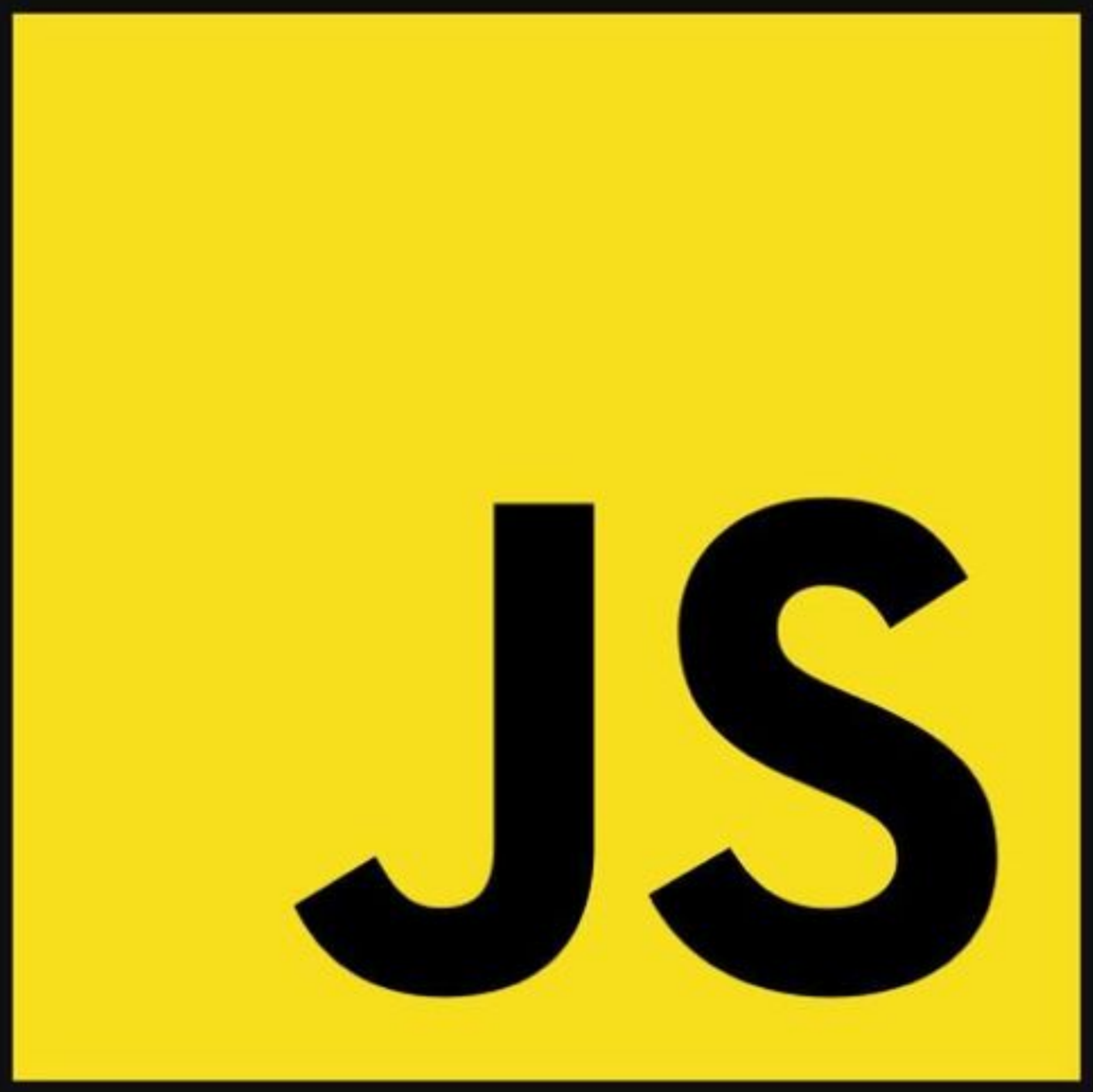
JS

Write Better JS Code

← Swipe



Joe Harrison
@frontendjoe

Readable Names

Although single letter variable names can have a use case, I'd avoid using them in this naming context.

```
const n = "Jeff"
const l = "Sheldon"
const a = 27
```

```
const name = "Jeff"
const location = "Sheldon"
const age = 27
```

I use single letter variable names all the time in React when working with redux and styled-components.

Joe Harrison
@frontendjoe

Avoid Unneeded Contexts

Don't add redundant context to variable names when already provided by it's containing object.

```
const user = {
  userName: "Jeff",
  userLocation: "Sheldon",
  userAge: 27,
}
```

```
const user = {
  name: "Jeff",
  location: "Sheldon",
  age: 27,
}
```

Joe Harrison
@frontendjoe

Avoid Hardcoding DRY

If the timeout value changes then you only have to change it in one place. Don't Repeat Yourself.

```
setTimeout(clearUserData, 500)
setTimeout(clearCache, 500)
```

```
const CLEAR_DURATION = 500

setTimeout(clearUserData, CLEAR_DURATION)
setTimeout(clearCache, CLEAR_DURATION)
```

Variables that are considered constants should be written in Screaming Snake Case.

Joe Harrison
@frontendjoe

Verbose Function Names

Lengthy function names are cool, as long as they describe what the function actually does.

```
const toggle = () => console.log("toggle")
const click = () => console.log("click")
```

```
const toggleDarkMode =
  () => console.log("toggleDarkMode")

const handleButtonClicked =
  () => console.log("handleButtonClicked")
```

handle and **toggle** are words I frequently use to prefix my function names – consistency improves readability.

Joe Harrison
@frontendjoe

Single Arguments

It's a controversial rule, but functions should have 0, 1, or 2 arguments. Destructuring a config object solves this.

```
const sendNotification(name, content, image) =>
  /* send logic */

sendNotification("Warning", "...", "warn.png"
)
```

```
const sendNotification({ name, content, image })
  /* send logic */

const notificationConfig = {
  name: "Warning",
  content: "...",
  image: "warn.png"
}

sendNotification(notificationConfig)
```

I love the readability of doing things this way 👌

Joe Harrison
@frontendjoe

Literals Over Switches

This will improve code readability and even overall performance.

```
const getColorByStatus = status =>
  switch (status) {
    case "success":
      return "green";
    case "warning":
      return "red";
    default:
      return "blue";
  }
```

```
const statusColors = {
  warning: "red",
  success: "green",
};

const getColorByStatus = status =>
  statusColors[status] || "blue"
```

Joe Harrison
@frontendjoe

Don’t Overcomment

Too many comments in your code can devalue their meaning. On the next example you will see how we can write more functions to help document our code.

```
const handleUserCreated = user => {
  // create user id
  const id =
    `id${Math.random().toString(16).slice(2)}`

  // apply new user id
  user.id = id

  // create user in database
  createUser(user)
}
```

Joe Harrison
@frontendjoe

Avoid Complex Functions

By writing an extra function there is no need to add a comment describing what it does. Making function names more verbose also contributes to better readability in your code.

```
const createUserId = () =>
  `id${Math.random().toString(16).slice(2)}`

const handleCreateUserClicked = user => {
  user.id = createUserId()
  createUserInDatabase(user)
}
```

Seeing many functions, each with a single use case, will make your code flow much better and be a lot easier to come back to in future.

Joe Harrison
@frontendjoe

Was It Useful?

Let me know in the comments