Improve your code

Artem Prokhatskyi

front-end team leader

soon	Betlab
2016-2017	WorldAPP / form.com
2013-2016	EPAM Systems
2012-2013	Dmytro Babych Studio
2009-2012	MC Design / freelance

What is the perfect code?

Signs of the good code

1. Modern
2. Readable
3. Self-describing
4. Precise
5. Effective
6. Scalable
7. Safe

```
var count = 10;

function checkCount() {
    if (count > 5) {
        console.log('do something');
    }

    // some code...

    var count = 2;
}

checkCount();
```

```
var count = 10;

function checkCount() {
    var count;

    if (count > 5) {
        console.log('do something');
    }

    // some code...

    count = 2;
}

checkCount();
```

Advice #0

Use modern standards.

```
function bar() {
    var a = 1;
    var a = 2; // It's OK
}
```

```
function bar() {
    let b = 1;
    let b = 2; // Identifier 'b' has already been declared
}
```

```
function bar() {
    c = 2;
    var c = 1; // It's OK
}
```

```
function bar() {
    d = 2; // d is not defined
    let d = 1;
}
```

https://babeljs.io/

Bad

```
class Person {
    getFullName() {
        return this.first_name + ' ' +
            this.last_name;
    }
}
```

Good

```
class Person {
    getFullName() {
        return `${this.firstName} ${this.lastName}`;
    }
}
```

Bad

```
function sum() {
    var total = 0, len = arguments.length;

    for (var i = 0; i < len; i++ ) {
        total += arguments[i];
    }

    return total;
}
```

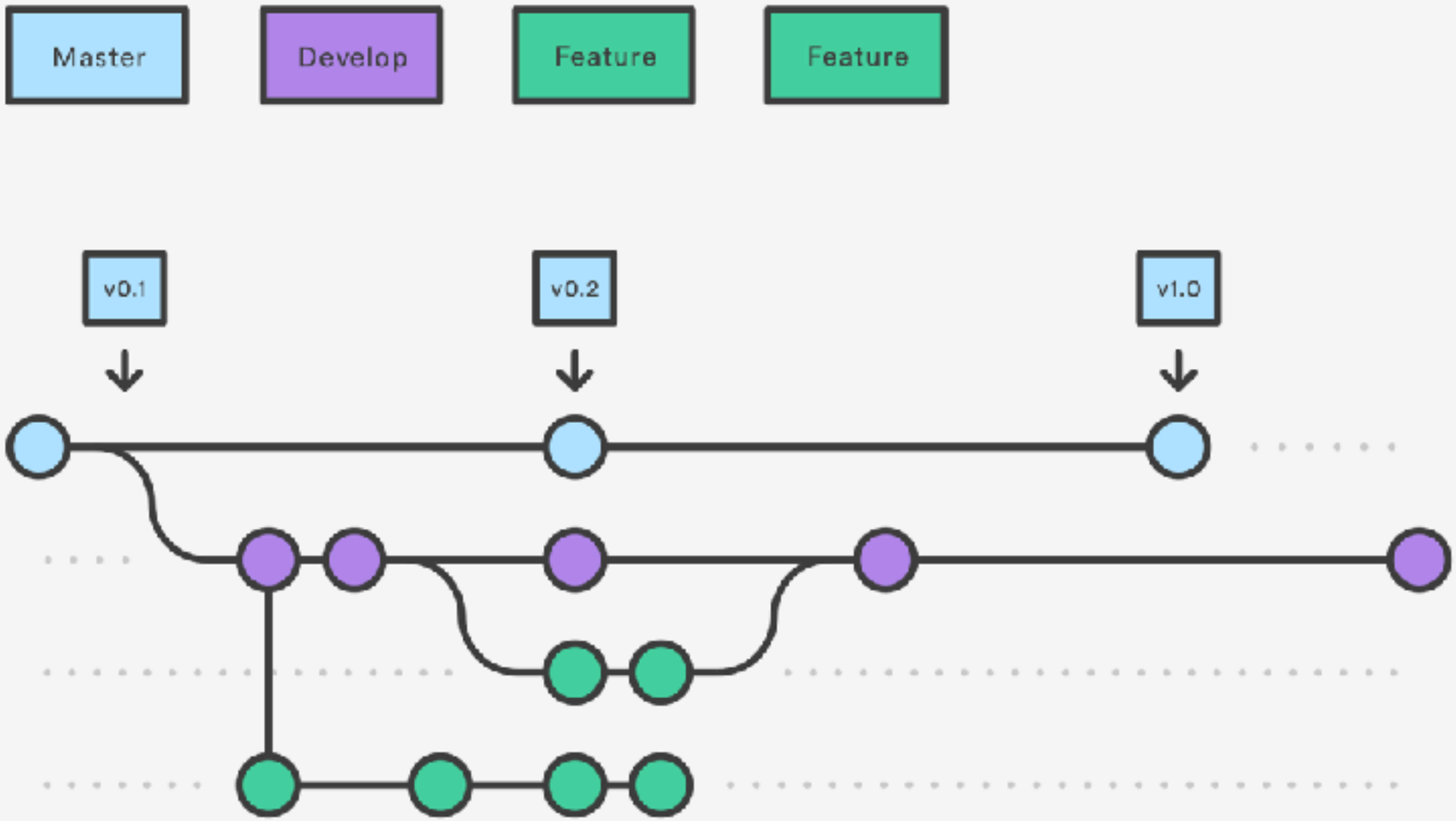
Good

```
function sum(...args) {
    return args.reduce((acc, i) => (
        acc + i
    ), 0);
}
```

Advice #1

Do code review.

A successful Git Branch model



Review rules

1. do design commit
2. choose two-four **appropriate** developers for review
3. take into account reviewers availability
4. remind yourself
5. write down review rules and share across the team
6. automate

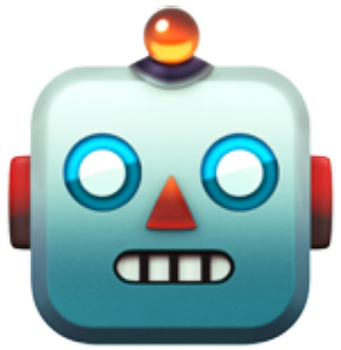
Code review automatization tools

https://github.com/devexp-org/review-bot

https://github.com/facebook/mention-bot

Advice #2

Use linting tools



Key points

1. ESlint (JSCS + JSlint)
2. put into a documentation all rules, try to explain them for newbies
3. check out airbnb rules, they are pretty cool https://github.com/airbnb/javascript

```
if (isValid === true) {
  // ...
}
```

Bad

```
if (isValid) {
  // ...
}
```

Good

```
if (name) {
  // ...
}
```

Bad

```
if (name !== '') {
  // ...
}
```

Good

```
if (collection.length) {
  // ...
}
```

Bad

```
if (collection.length > 0) {
  // ...
}
```

Good

Frontend DevConf

Илья Климов

«Медленный JS: сеанс черной магии с ее разоблачением»

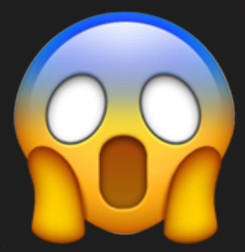
0:11 / 45:20

https://youtu.be/ZAJmJmKWNPw

Find the bug

```
1   function getNextDate(date) {
2       let newDate = new Date(date);
3
4       if (isNaN(newDate)) {
5           newDate = new Date();
6       }
7
8       newDate.setDate(newDate.getDate() + 1);
9
10      return newDate;
11  }
```

```
new Date();
new Date(1503307008737);
new Date('8/21/17');
new Date(year, month, date, hours, minutes, seconds, milliseconds);
new Date(Date);
```



indeterminacy

```
1   function doSomethingWithName(name) {
2       // ...
3
4
5       name.firstName = name.firstName.toUpperCase();
6       name.lastName = name.lastName.toUpperCase();
7   }
```

Advice #3

Write JSDoc

What is JSDoc?

JSDoc is a markup language used to annotate JavaScript source code files.

Using comments containing JSDoc, programmers can add documentation describing the application programming interface of the code they're creating.

This is then processed, by various tools, to produce documentation in accessible formats like HTML.

```
/**
 * Base page color
 * @constant
 * @type {string} - hex color code
 */
const BASE_COLOR = '#c00';
```

```
/**
 * @function showNotification
 * @async
 * @param {string} message - Text message to show in alert
 * @param {numver} [delay = 1000] - Message delay in miliseconds
 */
function showNotification(message, delay = 1000) {
    setTimeout(() => {
        alert(message)
    }, delay);
}
```

```
/**
 * Carried sum function
 * @param {number}
 * @return {number} - sum of two numbers
 * @example
 * // returns 3
 * sum(1)(2)
 */
function sum(a) {
    return function(b) {
        return a + b;
    }
}
```

```
/**
 * Represents text document
 * @class
 * @extends Document
 */
class TextDocument extends Document {
    /**
     * @this Document
     */
    handleSubmitClick = () => {
        this.isLoading = true;
    }
    /**
     * @param {object} e - The Event instance
     * @this Element
     */
    handleCloseClick(e) {
        e.currentTarget.innerText = 'Loading';
    }
}
```

```
/**
* @function setName
* @this User
* @fires Event#userChange
* @param {string} name - new name for user
*/
User.prototype.setName = function(name) {
    this.name = name;

    /**
     * Global userChange event
     * @event Event#userChange
     * @type {User}
     */
    Event.emit('userChange', this);
}
```

```
/**
 * Returns next day's date after given date.
 * @param {Date} date - current date
 * @returns {Date}
 */
function getNextDate(date) {
    let newDate = new Date(date);

    if (isNaN(newDate)) {
        newDate = new Date();
    }

    newDate.setDate(newDate.getDate() + 1);

    return newDate;
}


getNextDate('12/12/12');
```

Advice #4

Use strict typing

Tools:

1. Flow - https://flow.org/
2. TypeScript - https://www.typescriptlang.org/

Flow

1. Static type checker
2. Facebook project
3. Integration with babel
4. Stand-alone package
5. Low entry threshold
6. Not nullable types

Simple usage of Flow

```
1   // @flow
2   function getWords(sentence) {
3     return sentence.split(' ');
4   }
5
6   getWords(34); // Error!
```

```
1   // @flow
2   function getWords(sentence: string) : Array<string> {
3     return sentence.split(' ');
4   }
5
6   getWords(34); // Error!
```

Not Nullable type

```
1   // @flow
2   function getFullName(name: { firstName: string, lastName: string }) {
3       return `${name.firstName} ${name.lastName}`;
4   }
5
6   getFullName(null); // Error!
```

Maybe Type

```
1   // @flow
2   function getFullName(name: ?{ firstName: string, lastName: string }) {
3       return `${name.firstName} ${name.lastName}`;
4   }
5
6   getFullName(null); // Error!
```

Maybe Type

```
1   // @flow
2   function getFullName(name: ?{ firstName: string, lastName: string }) {
3       if (!name) {
4           return 'No Name';
5       }
6       return `${name.firstName} ${name.lastName}`;
7   }
8
9   getFullName(null); // Ok!
```

Typescript

1. Programming language
2. Microsoft project!
3. Based on ES2015
4. Stand-alone package
5. Not nullable types (`--strictNullChecks` in TypeScript 2)
6. A lot of documentation
7. Extensive support in IDEs

Simple usage of TypeScript

```
1
2    function getWords(sentence: string) : Array<string> {
3      return sentence.split(' ');
4    }
5
6    getWords(34); // Error!
```

TypeScript is programming language,
Flow is not 👆

```
1   // @flow
2   function getStringLength(x) /* : string */ {
3     return x.length;
4   }
5   getStringLength('Hello, world!');
```

Check it out

```
1   // @flow
2   function getNextDate(date: Date = new Date()) : Date {
3       date.setDate(date.getDate() + 1);
4
5       return date;
6   }
7
8   getNextDate(); // Ok
9   getNextDate(new Date('12/12/12')); // Ok
10  getNextDate('12/12/12'); // Error
11  getNextDate(1503327921764); // Error
```

Perfect or not?

```flow
// @flow
function getNextDate(date: Date = new Date()): Date {
    date.setDate(date.getDate() + 1);

    return date;
}

const currentDate = new Date('12/21/17');
const nextDate = getNextDate(currentDate);

console.log(nextDate); // Fri Dec 22 2017 00:00:00 GMT+0200 (EET) seems OK
console.log(currentDate); // Fri Dec 22 2017 00:00:00 GMT+0200 (EET) WTF?
```

Mutation! 

Advice #5

Learn the computer science

What else?

1. OOP
2. Functional programing
3. Design patterns
4. Algorithms and data structures
5. Databases
6. Computer security and cryptography
7. Computer networks
8. …

Pure functions

1. The function always evaluates the same result value given the same argument value.
2. Evaluation of the result does not cause any semantically observable side effect.

```
const user = {
    name: 'Oleg',
    aviability: false
}

function toggleAviability(user) {
    user.aviability = !user.aviability;
}
```

```
function showList(list) {
    if (list.length > 10) {
        Event.emit('showLongList');
    }

    Event.emit('showShortList');
}
```

```
function sum(a, b) {
    const result = a + b;
    console.log(result);
    return result;
}
```

```
const counter = (() => {
    let counter = 0;

    return () => {
        counter += 1;
        return counter;
    }
})()
```

```
function sum(...args) {
    return args.reduce((acc, i) => (acc + i), 0);
}
```

Final result

```
1   function getNextDate(date : Date) : Date {
2       const currentDate = new Date(date);
3       currentDate.setDate(date.getDate() + 1);
4
5       return currentDate;
6   }
```

What next?

Thank you!

Leave your feedback