Async / Await
Vs
Promises

JS

Async / Await

Synchronous

```
function fnWorkSecond() {
    console.log("Work Second");
}

function fnWorkFirst() {
    console.log("Work First");
}

fnWorkFirst();
fnWorkSecond();
```

Output

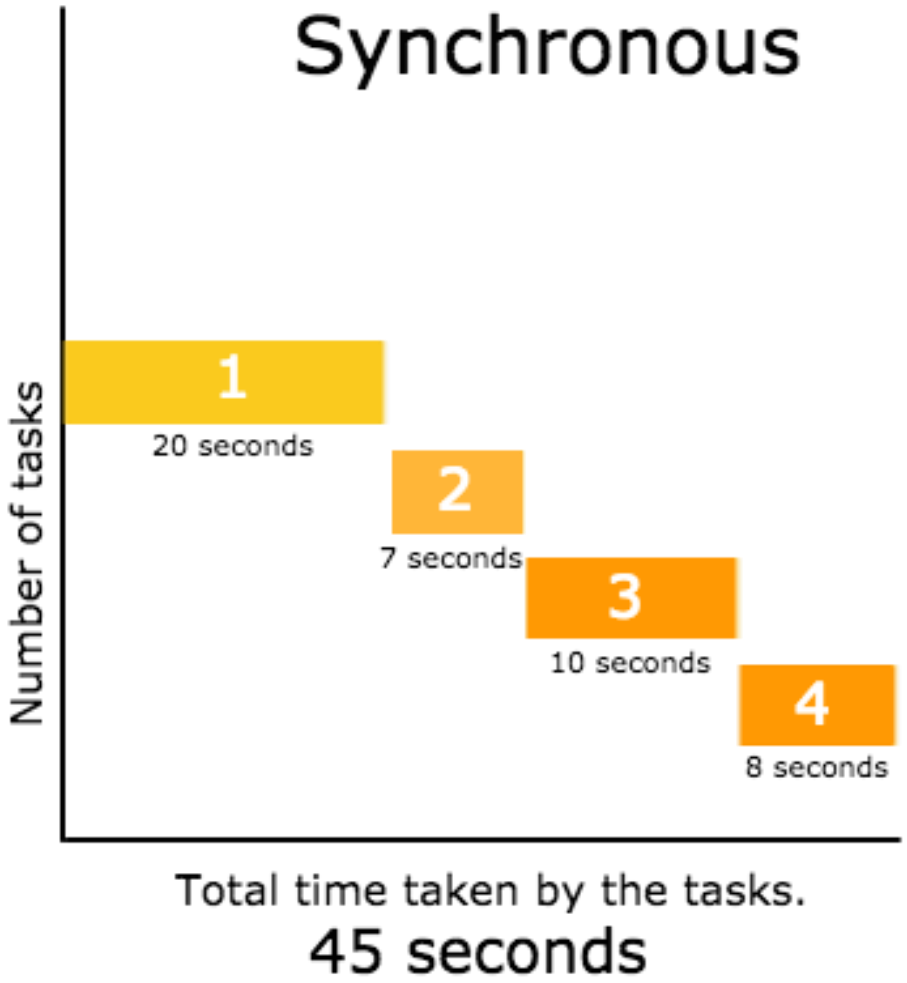
```
Work First
Work Second
```

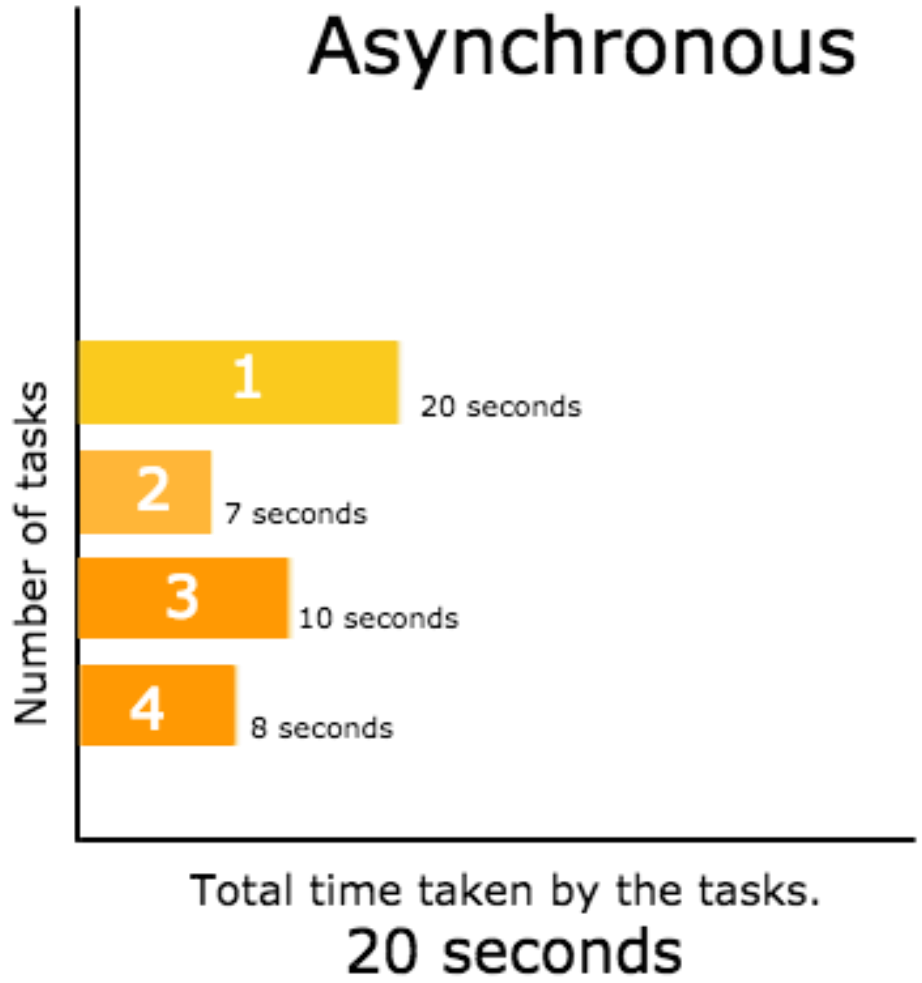
Asynchronous

```
function fnWorkSecond() {
     console.log("Work Second");
}

function fnWorkFirst() {
     console.log("Work First");
}

setTimeout(fnWorkFirst, 1000);
fnWorkSecond();
```

Output

```
Work Second
Work First
```


Synchronous
Number of tasks
1
20 seconds
2
7 seconds
3
10 seconds
4
8 seconds
Total time taken by the tasks.
45 seconds


Asynchronous
Number of tasks
1
20 seconds
2
7 seconds
3
10 seconds
4
8 seconds
Total time taken by the tasks.
20 seconds

JavaScript Callback

Callback

```
function fnWorkSecond() {
    console.log("Work Second");
}

function fnWorkFirst() {
    console.log("Work First");
    fnWorkSecond();
}

fnWorkFirst();
```

Output

```
Work First
Work Second
```

Callback(2)

```
function fnWorkSecond() {
    console.log("Work Second");
}

function fnWorkFirst(callback) {
    console.log("Work First");
    callback();
}

fnWorkFirst(fnWorkSecond);
```

Output

```
Work First
Work Second
```

Callback(3)

```
function fnWorkSecond(data_back) {
    console.log("Work Second");
    console.log(data_back);
}

function fnWorkFirst(callback) {
    console.log("Work First<");
    callback("send data back");
}

fnWorkFirst(fnWorkSecond);
```

Output

```
Work First
Work Second
send data back
```

Callback(4)

```
function fnWorkFirst(callback) {
    console.log("Work First");
    callback("send data back");
}

fnWorkFirst(function (data_back) {
    console.log("Work Second");
    console.log(data_back);
});
```

Output

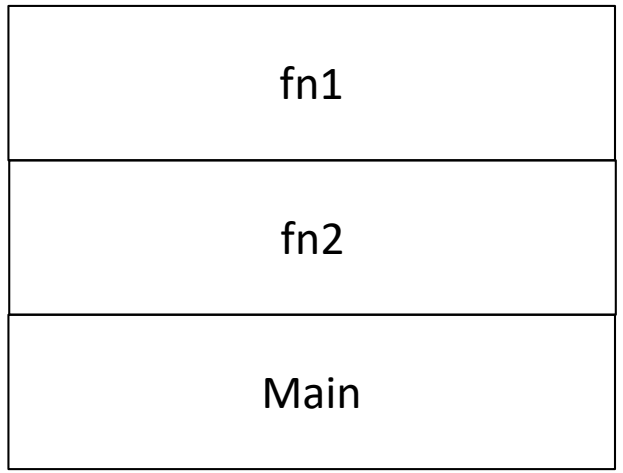
```
Work First
Work Second
send data back
```

Call Stack

Code

```
const fn1 = () => {
  console.log('fn1')
}

const fn2 = () => {
  fn1()
  console.log('fn2')
}

fn2()
```

Call Stack



Output

fn1

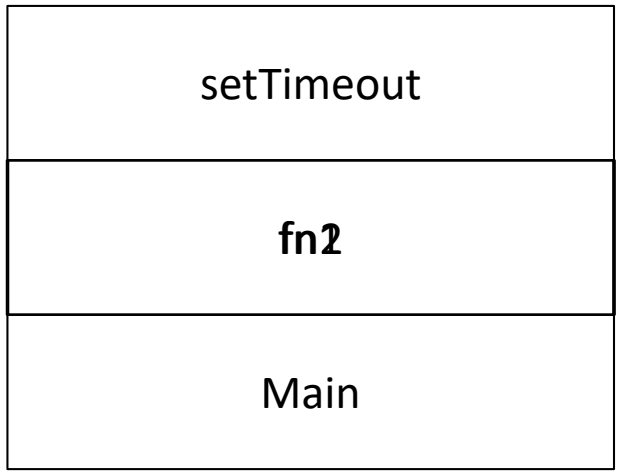
fn2

Call Stack + Callback

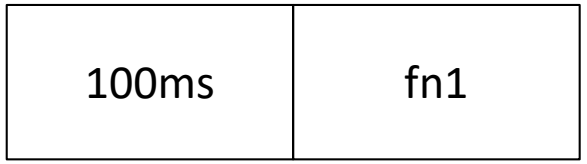
Code

```
const fn1 = () => {
    console.log('fn1')
}

const fn2 = () => {
    setTimeout(fn1, 100)
    console.log('fn2')
}

fn2()
```

Call Stack



Output



fn1

fn2

Callback Hell

```
function hell(win) {
 // for listener purpose
 return function() {
   loadLink(win, REMOTE_SRC+'/assets/css/style.css', function() {
     loadLink(win, REMOTE_SRC+'/lib/async.js', function() {
       loadLink(win, REMOTE_SRC+'/lib/easyXDM.js', function() {
         loadLink(win, REMOTE_SRC+'/lib/json2.js', function() {
           loadLink(win, REMOTE_SRC+'/lib/underscode.min.js', function() {
             loadLink(win, REMOTE_SRC+'/lib/backbone.min.js', function() {
               loadLink(win, REMOTE_SRC+'/dev/base_dev.js', function() {
                 loadLink(win, REMOTE_SRC+'/assets/js/deps.js', function() {
                   loadLink(win, REMOTE_SRC+'/src/' + win.loader_path + '/loader.js', function() {
                     async.eachSeries(SCRIPTS, function(src, callback) {
                       loadScript(win, BASE_URL+src, callback);
                     });
                   });
                 });
               });
             });
           });
         });
       });
     });
   });
 };
}
```

```
function loadNews(callback){
    //Simulate ajax load data
    setTimeout( () => {
        callback([
            {id:1, title: 'a'},
            {id:2, title: 'b'},
            {id:3, title: 'c'},
            {id:4, title: 'd'},
            {id:5, title: 'e'},
            {id:6, title: 'f'},
            {id:7, title: 'g'},
            {id:8, title: 'h'},
        ])
    } , 1000)
}

loadNews(function (result){
console.log(result)
})

console.log('Not Blocking');
```

Output

```
Not Blocking
[Object , Object ,
Object , Object ,
Object , Object ,
Object , Object]
```

```
function loadNews(callback){
    //Simulate ajax load data
    setTimeout( () => {
        callback([
        {id:1, title: 'a'},
        {id:2, title: 'b'},
        {id:3, title: 'c'},
        {id:4, title: 'd'},
        {id:5, title: 'e'},
        {id:6, title: 'f'},
        {id:7, title: 'g'},
        {id:8, title: 'h'},
        ])
    } , 1000)
}
```

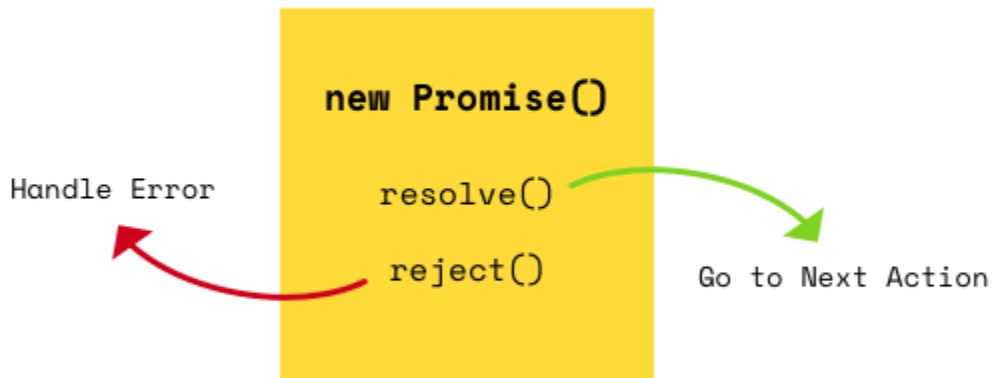
```
loadNews(function (result){
  validateNews(result, function(news){
    sortByLatest(news, function(latest){
      latest5News(latest, function(final){
        console.log('final');
        console.log(final)
      })
    })
  })
})

console.log('Not Blocking');
```

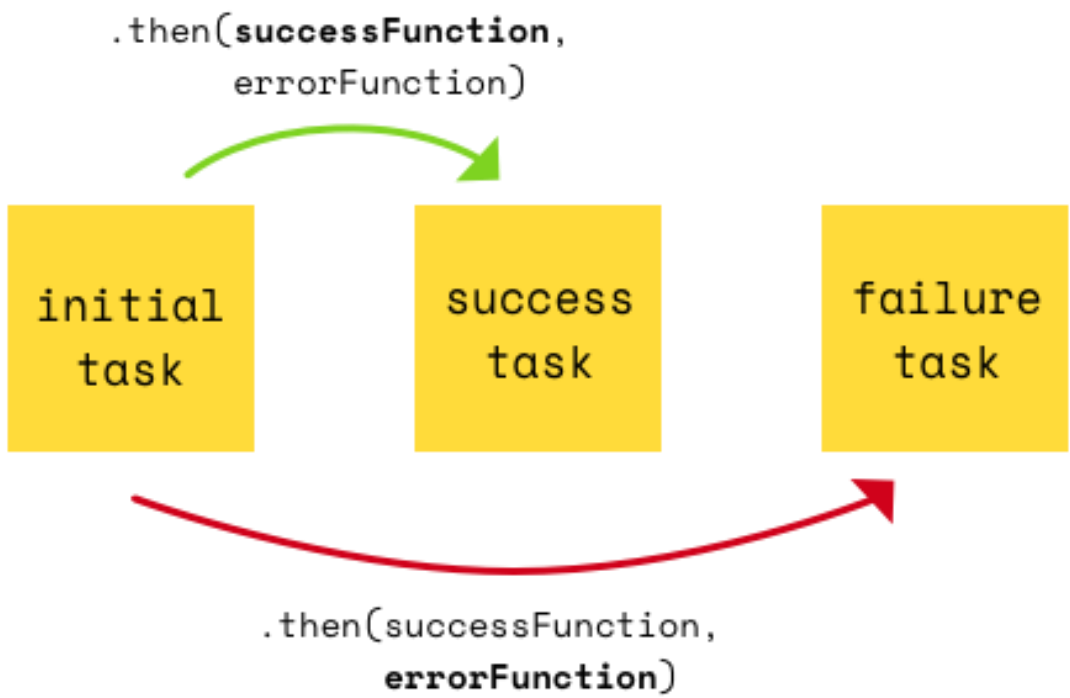
Output

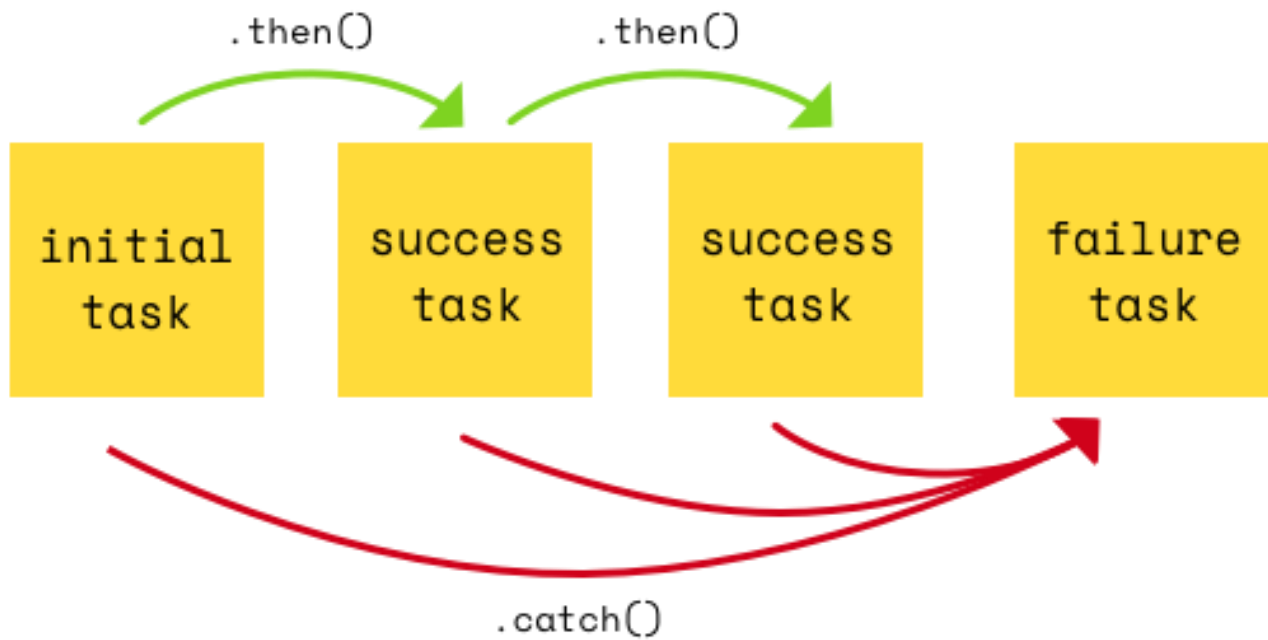
```
Not Blocking
…
```

JS

ES6
promises


new Promise()
resolve()
reject()
Handle Error
Go to Next Action


.then(successFunction,
errorFunction)
initial task
success task
failure task
.then(successFunction,
errorFunction)


.then()
.then()
initial task
success task
success task
failure task
.catch()

Promise Resolve

```
const a ='a'

const ap = new Promise( (resolve) => {
    setTimeout( () => {
    resolve('a')
    }, 1000)
})

console.log(a)
console.log(ap)

ap.then(e => {
  console.log(e)
  return 'xxx' + e
})
.then(e => {
  console.log(e)
})
```

Output

```
a
Promise{…}
a
xxxa
```

Promise Reject

```
const a ='a'
const ap = new Promise( (resolve, reject) => {
    setTimeout( () => {
    //resolve('a')
    reject('error')
    }, 1000)
})

console.log(a)
console.log(ap)

ap.then(e => {
console.log(e)
return 'xxx' + e
})
.then(e => {
console.log(e)
})
.catch(e => {
console.log(e)
})
```

Output

```
a
Promise{…}
error
```

Callback To Promise

```
function loadNews(callback){
    //Simulate ajax load data
    setTimeout( () => {
        callback([
        {id:1, title: 'a'},
        {id:2, title: 'b'},
        {id:3, title: 'c'},
        {id:4, title: 'd'},
        {id:5, title: 'e'},
        {id:6, title: 'f'},
        {id:7, title: 'g'},
        {id:8, title: 'h'},
        ])
    } , 1000)
}
```

```
function loadNewsP(){
    //Simulate ajax load data
    const news = new Promise((resolve) => {
    setTimeout( () => {
        resolve([
            {id:1, title: 'a'},
            {id:2, title: 'b'},
            {id:3, title: 'c'},
            {id:4, title: 'd'},
            {id:5, title: 'e'},
            {id:6, title: 'f'},
            {id:7, title: 'g'},
            {id:8, title: 'h'}
        ])
    } , 1000)
})
return news
```

Callback To Promise(2)

```
loadNews(function (result){
  validateNews(result, function(news){
    sortByLatest(news, function(latest){
      latest5News(latest, function(final){
        console.log('final');
        console.log(final)
      })
    })
  })
})

console.log('Not Blocking');
```

```
function validateNewsP(result){
//TODO SOMETING
return result
}

function sortByLatestP(result){
//TODO SOMETING
return result
}

const newsP = loadNewsP()
    .then(validateNewsP)
    .then(sortByLatestP)
    .then(e => {
        console.log('final')
        console.log(final)
        })
```

Fetch

Async/Await

Async Function

```
// Normal Function
function add(x,y){
  return x + y;
}

// Async Function
async function add(x,y){
  return x + y;
}
```

Promise

```
function doubleAfter2Seconds(x) {
  return new Promise(resolve => {
    setTimeout(() => {
      resolve(x * 2);
    }, 2000);
  });
}

function addPromise(x){
  return new Promise(resolve => {
    doubleAfter2Seconds(10).then((a) => {
      doubleAfter2Seconds(20).then((b) => {
        doubleAfter2Seconds(30).then((c) => {
          resolve(x + a + b + c);
        })
      })
    })
  });
}
```

```
addPromise(10).then((sum) => {
    console.log(sum);
});
```

Convert Promise To Await

```
function doubleAfter2Seconds(x) {
  return new Promise(resolve => {
    setTimeout(() => {
      resolve(x * 2);
    }, 2000);
  });
}

async function addAsync(x) {
  const a = await doubleAfter2Seconds(10);
  const b = await doubleAfter2Seconds(20);
  const c = await doubleAfter2Seconds(30);
  return x + a + b + c;
}
```

```
addAsync(10).then((sum) => {
    console.log(sum);
});
```


jQuery
write less, do more.

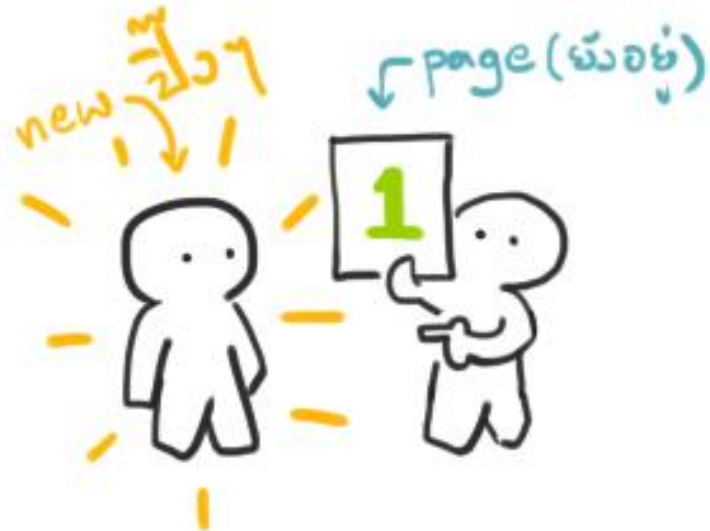

AJAX
Asynchronous Javascript And XML

แบบธรรมดา

แบบ Ajax

แบบธรรมดา

แบบ Ajax

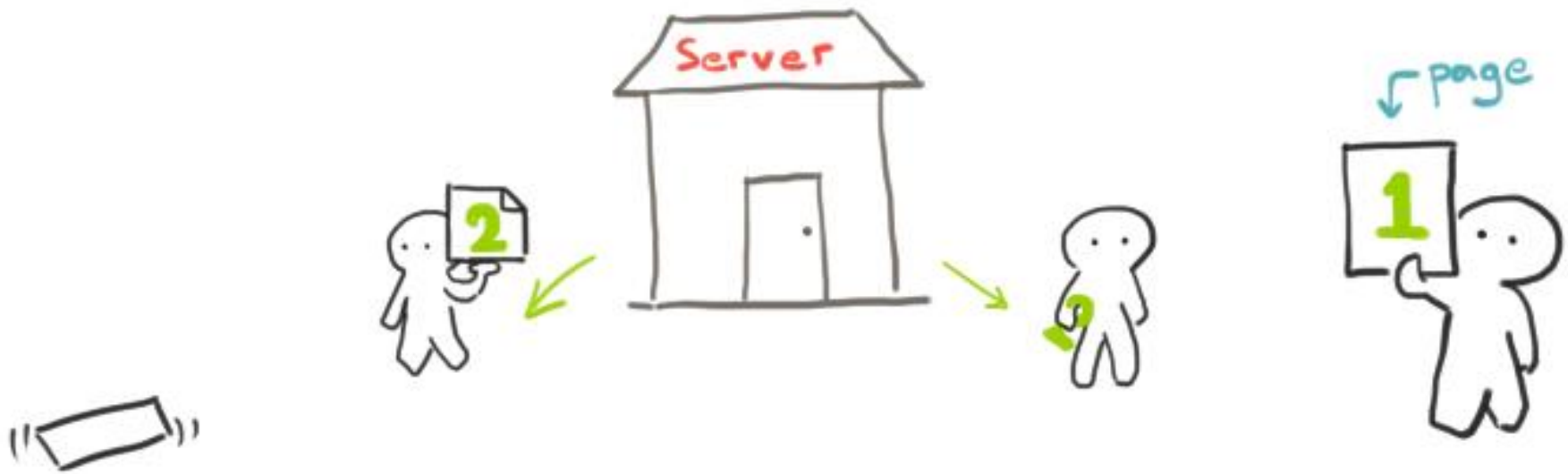


แบบธรรมดา

แบบ Ajax

แบบธรรมดา

แบบ Ajax



แบบธรรมดา

แบบ Ajax

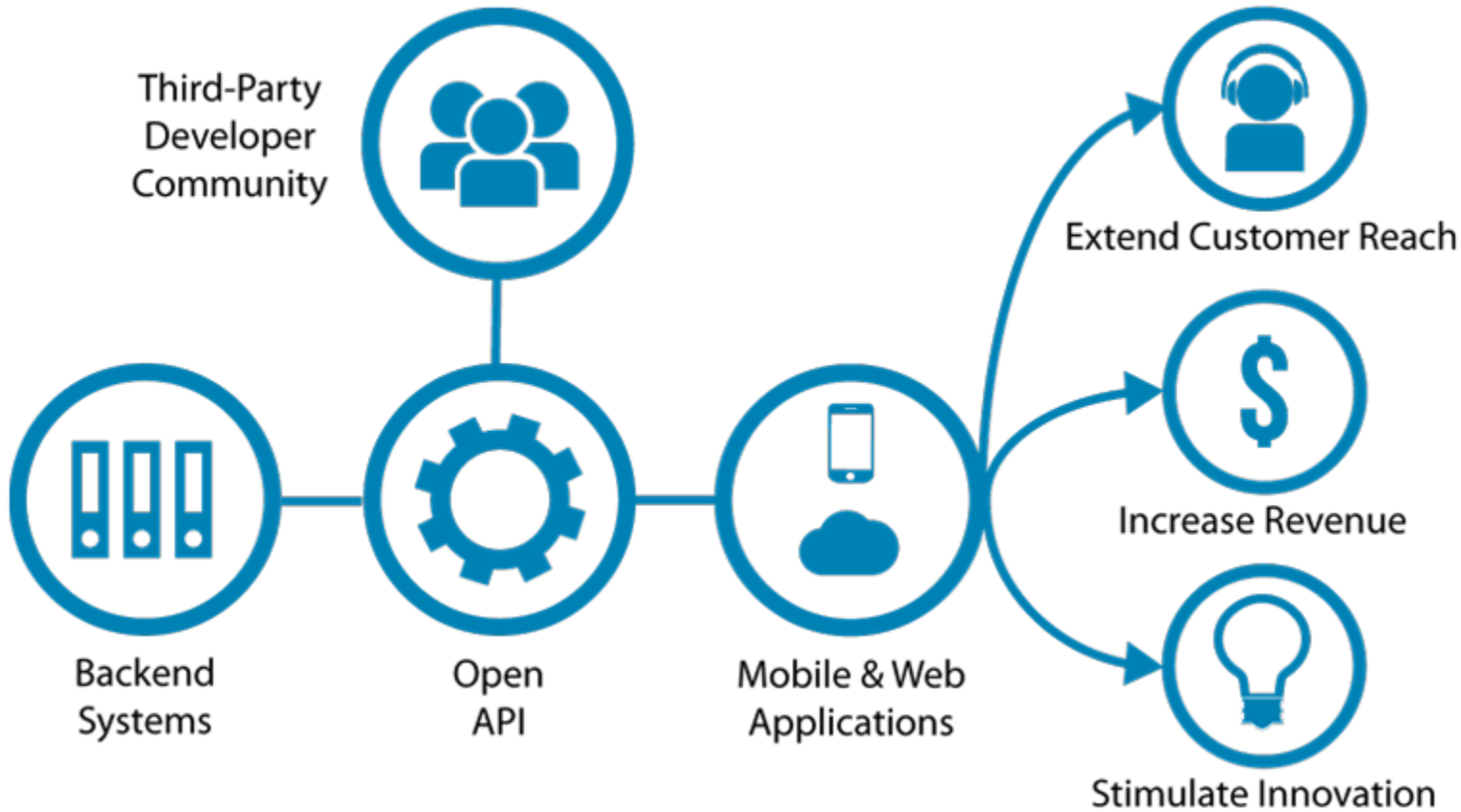
Ajax XHR

```
const xmlhttp = new XMLHttpRequest();

xmlhttp.onreadystatechange = function() {

    if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
        const response = JSON.parse(xmlhttp.response);
        console.log(response);
    }

};

xmlhttp.open('GET', 'https://www.reddit.com/.json', true);
xmlhttp.send();
```

jQuery Ajax

```
$.ajax({
        method: 'GET',
        url: 'https://www.reddit.com/.json'
})

.done((response) => {
        console.log(response);
});
```


API
Application Programming Interface


Third-Party
Developer
Community
Backend
Systems
Open
API
Mobile & Web
Applications
Extend Customer Reach
Increase Revenue
Stimulate Innovation

{fetch API}

Fetch

```
fetch(url) // Call the fetch function passing the url of the API as a parameter

.then(function() {
  // Your code for handling the data you get from the API
})

.catch(function() {
  // This is where you run code if the server returns any errors
});
```

```
fetch(url)
.then((resp) => resp.json()) // Transform the data into json
.then(function(data) {
    // Create and append the li's to the ul
    })
})
```

XHR vs jQuery vs Fetch

```
const xmlhttp = new XMLHttpRequest();

xmlhttp.onreadystatechange = function() {

    if (xmlhttp.readyState == 4
        && xmlhttp.status == 200) {
        const response = JSON.parse(xmlhttp.response);
        console.log(response);
    }

};

xmlhttp.open('GET', 'https://www.reddit.com/.json', true);
xmlhttp.send();
```

```
fetch('https://www.reddit.com/.json', { method: 'GET' })
.then((res) => res.json())
.then((data) => console.log(data));
```

```
$.ajax({
        method: 'GET',
        url:
'https://www.reddit.com/.json'
})

.done((response) => {
        console.log(response);
});
```