Why Elm?

Problem #1

Implicit Type Coercion & Weak Typing

Problem #1

Implicit Type Coercion & Weak Typing

```
function withDefaults(value, array) {
  for (let i =0; i < array.length; i += 1) {
    if (!array[i]) {
      array[i] = value;
    }
  }
  return array;
}

withDefaults(14, [1,2,3,4]);
withDefaults(14, [42,null,null,9]);
withDefaults(14, [0, 5, null]);
```

Problem #1

Implicit Type Coercion & Weak Typing

```
function withDefaults(value, array) {
  for (let i =0; i < array.length; i += 1) {
    if (!array[i]) {
      array[i] = value;
    }
  }
  return array;
}

withDefaults(14, [1,2,3,4]);
withDefaults(14, [42,null,null,9]);
withDefaults(14, [0, 5, null]);

// [1,2,3,4]
// [42,14,14,9]
// [14,5,14]
```

Solution #1

Strong Typing

Problem #2

Errors Only Raised at Runtime

Problem #2

Errors Only Raised at Runtime

```
function titleize(str) {
  let words = str.split(" ");
  return words
    .map(word => `${word[0].toUpperCase()}${word.slice(1)}`)
    .join(" ");
}

titleize("why elm?") // Why Elm?
```

Problem #2

Errors Only Raised at Runtime

```
function titleize(str) {
  let words = str.split(" ");
  return words
    .map(word => `${word[0].toUpperCase()}${word.slice(1)}`)
    .join(" ");
}

titleize("why elm?") // Why Elm?

titleize(null); // TypeError: Cannot read property 'split' of null
titleize(""); // TypeError: Cannot read property 'toUpperCase' of undefined
```

Solution #2

Static Typing

Problem #3

Refactoring & Maintaining Code

Problem #3

Refactoring & Maintaining Code

```
function request(url, callback) {
    ...
}
```

Problem #3

Refactoring & Maintaining Code

```
function request(url, callback) {
    ...
}
```

```
function request(url, { callback, headers }) {
    ...
}
```

Problem #3

Refactoring & Maintaining Code

```
function request(url, callback) {
    ...
}
```

```
function request(url, { callback, headers }) {
    ...
}
```

```
function request(url, { headers }) {
    return Promise(...);
}
```

Solution #3

(Friendly) Compiler Errors

Problem #4

Documentation & Code Hard to Keep in Sync

Problem #4

Documentation & Code Hard to Keep in Sync

```
/**
 * Create a secure password hash from a password string.
 *
 * @param {string} password - The password to hash
 * @param {Object} [opts] - See below
 * @param {string} [opts.salt] - The salt used to create the hash
 * @param {number} [opts.keylen=256] - Length of the underlying hashing key
 * @param {string} [opts.digest='sha256'] - The digest algorithm to use
 * @returns {Password} An opaque representation of a password and its
 *   corresponding hexadecimal digest.
 */
static hash(password, opts = {}) {
  const salt = opts.salt || crypto.randomBytes(64).toString('hex');
  const iterations = opts.iterations || Config.passwords.iterations;
  const keylen = opts.keylen || 512;
  const digest = opts.digest || 'sha512';

  const hash = crypto
    .pbkdf2Sync(password, salt, iterations, keylen, digest)
    .toString('base64');

  return new Password({ hash, salt, iterations, keylen, digest });
}
```

Solution #4

Types as Documentation

Problem #5

Imperative Code is Verbose and Error-Prone

Problem #5

Imperative Code is Verbose and Error-Prone

```
function sum(values) {
  let total = 0;
  for (let i = 0; i <= values.length; i += 1) { total += values[i]; }
  return total;
}

sum([1,2,3,4,5]);
```

Problem #5

Imperative Code is Verbose and Error-Prone

```
function sum(values) {
  let total = 0;
  for (let i = 0; i <= values.length; i += 1) { total += values[i]; }
  return total;
}

sum([1,2,3,4,5]);

// NaN
```

Solution #5

Declarative Approach

Problem #6

Mutable Data-Structures are Hard to Reason About

Problem #6

Mutable Data-Structures are Hard to Reason About

```
const re = new RegExp(/[a-f0-9]{7}/, 'g');

const commits = `
053eede Bump version to 4.0.1
15e4e82 Make token_type parsing case-insensitive
`

while(res = re.exec(commits)) {
  console.log(res[0]);
}
console.log(re.exec(commits)[0]);
```

Problem #6

Mutable Data-Structures are Hard to Reason About

```
const re = new RegExp(/[a-f0-9]{7}/, 'g');

const commits = `
053eede Bump version to 4.0.1
15e4e82 Make token_type parsing case-insensitive
`

while(res = re.exec(commits)) {
  console.log(res[0]);
}
console.log(re.exec(commits)[0]);

// 053eede
// 15e4e82
// 053eede
```

Problem #6

Mutable Data-Structures are Hard to Reason About

```
const re = new RegExp(/[a-f0-9]{7}/, 'g');

const commits = `
053eede Bump version to 4.0.1
15e4e82 Make token_type parsing case-insensitive
`

console.log(re.exec(commits)[0]);
console.log(re.exec(commits)[0]);
console.log(re.exec(commits)[0]);



// 053eede
// 15e4e82
// 053eede
```

Problem #6

Mutable Data-Structures are Hard to Reason About

```
const re = new RegExp(/[a-f0-9]{7}/, 'g');

const commits = `
053eede Bump version to 4.0.1
15e4e82 Make token_type parsing case-insensitive
`

console.log(re.exec(commits)[0]);
console.log(re.exec(commits)[0]);
console.log(re.exec(commits)[0]);



// 053eede
// 15e4e82
// TypeError: Cannot read property '0' of null
```

Solution #6

Immutable Data-Structures

Problem #7

Dealing With the Outside World

Problem #7

Dealing With the Outside World

```
function getJSON(url) {
  return new Promise(resolve => {
    let req = new XMLHttpRequest()
    req.addEventListener("Load", () => resolve(JSON.parse(req.responseText)));
    req.open("GET", url);
    req.send();
  });
}

getJSON("https://swapi.co/api/people/1").then(res => ...);
getJSON("https://swapi.co:1337/api/people/1").then(res => ...);
getJSON("https://swapi.co/api/patate/1").then(res => ...);
```

Solution #7

Managed Side-Effects

Problem #8

Upgrading Dependencies Without Breaking Things

Problem #8

Upgrading Dependencies Without Breaking Things

```
npm i react-dom@latest --please-dont-break-my-app-(too-much)
```

Solution #8

Enforced Semantic Versioning

Problem #9

Trade-off Between Dependencies & Assets Size

Problem #9

Trade-off Between Dependencies & Assets Size

Vue (2.5) 100kb

Angular (6) 93kb

React (16.4) 77kb

Problem #9

Trade-off Between Dependencies & Assets Size

Vue (2.5) 100kb

Angular (6) 93kb

React (16.4) 77kb

momentjs 82kb

react-router 31kb

material-ui 314kb

immutable-js 36kb

Problem #9

Trade-off Between Dependencies & Assets Size

Vue (2.5) 100kb

Angular (6) 93kb

React (16.4) 77kb

(elm (0.19) 29kb)

momentjs 82kb

react-router 31kb

material-ui 314kb

immutable-js 36kb

Solution #9

Dead-Code Elimination

Problem #10

On-boarding Newcomers Takes Time

Problem #10

On-boarding Newcomers Takes Time

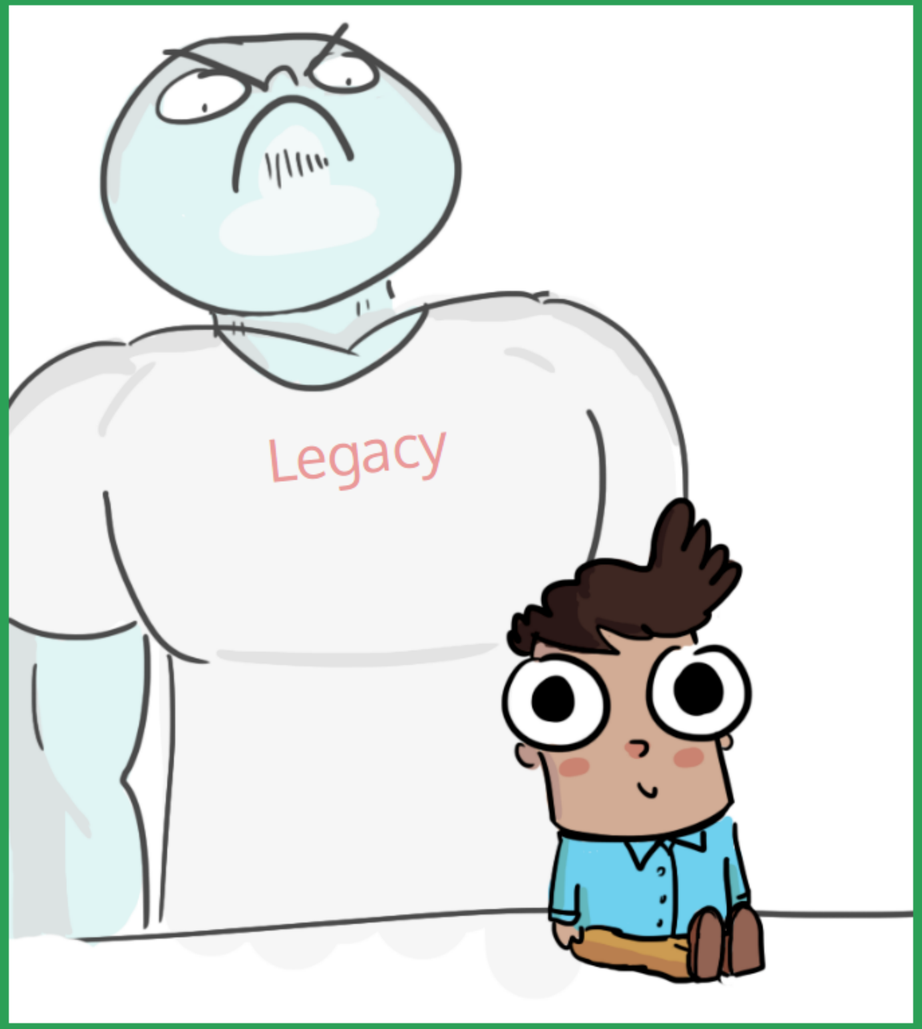

Image From: leftoversalad.com

Solution #10

Well-Defined & Common Standards

Why Elm?

- ♥ Strong & Static Typing ♥
- ♥ Side-Effects as Explicit Typed Commands ♥
- ♥ Readable & Composable Types Signatures ♥
- ♥ Enforced Semantic Versioning ♥
- ♥ Declarative, Stateless & Immutable Programming ♥
- ♥ Tiny Language, TEA, Small Learning Curve ♥
- ♥ Dead-code Elimination ♥