Automating Code Changes for 100 Repositories

Getting Started With Codemods 👨‍💻

@mrkosima

About me

Konstantin Klimashevich

Software Development Consultant at Xebia

Program Committee Member at GitNation

@mrkosima

klimashevich

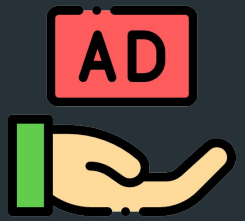
Marktplaats.nl - largest marketplace in NL

6.5M

visitors
per day

19M

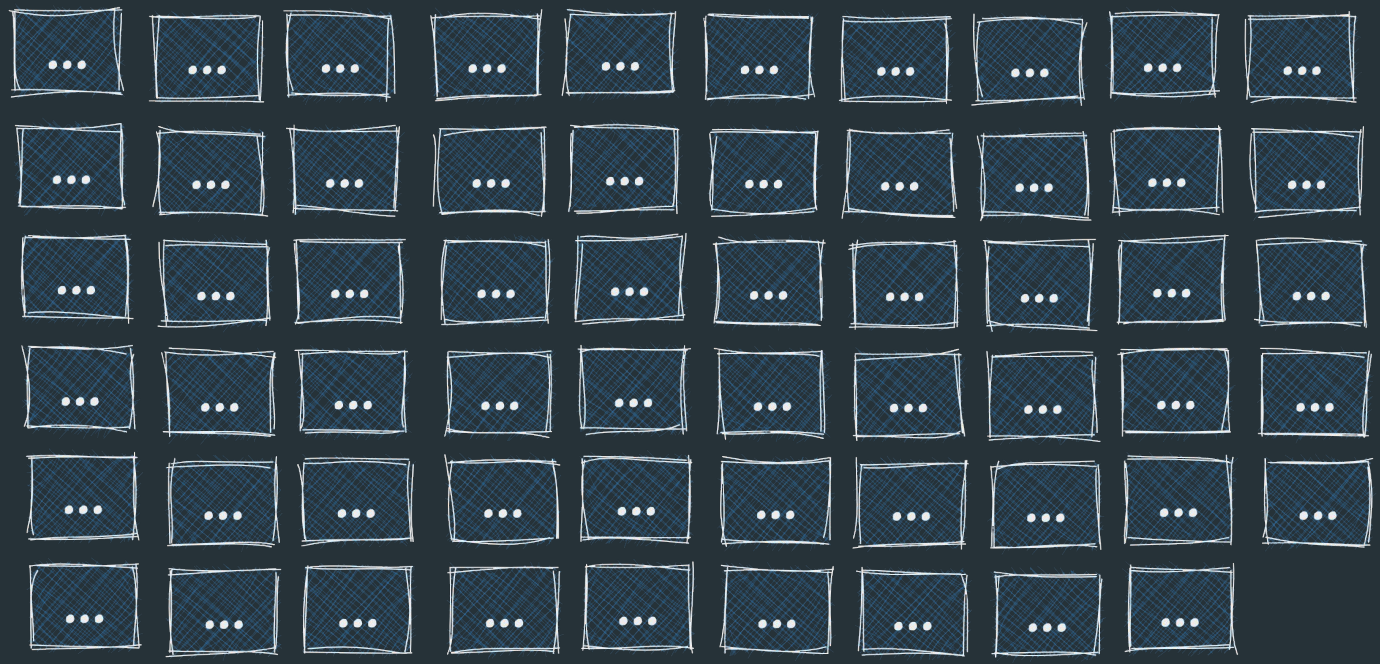
live ads
at any time



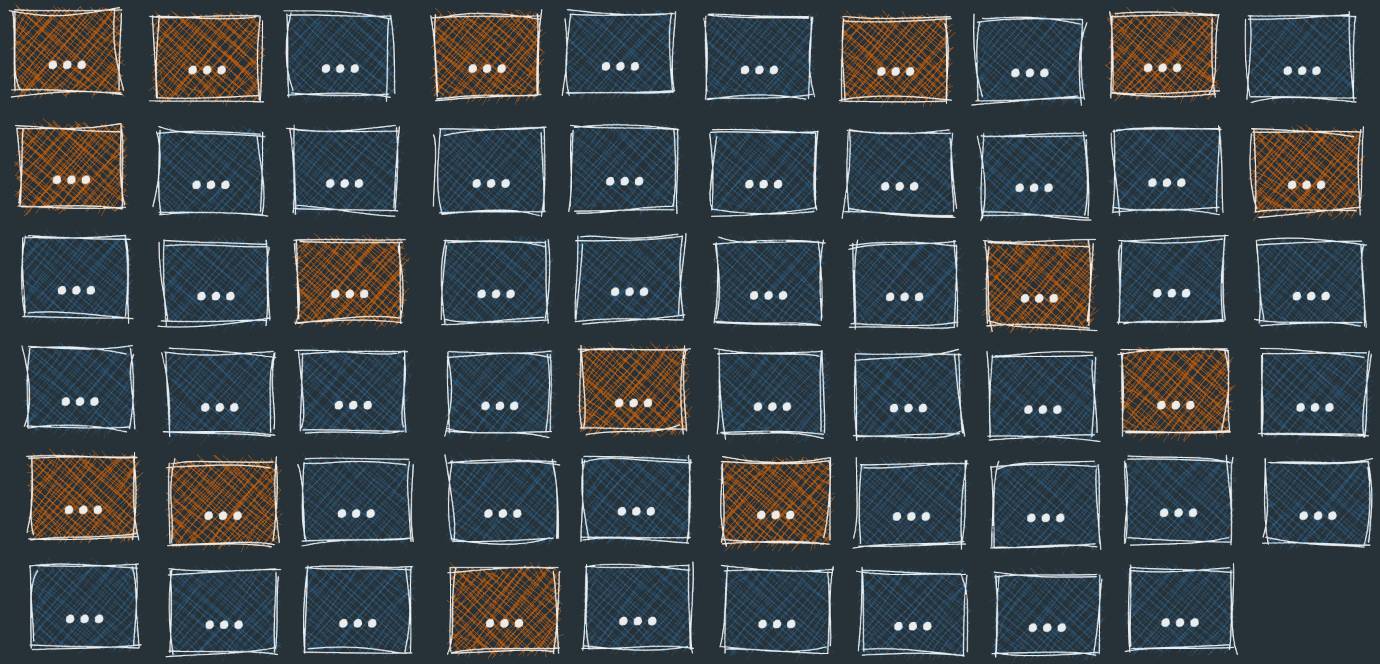
350K

new adv
per day

BFF services



BFF services - daily releases



59 Services in Production > 15 prod releases / day

JS/TS repositories

JS/TS repositories - code ownership

Domain FE Teams

FE Platform Team

FE Platform repositories

Change code

Frontend repositories

@mp/design-system@1.0.0

```
export const PrimaryButton = ({ onClose }) => {
    /*...*/
}
```

Toast.tsx

```
import { Card, PrimaryButton, Text } from '@mp/design-system';

export const Toast = ({ message, onClose }) => {
  return (
    <Card>
        <Text>{message}</Text>
        <PrimaryButton onClick={onClose}>
            Close
        </PrimaryButton>
    </Card>
  )
}
```

Toast.tsx

*@mp/design-system@**1.0.0***

```
import { Card, PrimaryButton, Text } from '@mp/design-system';

export const Toast = ({ message, onClose }) => {
  return (
    <Card>
        <Text>{message}</Text>
        <PrimaryButton onClick={onClose}>
            Close
        </PrimaryButton>
    </Card>
  )
}
```

@mp/design-system@2.0.0

```
export const Button = ({ kind, onClose }) => {
    /*...*/
}
```

Toast.tsx

*@mp/design-system@**2.0.0***

```
import { Card, Button, Text } from '@mp/design-system';

export const Toast = ({ message, onClose }) => {
  return (
    <Card>
        <Text>{message}</Text>
        <Button kind="primary" onClick={onClose}>
            Close
        </Button>
    </Card>
  )
}
```

Toast.tsx

```
- import { Card, PrimaryButton, Text } from '@mp/design-system';
+ import { Card, Button, Text } from '@mp/design-system';

export const Toast = ({ message, onClick }) => {
  return (
    <Card>
        <Text>{message}</Text>
-       <PrimaryButton onClick={onClose}>
+       <Button kind="primary" onClick={onClose}>
            Close
 -      </PrimaryButton>
 +      </Button>
    </Card>
  )
}
```

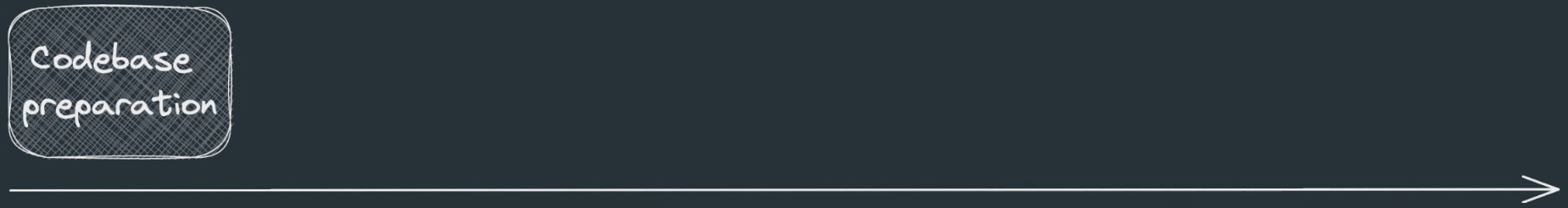
*.tsx

```
- import { Card, PrimaryButton, Text } from '@mp/design-system';
+ import { Card, Button, Text } from '@mp/design-system';

export const Toast = ({
  return (
    <Card>
      <Text>
-       <Prima
+       <Butto
            Cl
      -     </Prima
      +     </Butto
    </Card>
  )
}
```

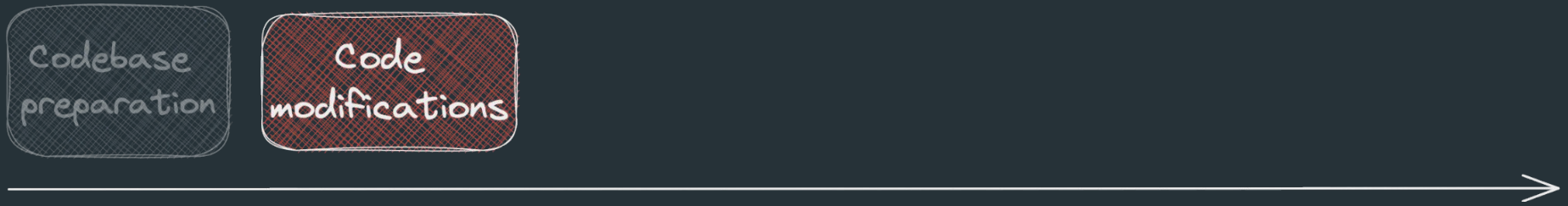
*.jsx

```
- import { Card, PrimaryButton, Text } from '@mp/design-system';
+ import { Car                  Text } from '@mp/design-system';

export const
  return (
    <Card>
      <Tex
-     <Pri
+     <But

 -    </Prima
 +    </Butto
    </Card>
  )
}
```

Automation

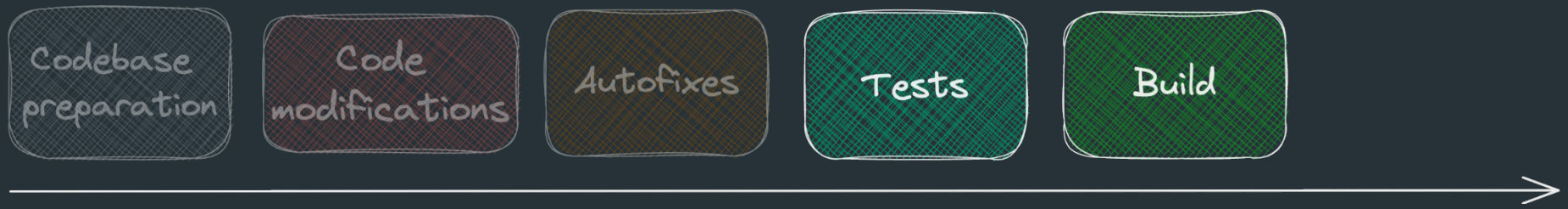


Automation



Automation

Automation


Codebase preparation
Code modifications
Autofixes
Tests
Build

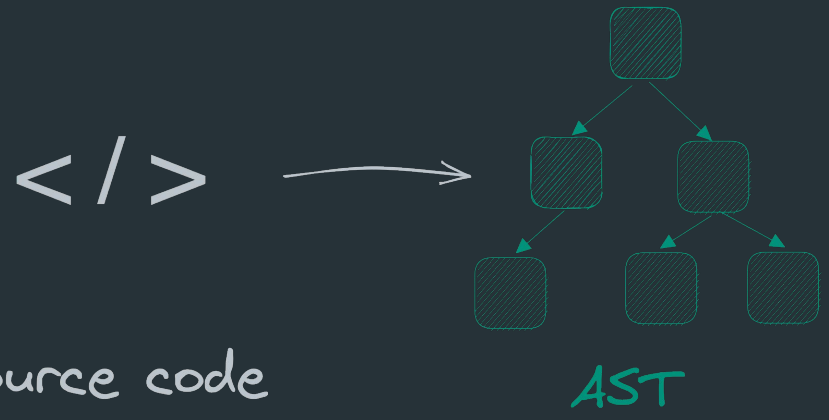
Automation


Codebase preparation
Code modifications
Autofixes
Tests
Build
Commit changes

Automation


Codebase preparation
Code modifications
Autofixes
Tests
Build
Commit changes

Abstract Syntax Tree (AST)



```
function greet() {
    return "Hi Berlin 👋!"
}
```

```
function greet() {
    return "Hi Berlin 👋!"
}
```

```
Program {
  - body: [
    - FunctionDeclaration {
      - id: Identifier {
          name: "greet"
        }
        params: [ ]
        body: BlockStatement {
          - body: [
            - ReturnStatement {
                - argument: Literal {
                    value: "Hi Berlin 👋!"
                  }
              }
            ]
        }
        expression: false
        generator: false
        async: false
      }
  ]
  sourceType: "module"
}
```

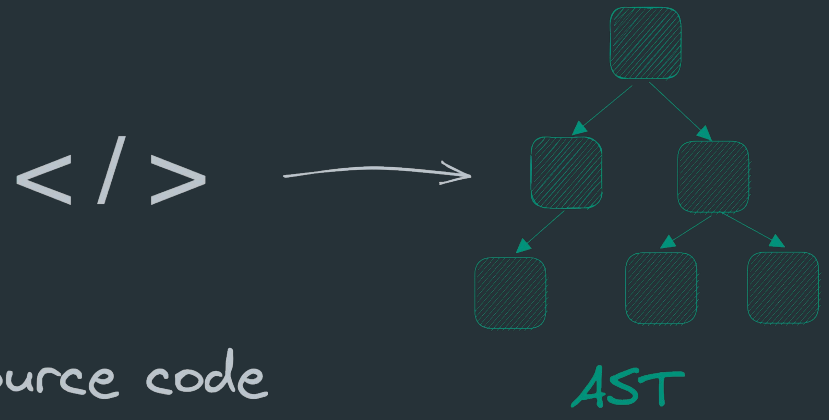
```
function greet() {
    return "Hi Berlin 👋!"
}
```

```
Program {
  - body: [
    - FunctionDeclaration {
      - id: Identifier {
          name: "greet"
        }
        params: [ ]
        body: BlockStatement {
          - body: [
            - ReturnStatement {
                - argument: Literal {
                    value: "Hi Berlin 👋!"
                  }
              }
            ]
        }
        expression: false
        generator: false
        async: false
      }
  ]
  sourceType: "module"
}
```

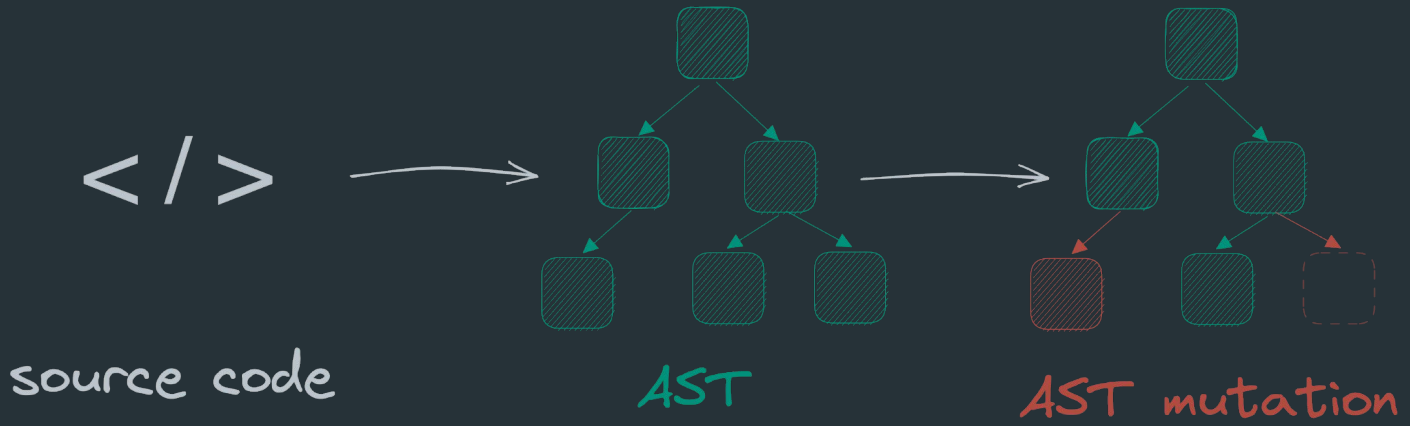
```
function greet() {
    return "Hi Berlin 👋!"
}
```

```
Program {
  - body: [
    - FunctionDeclaration {
      - id: Identifier {
          name: "greet"
        }
        params: [ ]
        body: BlockStatement {
          - body: [
            - ReturnStatement {
                - argument: Literal {
                    value: "Hi Berlin 👋!"
                  }
              }
            ]
        }
        expression: false
        generator: false
        async: false
      }
  ]
  sourceType: "module"
}
```

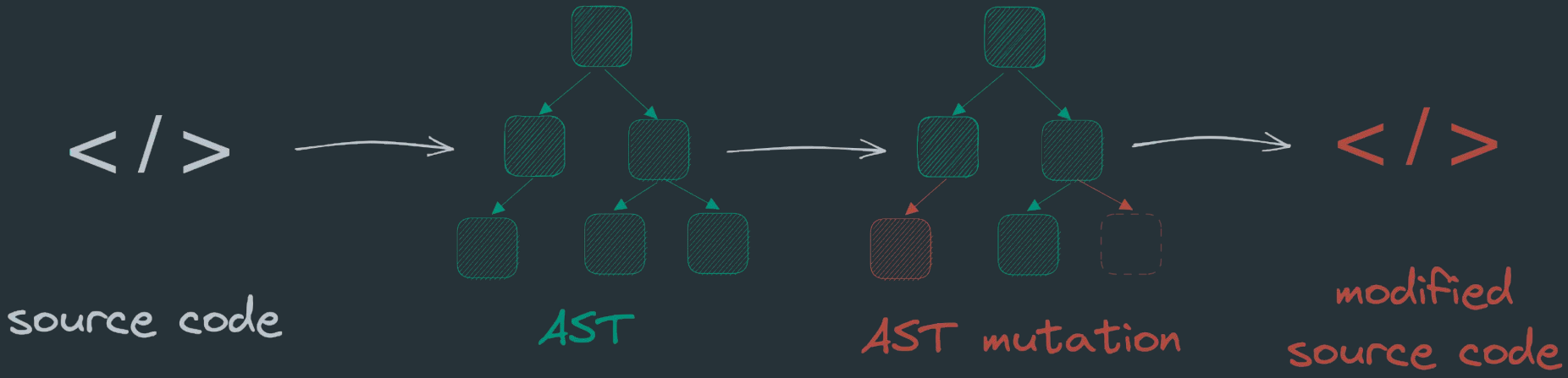
Abstract Syntax Tree (AST)



AST mutation



AST to source code



Toast.jsx

```
- import { Card, Button, Text } from '@mp/design-system';
+ import { Card, Button, Text } from '@mp/design-system';

export const Toast = ({ message, onClick }) => {
  return (
    <Card>
        <Text>{message}</Text>
-       <PrimaryButton onClick={onClick}>
+       <Button kind="primary" onClick={onClose}>
            Close
 -      </PrimaryButton>
 +      </Button>
    </Card>
  )
}
```

JScodeshift

```
npx jscodeshift -t transform.js Toast.jsx
```

JScodeshift

```
npx jscodeshift -t transform.js Toast.jsx
```

JScodeshift

```
npx jscodeshift -t transform.js Toast.jsx
```

transform.js

```
export default function transform(fileInfo, api, options) {
  const j = api.jscodeshift;
  const ast = j(fileInfo.source);

  ast
    .find(j.ImportDeclaration, { source: { value: "@mp/design-system" } })
    .find(j.ImportSpecifier)
    .filter((path) => path.node.imported.name === "PrimaryButton")
    .replaceWith(j.importSpecifier(j.identifier("Button")));

  ast
    .find(j.JSXElement)
    .filter(path => path.value.openingElement.name.name === "PrimaryButton")
    .forEach(element => {
      const newEl = j.jsxElement(
        j.jsxOpeningElement(j.jsxIdentifier("Button"), [
          j.jsxAttribute(j.jsxIdentifier("kind"), j.stringLiteral("primary")),
          ...element.node.openingElement.attributes
        ]),
        j.jsxClosingElement(j.jsxIdentifier("Button")),
        element.node.children
      );

      j(element).replaceWith(newEl);
    });

  return ast.toSource();
}
```

transform.js - API

```
export default function transform(fileInfo, api, options) {



}
```

transform.js - to AST and back

```
export default function transform(fileInfo, api, options) {
  const j = api.jscodeshift;
  const ast = j(fileInfo.source); // source code to AST

  /* AST mutations here */

  return ast.toSource();  // AST to source code
}
```

transform.js - mutations

```
export default function transform(fileInfo, api, options) {
  const j = api.jscodeshift;
  const ast = j(fileInfo.source); // source code to AST

  /* AST mutations here */

  return ast.toSource();  // AST to source code
}
```

Toast.tsx

```
import { Card, PrimaryButton, Text } from '@mp/design-system';
```

Toast.tsx

```
import { Card, PrimaryButton, Text } from '@mp/design-system';
```

```
...
- ImportDeclaration {
  - source: {
      value: "@mp/design-system"
    }
  - specifiers: [
      + ImportSpecifier {...}
      - ImportSpecifier {
        - imported {
            name: "PrimaryButton"
          }
        }
      + ImportSpecifier {...}
    ]
  }
```

transform.js - update import

```
import { Card, PrimaryButton, Text } from '@mp/design-system';

export default function transform(fileInfo, api, options) {
  /*...*/




  /*...*/
}
```

transform.js

```
import { Card, PrimaryButton, Text } from '@mp/design-system';
```

```
export default function transform(fileInfo, api, options) {
  /*...*/

  ast
    .find(j.ImportDeclaration, { source: { value: "@mp/design-system" }})
    .find(j.ImportSpecifier)
    .filter((path) => path.node.imported.name === "PrimaryButton")
    .replaceWith(j.importSpecifier(j.identifier("Button")));


  /*...*/
}
```

transform.js

```
import { Card, PrimaryButton, Text } from '@mp/design-system';
```

```
export default function transform(fileInfo, api, options) {
  /*...*/

  ast
    .find(j.ImportDeclaration, { source: { value: "@mp/design-system" }})
    .find(j.ImportSpecifier)
    .filter((path) => path.node.imported.name === "PrimaryButton")
    .replaceWith(j.importSpecifier(j.identifier("Button")));


  /*...*/
}
```

transform.js

```
import { Card, PrimaryButton, Text } from '@mp/design-system';
```

```
export default function transform(fileInfo, api, options) {
  /*...*/

  ast
    .find(j.ImportDeclaration, { source: { value: "@mp/design-system" }})
    .find(j.ImportSpecifier)
    .filter((path) => path.node.imported.name === "PrimaryButton")
    .replaceWith(j.importSpecifier(j.identifier("Button")));


  /*...*/
}
```

transform.js

```
import { Card, Button, Text } from '@mp/design-system';
```

```
export default function transform(fileInfo, api, options) {
  /*...*/

  ast
    .find(j.ImportDeclaration, { source: { value: "@mp/design-system" }})
    .find(j.ImportSpecifier)
    .filter((path) => path.node.imported.name === "PrimaryButton")
    .replaceWith(j.importSpecifier(j.identifier("Button")));


  /*...*/
}
```

transform.js

```
<PrimaryButton onClick={onClose}>Close<PrimaryButton>
```

```
export default function transform(fileInfo, api, options) {
  /*...*/









  /*...*/
}
```

transform.js

```
<PrimaryButton onClick={onClose}>Close<PrimaryButton>
```

```
export default function transform(fileInfo, api, options) {
  /*...*/

  ast
    .find(j.JSXElement)
    .filter(p => p.value.openingElement.name.name ==="PrimaryButton")
    .forEach(element => {

    });


  /*...*/
}
```

transform.js

```
<PrimaryButton onClick={onClose}>Close<PrimaryButton>
```

```
export default function transform(fileInfo, api, options) {
  /*...*/

  ast
    .find(j.JSXElement)
    .filter(p => p.value.openingElement.name.name ==="PrimaryButton")
    .forEach(element => {

    });


  /*...*/
}
```

transform.js

</>

```
export default function transform(fileInfo, api, options) {
  /*...*/
  const newEl = j.jsxElement(
    j.jsxOpeningElement(j.jsxIdentifier("Button"), [
      j.jsxAttribute(j.jsxIdentifier("kind"), j.stringLiteral("primary")),
      ...element.node.openingElement.attributes
    ]),
    j.jsxClosingElement(j.jsxIdentifier("Button")),
    element.node.children
  );
  j(element).replaceWith(newEl);
  /*...*/
}
```

transform.js

```
<Button />
```

```
export default function transform(fileInfo, api, options) {
  /*...*/
  const newEl = j.jsxElement(
    j.jsxOpeningElement(j.jsxIdentifier("Button"), [
      j.jsxAttribute(j.jsxIdentifier("kind"), j.stringLiteral("primary")),
      ...element.node.openingElement.attributes
    ]),
    j.jsxClosingElement(j.jsxIdentifier("Button")),
    element.node.children
  );
  j(element).replaceWith(newEl);
  /*...*/
}
```

transform.js

```
<Button kind="primary" />
```

```
export default function transform(fileInfo, api, options) {
  /*...*/
  const newEl = j.jsxElement(
    j.jsxOpeningElement(j.jsxIdentifier("Button"), [
      j.jsxAttribute(j.jsxIdentifier("kind"), j.stringLiteral("primary")),
      ...element.node.openingElement.attributes
    ]),
    j.jsxClosingElement(j.jsxIdentifier("Button")),
    element.node.children
  );
  j(element).replaceWith(newEl);
  /*...*/
}
```

transform.js

```
<Button kind="primary" onClick={onClose} />
```

```
export default function transform(fileInfo, api, options) {
  /*...*/
  const newEl = j.jsxElement(
    j.jsxOpeningElement(j.jsxIdentifier("Button"), [
      j.jsxAttribute(j.jsxIdentifier("kind"), j.stringLiteral("primary")),
      ...element.node.openingElement.attributes
    ]),
    j.jsxClosingElement(j.jsxIdentifier("Button")),
    element.node.children
  );
  j(element).replaceWith(newEl);
  /*...*/
}
```

transform.js

```
<Button kind="primary" onClick={onClose}></Button>
```

```
export default function transform(fileInfo, api, options) {
  /*...*/
  const newEl = j.jsxElement(
    j.jsxOpeningElement(j.jsxIdentifier("Button"), [
      j.jsxAttribute(j.jsxIdentifier("kind"), j.stringLiteral("primary")),
      ...element.node.openingElement.attributes
    ]),
    j.jsxClosingElement(j.jsxIdentifier("Button")),
    element.node.children
  );
  j(element).replaceWith(newEl);
  /*...*/
}
```

transform.js

```
<Button kind="primary" onClick={onClose}>Close</Button>
```

```
export default function transform(fileInfo, api, options) {
  /*...*/
  const newEl = j.jsxElement(
    j.jsxOpeningElement(j.jsxIdentifier("Button"), [
      j.jsxAttribute(j.jsxIdentifier("kind"), j.stringLiteral("primary")),
      ...element.node.openingElement.attributes
    ]),
    j.jsxClosingElement(j.jsxIdentifier("Button")),
    element.node.children
  );
  j(element).replaceWith(newEl);
  /*...*/
}
```

transform.js

```
<Button kind="primary" onClick={onClose}>Close</Button>
```

```
export default function transform(fileInfo, api, options) {
  /*...*/
  const newEl = j.jsxElement(
    j.jsxOpeningElement(j.jsxIdentifier("Button"), [
      j.jsxAttribute(j.jsxIdentifier("kind"), j.stringLiteral("primary")),
      ...element.node.openingElement.attributes
    ]),
    j.jsxClosingElement(j.jsxIdentifier("Button")),
    element.node.children
  );
  j(element).replaceWith(newEl);
  /*...*/
}
```

JScodeshift

```
npx jscodeshift -t transform.js src/**/*
```

Automation + codemod

Automation + codemod

Takeways

Automate repetitive tasks

Use power of AST for code changes

Come home and write your first codemod*

*if haven't done it yet

Thank you!

@mrkosima