Observable

Enhancing EventTarget

Observable

- Composable enhancement of Event Target
- Allows events to be coordinated in async functions

Consuming an EventTarget (ET) as an Observable

```
domElement.
  on('mousemove').
  forEach(e => console.log(e));
```

Consuming ET as Observable with Capture Options

```
domElement.
  on('mousemove', { capture: true }).
  forEach(e => console.log(e));
```

Consuming ET as Observable with Cancellation

```
const { token, cancel } = CancelToken.source();

const subscriptionPromise =
  domElement.
    on('mousemove', { capture: true }).
    forEach(e => console.log(e), token);



// unsubscribe later
cancel();
```

rejects

Composing Observables

Composing Observables

- Observables can be composed like Arrays (ex. map, filter, etc)
- Composition can be enabled with user-land libraries like lodash
- Future plan to add popular functions to Observable prototype

Contrived composition example

```
import { map, filter } from 'lodash-for-events';

function mouseMoveCoordinatesInRect(element, rect) {
  let events = element.on('mousemove');
  let coordinates = map(events, e => ({ x: e.clientX, y: e.clientY }));

  return filter(coordinates, ({ x, y }) => {
    return x >= rect.left && x <= rect.right &&
           y >= rect.top && y <= rect.bottom;
  });
}

mouseMoveCoordinatesInRect(document.body, new DOMRect(0, 0, 100, 100)).
  forEach(coords => drawPixelAt(coords));
```

If we improve Observable.prototype this becomes...

```
function mouseMoveCoordinatesInRect(element, rect) {
  return element.on('mousemove').
    map(e => ({ x: e.clientX, y: e.clientY })).
    filter(coordinates, ({ x, y }) => {
      return x >= rect.left && x <= rect.right &&
             y >= rect.top && y <= rect.bottom;
    });
}
```

Composition Use Case: Draw Signature on Canvas

```
async function drawSignature(signatureCanvas, cancelToken) {
  await.cancelToken = cancelToken;











  // snip...

}
```

[illegible]

Composition with user-land libraries

```
import { _ } from 'lodash-for-events';

async function drawSignature(signatureCanvas, cancelToken) {
  await.cancelToken = cancelToken;
  const sigMouseDowns = _(signatureCanvas.on('mousedown'));









  // snip...

}
```

[illegible]

Composition: Map Function

```
import { _ } from 'lodash-for-events';

async function getSignature(signatureCanvas, cancelToken) {
  await.cancelToken = cancelToken;
  const toPoint = e => ({ x: e.offsetX, y: e.offsetY });
  const sigMouseDowns = _(signatureCanvas.on('mousedown')).map(toPoint);











  // snip...

}
```

Composition: Await Events with First

```
import { _ } from 'lodash-for-events';

async function getSignature(signatureCanvas, cancelToken) {
  await.cancelToken = cancelToken;
  const toPoint = e => ({ x: e.offsetX, y: e.offsetY });
  const sigMouseDowns = _(signatureCanvas.on('mousedown')).map(toPoint);



  let lastPointClicked = await sigMouseDowns.first(cancelToken);







  // snip...

}
```

[illegible]

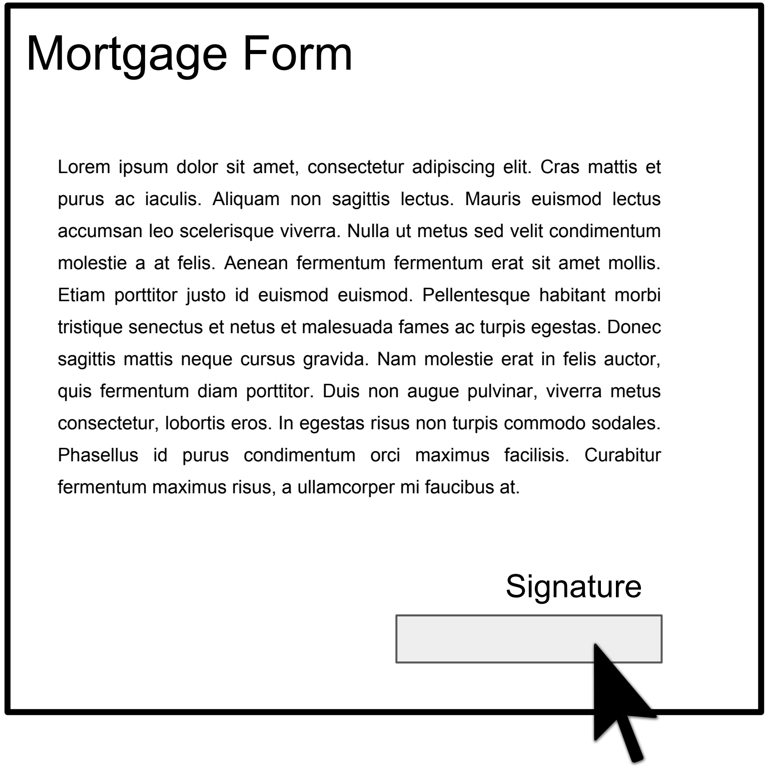
Composition: Handle Event 'til another Event Occurs

```
import { _ } from 'lodash-for-events';

async function getSignature(signatureCanvas, cancelToken) {
  await.cancelToken = cancelToken;
  const toPoint = e => ({ x: e.offsetX, y: e.offsetY });
  const sigMouseDowns = _(signatureCanvas.on('mousedown')).map(toPoint);
  const sigMouseMoves = _(signatureCanvas.on('mousemove')).map(toPoint);
  const sigMouseUps   = _(signatureCanvas.on('mouseup')).map(toPoint);

  let lastPointClicked = await sigMouseDowns.first(cancelToken);

  await sigMouseMoves.takeUntil(sigMouseUps).
    forEach(
      point => {
        strokeLine(signatureCanvas, lastPointClicked.x, lastPointClicked.y, point.x, point.y);
        lastPointClicked = point;
      },
      cancelToken);

}
```

[illegible]

Using Events in Async Workflows

Example: Form that collects e-signature

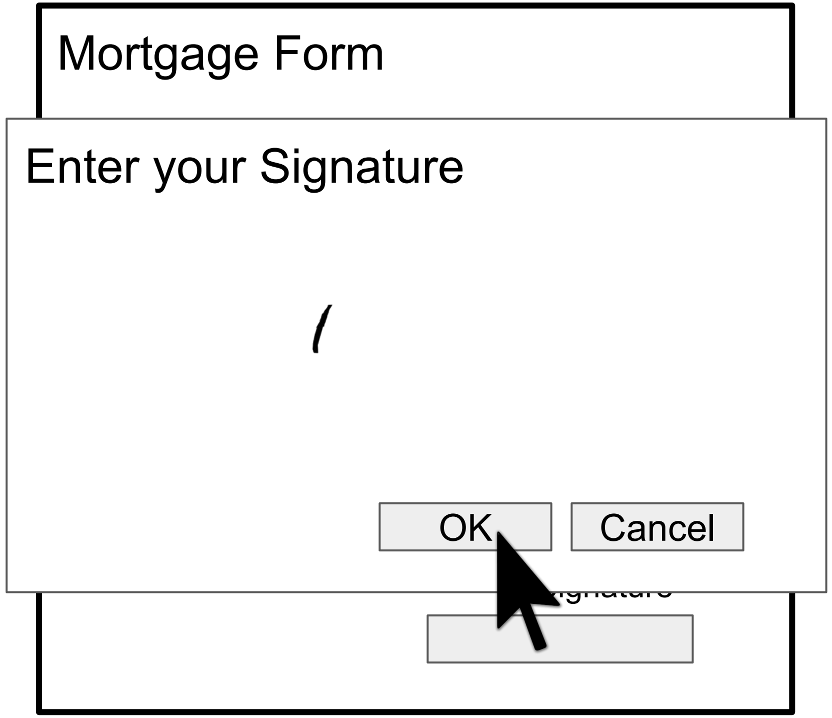


Mortgage Form

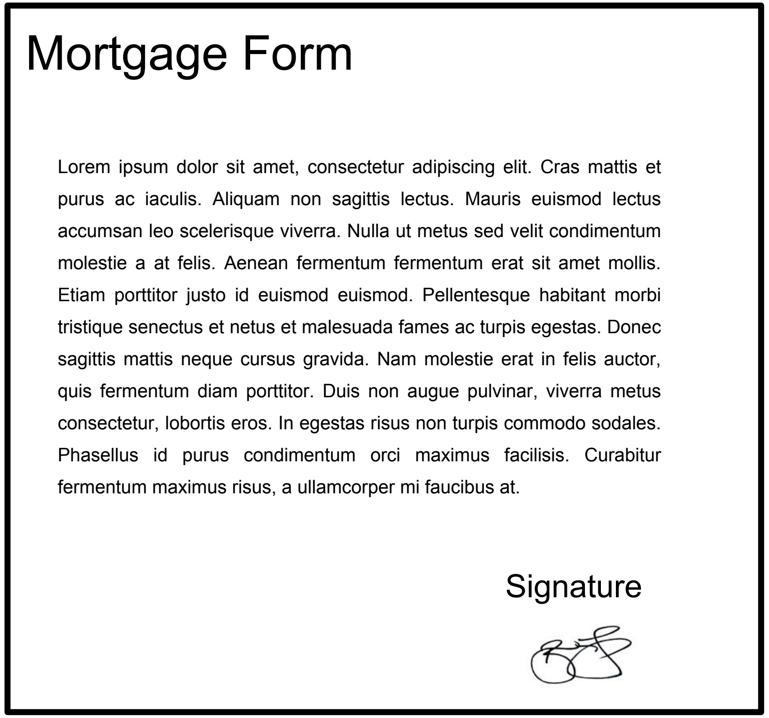
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras mattis et purus ac iaculis. Aliquam non sagittis lectus. Mauris euismod lectus accumsan leo scelerisque viverra. Nulla ut metus sed velit condimentum molestie a at felis. Aenean fermentum fermentum erat sit amet mollis. Etiam porttitor justo id euismod euismod. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec sagittis mattis neque cursus gravida. Nam molestie erat in felis auctor, quis fermentum diam porttitor. Duis non augue pulvinar, viverra metus consectetur, lobortis eros. In egestas risus non turpis commodo sodales. Phasellus id purus condimentum orci maximus facilisis. Curabitur fermentum maximus risus, a ullamcorper mi faucibus at.

Signature

Example: Form that collects e-signature



Example: Form that collects e-signature



Mortgage Form

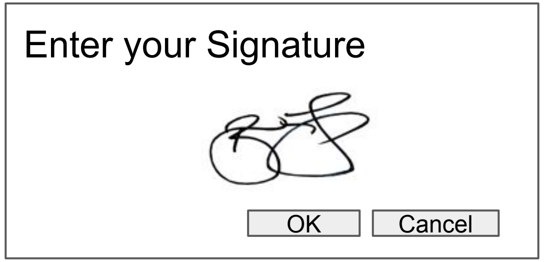
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras mattis et purus ac iaculis. Aliquam non sagittis lectus. Mauris euismod lectus accumsan leo scelerisque viverra. Nulla ut metus sed velit condimentum molestie a at felis. Aenean fermentum fermentum erat sit amet mollis. Etiam porttitor justo id euismod euismod. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec sagittis mattis neque cursus gravida. Nam molestie erat in felis auctor, quis fermentum diam porttitor. Duis non augue pulvinar, viverra metus consectetur, lobortis eros. In egestas risus non turpis commodo sodales. Phasellus id purus condimentum orci maximus facilisis. Curabitur fermentum maximus risus, a ullamcorper mi faucibus at.

Signature

[illegible]

Async Function: getSignature

```
import { _ } from 'lodash-for-events';

async function getSignature(token) {
  const signatureDialog = createAndDisplaySignatureDialog();
  try {
    const signatureCanvas = signatureDialog.querySelector('.signatureCanvas');

    const okButton = signatureDialog.querySelector('.okbutton');
    const cancelButton = signatureDialog.querySelector('.cancelbutton');
    const ok = _(okButton.on('click')).first(token);
    const cancel = _(cancelButton.on('click')).first(token);

    // concurrently handle signature draws, an ok click, and a cancel click
    return await Promise.race([
      drawSignature(signatureCanvas, okButton, token),
      ok.then(() => signatureCanvas.toDataURL()),
      cancel.then(() => undefined))
    ]);
  }
  finally {
    signatureDialog.remove();
  }
}
```



Using Events in Async Functions: drawSignature

```
import { _ } from 'lodash-for-events';

async function drawSignature(signatureCanvas, okButton, token) {
  await.cancelToken = cancelToken;
  const context = signatureCanvas.getContext('2d');
  const toPoint = e => ({ x: e.offsetX, y: e.offsetY });
  const sigMouseDowns = _(signatureCanvas.on('mousedown')).map(toPoint);
  const sigMouseMoves = _(signatureCanvas.on('mousemove')).map(toPoint);
  const sigMouseUps   = _(signatureCanvas.on('mouseup')).map(toPoint);

  while(true) {
    let lastPoint = await sigMouseDowns.first(token);

    await sigMouseMoves.takeUntil(sigMouseUps).
      forEach(
        point => {
          strokeLine(context, lastPoint.x, lastPoint.y, point.x, point.y);
          okButton.disabled = false;
          lastPoint = point;
        },
        token);
  }
}
```

Benefits of Observable

- Coordinate event streams in async functions
- takeUntil: combine multiple infinite event streams into a finite stream which can be **awaited**
- Can be adapted into Asynchronous Iterators

Appendix

Observable Class

```
class Observable<T> {
    constructor(subscribeDefn: Function)

    subscribe(observer: Observer, token: CancelToken): void

   forEach(nextFn: Function, token: CancelToken): Promise

    [Symbol.observable](): Observable

    static of(...items): Observable

    static from(ObservableLike: Object): Observable
}
```

Observer Interface

```
interface Observer {

    next(value),

   // try/else equivalent
    else(error),

    complete(value)

    // receives object sent during unsubscription/cancellation
   // try/catch equivalent
    catch(cancel)
}
```

Adapting EventTarget to Observable

```
EventTarget.prototype.on = function(name, options) {
    // constructor passed Observable.prototype.subscribe(observer, token) defn
    return new Observable((observer, token) => {
        const handler = e => {
            if (token.reason === undefined) {
                observer.next(e);
            }
        };

        this.addEventListener(name, handler, options);

        token.promise.then(cancel => {
            this.removeEventListener(name, handler, options);
            observer.catch(cancel);
        });
    });
}
```