// Set oninput callback for each slider
sliders.forEach((slider) =>
    slider.addEventListener("input", update));

const update = function() {
    colorBox.style.background = `rgb(${sliders[0].value}, ${sliders[1].value}, ${sliders[2].value})`;
    sliders.forEach((slider, index) =>
        labels[index].innerHTML = slider.value);
};
Client-Server
(e.g. browser)

Internet

Site

HTTP & URI

HTML, JSON, ...

3-tier Architecture

Web Server
(e.g., Apache, NGinx)

App. Server
(e.g., NodeJS)

Database
(e.g., SQLite, PostgreSQL)

Presentation Tier

Logic Tier

Persistence Tier

Routing & Controllers
(e.g., NextJS)

Models
(e.g., knex, objection)

MVC

Client

Internet

Site
const wrapValue = (n) => { // function(n) {
    let local = n;
    return () => local; // function () { return local; }
}

let wrap1 = wrapValue(1); // () => 1
let wrap2 = wrapValue(2); // () => 2
console.log(wrap1()); // What will print here?
console.log(wrap2()); // What will print here?
function ColorPicker() {
  const [red, setRed] = React.useState(0);
  const [green, setGreen] = React.useState(0);
  const [blue, setBlue] = React.useState(0);

  const color = {
    background: `rgb(${red}, ${green}, ${blue})`
  };
  return (
    <div>
      <div className="color-swatch" style={color} />
      <LabeledSlider label="Red" value={red} setValue={value => setRed(value)} />
      <LabeledSlider label="Green" value={green} setValue={value => setGreen(value)} />
      <LabeledSlider label="Blue" value={blue} setValue={value => setBlue(value)} />
    </div>
  );
}

Props down!  Callbacks up!

Single source of truth!
<table>
<thead>
<tr>
<th>Route</th>
<th>Controller Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST /api/films</td>
<td>Create new movie from request data</td>
</tr>
<tr>
<td>GET /api/films/:id</td>
<td>Read data of movie with id == :id</td>
</tr>
<tr>
<td>PUT /api/films/:id</td>
<td>Update movie with id == :id from request data</td>
</tr>
<tr>
<td>DELETE /api/films/:id</td>
<td>Delete movie with id == :id</td>
</tr>
<tr>
<td>GET /api/films</td>
<td>List (read) all movies</td>
</tr>
</tbody>
</table>

```javascript
nc.get(async (req, res) => {
  const films = Film.query().withGraphFetched('genres');
  res.status(200).send(films);
});
```
<table>
<thead>
<tr>
<th></th>
<th>Relational (RDBMS)</th>
<th>Non-Relational</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data</strong></td>
<td>Table-oriented</td>
<td>Document-oriented, key-value, graph-based, column-oriented, ...</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Fixed schema</td>
<td>Dynamic schema</td>
</tr>
<tr>
<td><strong>Joins</strong></td>
<td>Used extensively</td>
<td>Used infrequently</td>
</tr>
<tr>
<td><strong>Interface</strong></td>
<td>SQL</td>
<td>Custom query language</td>
</tr>
<tr>
<td><strong>Transactions</strong></td>
<td>ACID</td>
<td>CAP</td>
</tr>
</tbody>
</table>

SELECT * FROM people WHERE age > 25;

db.people.find(
    { age: { $gt: 25 } }
)
Google testing blog

Kent C Dodds “Write tests. Not too many. Mostly integration.”
Main is always “deployable”
• Tests pass
• No incomplete features

Short-lived branch for single feature

```plaintext
main
feature
```

```plaintext
git checkout -b feature

feature

main

```

```plaintext
git checkout main
git merge feature

Make sure tests pass
git commit -m "..."
...
```
<table>
<thead>
<tr>
<th>Film</th>
</tr>
</thead>
<tbody>
<tr>
<td>Responsibility</td>
</tr>
<tr>
<td>Knows its title</td>
</tr>
<tr>
<td>Knows its plot overview</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Know which genres it is</td>
</tr>
</tbody>
</table>

(Figure 4.4, *Engineering Long Lasting Software* by Armando Fox and David Patterson, Alpha edition, 2012.)
“Scrum-ish” (in a nutshell)

- Sprint Goal
- Sprint Planning
- Product Backlog

Sprint Backlog

- Feature
- Feature

Sprint Planning

- Feature
- Feature
- Feature
- Feature

Sprint (2-4 weeks)

- 24 hours between “standup” meetings
- “Deployable” product increment

Sprint Demo & Retrospective

Frequent feedback!
Iterative Incremental

http://itsadeliverything.com/revisiting-the-iterative-incremental-mona-lisa
2009 World Touring Car Championship

2013 McGrath Cup
2015 Canadian Grand Prix

2009 World Touring Car Championship

The 2009 World Touring Car Championship season was the sixth FIA World Touring Car Championship season, the fifth since its 2005 return. It began on 8 March, and ended on 22 November, after twenty-four races. The championship, which was reserved for Super 2000 Cars and Diesel 2000 Cars, comprised two titles, the FIA World Touring Car Champion for Drivers and the FIA World Touring Car Champion for Manufacturers. Italian Gabriele Tarquini won the Drivers' Championship by four points from SEAT Sport team-mate Yvan Muller, with BMW Team Germany's Augusto Farfus finishing third ten points behind Muller. The Manufacturers' Championship was retained by SEAT beating BMW by just three points. Tom Coronel took the Independents' Trophy for SUNRED Engineering, which won the Teams' Trophy.

11/8/2016, 8:04:50 AM

CS 312 Practical: CSS Frameworks
SMALL (So Many Acronyms Littering the Lectures)

- F.I.R.S.T.
- I.N.V.E.S.T.
- R.A.S.P.
- DRY
- SoC
- SOFA
- SOLID
- SaaS
- TDD, BDD
- MVC
- WISBNWIW
- ACID
- CRUD(L)

- HTML / DOM / CSS / JSX
- CI / CD
- UI
- AJAX
- REST API
- URI / URL
- TCP/IP
- JSON
- CRC
- ORM
- POJO
- SQL / RDMS
- VCS
- SLO / SLA
Take-aways

• Behind every design decision there should be a user story (a stakeholder and a motivation!)
• Testing, not just a class requirement, it’s a good idea
• Develop iteratively and incrementally
• There should be one source of truth
• Don’t repeat yourself
• Don’t mutate props or state
• Do really read error messages
• Automate all the things
• Don’t break the “Build”
• Program strategically not tactically