer">
            <div class="wrapper">
                <div>answer</div>
            </div>
        </div>
    </div>
    <div class="row">
        <div class="question">question?</div>
        <div class="answer">
            <div class="wrapper">
                <div>answer</div>
            </div>
        </div>
    </div>
</div>
```

We introduced a new nested `div.wrapper` which wraps whole content of the answer and is the place where you want to apply stuff like padding etc. That strikes out additional problems with animations and box layout.

The main thing here is to set `.row` as a grid and have `.question` and `.answer` as grid columns. Now we can set the grid template that will represent our inital state.

```
.row {
    display: grid;
    grid-template-rows: min-content 0fr;
}
```

Here we literally say “`.head` is gonna wrap it’s content and the other piece which is `.answer` is gonna take 0 space (fraction)”. That’s how you get the state where all answers are collapsed.

Now we need to introduce a new class - `active`. This class is gonna sit on `.row` and gets toggled once user clicks on the `.question` block. This is the time where we write a few lines of Javascript.

```
document.querySelectorAll(".accordion .question").forEach(e => {
    e.addEventListener("click", function() {
        this.parentElement.classList.toggle("active")})
})
```

Once the `.row` is `.active` we can prepare CSS for that state:

```
.row.active {
    grid-template-rows: min-content 1fr;
}
```

That changes the grid from “content + nothing” to “content + 1 fraction” which exactly wraps around it’s content and additional padding you have set.

Adding nice annimation is just one line of CSS:

```
.row {
    transition: grid-template-rows 1s;
}
```

We are done here.