

Unbreakable

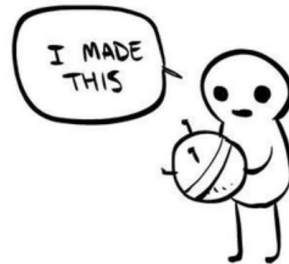
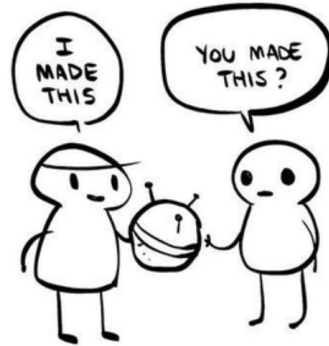
Unbreakable Craftsmanship



Steve Wozniak (left) and Steve Jobs (right) invents Apple II (1977) i.imgur.com

Submitted 1 month ago by Strazdas1

97 comments share save hide give gold report



“Woz designed all the hardware *and* all the circuit boards *and* all the software that went into the Apple II... not one bug has ever been found ... the circuit design of the Apple II is widely considered to be astonishingly beautiful, as close to perfection as one can get in engineering.”

-- Vikram Chandra *Geek Sublime*

Unbreakable

Unbreakable Craftsmanship

Unbreakable

Craftsmanship

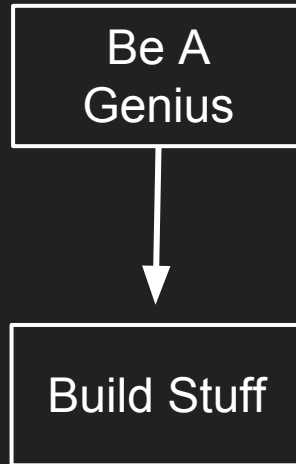
Doing something good for its own sake





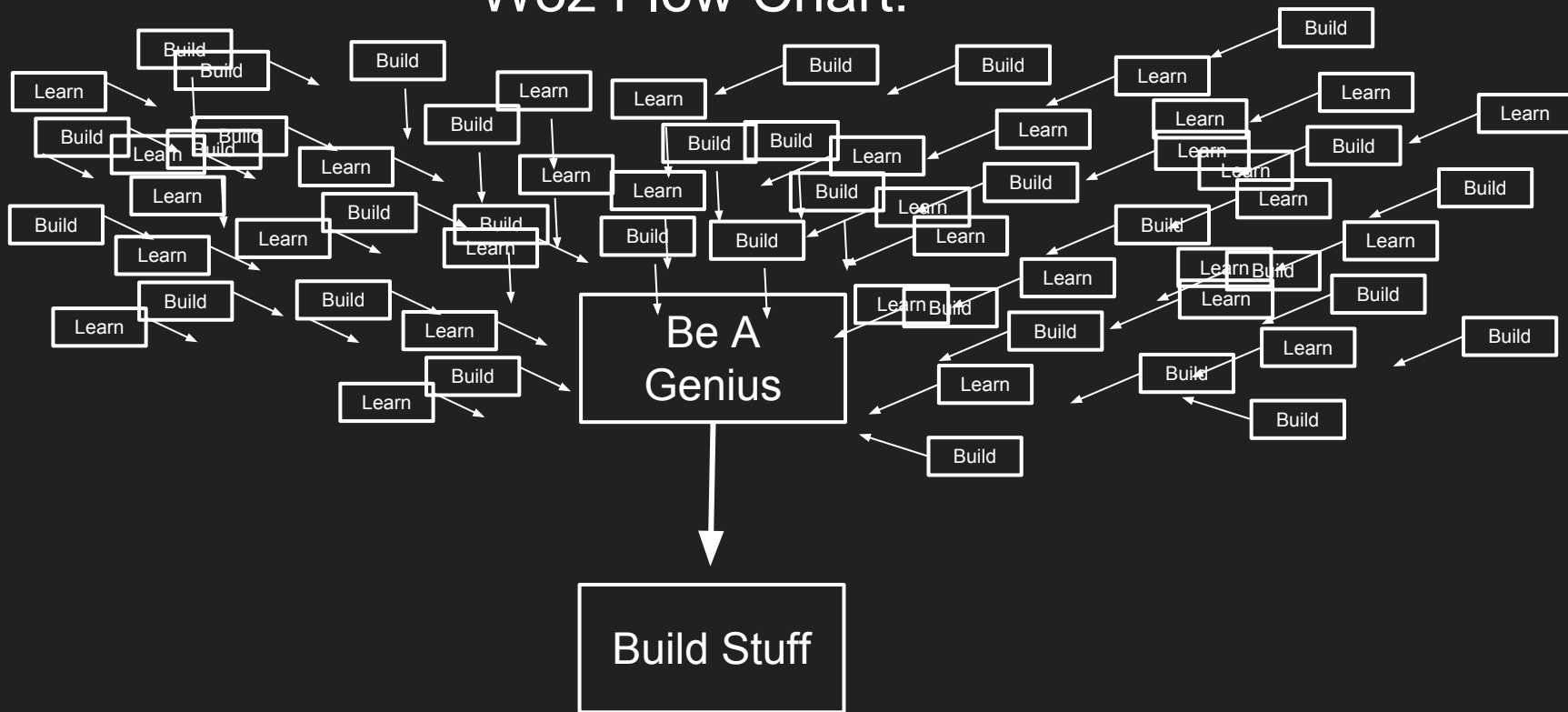
Woz Flow Chart:

Woz Flow Chart:





Woz Flow Chart:



Joe Morgan

Joe Morgan

@joesmorgan

Joe Morgan

@joesmorgan

Lawrence, KS

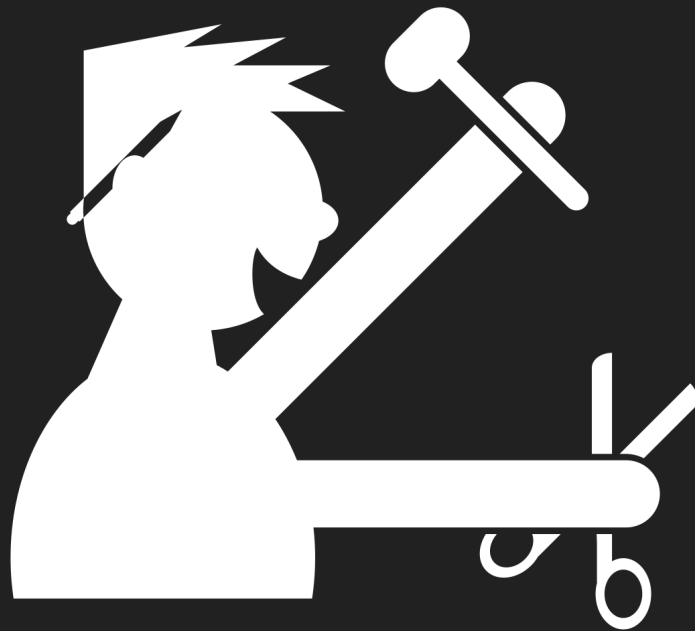
I write code

I write code

I try to get a little better everyday

Stages on craftsmanship

Apprentice



Apprentice



Apprentice: Guidance



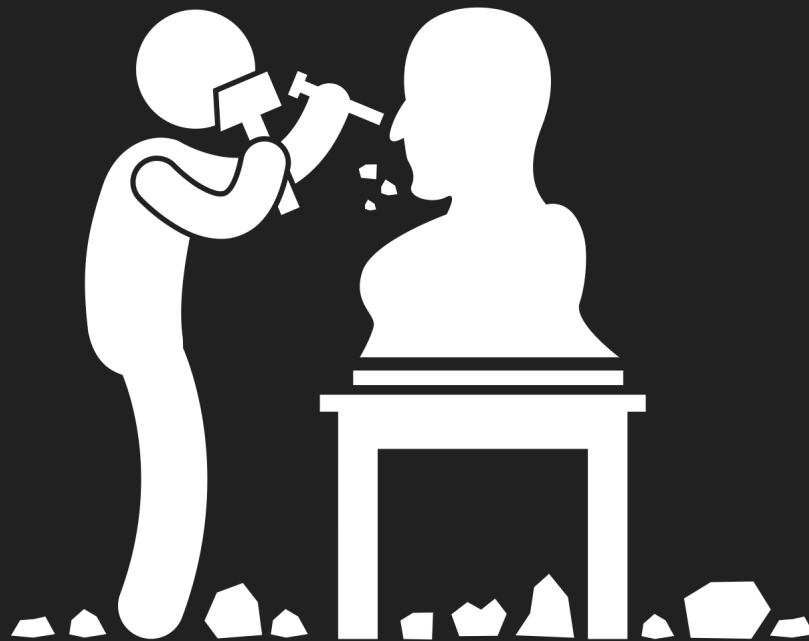
Journeyman



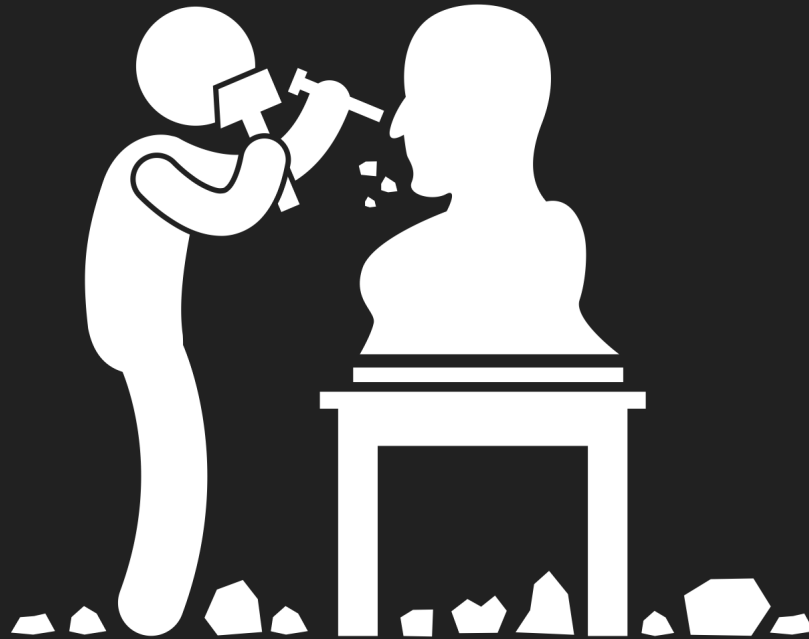
Journeymen: Experience and Intuition



Master



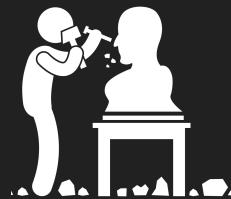
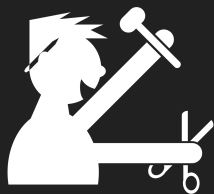
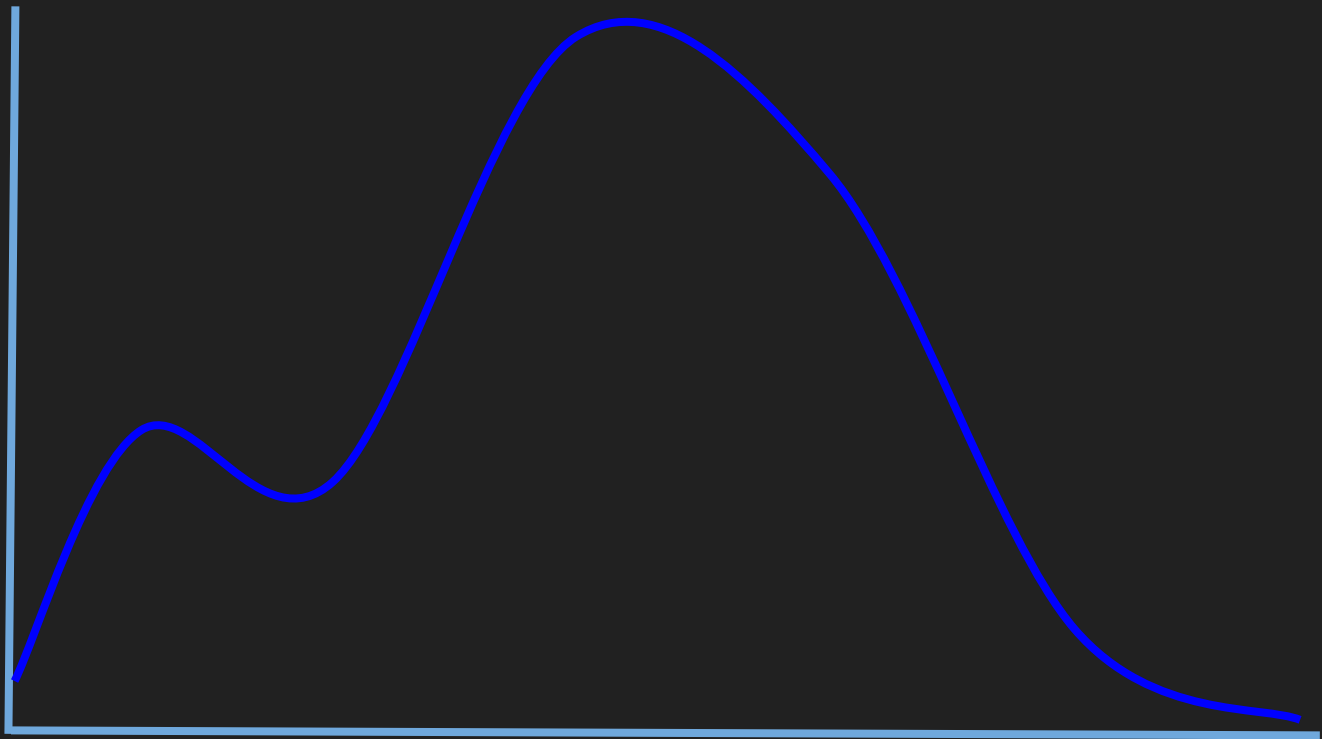
Master: Creativity

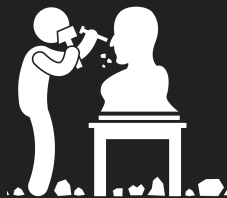
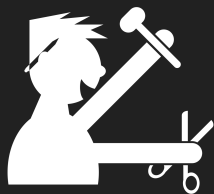
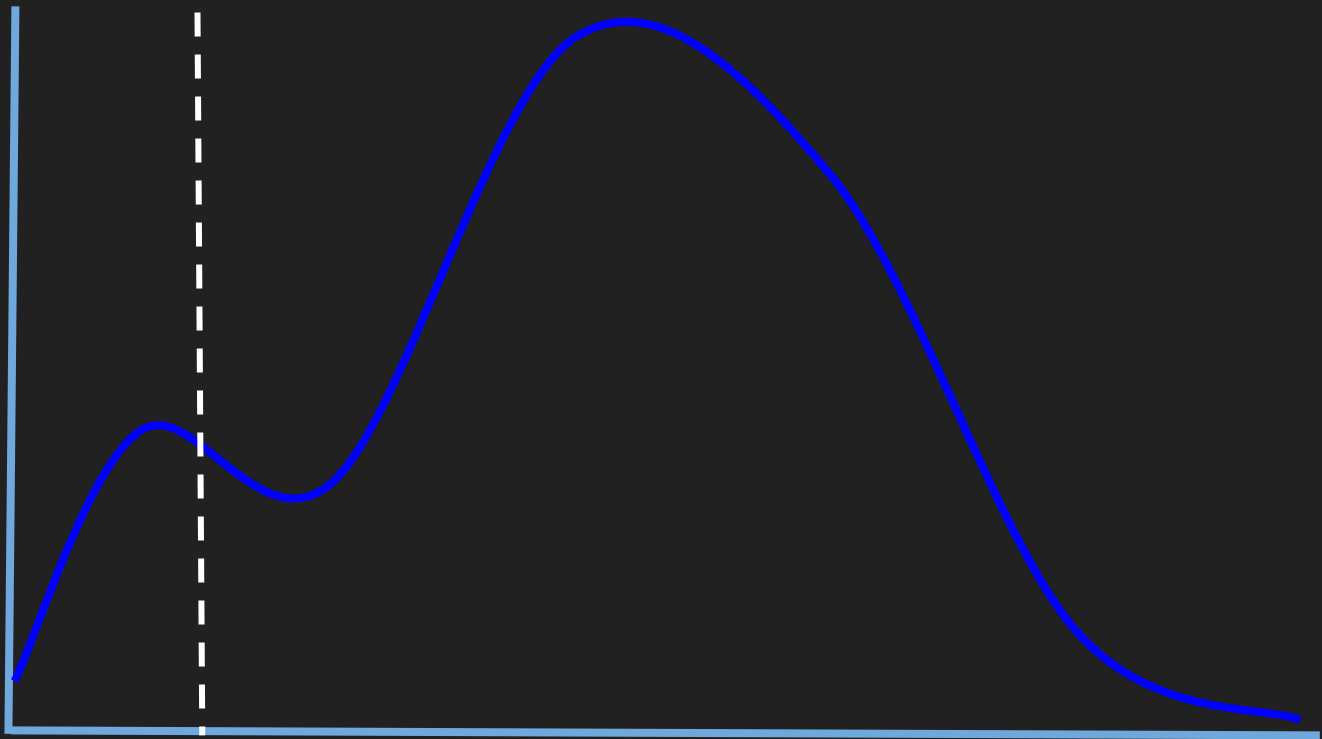


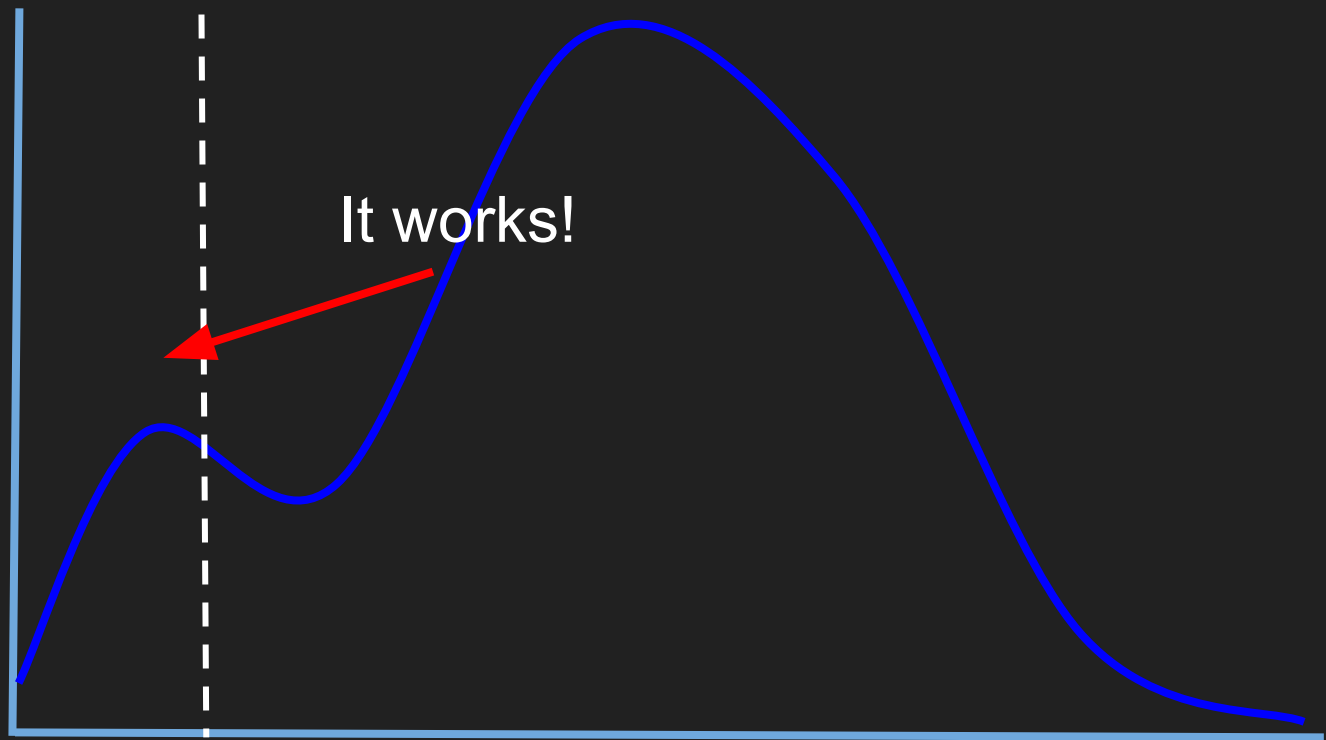
How does that relate to software?

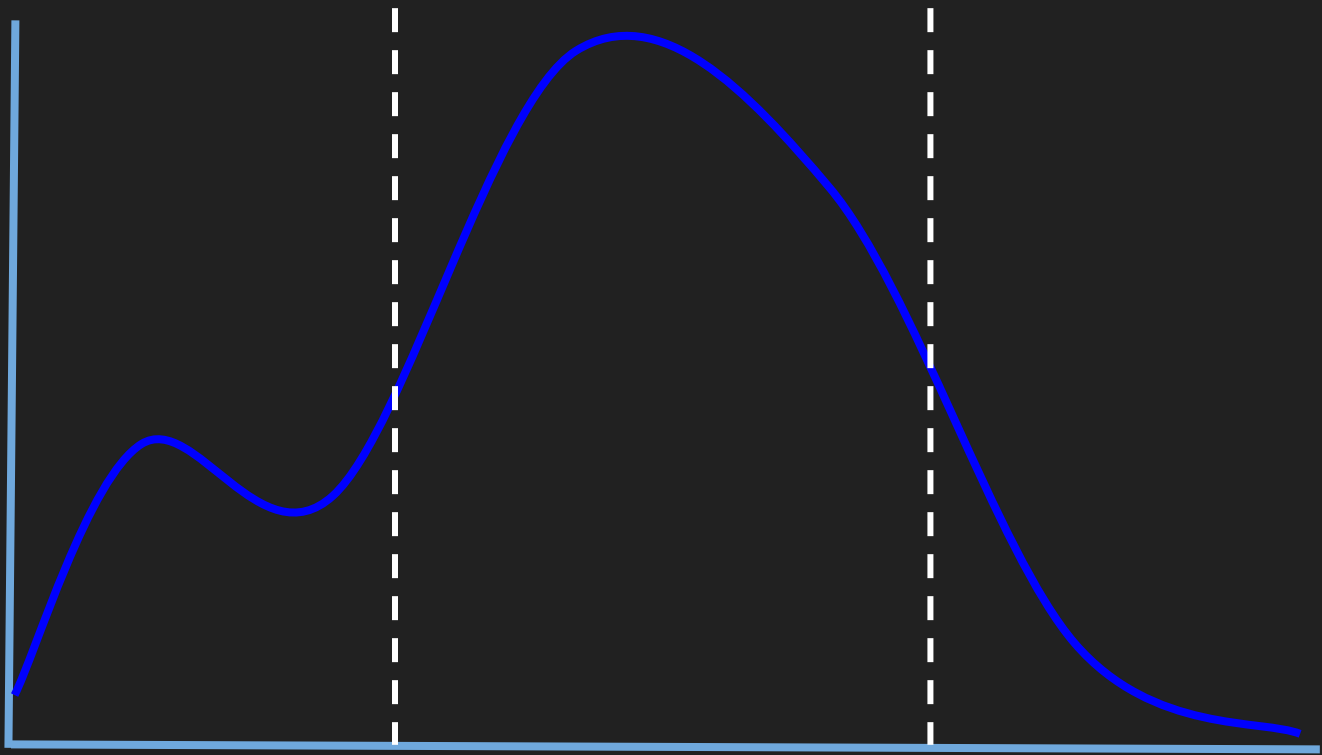
of Devs

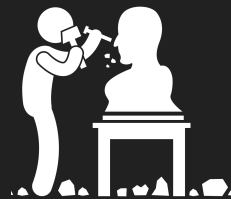
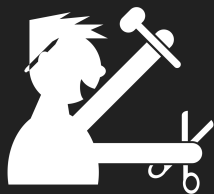
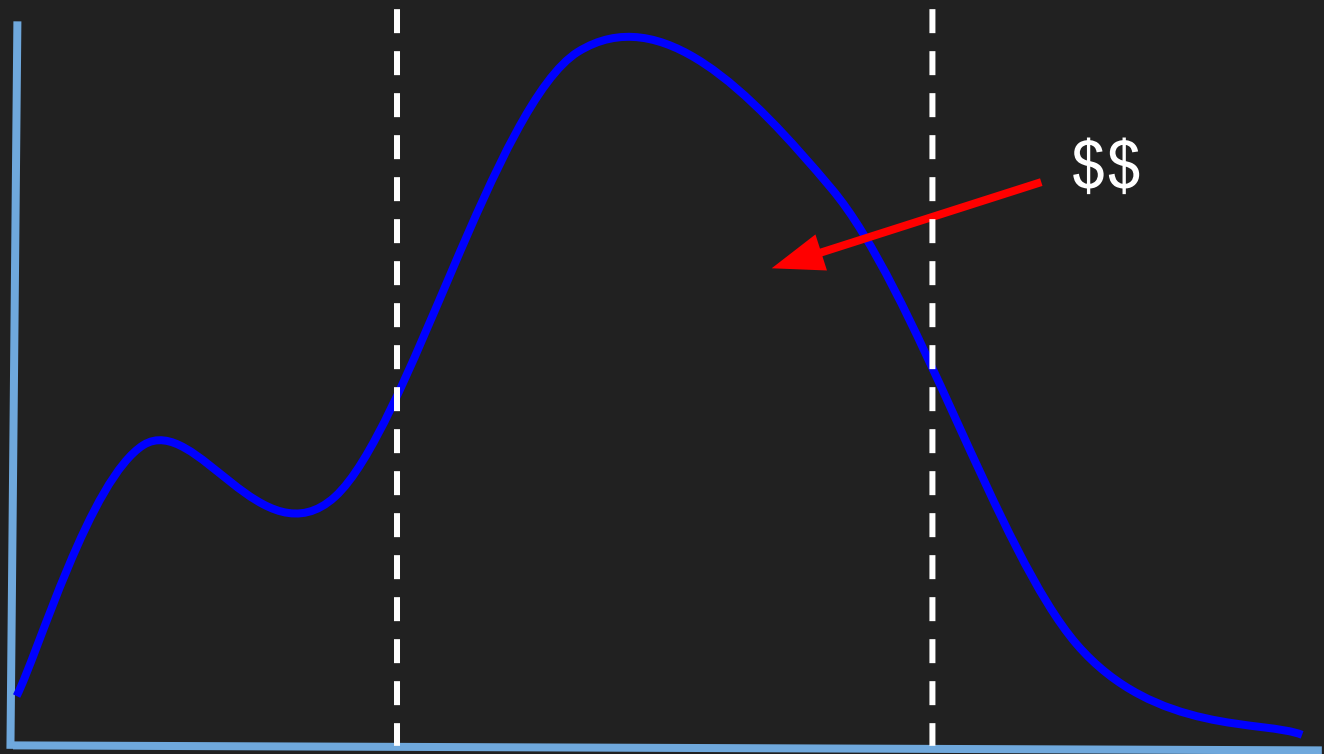


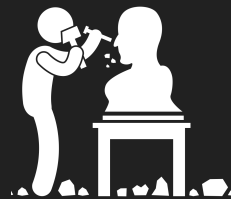
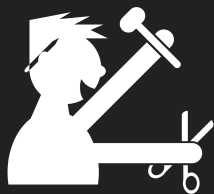
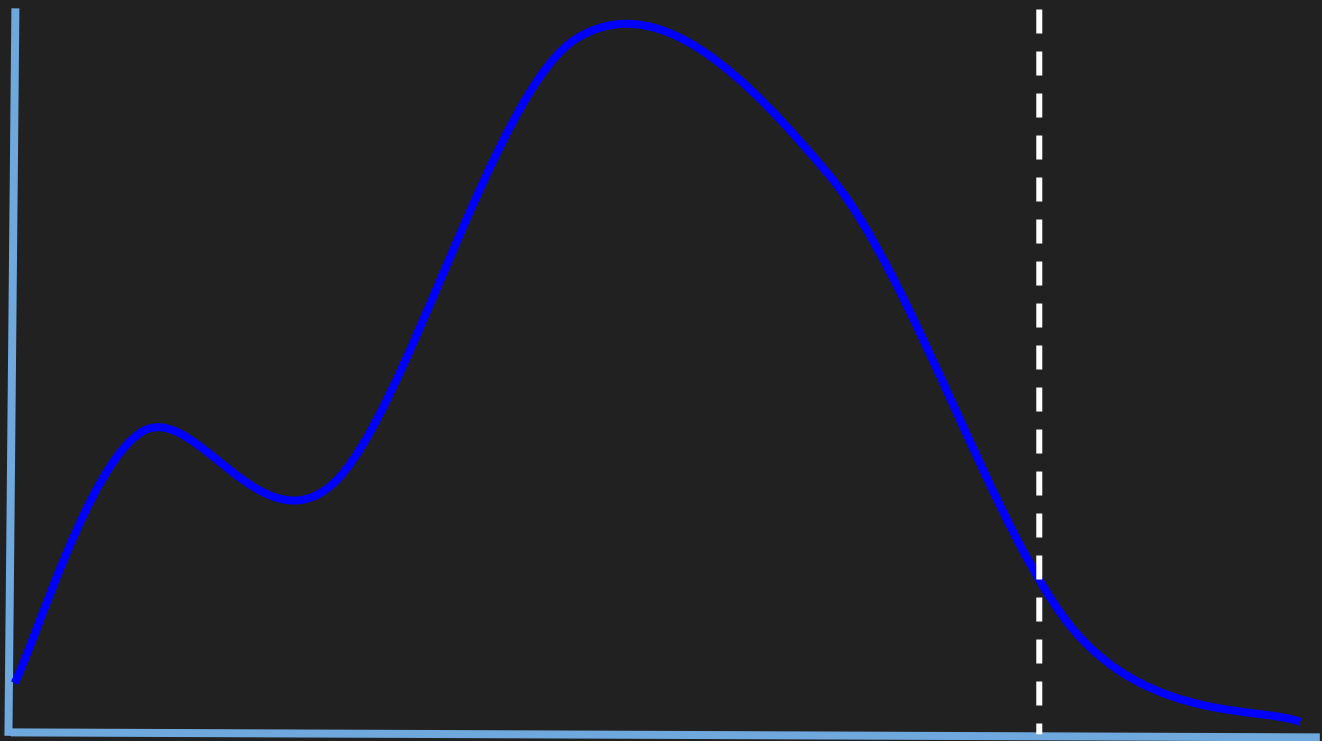


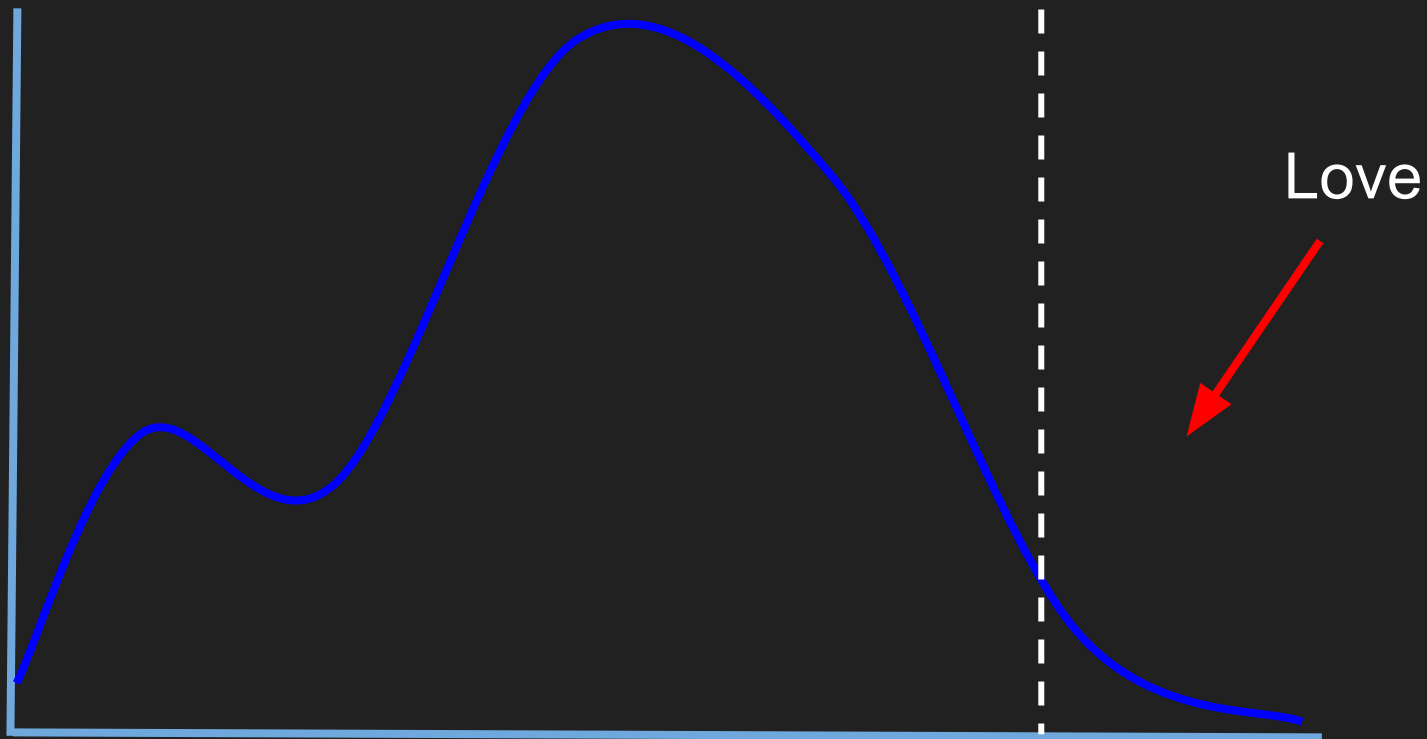




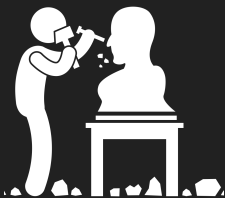








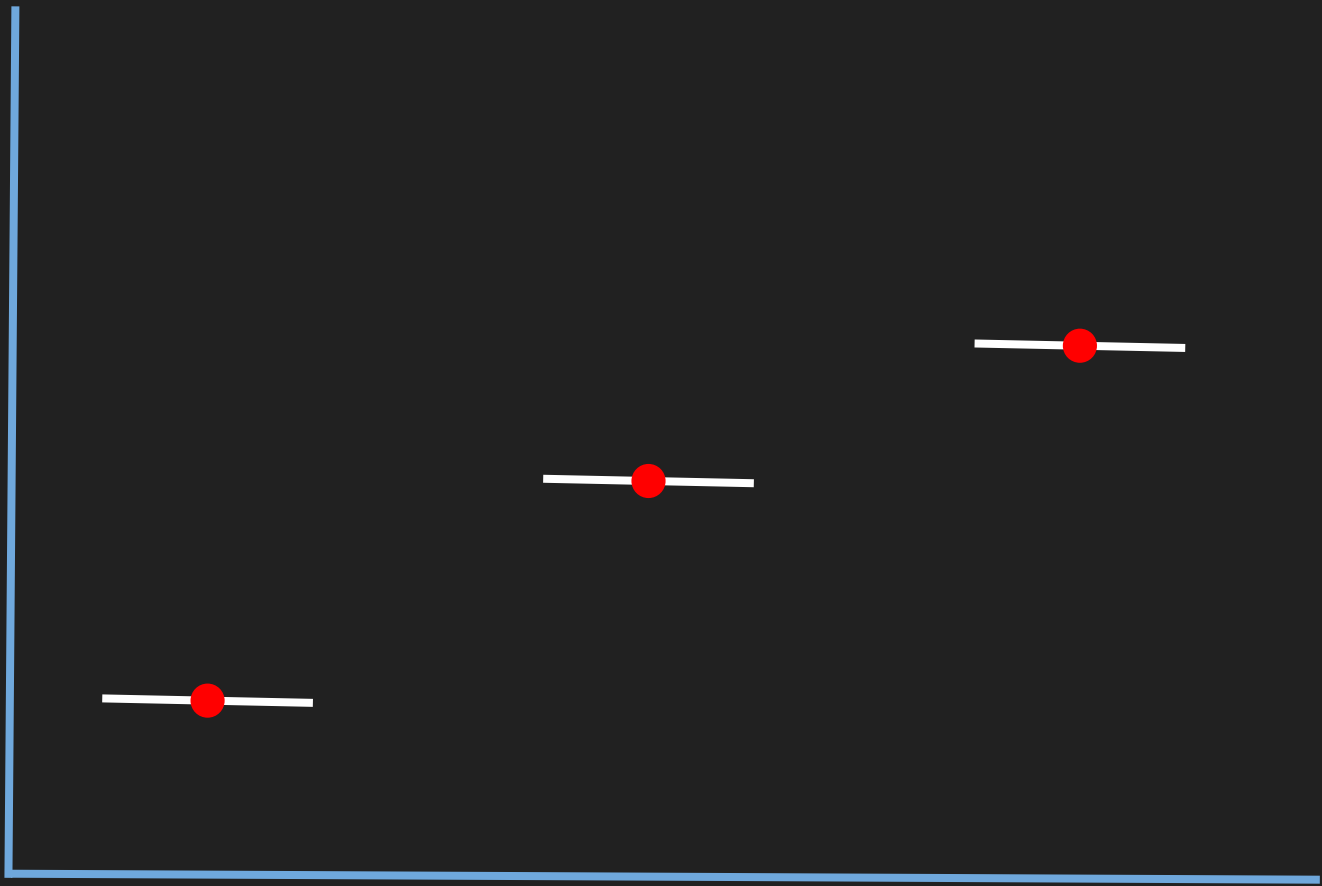
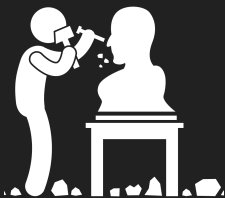
Self Evaluation



Servers

Server Code

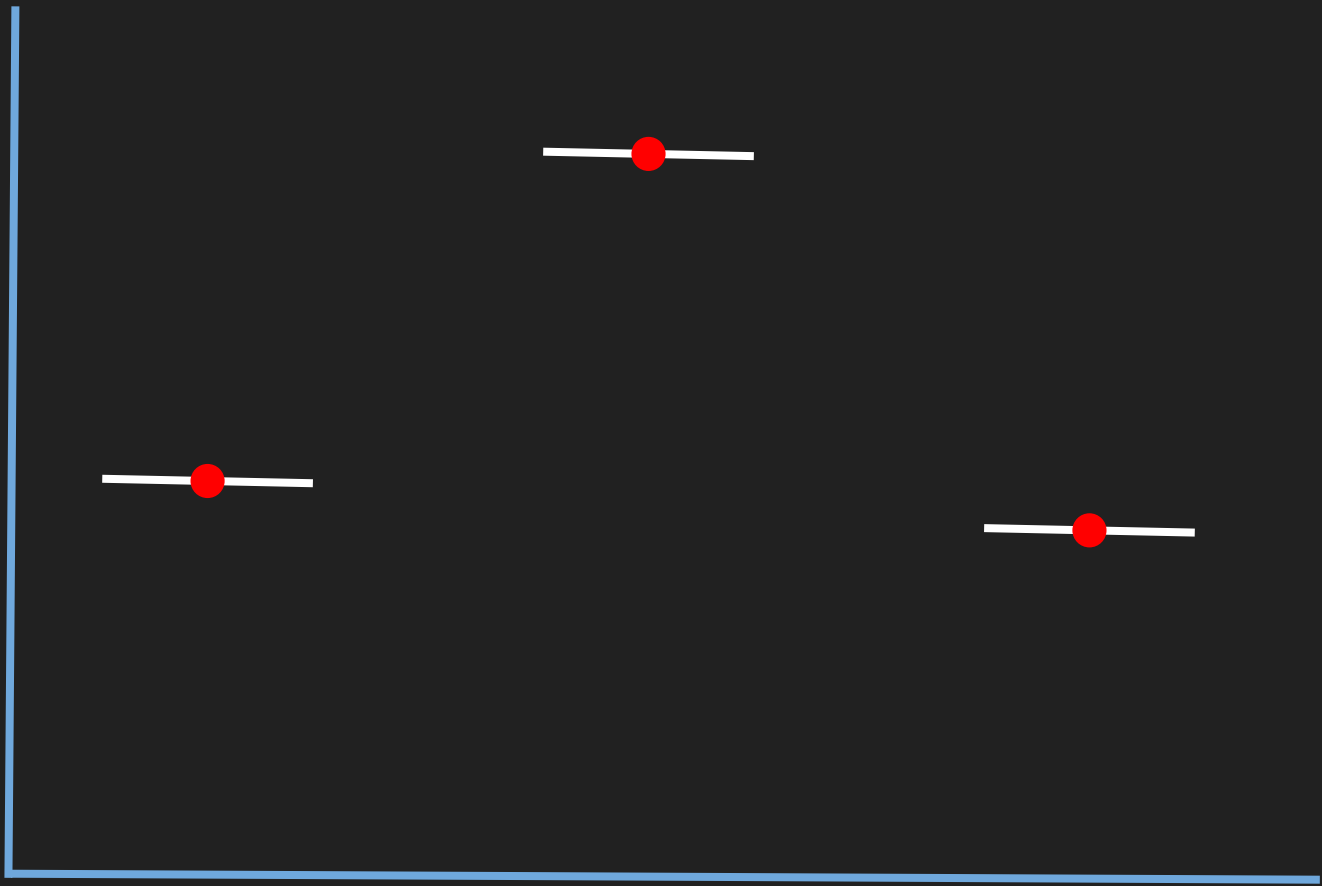
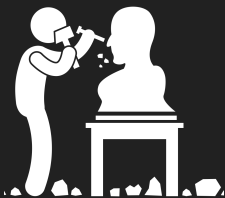
Frontend Code



Servers

Server Code

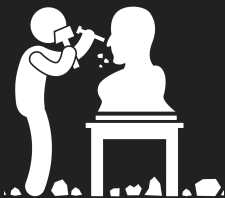
Frontend Code



Servers

Server Code

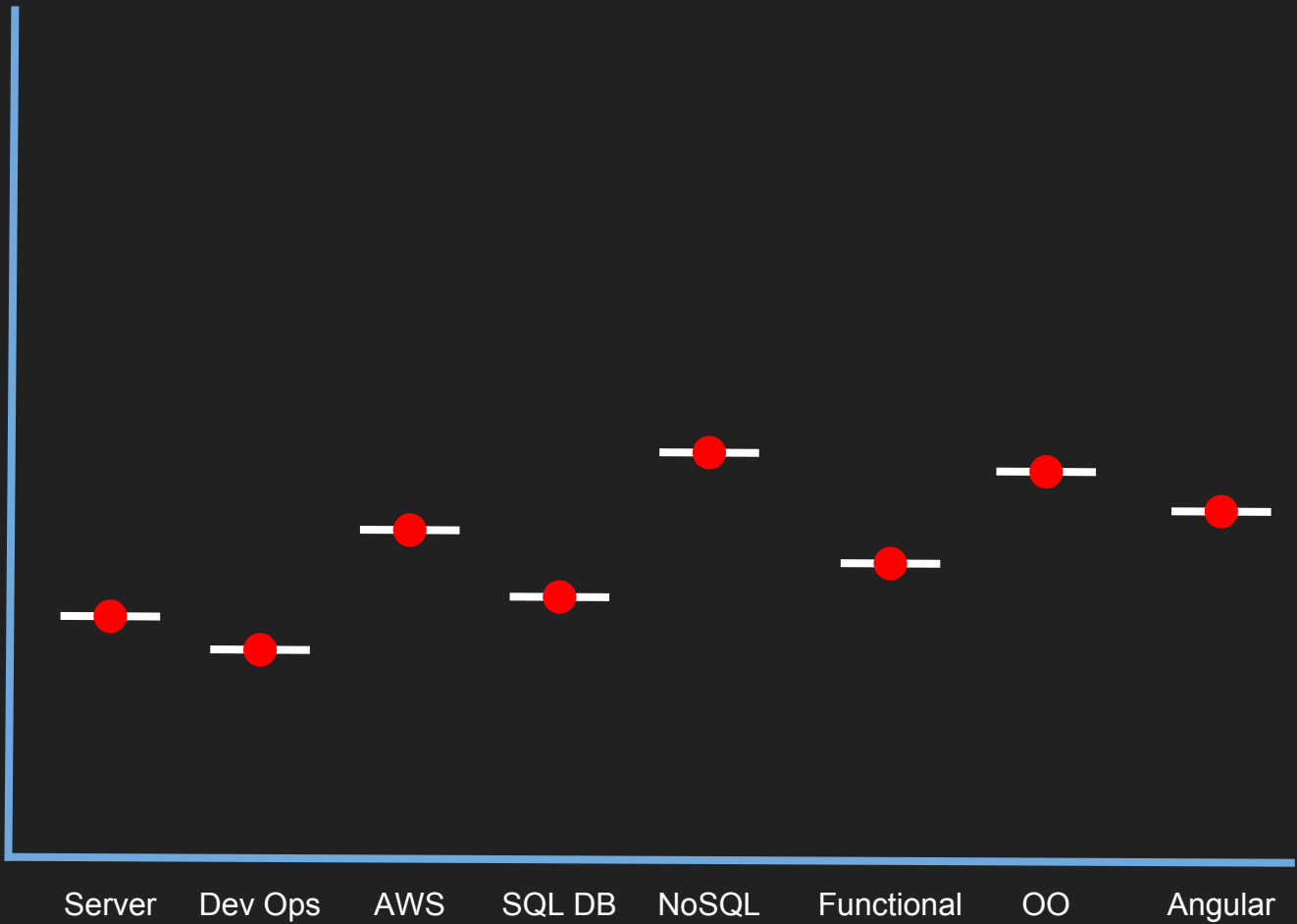
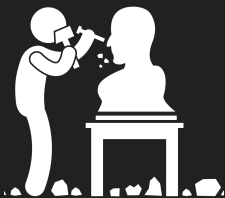
Frontend Code

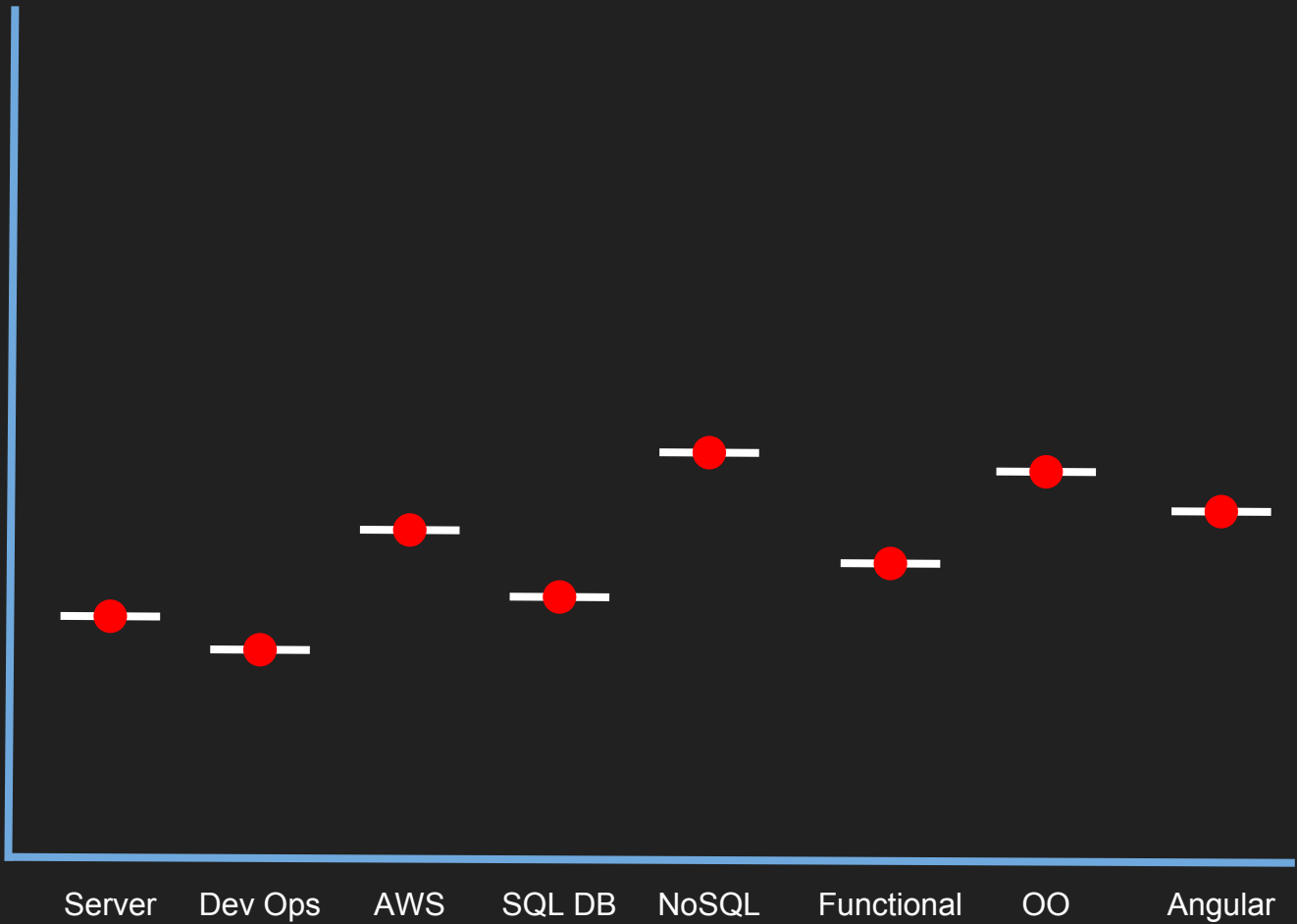
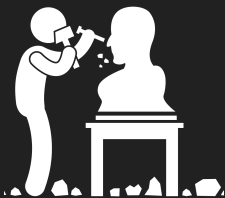


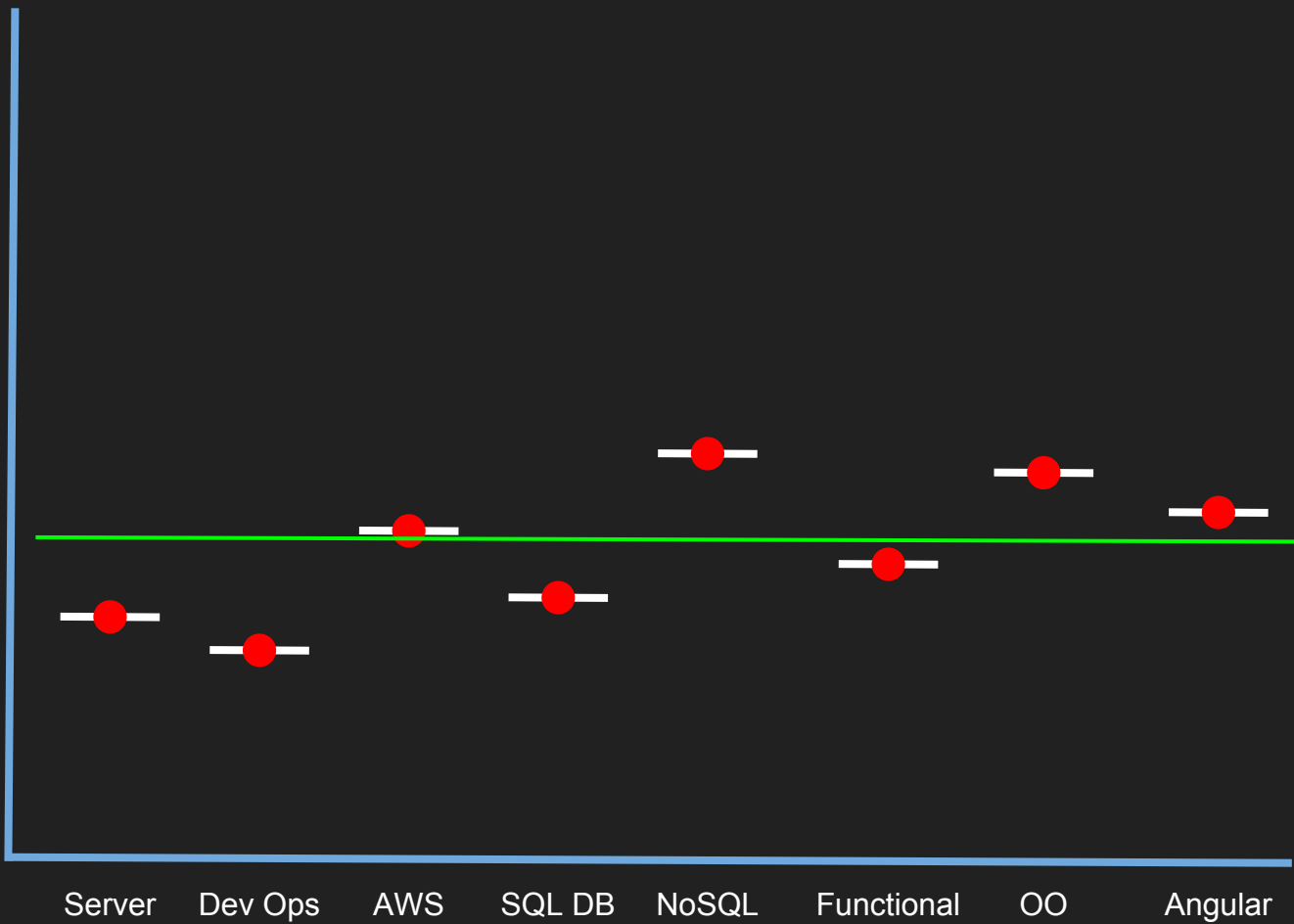
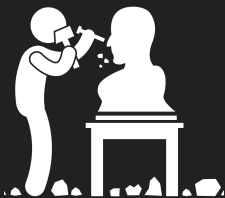
Servers

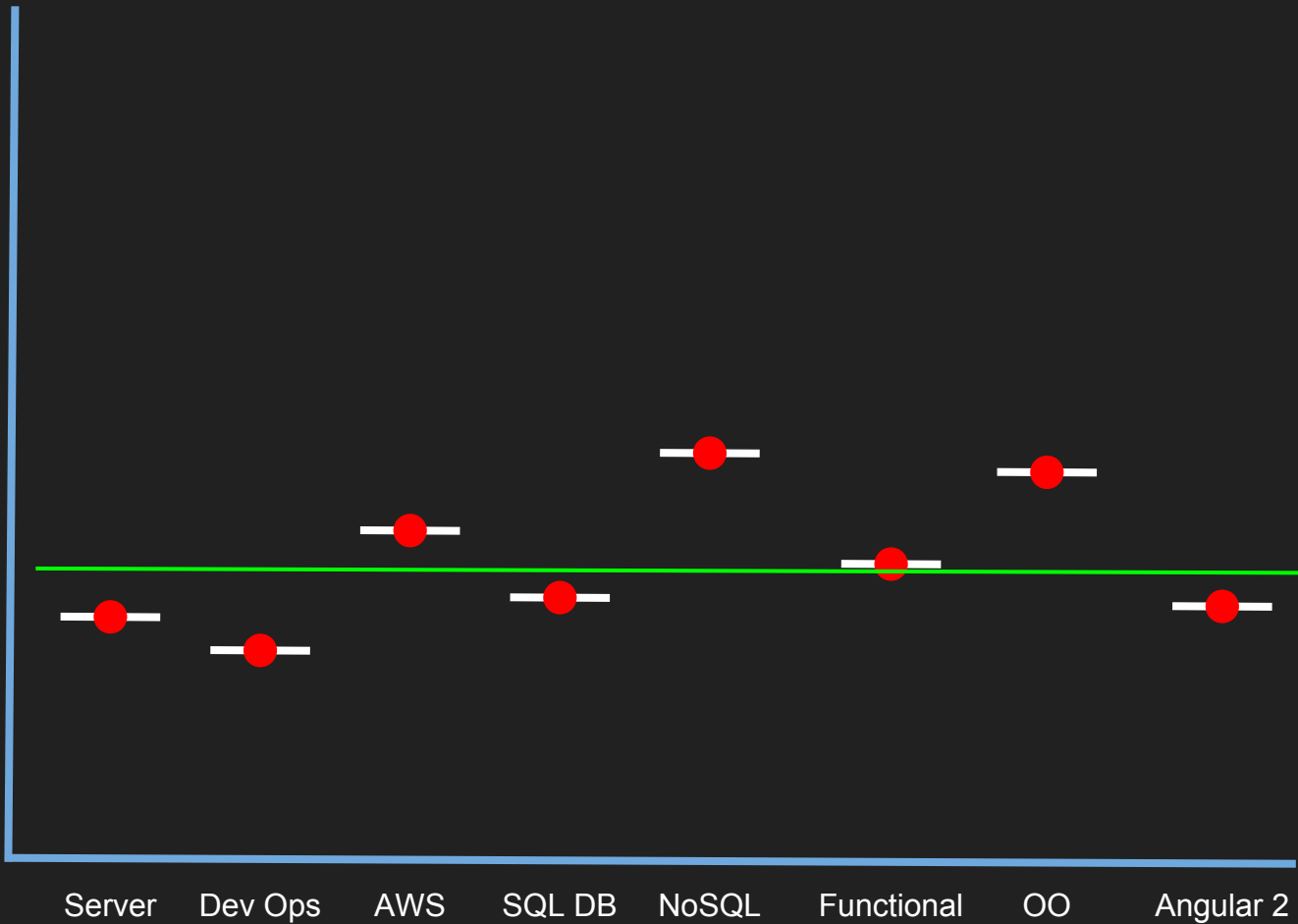
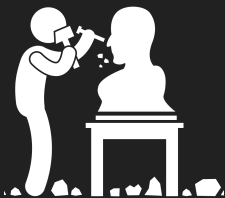
Server Code

Frontend Code









How do we improve?



Eye: What can you see?



Eye: What can you see?



Hand: What can you build?



Eye: What can you see?



Hand: What can you build?



Mind: What do you know?





Cheap: It's Chic, but Is It Good?

By WILLIAM L. HAMILTON OCT. 20, 2005

IT was probably only a matter of time before the big three big-box retailers -- Wal-Mart, Kmart and Target -- got interested in selling furniture.

Having succeeded in most other areas, with close to \$400 billion combined sales in the last fiscal year, they now have their eye on the last big apple on the tree, swinging seductively and ready to pick.

Wal-Mart, by virtue of its size, is already the largest furniture retailer in the country. Like Kmart and Target, it has traditionally offered inexpensive ready-to-assemble home-office furniture and storage items.

But in the last year the big three have made concerted efforts to sell stylish furniture for every part of the house at prices in line with the bargain-minded goods in their other aisles and online.





[The chipping paint] could be an example of machine-cutting "without sharp blades," adding "it looks like they painted before they actually cut the piece."





[The chipping paint] could be an example of machine-cutting "without sharp blades," adding "it looks like they painted before they actually cut the piece."









[The chipping paint] could be an example of machine-cutting "without sharp blades," adding "it looks like **they painted before they actually cut the piece.**"





```
$offset = $argv[0];
```

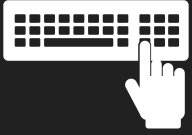
```
$query = "SELECT id, name FROM products ORDER BY name LIMIT 20  
        OFFSET $offset;;"
```

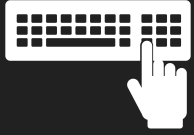
“Woz designed all the hardware *and* all the circuit boards *and* all the software that went into the Appie II... not one bug has ever been found ... the circuit design of the Apple II is widely considered to be astonishingly beautiful, as close to perfection as one can get in engineering.”

-- Vikram Chandra *Geek Sublime*

“Woz designed all the hardware *and* all the circuit boards *and* all the software that went into the Apple II... **not one bug** has ever been found ... the circuit design of the Apple II is widely considered to be **astonishingly beautiful**, as close to perfection as one can get in engineering.”

-- Vikram Chandra *Geek Sublime*





Hand: What we normally associate with craft



```
function createThunkMiddleware(extraArgument) {  
  return ({ dispatch, getState }) => next => action => {  
    if (typeof action === 'function') {  
      return action(dispatch, getState, extraArgument);  
    }  
  
    return next(action);  
  };  
}  
  
const thunk = createThunkMiddleware();  
thunk.withExtraArgument = createThunkMiddleware;  
  
export default thunk;
```



```
function createThunkMiddleware(extraArgument) {  
  return ({ dispatch, getState }) => next => action => {  
    if (typeof action === 'function') {  
      return action(dispatch, getState, extraArgument);  
    }  
  
    return next(action);  
  };  
}  
  
const thunk = createThunkMiddleware();  
thunk.withExtraArgument = createThunkMiddleware;  
  
export default thunk;
```




```
function createThunkMiddleware(extraArgument) {  
  return ({ dispatch, getState }) => next => action => {  
    if (typeof action === 'function') {  
      return action(dispatch, getState, extraArgument);  
    }  
  
    return next(action);  
  };  
}  
  
const thunk = createThunkMiddleware();  
thunk.withExtraArgument = createThunkMiddleware;  
  
export default thunk;
```



Mind: How much you understand



Mind: ~~How much you understand~~
We can test you on this!



FizzBuzz:

Write a function that prints the numbers from 1 to 20.

Print “Fizz” for multiples of 3

Print “Buzz” for multiples of 5

Print “FizzBuzz” for multiples of 3 and 5



FizzBuzz:

```
for (let i=1; i <= 20; i++) {  
  if (i % 3 === 0 && i % 5 === 0) {  
    console.log("FizzBuzz");  
  } else if (i % 3 === 0) {  
    console.log("Fizz");  
  } else if (i % 5 === 0) {  
    console.log("Buzz");  
  } else {  
    console.log(i);  
  }  
}
```



FizzBuzz:

```
for (let i=1; i <= 20; i++) {  
  if (i % 3 === 0 && i % 5 === 0) {  
    console.log("FizzBuzz");  
  } else if (i % 3 === 0) {  
    console.log("Fizz");  
  } else if (i % 5 === 0) {  
    console.log("Buzz");  
  } else {  
    console.log(i);  
  }  
}
```

Who cares?

```
function load($array) {
    if(is_array($array)) {
        foreach($array as $key=>$value) {
            $this->vars[$key] = $value;
        }
    }
}

function unload($vars = '') {
    if($vars) {
        if(is_array($vars)) {
            foreach($vars as $var) {
                unset($this->vars[$var]);
            }
        }
        else {
            unset($this->vars[$vars]);
        }
    }
    else {
        $this->vars = array();
    }
}

function fetch() {
    $name= $argv[0];
    $query = "SELECT id, $name FROM products;";
    $result = pg_query($conn, $query);
}

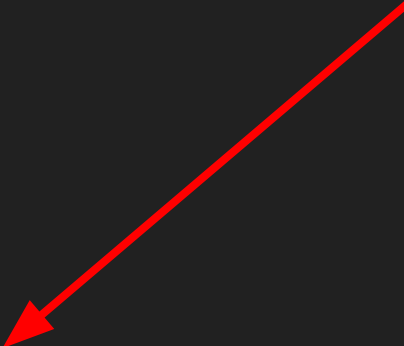
function get_all() {
    if($this->isAdmin) {
        return $this->vars;
    }
}
```

```
function load($array) {
    if(is_array($array)) {
        foreach($array as $key=>$value) {
            $this->vars[$key] = $value;
        }
    }
}

function unload($vars = '') {
    if($vars) {
        if(is_array($vars)) {
            foreach($vars as $var) {
                unset($this->vars[$var]);
            }
        }
        else {
            unset($this->vars[$vars]);
        }
    }
    else {
        $this->vars = array();
    }
}

function fetch() {
    $name= $argv[0];
    $query = "SELECT id, $name FROM products;";
    $result = pg_query($conn, $query);
}

function get_all() {
    if($this->isAdmin) {
        return $this->vars;
    }
}
```

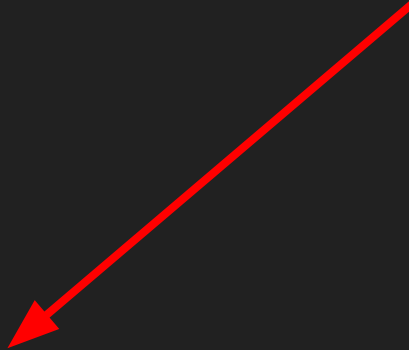


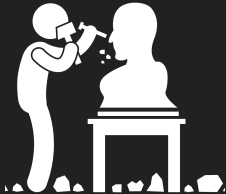

```
function load($array) {
    if(is_array($array)) {
        foreach($array as $key=>$value) {
            $this->vars[$key] = $value;
        }
    }
}

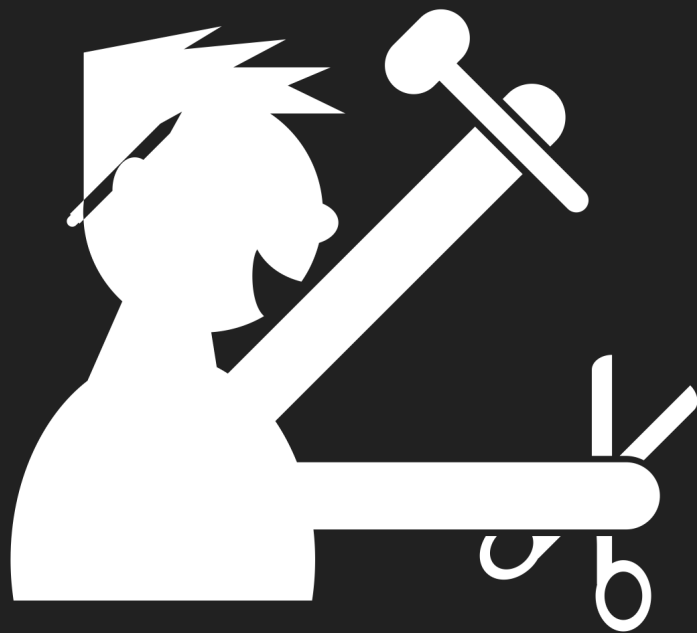
function unload($vars = '') {
    if($vars) {
        if(is_array($vars)) {
            foreach($vars as $var) {
                unset($this->vars[$var]);
            }
        }
        else {
            unset($this->vars[$vars]);
        }
    }
    else {
        $this->vars = array();
    }
}

function fetch() {
    $name= $argv[0];
    $query = "SELECT id, $name FROM products;";
    $result = pg_query($conn, $query);
}

function get_all() {
    if($this->isAdmin) {
        return $this->vars;
    }
}
```









Understand Style



Understand Style

“When program statements were arranged in a sensible order, experts were able to remember them better than novices. When statements were shuffled, the experts’ superiority was reduced”

-- Steve McConnell *Code Complete*



Understand Style

```
function email() {  
    var form =  
document.getElementById('contact-form');  
var re =  
/^[^<>()\\[\]\\. , ; : \s@"]+(\.[^<>()\\[\]\\. , ; : \s@"]+)*|(".+")@((\[[0-9]{1,3}\. [0-9]{1,3}\. [0-9]{1,3}\. [0-9]{1,3}\)|((\b[a-zA-Z-0-9]+\b)+[a-zA-Z]{2,}))$ /;  
  
for(input of form.elements) {if(input.test(re)) {  
    return input;  
}  
}  
  
else{return false;  
}  
}
```



```
function AddToarray(arr, x)
{
    var isInteger, updatedArr;
    var val

    isInteger = typeof x == 'number';
    if (isInteger)
        x = x + ''

    var shoul_Add
    if(Array.isArray(x)) {shoul_Add = false}
    else {
        shoul_Add=true
    }
    if(shoul_Add) {
        if(x !== '14') {
            arr.push(x);
        }
    }
    return arr
}
```



```
function AddToArray(arr, x)
{
    var isInteger, updatedArr;
    var val

    isInteger = typeof x == 'number';
    if (isInteger)
        x = x + ''

    var shoul_Add
    if(Array.isArray(x)) {shoul_Add = false}
    else {
        shoul_Add=true
    }
    if(shoul_Add) {
        if(x !== '14') {
            arr.push(x);
        }
    }
    return arr
}
```

It works!



```
function AddToarray(arr, x)
{
    var isInteger, updatedArr;
    var val

    isInteger = typeof x == 'number';
    if (isInteger)
        x = x + ''

```

Cultivate Dissatisfaction

```
var shoul_Add = true;
if(Array.isArray(x)) {shoul_Add = false}
else {
    shoul_Add=true
}
if(shoul_Add) {
    if(x !== '14') {
        arr.push(x);
    }
}
return arr
}
```



```
function AddToarray(arr, x)
{
    var isInteger, updatedArr;
    var val

    isInteger = typeof x == 'number';
    if (isInteger)
        x = x + ''

    var shoul_Add
    if(Array.isArray(x)) {shoul_Add = false}
    else {
        shoul_Add=true
    }
    if(shoul_Add) {
        if(x !== '14') {
            arr.push(x);
        }
    }
    return arr
}
```



```
function AddToArray(arr, x)
{
    var isInteger, updatedArr;
    var val

    isInteger = typeof x == 'number';
    if (isInteger)
        x = x + ''
```

Learn Community Standards

```
    var shoul_Add;
    if(Array.isArray(x)) {shoul_Add = false}
    else {
        shoul_Add=true
    }
    if(shoul_Add) {
        if(x !== '14') {
            arr.push(x);
        }
    }
    return arr
}
```



```
function AddToArray(arr, x)
{
    var isInteger, updatedArr;
    var val;

    isInteger = typeof x == 'number';
    if (isInteger)
        x = x + ''
```

Learn Community Standards: Read Code

```
    var shoul_Add = true;
    if(Array.isArray(x)) {shoul_Add = false}
    else {
        shoul_Add = true;
    }
    if(shoul_Add) {
        if(x !== '14') {
            arr.push(x);
        }
    }
    return arr
}
```



```
function AddToarray(arr, x)
{
    var isInteger, updatedArr;
    var val

    isInteger = typeof x == 'number';
    if (isInteger)
        x = x + ''

    var shoul_Add
    if(Array.isArray(x)) {shoul_Add = false}
    else {
        shoul_Add=true
    }
    if(shoul_Add) {
        if(x !== '14') {
            arr.push(x);
        }
    }
    return arr
}
```

Use A Linter



```
function addStringToArray(arr, value) {  
  const blackListedValue = '14';  
  
  if (Array.isArray(value)) {  
    return arr;  
  }  
  
  if (typeof value === 'number') {  
    value = value + '';  
  }  
  
  if (value !== blackListedValue) {  
    arr.push(value);  
  }  
  
  return arr;  
}
```



```
function addStringToArray(arr, value) {  
  const blackListedValue = '14';  
  
  if (Array.isArray(value)) {  
    return arr;  
  }  
  
  if (typeof value !== 'number') {  
    value = 0;  
  }  
  
  if (value !== blackListedValue) {  
    arr.push(value);  
  }  
  
  return arr;  
}
```

Readable.
Not Perfect.



Get Feedback



```
<?
class House {
    public $rooms = 3;

    public function getRooms() {
        return $this->rooms;
    }
}

$h = new House();
$h->getRooms();
// 3
```



```
<?
class House {
    public $rooms = 3;
    static $bathrooms = 1;

    public function getRooms() {
        return $this->rooms;
    }

    static function getBathrooms() {
        return self::$bathrooms;
    }
}

$h = new House();
$h->getRooms();
// 3

$h::getBathroom();
// 1
```



```
<?
class House {
    public $rooms = 3;
    static $bathrooms = 1;

    public function getRooms() {
        return $this->rooms;
    }

    static function getBathrooms() {
        return self::$bathrooms;
    }
}
```

Code Review

```
$h = new House();
$h->getRooms();
// 3
```

```
$h::getBathroom();
// 1
```



```
<?
class House {
    public $rooms = 3;
    static $bathrooms = 1;

    public function getRooms() {
        return $this->rooms;
    }

    static function getBathrooms() {
        return self::$bathrooms;
    }
}
```

```
$h = new House();
$h->getRooms();
// 3
```

```
$h::getBathroom();
// 1
```

Why is this static?





```
<?
class House {
    public $rooms = 3;
    static $bathrooms = 1;

    public function getRooms() {
        return $this->rooms;
    }

    static function getBathrooms() {
        return self::$bathrooms;
    }
}
```

Uhh.....

```
$h = new House();
$h->getRooms();
// 3
```

```
$h::getBathroom();
// 1
```



```
<?
class House {
    public $rooms = 3;
    static $bathrooms = 1;

    public function getRooms() {
        return $this->rooms;
    }

    static function getBathrooms() {
        return self::$bathrooms;
    }
}
```

Be Deliberate

```
$h = new House();
$h->getRooms();
// 3
```

```
$h::getBathroom();
// 1
```



```
<?
class House {
    public $rooms = 3;

    public function getRooms() {
        return $this->rooms;
    }

    static $bath_rooms = '1';
    static function get_Bathrooms() {
        $arr = [];
        $arr.push(self::$bath_rooms);
        if(false){
            return self;
        }
        return self::$bath_rooms;
    }
}
```



```
<?
class House {
    public $rooms = 3;

    public function getRooms() {
        return $this->rooms;
    }

    static $bath_rooms = '1';
    static function get_Bathrooms() {
        $arr = [];
        $arr.push(self::$bath_rooms);
        if(false){
            return self;
        }
        return self::$bath_rooms;
    }
}
```

Do it again. Then
give it back to me.



```
<?
class House {
    public $rooms = 3;

    public function getRooms() {
        return $this->rooms;
    }

    static $bath_rooms = [];
    static function get_bathrooms() {
        $arr = [];
        $arr.push(self::$bath_rooms);
        if(false){
            return self;
        }
        return self::$bath_rooms;
    }
}
```

Reviewers



```
<?
class House {
    public $rooms = 3;

    public function getRooms() {
        return $this->rooms;
    }
}
```

```
static $bath_rooms = 0;
static function get_bathrooms() {
    $arr = [];
    $arr['rooms'] = $rooms;
    if(false){
        return self::get_rooms();
    }
    return self::$bath_rooms;
}
}
```

**Reviewers: You are
not being nice when
you withhold
feedback.**



```
<?
class House {
    public $rooms = 3;

    public function getRooms() {
        return $this->rooms;
    }
}
```

No feedback?

```
static::$bath_rooms = [];
static function get_bathrooms() {
    $arr = [];
    $arr.push(self::$bath_rooms);
    if(false){
        return self;
    }
    return self::$bath_rooms;
}
}
```



```
<?
class House {
    public $rooms = 3;

    public function getRooms() {
        return $this->rooms;
    }
}
```

No feedback? Write Test

```
static $bath_rooms = 1;
static function get_bathrooms() {
    $arr = [];
    $arr['bath_rooms'] = self::$bath_rooms;
    if(false){
        return self;
    }
    return self::$bath_rooms;
}
}
```



```
function addAuthor() {  
  fetch('http://foo.com')  
    .then(data => {  
      return data.json();  
    })  
    .then(author => {  
      const { name } = author;  
      if(!name) {  
        dispatchState('Unknown')  
      } else {  
        if(name.indexOf(',') > -1) {  
          const names = name.split(',');  
          dispatchState(`${name[0]} ${names[1]}`)  
        } else {  
          dispatchState(name);  
        }  
      }  
    })  
    .then(fetchPostsByAuthor(author.id));  
}
```



```
function addAuthor() {
  fetch('http://foo.com')
    .then(data => {
      return data.json();
    })
    .then(author => {
      const { name } = author;
      if(!name) {
        dispatchState('Unknown')
      } else {
        if(name.indexOf(',') > -1) {
          const names = name.split(',');
          dispatchState(`${name[0]} ${names[1]}`)
        } else {
          dispatchState(name);
        }
      }
    })
    .then(() => {
      fetchPostsByAuthor(author.id);
    })
}
```

Test



```
function addAuthor() {  
  fetch('http://foo.com')  
    .then(data => {  
      return data.json();  
    })  
    .then(author => {  
      const { name } = author;  
      if(!name) {  
        dispatchState('Unknown')  
      } else {  
        if(name.indexOf(',') > -1) {  
          const names = name.split(',');  
          dispatchState(`${name[0]} ${names[1]}`)  
        } else {  
          dispatchState(name);  
        }  
      }  
    })  
    .then(fetchPostsByAuthor(author.id));  
}
```

Async

Lots of conditionals

Calls another method

Calls another method



```
function addAuthor() {  
  fetch('http://foo.com')  
    .then(data => {  
      return data.json();  
    })  
    .then(author => {  
      const {name} = author;  
      if(!name) {  
        dispatchState('Unknown')  
      } else {  
        if(name.indexOf(',') > -1) {  
          const names = name.split(',');  
          dispatchState(`${name[0]} ${names[1]}`)  
        } else {  
          dispatchState(name);  
        }  
      }  
    })  
    .then(() => {  
      fetchPostsByAuthor(author.id);  
    })  
}
```

Every test and every
assertion is a statement
of why code exists



```
function addAuthor() {  
  fetch('http://foo.com')  
    .then(data => {  
      return data.json();  
    })  
    .then(author => {  
      const { name } = author;  
      if(!name) {  
        dispatchState('Unknown')  
      } else {  
        if(name.indexOf(',') > -1) {  
          const names = name.split(',');  
          dispatchState(`${name[0]} ${names[1]}`)  
        } else {  
          dispatchState(name);  
        }  
      }  
    })  
    .then(fetchPostsByAuthor(author.id);  
  })  
}
```

If it's hard to test



```
function addAuthor() {  
  fetch('http://foo.com')  
    .then(data => {  
      return data.json();  
    })  
    .then(author => {  
      const { name } = author;  
      if(!name) {  
        dispatchState('Unknown')  
      } else {  
        if(name.indexOf(',') > 0) {  
          const names = name.split(',');  
          dispatchState(`${names[0]} ${names[1]}`);  
        } else {  
          dispatchState(name);  
        }  
      }  
    })  
    .then(fetchPostsByAuthor(author.id);  
  })  
}
```

If it's hard to test,
make it easier to test



Mind:



Mind: Not as important as you think



Mind: Not as important as you think

In Shakespeare:

The top 10 most frequently occurring words make up 21.4% of all words.

The top 100 most frequently occurring words make up 53.9% of all words.



Mind: Not as important as you think

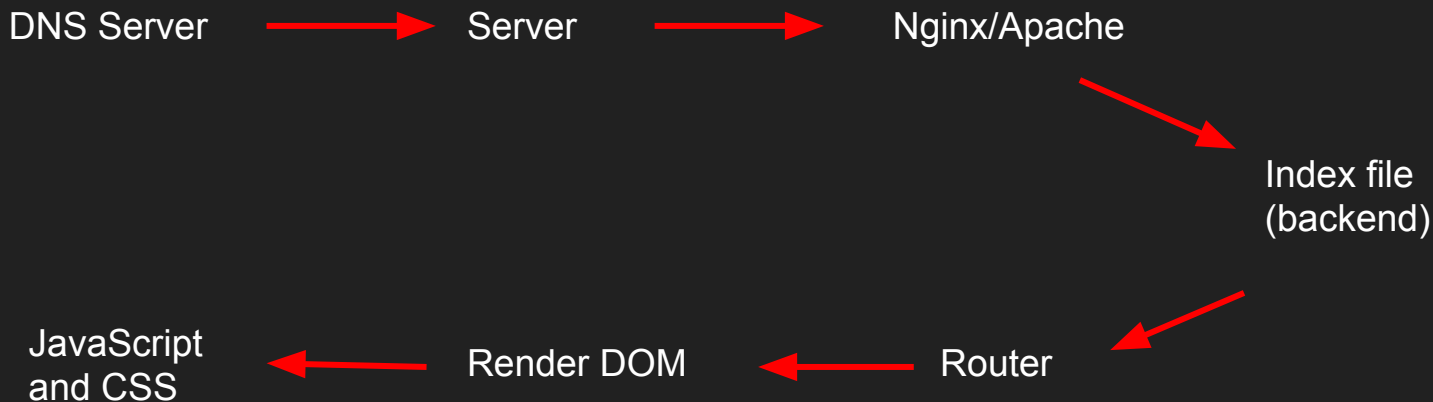
- ✓ Basics
- ✓ Things work (at least it seems so)
- ✓ Learning by reading/doing



Relationships between parts



Relationships between parts





Relationships between parts

/etc

/sbin

/var

/usr

/sys



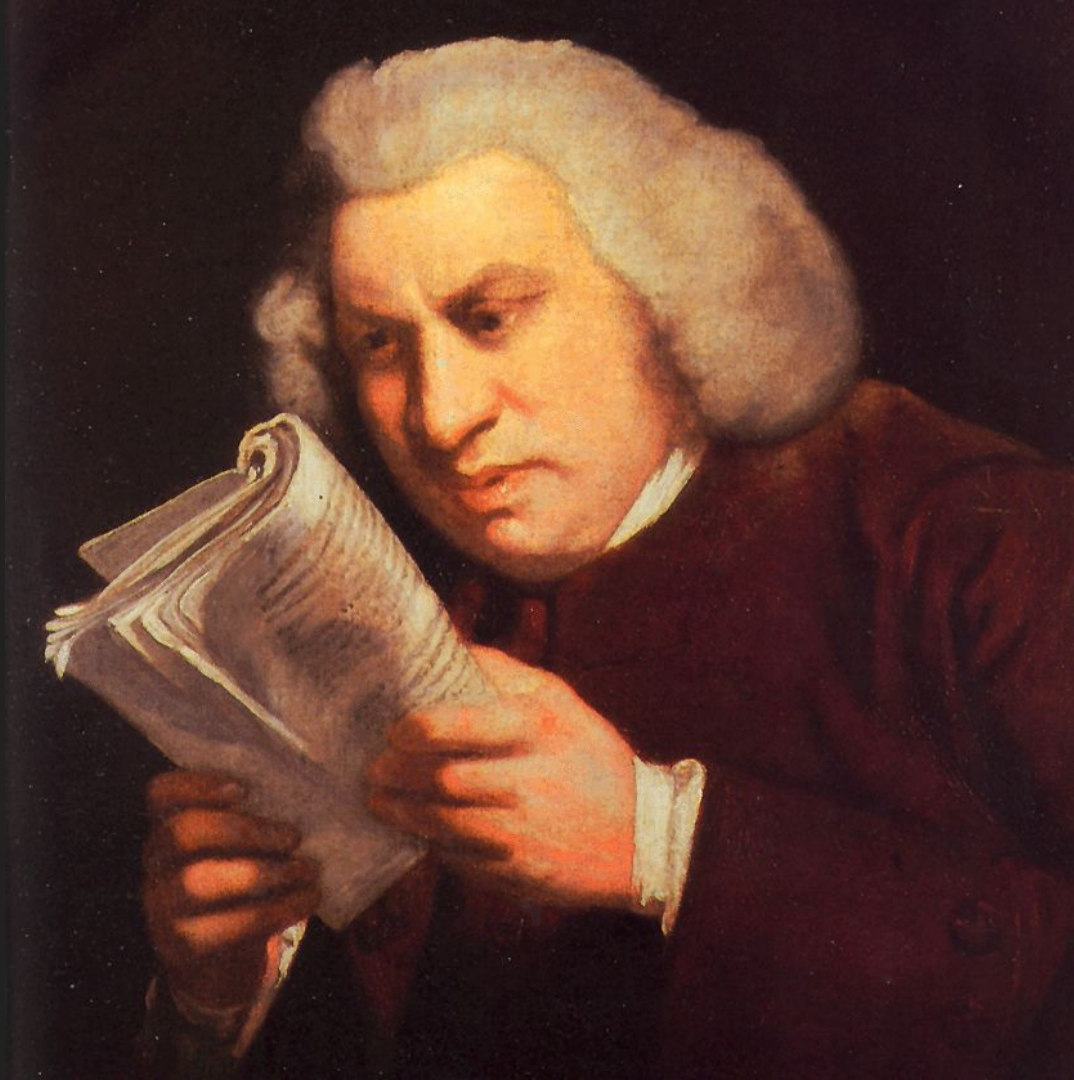


Make Code Beautiful



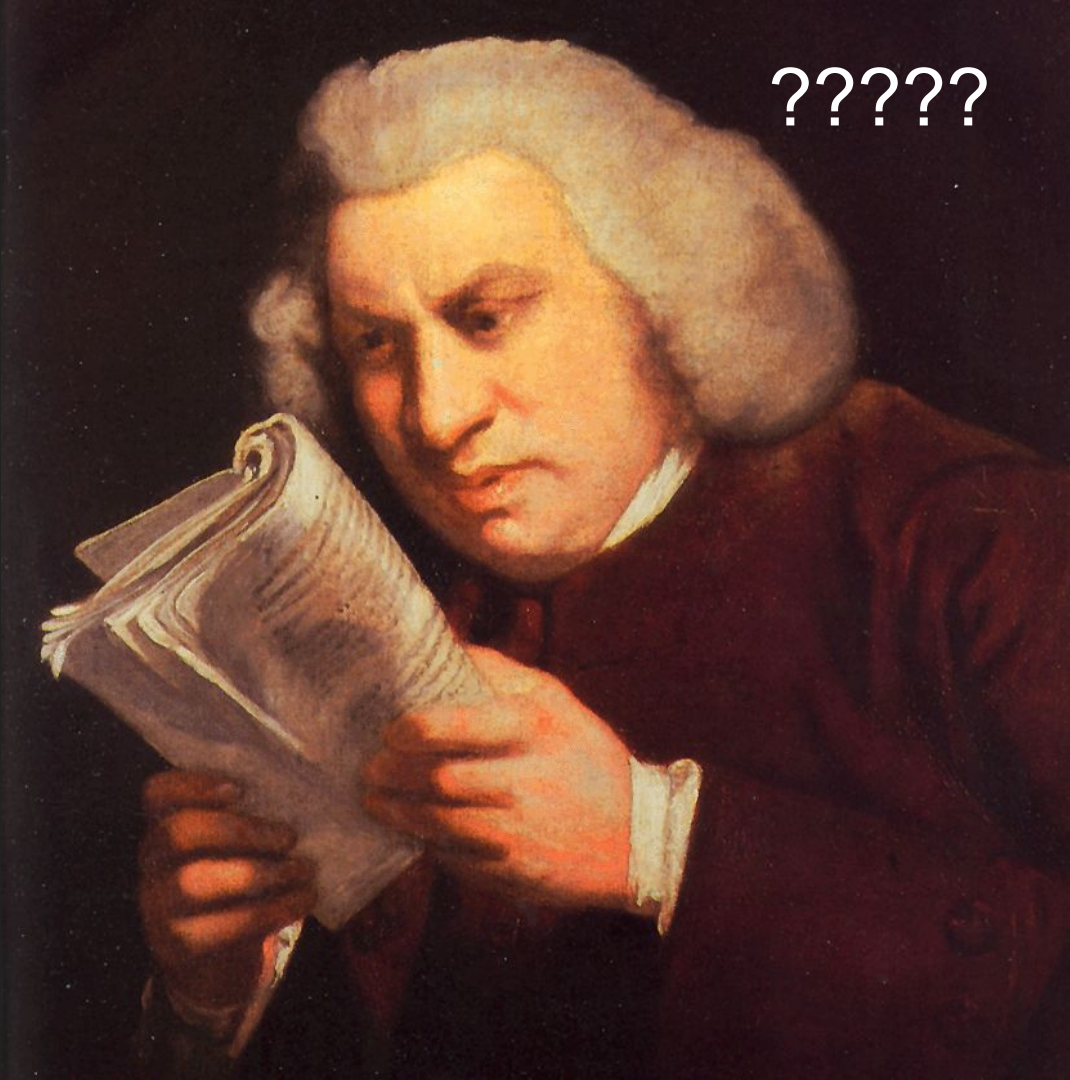
```
function formatPathStr(config, preferences, defaultPath, alternativePath, reroute, cb) {
  var path;
  // We need to check if they want
  if(config && config.path && !config.path.match(/tmp/) && (defaultPath ||
alternativePath)) {
    if(preferences.reroute && alternativePath) {
      path = alternativePath + '/zone2/' + config.path;
      return reroute(path); //Promise
    }
    if(!preferences.reroute && alternativePath) {
      if(!alternativePath.match(/user/)) {
        if(preferences.send) {
          path = alternativePath + '/zone3' + config.path;
          cb();
          return;
        }
        path = alternativePath + '/zone2' + config.path;
      }
    }
    // This goes on and on

  }
  return config.name + '/zone5/' + config.path;
}
```





?????







```
function formatPathStr(config, preferences, defaultPath, alternativePath, reroute, cb) {
  var path;
  // We need to check if they want
  if(config && config.path && !config.path.match(/tmp/) && (defaultPath ||
alternativePath)) {
    if(preferences.reroute && alternativePath) {
      path = alternativePath + '/zone2/' + config.path;
      return reroute(path); //Promise
    }
    if(!preferences.reroute && alternativePath) {
      if(!alternativePath.match(/user/)) {
        if(preferences.send) {
          path = alternativePath + '/zone3' + config.path;
          cb();
          return;
        }
        path = alternativePath + '/zone2' + config.path;
      }
    }
    // This goes on and on

  }
  return config.name + '/zone5/' + config.path;
}
```




```
function formatPathStr(config, preferences, defaultPath, alternativePath, reroute, cb) {
  var path;
  // We need to check if they want
  if(config && config.path && !config.path.match(/tmp/) && (defaultPath ||
alternativePath)) {
    if(preferences.reroute && alternativePath) {
      path = alternativePath + '/zone2/' + config.path;
      return reroute(path); //Promise
    }
    if(!preferences.reroute && alternativePath) {
      if(!alternativePath.match(/tmp/)) {
        if(preferences.send) {
          path = alternativePath + '/zone3' + config.path;
          cb();
          return;
        }
        path = alternativePath + '/zone2' + config.path;
      }
    }
    // This goes on and on
  }
  return config.name + '/zone5/' + config.path;
}
```

Smelly Code



```
function formatPathStr(config, preferences, defaultPath, alternativePath, reroute, cb) {  
  var path;  
  // We need to check if they want  
  if(config && config.path && !config.path.match(/tmp/) && (defaultPath ||  
alternativePath)) {  
    if(preferences.reroute && alternativePath) {  
      path = alternativePath + '/zone2/' + config.path;  
      return reroute(path); //Promise  
    }  
    if(!preferences.reroute && alternativePath) {  
      if(!alternativePath.match(/user/)) {  
        if(preferences.send) {  
          path = alternativePath + '/zone3' + config.path;  
          cb();  
          return;  
        }  
        path = alternativePath + '/zone2' + config.path;  
      }  
    }  
    // This goes on and on  
  }  
  return config.name + '/zone5/' + config.path;  
}
```

Long parameter list

Complex conditional blocks

Differing return statements

Uncommunicative name



```
function formatPathStr(config, preferences, defaultPath, alternativePath, reroute, cb) {
  var path;
  // We need to check if they want
  if(config && config.path && !config.path.match(/tmp/) && (defaultPath ||
alternativePath)) {
    if(preferences.reroute && alternativePath) {
      path = alternativePath + '/zone2/' + config.path;
      return reroute(path); //Promise
    }
    if(!preferences.reroute && alternativePath) {
      if(!alternativePath.match(/usr/)) {
        if(preferences.send) {
          path = alternativePath + '/zone3' + config.path;
          cb();
          return;
        }
        path = alternativePath + '/zone2' + config.path;
      }
    }
    // This goes on and on
  }
  return config.name + '/zone5/' + config.path;
}
```

Coding Horror Code Smells



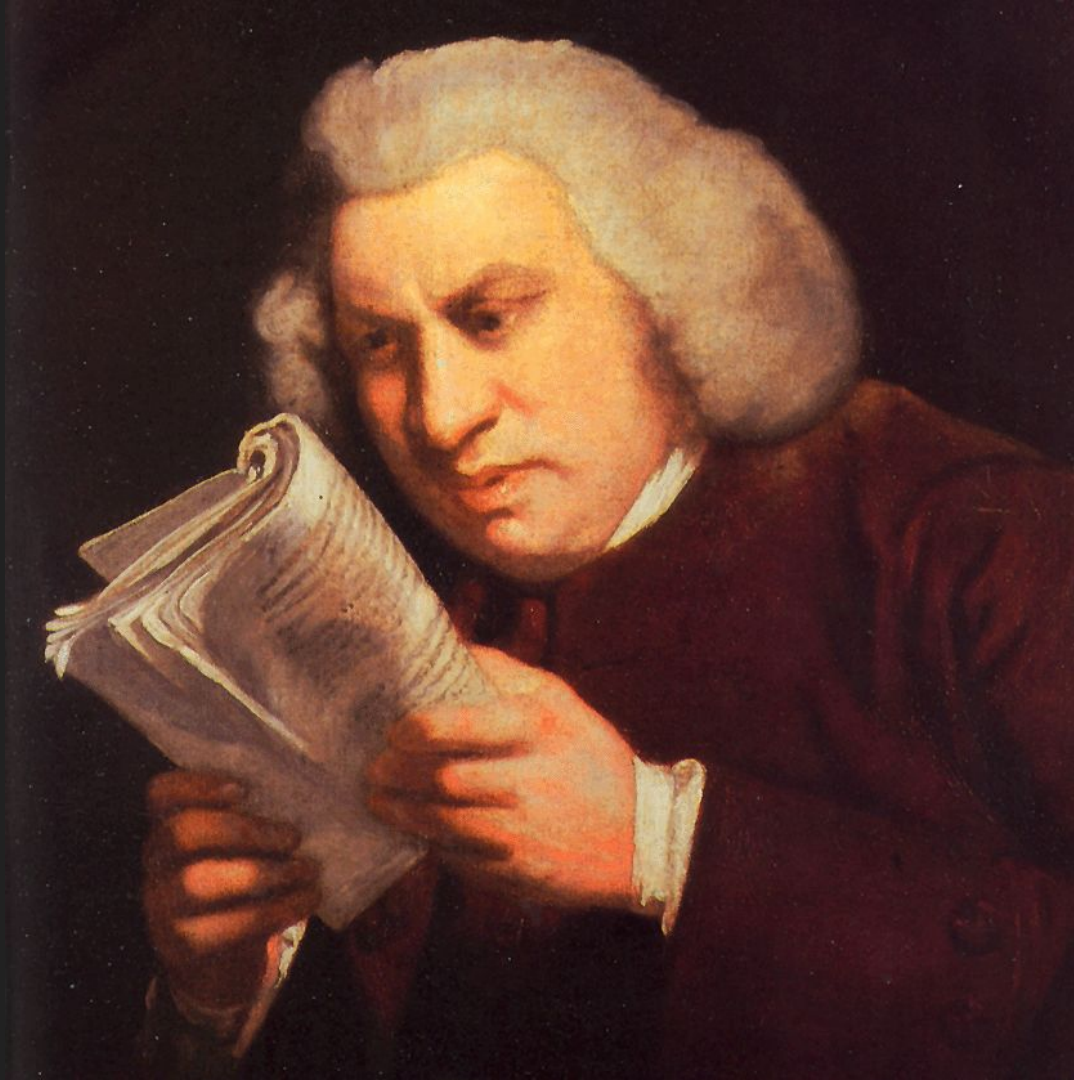
```
function formatPathStr(config, preferences, defaultPath, alternativePath, reroute, cb) {
  var path;
  // We need to check if they want
  if(config && config.path && !config.path.match(/tmp/) && (defaultPath ||
alternativePath)) {
    if(preferences.reroute && alternativePath) {
      path = alternativePath + '/zone2/' + config.path;
      return new Promise((resolve, reject) => {
        // This goes on and on
      });
    }
    if(!preferences.reroute && alternativePath) {
      if(!alternativePath.match(/user/)) {
        if(preferences.send) {
          path = alternativePath + '/zone' + config.path;
          cb();
          return;
        }
        path = alternativePath + '/zone2' + config.path;
      }
    }
  }
  return config.name + '/zone5/' + config.path;
}
```

Don't
memorize
smells.



```
function formatPathStr(config, preferences, defaultPath, alternativePath, reroute, cb) {
  var path;
  // We need to check if they want
  if(config && config.path && !config.path.match(/tmp/) && (defaultPath ||
alternativePath)) {
    if(preferences.reroute && alternativePath) {
      path = alternativePath + '/zone2/' + config.path;
      return reroute(path); //Promise
    }
    if(!preferences.reroute && alternativePath) {
      if(!alternativePath.match(/user/)) {
        if(preferences.send) {
          path = alternativePath + '/zone3' + config.path;
          cb();
          return;
        }
        path = alternativePath + '/zone2' + config.path;
      }
    }
    // This goes on and on
  }
  return config.name + '/zone5/' + config.path;
}
```

Trust yourself





```
function createThunkMiddleware(extraArgument) {  
  return ({ dispatch, getState }) => next => action => {  
    if (typeof action === 'function') {  
      return action(dispatch, getState, extraArgument);  
    }  
  
    return next(action);  
  };  
}  
  
const thunk = createThunkMiddleware();  
thunk.withExtraArgument = createThunkMiddleware;  
  
export default thunk;
```



READ THE SOURCE LUKE



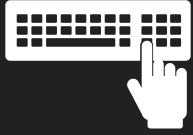


Build Intuition and Refactoring



Build Intuition and Refactoring

Pop Quiz



1. Get an array of unique items
2. Determines if a string starts with an uppercase letter
3. Calculate the area of a circle given radius



```
function getUniqueItems(arr) {  
  let unique = [];  
  for(let item of arr) {  
    if(!unique.includes(item)) {  
      unique.push(item);  
    }  
  }  
  return unique;  
}
```



```
function getUniqueItems(arr) {  
  let unique = [];  
  for(let item of arr) {  
    if (!unique.includes(item)) {  
      unique.push(item);  
    }  
  }  
  return unique;  
}
```

Cultivate Disatisfaction



```
function getUniqueItems(arr) {  
  let unique = [];  
  for(let item of arr) {  
    if(!unique.includes(item))  
      unique.push(item);  
  }  
}  
return unique;  
}
```

You can look up
solved problems



```
function getUniqueItems(arr) {  
  return arr.reduce((unique, item) => {  
    return unique.includes(item) ? unique : [...unique, item];  
  }, [])  
}
```



```
function getUniqueItems(arr) {  
  return arr.reduce((unique, item) => {  
    return unique.includes(item) ? unique : [...unique, item];  
  }, [])  
}
```

Iterate Over Ideas



```
function getUniqueItems(items) {  
  return [...new Set(items)];  
}
```



```
function getUniqueItems(items) {  
  return [...new Set(items)];  
}
```

Test



Tools, Principles, Patterns



Why do things exist



Why do things exist





Why do things exist

```
const x = ['a', 'b', 'c'];
```

```
...x  
// 'a', 'b', 'c'
```



Solve the problem

```
const x = ['a', 'b', 'c'];
```

```
x.push('d');
```

```
x;
```

```
['a', 'b', 'c', 'd'];
```



What is the [X] way to solve the problem?

```
const x = ['a', 'b', 'c'];
```

```
x.push('d');
```

```
x;
```

```
['a', 'b', 'c', 'd'];
```




What is the JavaScript way to solve the problem?

```
const x = ['a', 'b', 'c'];
```

```
x.push('d');
```

```
x;
```

```
['a', 'b', 'c', 'd'];
```



What is the JavaScript way to solve the problem?

```
const x = ['a', 'b', 'c'];
```

```
x.push('d');
```

```
x;
```

```
['a', 'b', 'c', 'd'];
```



mutation



What is the JavaScript way to solve the problem?

```
const x = ['a', 'b', 'c'];
```

```
const y = [...x, 'd'];
```

```
y;  
['a', 'b', 'c', 'd'];
```

```
x;  
['a', 'b', 'c'];
```



What is the JavaScript way to solve the problem?

```
const x = ['a', 'a', 'b', 'c'];
```

```
const copyX = [...x];
```

```
const combined = [...x, ...y];
```

```
const unique = [...new Set(x)];
```



What is the JavaScript way to solve the problem?

```
const x = ['a', 'a', 'b', 'c'];
```

No Mutations

```
const copyX = [...x];
```

```
const combined = [...x, ...y];
```

```
const unique = [...new Set(x)];
```



What is the Python way to solve the problem?



Suspicious Python Devs



```
171         return ' '.join(bits)
172
173
174     class BadOptionUsage(UsageError):
175         """Raised if an option is generally supplied but the use of the option
176         was incorrect. This is for instance raised if the number of arguments
177         for an option is not correct.
178
179         .. versionadded:: 4.0
180
181         :param option_name: the name of the option being used incorrectly.
182         """
183
184         def __init__(self, option_name, message, ctx=None):
185             UsageError.__init__(self, message, ctx)
186             self.option_name = option_name
187
188
189     class BadArgumentUsage(UsageError):
190         """Raised if an argument is generally supplied but the use of the argument
191         was incorrect. This is for instance raised if the number of values
192         for an argument is not correct.
193
194         .. versionadded:: 6.0
195         """
196
197         def __init__(self, message, ctx=None):
198             UsageError.__init__(self, message, ctx)
199
200
201     class FileError(ClickException):
202         """Raised if a file cannot be opened."""
203
204         def __init__(self, filename, hint=None):
205             ui_filename = filename_to_ui(filename)
206             if hint is None:
```




python multiple classes per file



All

Shopping

News

Videos

Images

More

Settings

Tools

About 869,000 results (0.77 seconds)

class - Are multiple Python classes in a single file recommended ...

<https://stackoverflow.com/.../are-multiple-python-classes-in-a-single-file-recommende...> ▼

Jul 7, 2009 - Modules are just as likely to contain functions (which are first-class objects in **Python**) as **classes**. In Java, the unit of decomposition is the **class**. Hence, **Python** has one module=one file, and Java has one (public) **class**=one file.

Multiple classes in a Python module - Stack Overflow

<https://stackoverflow.com/questions/2634394/multiple-classes-in-a-python-module> ▼

Apr 14, 2010 - Here is a useful rule of thumb from what I have seen of typical Java projects: ... Is this something that is normally done or should I stick with 1 class per module? ... The bottom-most package in Java should be a **file** in **Python**.

class - How many Python classes should I put in one file? - Stack ...

<https://stackoverflow.com/.../how-many-python-classes-should-i-put-in-one-file> ▼

Sep 20, 2008 - A **Python file** is called a "module" and it's one way to organize your software so that it ... In a big application, however, there are always **multiple** dimensions of analysis and one person will split ... If you have a bunch of **classes** grouped into a **single file**, it may not be obvious to other developers that there are ...

Python: one single module (file .py) for each class? - Stack Overflow

<https://stackoverflow.com/questions/.../python-one-single-module-file-py-for-each-cla...> ▼

Apr 28, 2015 - In Java, you cannot - by design - have more than one **class** in a **file**. In **Python**, if you group related **classes** in a **single file**, you are on the save side. Take a look at the **Python** standard library: many modules contain **multiple classes** in a **single file**.

python - Is it considered Pythonic to have multiple classes defined in ...

<https://softwareengineering.stackexchange.com/.../is-it-considered-pythonic-to-have-...> ▼

<https://www.google.com/webhp?hl=en&sa=X&ved=0ahUKewie-ZL...> for the first time, I've found that I end up Moreover, having



Learn Principles



Learn Principles

... BUT they should really be there to solidify what you've been seeing



Learn Principles

This was the book that showed me there were names for so many patterns that I'd discovered purely through fumbling around with my own code.

-- Rebecca Murphey on *JavaScript Patterns*



Learn Principles

... they are there to help
communicate



Learn Principles

Experience without theory is blind, but theory without experience is mere intellectual play -- Immanuel Kant



Learn Principles

... run everything through your
hands or eyes



Learn Principles

... Before I created Rails, I redrew many of the diagrams in OmniGraffle for Martin Fowler because I liked the book so much

-- David on *Patterns of Enterprise Architecture*



Learn Principles

...The trick to reading this book is to carefully read through every single refactoring pattern and then try to apply it on your code base (you don't have to commit if it doesn't fix things). You can't just blow through it or you won't really learn it.

-- David on *Refactoring*



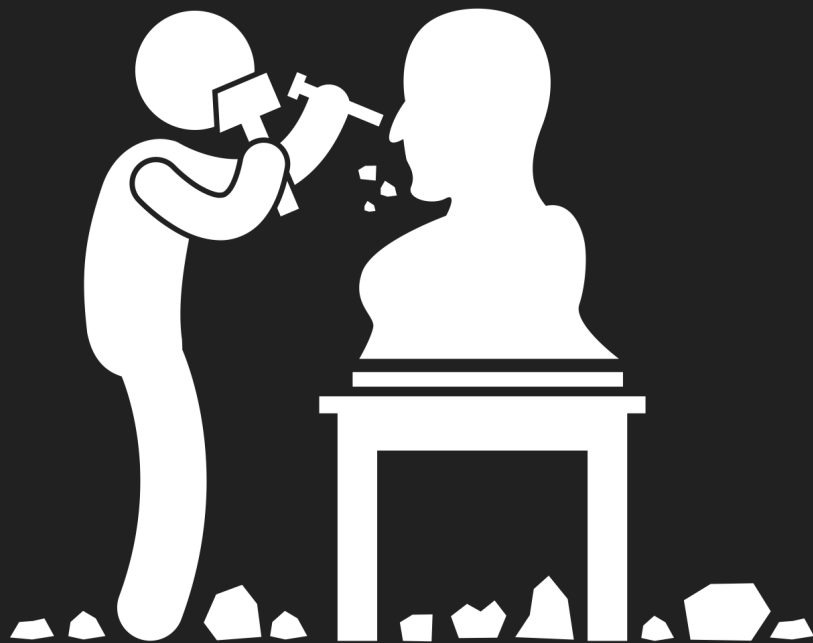
Learn Principles

- ✓ Algorithms
- ✓ Mutations/Side Effects
- ✓ Patterns (Gang of Four)
- ✓ SOLID



READ THE SOURCE LUKE







See Abstractions Across Projects



Dan Abramov @dan_abramov · Nov 26

People are reading *way* too much into Presentational vs Container components divide. Almost regretting I wrote that.



45



97



325



Dan Abramov

@dan_abramov



Following

It's just a pattern I noticed in a codebase. We didn't follow any "rules". Components often flipped back and forth as we worked on them.

RETWEET

1

LIKES

32



10:51 AM - 26 Nov 2016

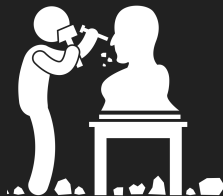
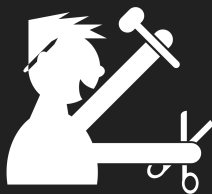


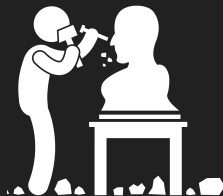
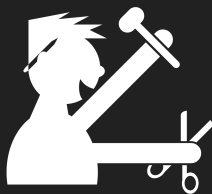
“We have included only designs that have been applied more than once in different systems... Most ... have never been documented before”

-- Gang of Four *Design Patterns*

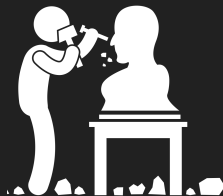
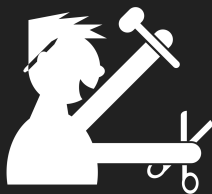


“How many times have you had design *déjà vu* -- that feeling that you’ve solved a problem before but not knowing exactly where or how?



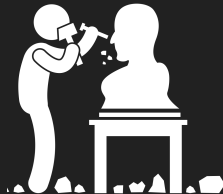
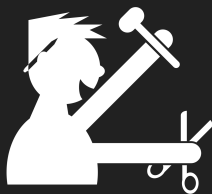


Foo.js



Foo.js

```
-- src
-- components
--   button.js
--   link.js
--   footer.js
-- util
-- colors
```



Foo.js

```
-- src
-- components
--   button.js
--   link.js
--   footer.js
-- util
-- colors
```





Reusable



Rule of three:

“The first time you do something, you just do it. The second time you do something similar, you wince at the duplication, but you do the duplicate thing anyway. Third you do something similar, you refactor.”

-- Martin Fowler *Refactoring*



Decoupled




```
import fetch from 'isomorphic-fetch';

function updateInventory(store) {
  return fetch(`http://foo.com/inventory?key=abc&name=${store}`)
    .then(response => response.json())
    .then((result) => {
      const count = result.inventory.toLocaleString();
      return `Inventory is ${count}`;
    });
}

export default updateInventory;
```

```
import fetch from 'isomorphic-fetch';

function updateInventory(store) {
  return fetch(`http://foo.com/inventory?key=abc&name=${store}`)
    .then(response => response.json())
    .then((result) => {
      const count = result.inventory.toLocaleString();
      return `Inventory is ${count}`;
    });
}

export default updateInventory;
```



Tightly coupled to api

```
import expect from 'expect';
import nock from 'nock';
import updateInventory from './coupled1';

describe('inventory update', () => {
  it('should update inventory', () => {
    nock('http://foo.com')
      .get('/inventory')
      .query({
        key: 'abc',
        name: 'bar',
      })
      .reply(200, { inventory: 1350 });

    return updateInventory('bar').then((result) => {
      expect(result).toEqual('Inventory is 1,350.');
```

```
    });
  });
});
```

```
import expect from 'expect';
import nock from 'nock';
import updateInventory from './coupled1';
```

```
describe('inventory update', () => {
  it('should update inventory', () => {
    nock('http://foo.com')
      .get('/inventory')
      .query({
        key: 'abc',
        name: 'bar',
      })
      .reply(200, { inventory: 1350 });

    return updateInventory('bar').then((result) => {
      expect(result).toEqual('Inventory is 1,350.');
```



Hijack the http request

```
import { getInventory } from './inventoryManager';

function updateInventory(store) {
  return getInventory(store)
    .then((result) => {
      const count = result.inventory.toLocaleString();
      return `Inventory is ${count}.`;
    });
}

export default updateInventory;
```

```
import { getInventory } from '../inventoryManager';
```

```
function updateInventory(store) {
```

```
  return getInventory(store)
```

```
    .then((result) => {
```

```
      const count = result.inventory.toLocaleString();
```

```
      return `Inventory is ${count}.`;
```

```
    });
```

```
}
```

```
export default updateInventory;
```



Endpoint removed

```
import expect from 'expect';
import sinon from 'sinon';

import updateInventory from './coupled';
import * as i from './inventoryManager';

describe('inventory update', () => {
  beforeEach(() => {
    const fetchInventory = new Promise(resolve => resolve({ inventory: 1350 }));
    sinon.stub(i, 'getInventory').returns(fetchInventory);
  });

  afterEach(() => {
    i.getInventory.restore();
  });


  it('should update inventory', () => updateInventory('bar').then((result) => {
    expect(result).toEqual('Inventory is 1,350.');
```

```
import expect from 'expect';  
import sinon from 'sinon';
```

```
import updateInventory from './coupled';  
import * as i from './inventoryManager';
```

Mocking

```
describe('inventory update', () => {  
  beforeEach(() => {  
    const fetchInventory = new Promise(resolve => resolve({ inventory: 1350 }));  
    sinon.stub(i, 'getInventory').returns(fetchInventory);  
  });  
  
  afterEach(() => {  
    i.getInventory.restore();  
  });  
  
  it('should update inventory', () => updateInventory('bar').then((result) => {  
    expect(result).toEqual('Inventory is 1,350.');  }));  
});
```




```
function updateInventory(store, fetchInventory) {  
  return fetchInventory  
    .then((result) => {  
      const count = result.inventory.toLocaleString();  
      return `Inventory is ${count}.`;   
    });  
}
```

```
export default updateInventory;
```

```
function updateInventory(store, fetchInventory) {  
  return fetchInventory  
    .then((result) => {  
    const count = result.inventory.toLocaleString();  
    return `Inventory is ${count}.`;   
  });  
}
```

 Inject everything

```
export default updateInventory;
```

```
import expect from 'expect';
import updateInventory from './coupled3';

describe('inventory update', () => {
  it('should update inventory', () => {
    const fetchInventory = new Promise(resolve => {
      resolve({ inventory: 1350 })
    });
    return updateInventory('bar', fetchInventory).then((result) => {
      expect(result).toEqual('Inventory is 1,350.');
```





Creatively Apply Ideas



Master the why of your language
first



Redux



Redux

Dan Abramov

Wanted to speak at a JS
conference in Europe.



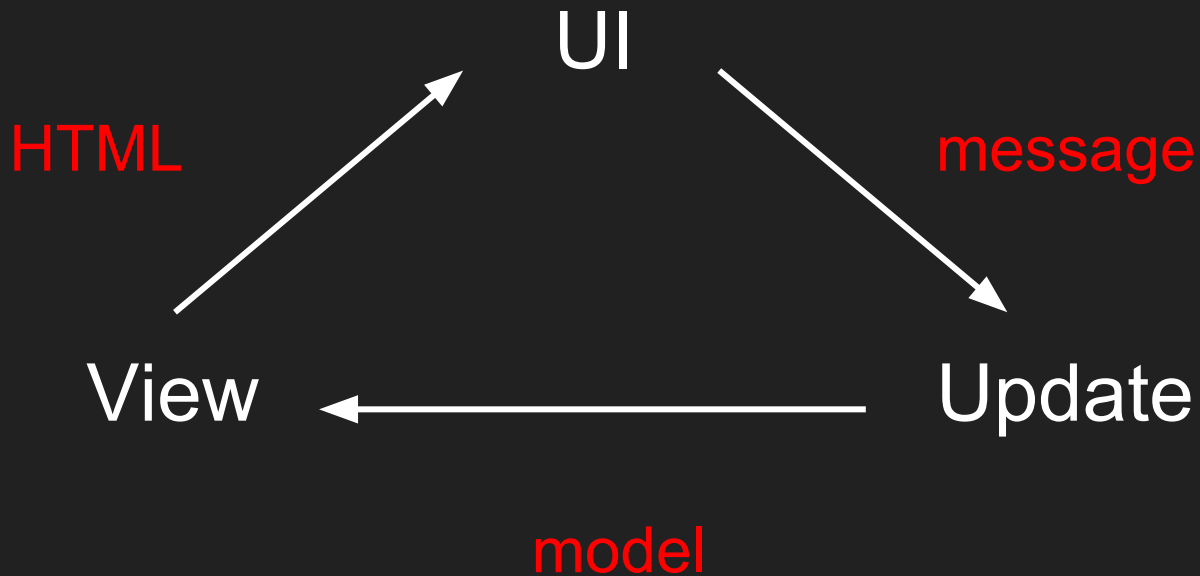
Redux

Dan Abramov

Applied Elm architecture in React

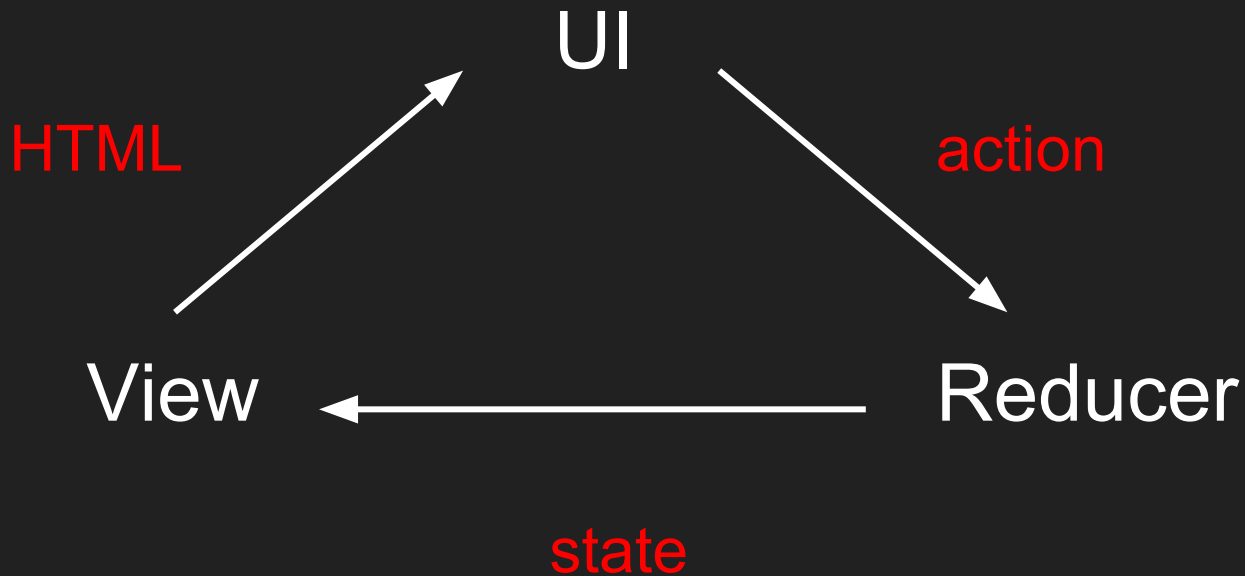


Elm





Redux





JavaScript

Brendan Eich was recruited to make Scheme into a scripting language. Netscape partner Sun Systems demanded it be similar to Java.

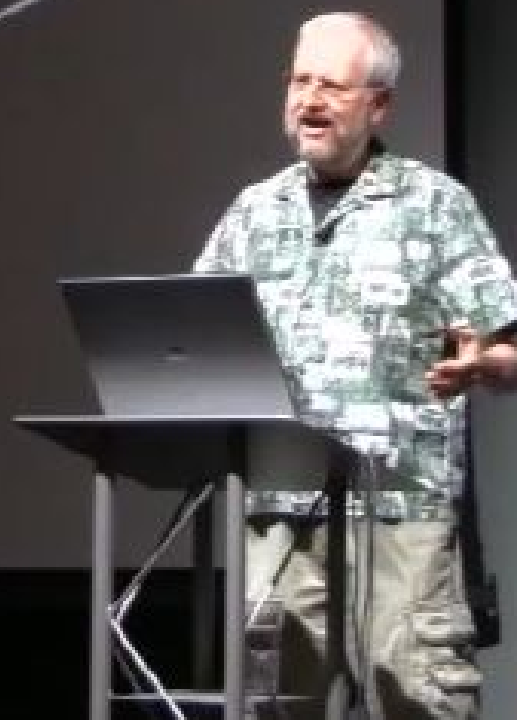
Brendan Eich wrote it in 10 days.

Java

Scheme

Self

JavaScript





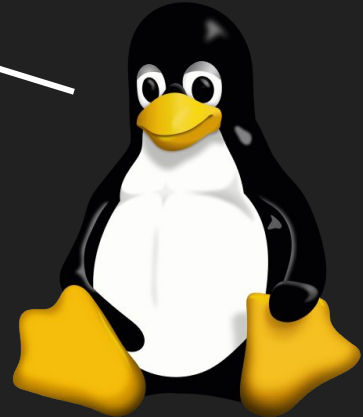
Linux

Linus Torvalds wanted to try out a new chip.



Linux

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu)... This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system...





Why

Bored.

Curious.

Dictated.



Why

They understood their inspirational material enough to take what they needed.



Style

Feedback

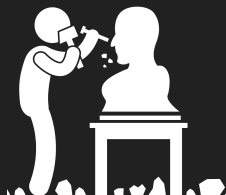
Flow



Beautiful

Refactor

Patterns



Abstract

Reusable

Creativity

Why?



Joe Morgan
@joesmorgan | thejoemorgan.com