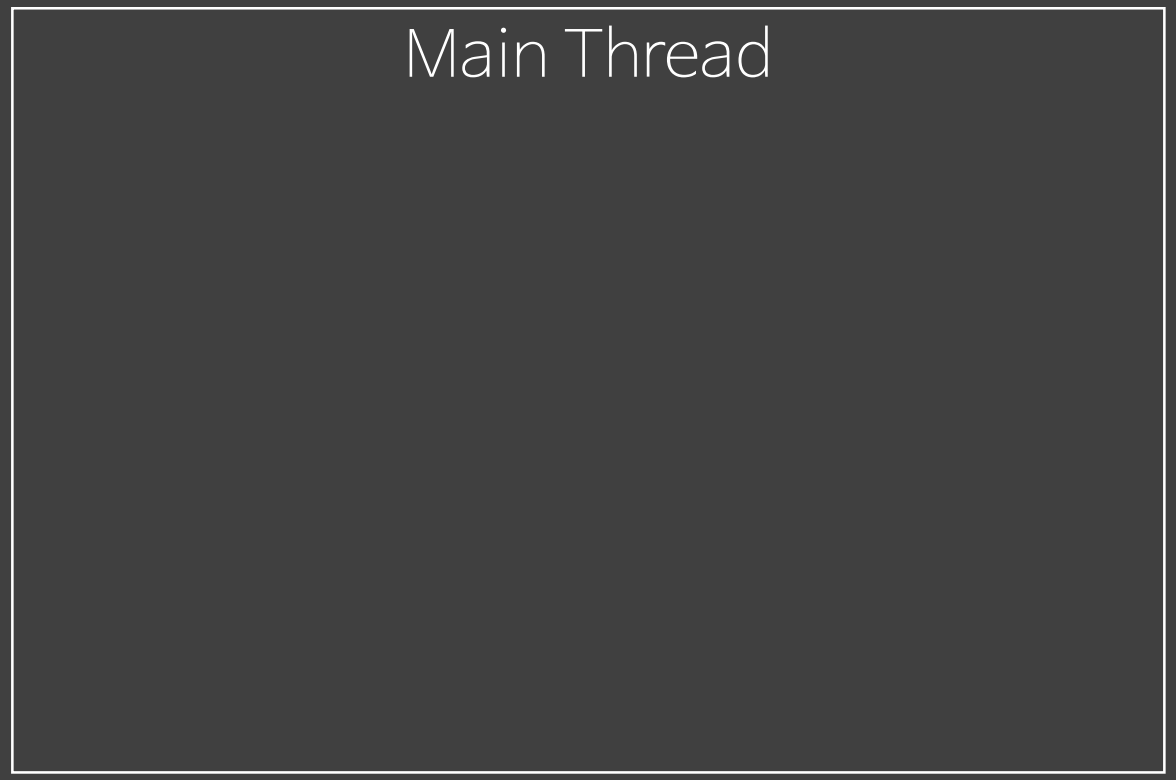
NONE BLOCKING JAVASCRIPT

1 2 3 4 5 6

WebWorker


Main Thread

Main Thread

```
const worker = new Worker("a.js");
```

Main Thread

```
const worker = new Worker("a.js");
```

Background Thread
a.js

Main Thread

```
const worker = new Worker("a.js");
worker.postMessage("hello");
```

Background Thread
a.js

Main Thread

```
const worker = new Worker("a.js");
worker.postMessage("hello");
```

Background Thread
a.js

```
onmessage = ({data})=>{
    console.log(data);

};
```

Main Thread

```
const worker = new Worker("a.js");
worker.postMessage("hello");
```

Background Thread
a.js

```
onmessage = ({data})=>{
    console.log(data);
    postMessage("world")
};
```

Main Thread

```
const worker = new Worker("a.js");
worker.postMessage("hello");
onmessage = ({data})=>{
    console.log(data);
};
```

Background Thread
a.js

```
onmessage = ({data})=>{
    console.log(data);
    postMessage("world")
};
```

URL & Blob

```
onmessage = ({data})=>{
    console.log(data);
    postMessage("world")
};
```

```
`onmessage = ({data})=>{
    console.log(data);
    postMessage("world")
};`
```

```
new Blob([
`onmessage = ({data})=>{
    console.log(data);
    postMessage("world")
};`
], {type:'text/javascript'})
```

```
URL.createObjectURL(
new Blob([
`onmessage = ({data})=>{
    console.log(data);
    postMessage("world")
};`
], {type:'text/javascript'})
)
```

```
URL.createObjectURL(
new Blob([
`onmessage = ({data})=>{
    console.log(data);
    postMessage("world")
};`
], {type:'text/javascript'})
)
```

blob:http://localhost:63342/757c5194-0897-4166-817d-ddd4fa684e01

```
const worker = new Worker(
URL.createObjectURL(
new Blob([
`onmessage = ({data})=>{
    console.log(data);
    postMessage("world")
};`
], {type:'text/javascript'})
)
);
```

```
const worker = new Worker(
URL.createObjectURL(
new Blob([
`onmessage = ({data})=>{
    console.log(data);
    postMessage("world")
};`
], {type:'text/javascript'})
)
);
```

```
const worker = new Worker("a.js");
```

workerPromise

```
const mine = {js:{type:'text/javascript'}};
```

```
const mine = {js:{type:'text/javascript'}};

const WorkerPromise =f=>{
  let resolve, reject;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>resolve(e.data), onerror:e=>reject(e.data)});
  return data=>new Promise((res, rej)=>{
    resolve = res;
    reject = rej;
    worker.postMessage(data);
  });
};
```

```
const mine = {js:{type:'text/javascript'}};

const WorkerPromise =f=>{
  let resolve, reject;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>resolve(e.data), onerror:e=>reject(e.data)});
  return data=>new Promise((res, rej)=>{
    resolve = res;
    reject = rej;
    worker.postMessage(data);
  });
};
```

```
const mine = {js:{type:'text/javascript'}};

const WorkerPromise =f=>{
  let resolve, reject;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>resolve(e.data), onerror:e=>reject(e.data)});
  return data=>new Promise((res, rej)=>{
    resolve = res;
    reject = rej;
    worker.postMessage(data);
  });
};
```

```
const mine = {js:{type:'text/javascript'}};

const WorkerPromise =f=>{
  let resolve, reject;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>resolve(e.data), onerror:e=>reject(e.data)});
  return data=>new Promise((res, rej)=>{
    resolve = res;
    reject = rej;
    worker.postMessage(data);
  });
};
```

```
const mine = {js:{type:'text/javascript'}};

const WorkerPromise =f=>{
  let resolve, reject;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>resolve(e.data), onerror:e=>reject(e.data)});
  return data=>new Promise((res, rej)=>{
    resolve = res;
    reject = rej;
    worker.postMessage(data);
  });
};
```

```
const mine = {js:{type:'text/javascript'}};

const WorkerPromise =f=>{
  let resolve, reject;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>resolve(e.data), onerror:e=>reject(e.data)});
  return data=>new Promise((res, rej)=>{
    resolve = res;
    reject = rej;
    worker.postMessage(data);
  });
};
```

```
const mine = {js:{type:'text/javascript'}};

const WorkerPromise =f=>{
  let resolve, reject;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>resolve(e.data), onerror:e=>reject(e.data)});
  return data=>new Promise((res, rej)=>{
    resolve = res;
    reject = rej;
    worker.postMessage(data);
  });
};
```

```
const mine = {js:{type:'text/javascript'}};

const WorkerPromise =f=>{
  let resolve, reject;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>resolve(e.data), onerror:e=>reject(e.data)});
  return data=>new Promise((res, rej)=>{
    resolve = res;
    reject = rej;
    worker.postMessage(data);
  });
};
```

```
const mine = {js:{type:'text/javascript'}};

const WorkerPromise =f=>{
  let resolve, reject;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>resolve(e.data), onerror:e=>reject(e.data)});
  return data=>new Promise((res, rej)=>{
    resolve = res;
    reject = rej;
    worker.postMessage(data);
  });
};
```

```
const mine = {js:{type:'text/javascript'}};

const WorkerPromise =f=>{
  let resolve, reject;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>resolve(e.data), onerror:e=>reject(e.data)});
  return data=>new Promise((res, rej)=>{
    resolve = res;
    reject = rej;
    worker.postMessage(data);
  });
};
```

```
const mine = {js:{type:'text/javascript'}};

const WorkerPromise =f=>{
  let resolve, reject;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>resolve(e.data), onerror:e=>reject(e.data)});
  return data=>new Promise((res, rej)=>{
    resolve = res;
    reject = rej;
    worker.postMessage(data);
  });
};
```

```
const mine = {js:{type:'text/javascript'}};

const WorkerPromise =f=>{
  let resolve, reject;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>resolve(e.data), onerror:e=>reject(e.data)});
  return data=>new Promise((res, rej)=>{
    resolve = res;
    reject = rej;
    worker.postMessage(data);
  });
};

const addWorld = WorkerPromise(str=>str + " world");
addWorld("hello").then(console.log);
```

greyscale

```
const greyscale = WorkerPromise(imgData=>{
    for(let i = 0; i < imgData.length; i += 4){
        const v = .34 * imgData[i] + .5 * imgData[i + 1] + .16 * imgData[i + 2];
        imgData[i] = imgData[i + 1] = imgData[i + 2] = v;
    }
    return imgData;
});
```

```
const greyscale = WorkerPromise(imgData=>{
    for(let i = 0; i < imgData.length; i += 4){
        const v = .34 * imgData[i] + .5 * imgData[i + 1] + .16 * imgData[i + 2];
        imgData[i] = imgData[i + 1] = imgData[i + 2] = v;
    }
    return imgData;
});
```

```
const greyscale = WorkerPromise(imgData=>{
    for(let i = 0; i < imgData.length; i += 4){
        const v = .34 * imgData[i] + .5 * imgData[i + 1] + .16 * imgData[i + 2];
        imgData[i] = imgData[i + 1] = imgData[i + 2] = v;
    }
    return imgData;
});
```

```
const greyscale = WorkerPromise(imgData=>{
    for(let i = 0; i < imgData.length; i += 4){
        const v = .34 * imgData[i] + .5 * imgData[i + 1] + .16 * imgData[i + 2];
        imgData[i] = imgData[i + 1] = imgData[i + 2] = v;
    }
    return imgData;
});
```

```
const greyscale = WorkerPromise(imgData=>{
    for(let i = 0; i < imgData.length; i += 4){
        const v = .34 * imgData[i] + .5 * imgData[i + 1] + .16 * imgData[i + 2];
        imgData[i] = imgData[i + 1] = imgData[i + 2] = v;
    }
    return imgData;
});

img.onload = ({target})=>{
  const {width, height} = target;
  const ctx = Object.assign(canvas, {width, height}).getContext("2d");
  ctx.drawImage(target, 0, 0);
  const imgData = ctx.getImageData(0, 0, width, height).data;
  greyscale(imgData).then(v=>ctx.putImageData(new ImageData(v, width, height), 0, 0));
}
```

```
const greyscale = WorkerPromise(imgData=>{
    for(let i = 0; i < imgData.length; i += 4){
        const v = .34 * imgData[i] + .5 * imgData[i + 1] + .16 * imgData[i + 2];
        imgData[i] = imgData[i + 1] = imgData[i + 2] = v;
    }
    return imgData;
});

img.onload = ({target})=>{
  const {width, height} = target;
  const ctx = Object.assign(canvas, {width, height}).getContext("2d");
  ctx.drawImage(target, 0, 0);
  const imgData = ctx.getImageData(0, 0, width, height).data;
  greyscale(imgData).then(v=>ctx.putImageData(new ImageData(v, width, height), 0, 0));
}
```

```
const greyscale = WorkerPromise(imgData=>{
    for(let i = 0; i < imgData.length; i += 4){
        const v = .34 * imgData[i] + .5 * imgData[i + 1] + .16 * imgData[i + 2];
        imgData[i] = imgData[i + 1] = imgData[i + 2] = v;
    }
    return imgData;
});

img.onload = ({target})=>{
  const {width, height} = target;
  const ctx = Object.assign(canvas, {width, height}).getContext("2d");
  ctx.drawImage(target, 0, 0);
  const imgData = ctx.getImageData(0, 0, width, height).data;
  greyscale(imgData).then(v=>ctx.putImageData(new ImageData(v, width, height), 0, 0));
}
```

```
const greyscale = WorkerPromise(imgData=>{
    for(let i = 0; i < imgData.length; i += 4){
        const v = .34 * imgData[i] + .5 * imgData[i + 1] + .16 * imgData[i + 2];
        imgData[i] = imgData[i + 1] = imgData[i + 2] = v;
    }
    return imgData;
});

img.onload = ({target})=>{
  const {width, height} = target;
  const ctx = Object.assign(canvas, {width, height}).getContext("2d");
  ctx.drawImage(target, 0, 0);
  const imgData = ctx.getImageData(0, 0, width, height).data;
  greyscale(imgData).then(v=>ctx.putImageData(new ImageData(v, width, height), 0, 0));
}
```

```
const greyscale = WorkerPromise(imgData=>{
    for(let i = 0; i < imgData.length; i += 4){
        const v = .34 * imgData[i] + .5 * imgData[i + 1] + .16 * imgData[i + 2];
        imgData[i] = imgData[i + 1] = imgData[i + 2] = v;
    }
    return imgData;
});
```

```
img.onload = ({target})=>{
  const {width, height} = target;
  const ctx = Object.assign(canvas, {width, height}).getContext("2d");
  ctx.drawImage(target, 0, 0);
  const imgData = ctx.getImageData(0, 0, width, height).data;
  greyscale(imgData).then(v=>ctx.putImageData(new ImageData(v, width, height), 0, 0));
}
```

```
const greyscale = WorkerPromise(imgData=>{
    for(let i = 0; i < imgData.length; i += 4){
        const v = .34 * imgData[i] + .5 * imgData[i + 1] + .16 * imgData[i + 2];
        imgData[i] = imgData[i + 1] = imgData[i + 2] = v;
    }
    return imgData;
});
```

```
img.onload = ({target})=>{
  const {width, height} = target;
  const ctx = Object.assign(canvas, {width, height}).getContext("2d");
  ctx.drawImage(target, 0, 0);
  const imgData = ctx.getImageData(0, 0, width, height).data;
  greyscale(imgData).then(v=>ctx.putImageData(new ImageData(v, width, height), 0, 0));
}
```

```
const greyscale = WorkerPromise(imgData=>{
    for(let i = 0; i < imgData.length; i += 4){
        const v = .34 * imgData[i] + .5 * imgData[i + 1] + .16 * imgData[i + 2];
        imgData[i] = imgData[i + 1] = imgData[i + 2] = v;
    }
    return imgData;
});
```

```
img.onload = ({target})=>{
  const {width, height} = target;
  const ctx = Object.assign(canvas, {width, height}).getContext("2d");
  ctx.drawImage(target, 0, 0);
  const imgData = ctx.getImageData(0, 0, width, height).data;
  greyscale(imgData).then(v=>ctx.putImageData(new ImageData(v, width, height), 0, 0));
}
```

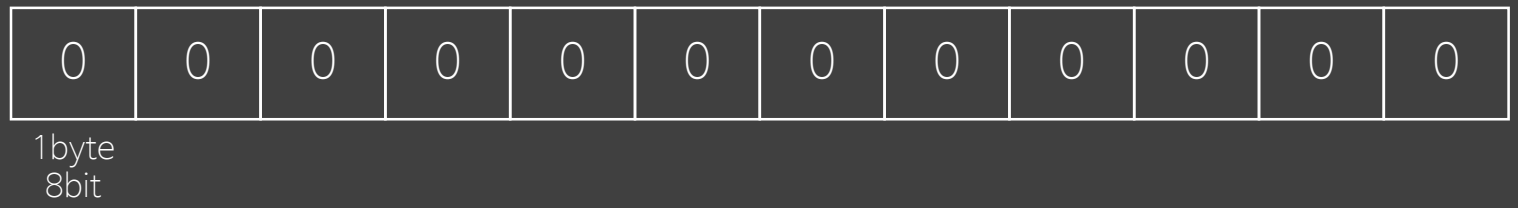
ArrayBuffer

```
new ArrayBuffer(12);
```

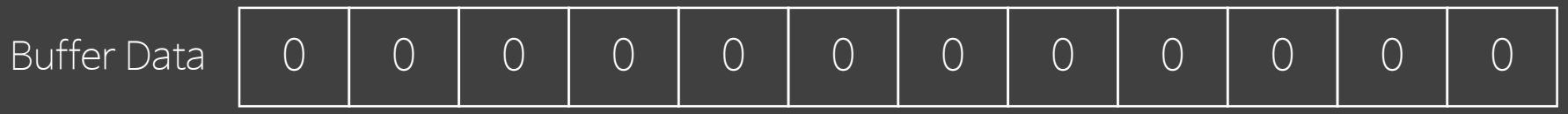
```
new ArrayBuffer(12);
```

1byte
8bit

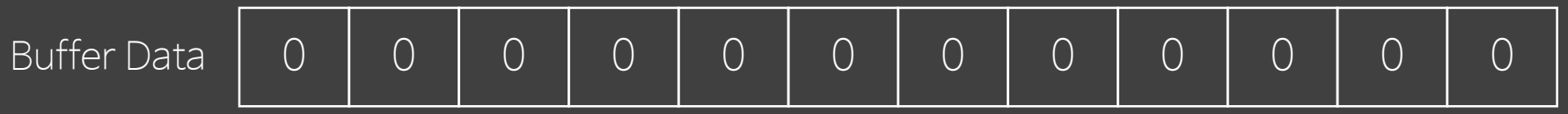
```
new ArrayBuffer(12);
```



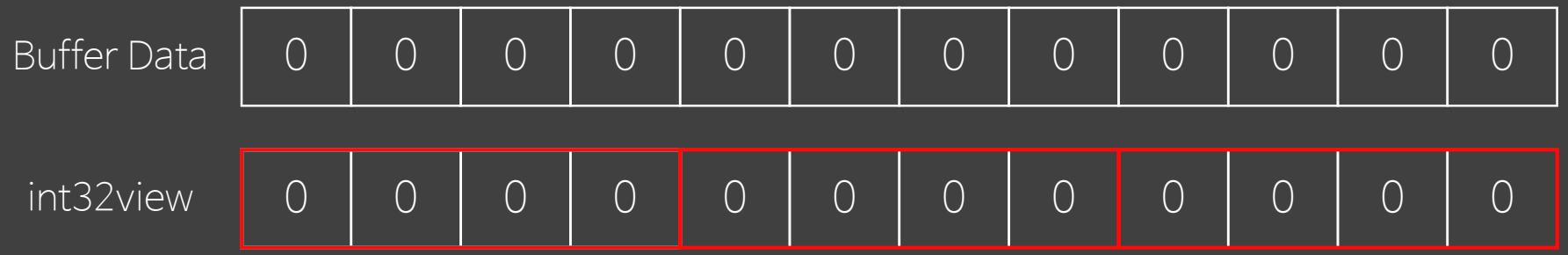
```
new ArrayBuffer(12);
```



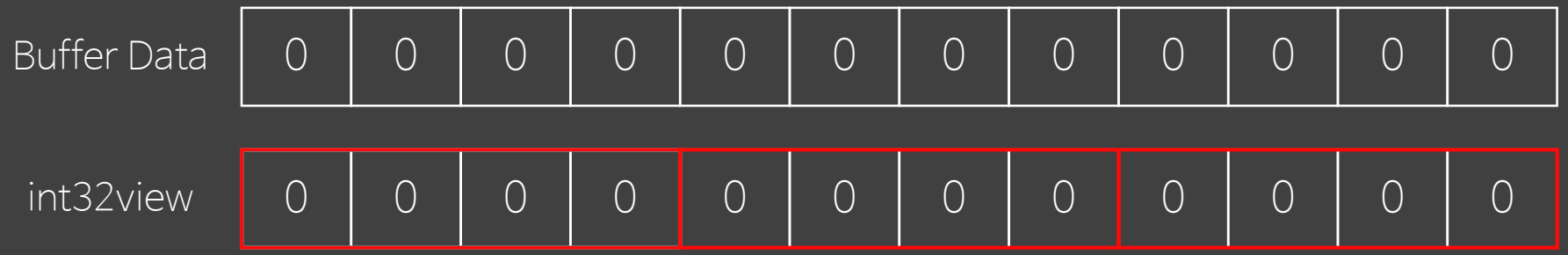
```
const intView = new Int32Array(new ArrayBuffer(12));
```



```
const intView = new Int32Array(new ArrayBuffer(12));
```

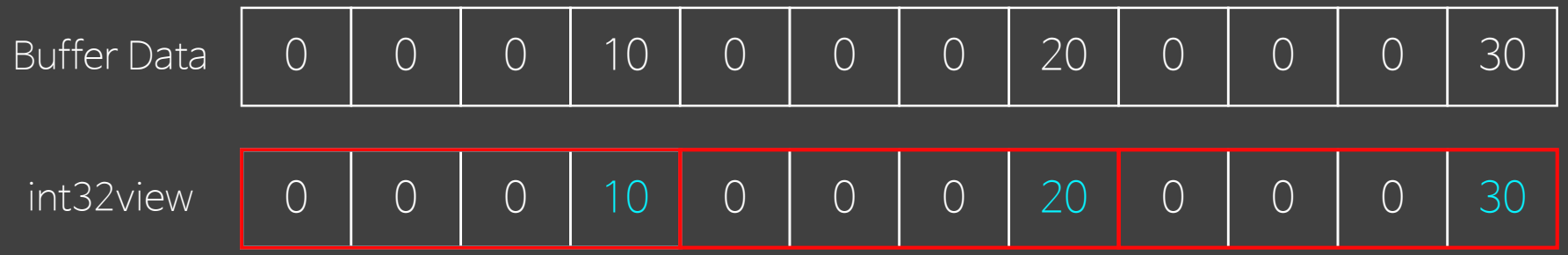


```
const intView = new Int32Array(new ArrayBuffer(12));
```

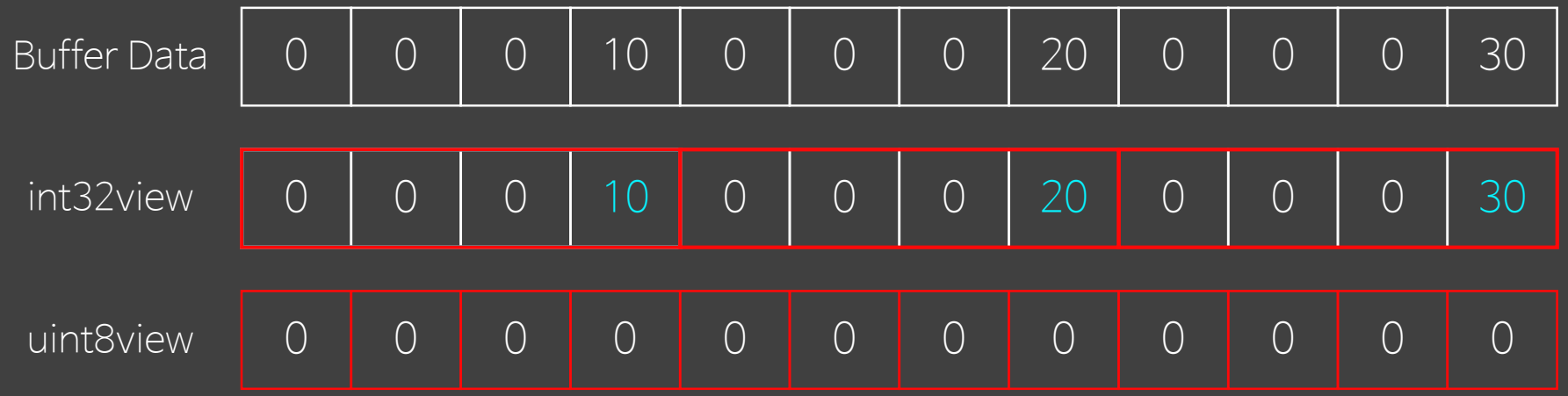



```
intView[0] = 10;
intView[1] = 20;
intView[2] = 30;
```

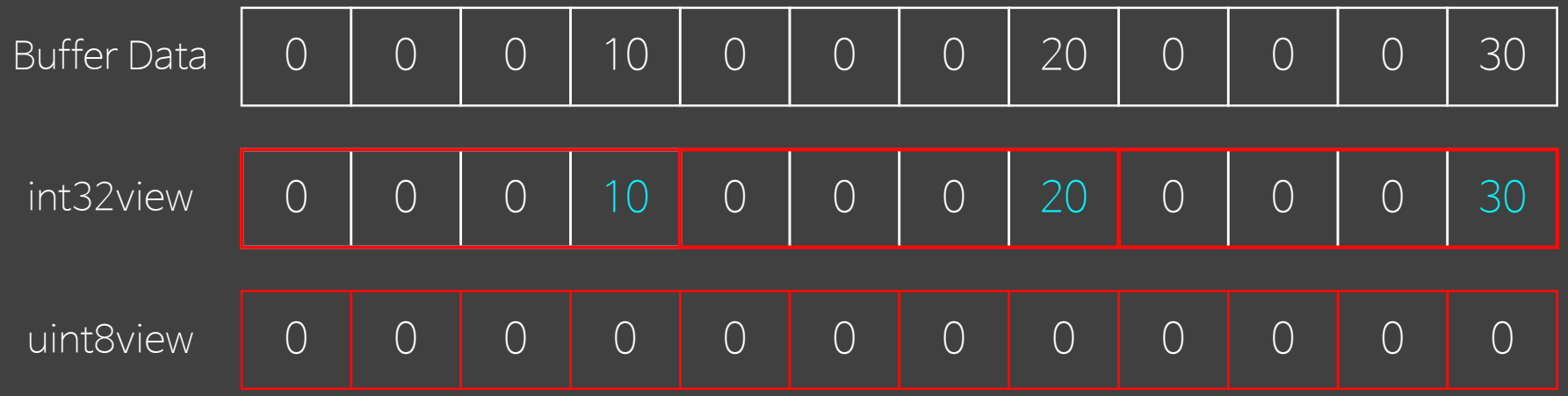
```
const intView = new Int32Array(new ArrayBuffer(12));
```



```
const utiny = new Uint8ClampedArray(new ArrayBuffer(12));
```

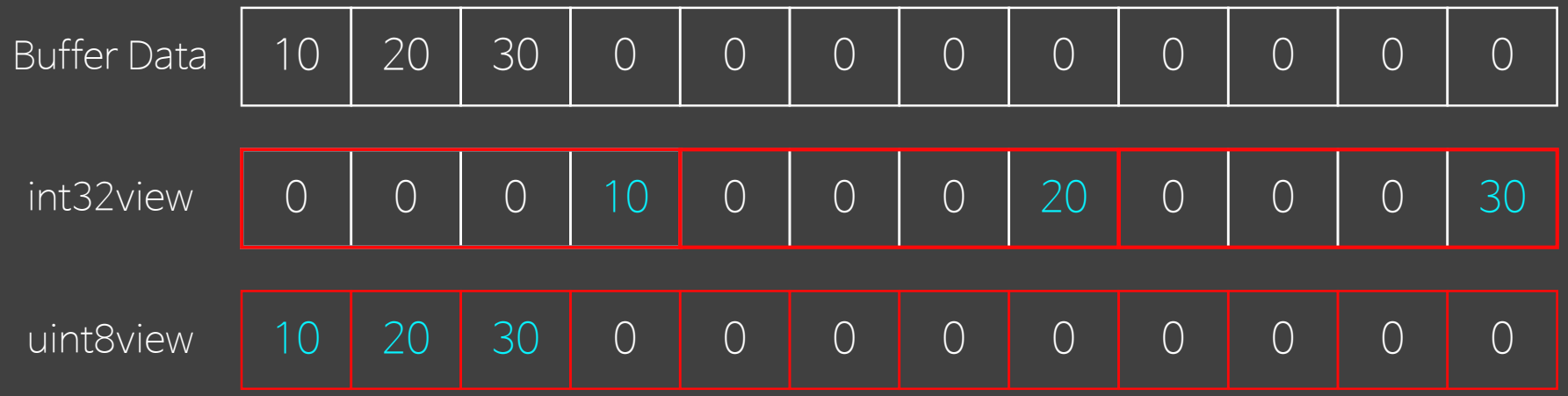


```
const utiny = new Uint8ClampedArray(new ArrayBuffer(12));
```



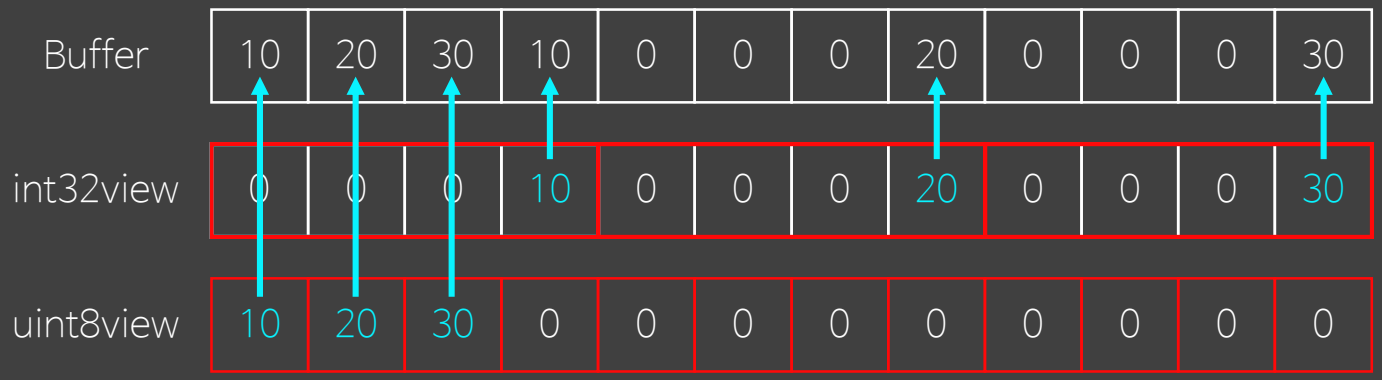

```
utiny[0] = 10;
utiny[1] = 20;
utiny[2] = 30;
```

```
const utiny = new Uint8ClampedArray(new ArrayBuffer(12));
```




```
utiny[0] = 10;
utiny[1] = 20;
utiny[2] = 30;
```

```
const buffer = new ArrayBuffer(12);
const intView = new Int32Array(buffer);
intView[0] = 10;
intView[1] = 20;
intView[2] = 30;
const utiny = new Uint8ClampedArray(buffer);
utiny[0] = 10;
utiny[1] = 20;
utiny[2] = 30;
```




```
const buffer = new ArrayBuffer(12);
const intView = new Int32Array(buffer);
intView[0] = 10;
intView[1] = 20;
intView[2] = 30;
const utiny = new Uint8ClampedArray(buffer);
utiny[0] = 10;
utiny[1] = 20;
utiny[2] = 30;
```

SharedArrayBuffer

```
img.onload = ({target})=>{
  const {width, height} = target;
  const ctx = Object.assign(canvas, {width, height}).getContext("2d");
  ctx.drawImage(target, 0, 0);
  const sObj = new SharedArrayBuffer(width * height * 4);
  const u8c = new Uint8ClampedArray(sObj);
  const imgData = ctx.getImageData(0, 0, width, height).data;
  u8c.set(imgData);
  greyscale(sObj).then(_=>{
    const r = new Uint8ClampedArray(u8c.byteLength);
    r.set(u8c);
    ctx.putImageData(new ImageData(r, width, height), 0, 0)
  });
}
```

```
img.onload = ({target})=>{
  const {width, height} = target;
  const ctx = Object.assign(canvas, {width, height}).getContext("2d");
  ctx.drawImage(target, 0, 0);
  const sObj = new SharedArrayBuffer(width * height * 4);
  const u8c = new Uint8ClampedArray(sObj);
  const imgData = ctx.getImageData(0, 0, width, height).data;
  u8c.set(imgData);
  greyscale(sObj).then(_=>{
    const r = new Uint8ClampedArray(u8c.byteLength);
    r.set(u8c);
    ctx.putImageData(new ImageData(r, width, height), 0, 0)
  });
}
```

```
img.onload = ({target})=>{
  const {width, height} = target;
  const ctx = Object.assign(canvas, {width, height}).getContext("2d");
  ctx.drawImage(target, 0, 0);
  const sObj = new SharedArrayBuffer(width * height * 4);
  const u8c = new Uint8ClampedArray(sObj);
  const imgData = ctx.getImageData(0, 0, width, height).data;
  u8c.set(imgData);
  greyscale(sObj).then(_=>{
    const r = new Uint8ClampedArray(u8c.byteLength);
    r.set(u8c);
    ctx.putImageData(new ImageData(r, width, height), 0, 0)
  });
}
```

```
img.onload = ({target})=>{
  const {width, height} = target;
  const ctx = Object.assign(canvas, {
  ctx.drawImage(target, 0, 0);
  const sObj = new SharedArrayBuffer(
  const u8c = new Uint8ClampedArray(s
  const imgData = ctx.getImageData(0,
  u8c.set(imgData);
  greyscale(sObj).then(_=>{
    const r = new Uint8ClampedArray(u8c.byteLength);
    r.set(u8c);
    ctx.putImageData(new ImageData(r, width, height), 0, 0)
  });
}
```

```
const greyscale = WorkerPromise(imgData=>{
    for(let i = 0; i < imgData.length; i += 4){
        const v = .34 * imgData[i] + .5 * imgData[i + 1] + .16 * imgData[i + 2];
        imgData[i] = imgData[i + 1] = imgData[i + 2] = v;
    }
    return imgData;
});
```

```
img.onload = ({target})=>{
  const {width, height} = target;
  const ctx = Object.assign(canvas, {width, height}).ge
  ctx.drawImage(target, 0, 0);
  const sObj = new SharedArrayBuffer(width * height * 4
  const u8c = new Uint8ClampedArray(sObj);
  const imgData = ctx.getImageData(0, 0, width, height)
  u8c.set(imgData);
  greyscale(sObj).then(_=>{
    const r = new Uint8ClampedArray(u8c.byteLength);
    r.set(u8c);
    ctx.putImageData(new ImageData(r, width, height), 0, 0)
  });
}
```

```
const greyscale = WorkerPromise(sObj=>{
    const v = new Uint8ClampedArray(sObj);
    for(let i = 0; i < v.byteLength; i += 4){
        const j = .34 * v[i] + .5 * v[i + 1] + .16 * v[i + 2];
        v[i] = v[i + 1] = v[i + 2] = j;
    }
    return sObj;
});
```

```
const brightness = WorkerPromise(({rate, sObj})=>{
    const v = new Uint8ClampedArray(sObj);
    for(let i = 0; i < v.byteLength; i += 4){
        v[i] = v[i] * (1 + rate);
        v[i + 1] = v[i + 1] * (1 + rate);
        v[i + 2] = v[i + 2] * (1 + rate);
    }
    return sObj;
});
```

```
const greyscale = WorkerPromise(sObj=>{
    const v = new Uint8ClampedArray(sObj);
    for(let i = 0; i < v.byteLength; i += 4){
        const j = .34 * v[i] + .5 * v[i + 1] + .16 * v[i + 2];
        v[i] = v[i + 1] = v[i + 2] = j;
    }
    return sObj;
});
```

```
  const copy =_=>{
    const r = new Uint8ClampedArray(u8c.byteLength);
    r.set(u8c);
    ctx.putImageData(new ImageData(r, width, height), 0, 0);
  };
  greyscale(sObj).then(copy);
}
```

```
const brightness = WorkerPromise(({rate, sObj})=>{
    const v = new Uint8ClampedArray(sObj);
    for(let i = 0; i < v.byteLength; i += 4){
        v[i] = v[i] * (1 + rate);
        v[i + 1] = v[i + 1] * (1 + rate);
        v[i + 2] = v[i + 2] * (1 + rate);
    }
    return sObj;
});
```

```
const greyscale = WorkerPromise(sObj=>{
    const v = new Uint8ClampedArray(sObj);
    for(let i = 0; i < v.byteLength; i += 4){
        const j = .34 * v[i] + .5 * v[i + 1] + .16 * v[i + 2];
        v[i] = v[i + 1] = v[i + 2] = j;
    }
    return sObj;
});
```

```
  const copy =_=>{
    const r = new Uint8ClampedArray(u8c.byteLength);
    r.set(u8c);
    ctx.putImageData(new ImageData(r, width, height), 0, 0);
  };
  greyscale(sObj).then(copy);
  brightness({rate:-.1, sObj}).then(copy);
}
```

```
const brightness = WorkerPromise(({rate, sObj})=>{
    const v = new Uint8ClampedArray(sObj);
    for(let i = 0; i < v.byteLength; i += 4){
        v[i] = v[i] * (1 + rate);
        v[i + 1] = v[i + 1] * (1 + rate);
        v[i + 2] = v[i + 2] * (1 + rate);
    }
    return sObj;
});
```

```
const greyscale = WorkerPromise(sObj=>{
    const v = new Uint8ClampedArray(sObj);
    for(let i = 0; i < v.byteLength; i += 4){
        const j = .34 * v[i] + .5 * v[i + 1] + .16 * v[i + 2];
        v[i] = v[i + 1] = v[i + 2] = j;
    }
    return sObj;
});
```

```
  const copy =_=>{
    const r = new Uint8ClampedArray(u8c.byteLength);
    r.set(u8c);
    ctx.putImageData(new ImageData(r, width, height), 0, 0);
  };
  greyscale(sObj).then(copy);
  brightness({rate:-.1, sObj}).then(copy);
}
```

```
greyscale(sObj).then(_=>{
    copy();
    return brightness({rate:-.1, sObj})
}).then(copy);
```

Schedule Queue

```
const WorkerPromise =f=>{
  let resolve, reject;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>resolve(e.data), onerror:e=>reject(e.data)});
  return data=>new Promise((res, rej)=>{
    resolve = res;
    reject = rej;
    worker.postMessage(data);
  });
};
```

```
const WorkerPromise =f=>{
  let resolve, reject, start, end;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>resolve(e.data), onerror:e=>reject(e.data)});
  return data=>new Promise((res, rej)=>{
    resolve = res;
    reject = rej;
    worker.postMessage(data);
  });
};
```

```
const WorkerPromise =f=>{
  let resolve, reject, start, end;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>resolve(e.data), onerror:e=>reject(e.data)});
  return data=>new Promise((res, rej)=>{
    const v = {data, resolve:res, reject:rej};
    if(end) end = end.next = v;
    else{
      start = end = v;
      resolve = res;
      reject = rej;
      worker.postMessage(data);
    }
  });
};
```

```
const WorkerPromise =f=>{
  let resolve, reject, start, end;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>resolve(e.data), onerror:e=>reject(e.data)});
  return data=>new Promise((res, rej)=>{
    const v = {data, resolve:res, reject:rej};
    if(end) end = end.next = v;
    else{
      start = end = v;
      resolve = res;
      reject = rej;
      worker.postMessage(data);
    }
  });
};
```

```
const WorkerPromise =f=>{
  let resolve, reject, start, end;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>resolve(e.data), onerror:e=>reject(e.data)});
  return data=>new Promise((res, rej)=>{
    const v = {data, resolve:res, reject:rej};
    if(end) end = end.next = v;
    else{
      start = end = v;
      resolve = res;
      reject = rej;
      worker.postMessage(data);
    }
  });
};
```

```
const WorkerPromise =f=>{
  let resolve, reject, start, end;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>(resolve(e.data), next()), onerror:e=>(reject(e.data), next())});
  return data=>new Promise((res, rej)=>{
    const v = {data, resolve:res, reject:rej};
    if(end) end = end.next = v;
    else{
      start = end = v;
      resolve = res;
      reject = rej;
      worker.postMessage(data);
    }
  });
};
```

```
const WorkerPromise =f=>{
  let resolve, reject, start, end;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>(resolve(e.data), next()), onerror:e=>(reject(e.data), next())});
  const next =_=>{
    if(!start.next) return;
    ({data, resolve, reject} = start.next);
    start = start.next;
    worker.postMessage(data);
  };
  return data=>new Promise((res, rej)=>{
    const v = {data, resolve:res, reject:rej};
    if(end) end = end.next = v;
    else{
      start = end = v;
      resolve = res;
      reject = rej;
      worker.postMessage(data);
    }
  });
};
```

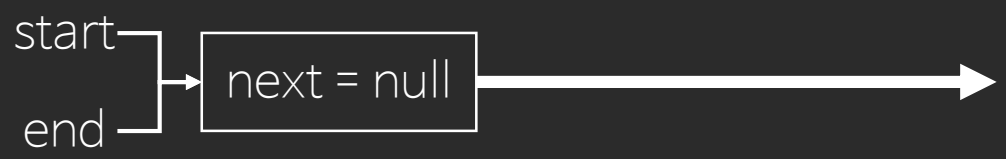
```
const WorkerPromise =f=>{
  let resolve, reject, start, end;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>(resolve(e.data), next()), onerror:e=>(reject(e.data), next())});
  const next =_=>{
    if(!start.next) return;
    ({data, resolve, reject} = start.next);
    start = start.next;
    worker.postMessage(data);
  };
  return data=>new Promise((res, rej)=>{
    const v = {data, resolve:res, reject:rej};
    if(end) end = end.next = v;
    else{
      start = end = v;
      resolve = res;
      reject = rej;
      worker.postMessage(data);
    }
  });
};
```

```
const WorkerPromise =f=>{
  let resolve, reject, start, end;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>(resolve(e.data), next()), onerror:e=>(reject(e.data), next())});
  const next =_=>{
    if(!start.next) return;
    ({data, resolve, reject} = start.next);
    start = start.next;
    worker.postMessage(data);
  };
  return data=>new Promise((res, rej)=>{
    const v = {data, resolve:res, reject:rej};
    if(end) end = end.next = v;
    else{
      start = end = v;
      resolve = res;
      reject = rej;
      worker.postMessage(data);
    }
  });
};
```

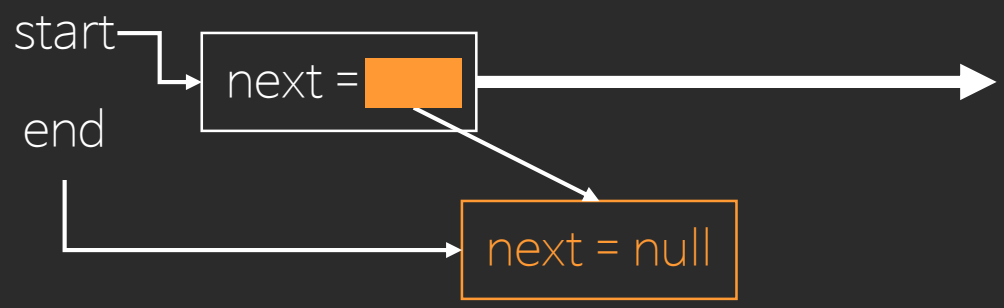
```
const WorkerPromise =f=>{
  let resolve, reject, start, end;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>(resolve(e.data), next()), onerror:e=>(reject(e.data), next())});
  const next =_=>{
    if(!start.next) return;
    ({data, resolve, reject} = start.next);
    start = start.next;
    worker.postMessage(data);
  };
  return data=>new Promise((res, rej)=>{
    const v = {data, resolve:res, reject:rej};
    if(end) end = end.next = v;
    else{
      start = end = v;
      resolve = res;
      reject = rej;
      worker.postMessage(data);
    }
  });
};
```

```
const WorkerPromise =f=>{
  let resolve, reject, start, end;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>(resolve(e.data), next()), onerror:e=>(reject(e.data), next())});
  const next =_=>{
    if(!start.next) return;
    ({data, resolve, reject} = start.next);
    start = start.next;
    worker.postMessage(data);
  };
  return data=>new Promise((res, rej)=>{
    const v = {data, resolve:res, reject:rej};
    if(end) end = end.next = v;
    else{
      start = end = v;
      resolve = res;
      reject = rej;
      worker.postMessage(data);
    }
  });
};
```

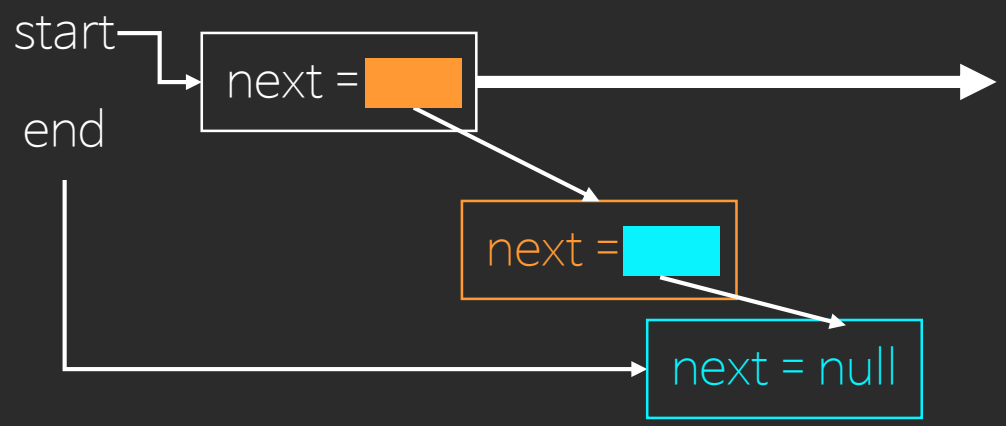
```
const WorkerPromise =f=>{
  let resolve, reject, start, end;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>(resolve(e.data), next()), onerror:e=>(reject(e.data), next())});
  const next =_=>{
    if(!start.next) return;
    ({data, resolve, reject} = start.next);
    start = start.next;
    worker.postMessage(data);
  };
  return data=>new Promise((res, rej)=>{
    const v = {data, resolve:res, reject:rej};
    if(end) end = end.next = v;
    else{
      start = end = v;
      resolve = res;
      reject = rej;
      worker.postMessage(data);
    }
  });
};
```



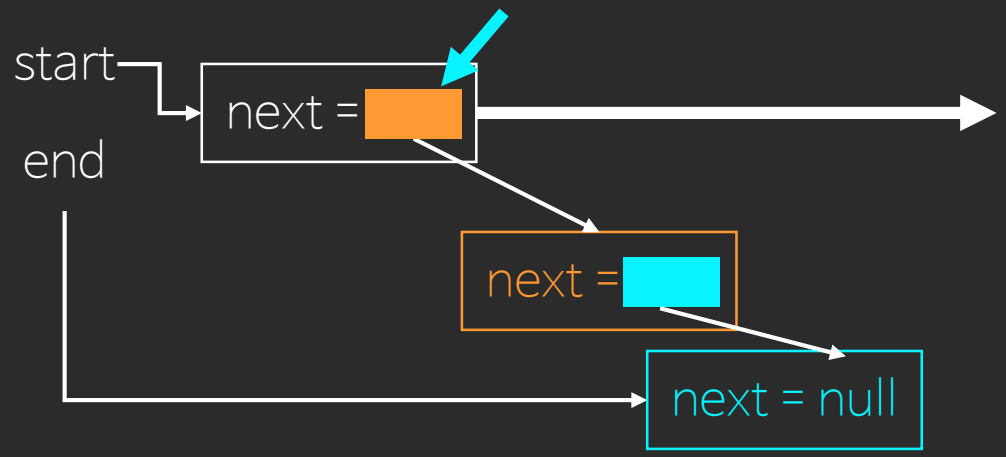
```
const WorkerPromise =f=>{
  let resolve, reject, start, end;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>(resolve(e.data), next()), onerror:e=>(reject(e.data), next())});
  const next =_=>{
    if(!start.next) return;
    ({data, resolve, reject} = start.next);
    start = start.next;
    worker.postMessage(data);
  };
  return data=>new Promise((res, rej)=>{
    const v = {data, resolve:res, reject:rej};
    if(end) end = end.next = v;
    else{
      start = end = v;
      resolve = res;
      reject = rej;
      worker.postMessage(data);
    }
  });
};
```



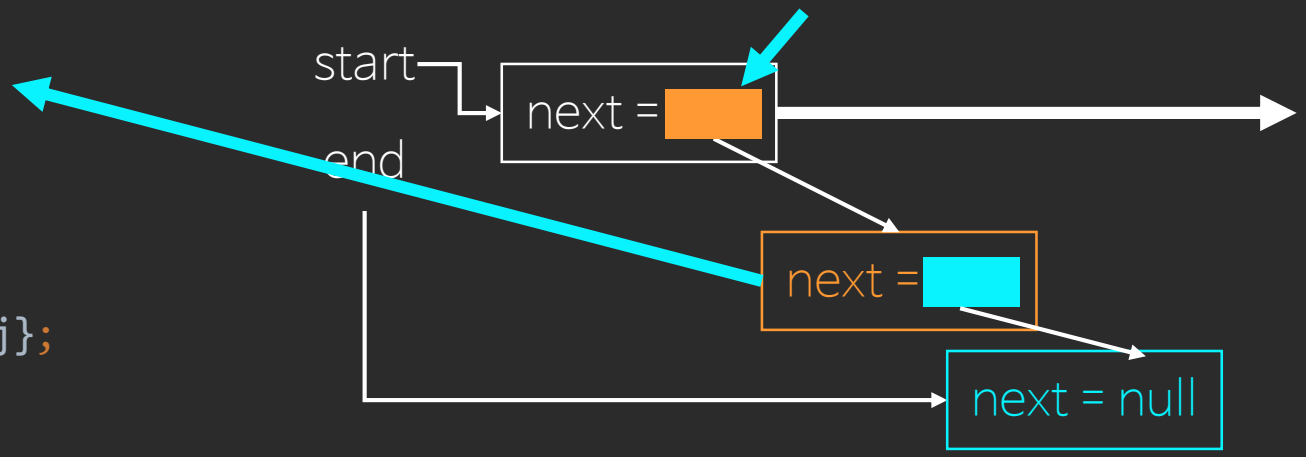
```
const WorkerPromise =f=>{
  let resolve, reject, start, end;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>(resolve(e.data), next()), onerror:e=>(reject(e.data), next())});
  const next =_=>{
    if(!start.next) return;
    ({data, resolve, reject} = start.next);
    start = start.next;
    worker.postMessage(data);
  };
  return data=>new Promise((res, rej)=>{
    const v = {data, resolve:res, reject:rej};
    if(end) end = end.next = v;
    else{
      start = end = v;
      resolve = res;
      reject = rej;
      worker.postMessage(data);
    }
  });
};
```



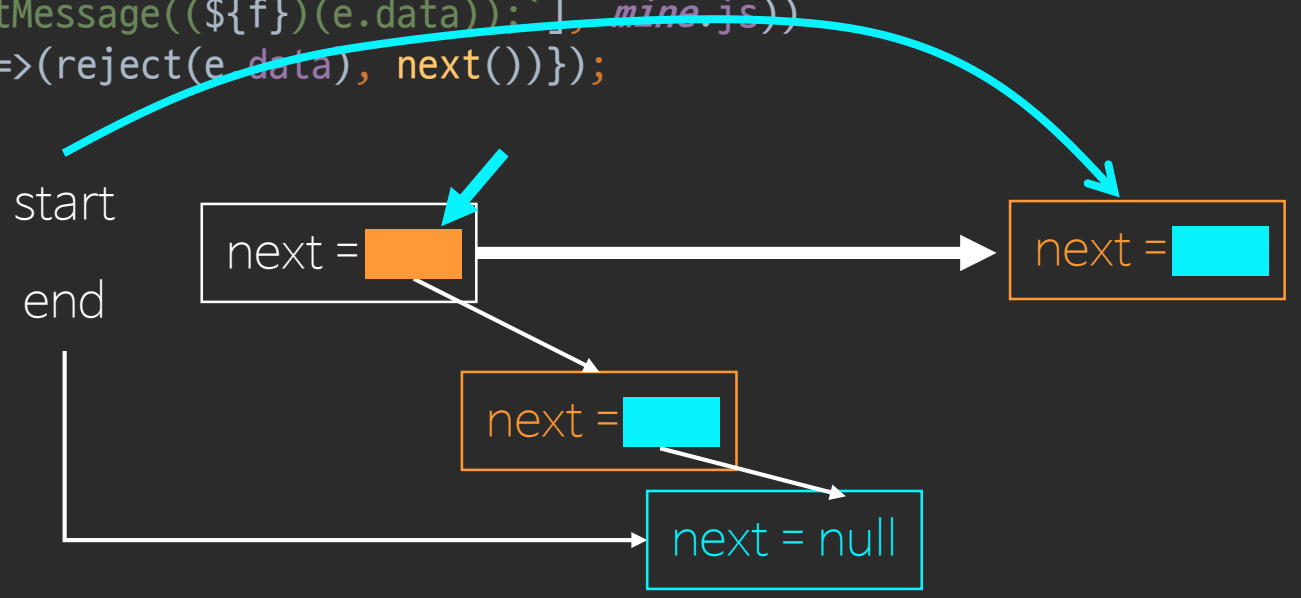
```
const WorkerPromise =f=>{
  let resolve, reject, start, end;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>(resolve(e.data), next()), onerror:e=>(reject(e.data), next())});
  const next =_=>{
    if(!start.next) return;
    ({data, resolve, reject} = start.next);
    start = start.next;
    worker.postMessage(data);
  };
  return data=>new Promise((res, rej)=>{
    const v = {data, resolve:res, reject:rej};
    if(end) end = end.next = v;
    else{
      start = end = v;
      resolve = res;
      reject = rej;
      worker.postMessage(data);
    }
  });
};
```



```
const WorkerPromise =f=>{
  let resolve, reject, start, end;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>(resolve(e.data), next()), onerror:e=>(reject(e.data), next())});
  const next =_=>{
    if(!start.next) return;
    ({data, resolve, reject} = start.next);
    start = start.next;
    worker.postMessage(data);
  };
  return data=>new Promise((res, rej)=>{
    const v = {data, resolve:res, reject:rej};
    if(end) end = end.next = v;
    else{
      start = end = v;
      resolve = res;
      reject = rej;
      worker.postMessage(data);
    }
  });
};
```



```
const WorkerPromise =f=>{
  let resolve, reject, start, end;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], {type:'text/javascript'}))
    ), {onmessage:e=>(resolve(e.data), next()), onerror:e=>(reject(e.data), next())});
  const next =_=>{
    if(!start.next) return;
    ({data, resolve, reject} = start.next);
    start = start.next;
    worker.postMessage(data);
  };
  return data=>new Promise((res, rej)=>{
    const v = {data, resolve:res, reject:rej};
    if(end) end = end.next = v;
    else{
      start = end = v;
      resolve = res;
      reject = rej;
      worker.postMessage(data);
    }
  });
};
```



```
const WorkerPromise =f=>{
  let resolve, reject, start, end;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>(resolve(e.data), next()), onerror:e=>(reject(e.data), next())});
  const next =_=>{
    if(!start.next) return;
    ({data, resolve, reject} = start.next);
    start = start.next;
    worker.postMessage(data);
  };
  return data=>new Promise((res, rej)=>{
    const v = {data, resolve:res, reject:rej};
    if(end) end = end.next = v;
    else{
      start = end = v;
      resolve = res;
      reject = rej;
      worker.postMessage(data);
    }
  });
};
```

```
let i = 10;
while(i--) brightness({rate:-.1, sObj}).then(copy);
```

```
const WorkerPromise =f=>{
  let resolve, reject, start, end;
  const worker = Object.assign(new Worker(
        URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], mine.js))
    ), {onmessage:e=>(resolve(e.data), next()), onerror:e=>(reject(e.data), next())});
  const next =_=>{
    if(!start.next) return;
    ({data, resolve, reject} = start.next);
    start = start.next;
    worker.postMessage(data);
  };
  return data=>new Promise((res, rej)=>{
    const v = {data, resolve:res, reject:rej};
    if(end) end = end.next = v;
    else{
      start = end = v;
      resolve = res;
      reject = rej;
      worker.postMessage(data);
    }
  });
};
```

```
greyscale(sObj).then(copy);
let i = 10;
while(i--) brightness({rate:-.1, sObj}).then(copy);
```

```
const SerialWorkerPromise =(()=>{
  let resolve, reject, start, end;
  const next =_=>{
    if(!start.next) return;
    ({data, resolve, reject, worker} = start.next);
    start = start.next;
    worker.postMessage(data);
  };
  return f=>{
    const worker = Object.assign(new Worker(
      URL.createObjectURL(new Blob([`onmessage = e=>postMessage((${f})(e.data));`], {type:'text/javascript'}))
    ), {onmessage:e=>(resolve(e.data), next()), onerror:e=>(reject(e.data), next())});
    return data=>new Promise((res, rej)=>{
      const v = {data, worker, resolve:res, reject:rej};
      if(end) end = end.next = v;
      else{
        start = end = v;
        resolve = res;
        reject = rej;
        worker.postMessage(data);
      }
    });
  };
})();
```

```
greyscale(sObj).then(copy);
let i = 10;
while(i--) brightness({rate:-.1, sObj}).then(copy);
```